sensors

Article

Minimizing Delay and Power Consumption at the Edge

Erol Gelenbe [1,2,3]

[1] Institute of Theoretical & Applied Informatics, Polish Academy of Sciences (IITiS-PAN), 44-100 Gliwice, Poland; seg@iitis.pl
[2] Université Côte d'Azur, CNRS I3S, 06107 Nice, France
[3] Department of Engineering, King's College, London SE1 8WA, UK

Abstract: Edge computing systems must offer low latency at low cost and low power consumption for sensors and other applications, including the IoT, smart vehicles, smart homes, and 6G. Thus, substantial research has been conducted to identify optimum task allocation schemes in this context using non-linear optimization, machine learning, and market-based algorithms. Prior work has mainly focused on two methodologies: (i) formulating non-linear optimizations that lead to NP-hard problems, which are processed via heuristics, and (ii) using AI-based formulations, such as reinforcement learning, that are then tested with simulations. These prior approaches have two shortcomings: (a) there is no guarantee that optimum solutions are achieved, and (b) they do not provide an explicit formula for the fraction of tasks that are allocated to the different servers to achieve a specified optimum. This paper offers a radically different and mathematically based principled method that explicitly computes the optimum fraction of jobs that should be allocated to the different servers to (1) minimize the average latency (delay) of the jobs that are allocated to the edge servers and (2) minimize the average energy consumption of these jobs at the set of edge servers. These results are obtained with a mathematical model of a multiple-server edge system that is managed by a task distribution platform, whose equations are derived and solved using methods from stochastic processes. This approach has low computational cost and provides simple linear complexity formulas to compute the fraction of tasks that should be assigned to the different servers to achieve minimum latency and minimum energy consumption.

Keywords: edge computing; sensor networks; edge computing; latency minimization; reducing energy consumption; G-networks; analytical solution

Academic Editors: Yuh-Shyan Chen and Wei Yi

Received: 16 October 2024
Revised: 3 January 2025
Accepted: 9 January 2025
Published: 16 January 2025

Citation: Gelenbe, E. Minimizing Delay and Power Consumption at the Edge. *Sensors* **2025**, *25*, 502. https://doi.org/10.3390/s25020502

1. Introduction

The advent of the Internet of Things (IoT) and related technologies, such as smart homes, smart vehicles, 5th generation (5G) networks, and beyond 5G, increase the need for high throughput, low task delays, and low energy consumption through the development of systems that provide computing and communication services at the edge [1,2]. While radio access networks (RANs) and mobile base stations can massively increase the bandwidth and throughput that is offered to end users through these technologies, applications are also being moved from cloud computing platforms to the edge of the Internet [3–5] to achieve high throughput with low latency and lower energy consumption [6,7]. Motivated by these developments, much research has been conducted to allocate tasks in edge systems in a manner that attempts to minimize latency and energy consumption using non-linear optimization techniques [8,9] leading to NP-hard problems, which are processed with various heuristics and approximations, or with AI-based approaches [10,11], such as reinforcement learning. These previous approaches have some shortcomings: (a) there is no guarantee that optimum solutions are achieved, and (b) they do not provide a clear indication of the fraction of tasks that should be allocated to the different servers to achieve a specified optimum. Also, the parameters that are used by these methods must be measured and updated to construct the required algorithms; the methods are computationally costly, with additional overhead and energy consumption required for lightweight edge systems. In addition, these approaches do not provide insight into the key parameters, such as the task allocation rates or proportion of tasks that should be sent to different servers, to guarantee that the system will operate at or near its optimum point.

Thus, this paper proposes a radically different, mathematically based, and principled approach that explicitly computes the optimum fraction of jobs that should be allocated to the different servers to either (1) minimize the average latency (delay) of the jobs that are allocated to the edge servers or (2) minimize the average energy consumption of the jobs that use the edge servers. To achieve these objectives, this paper develops a mathematical model of a multiple-server system that is managed by a task dispatching platform (DP). The model equations are derived and solved using methods from stochastic processes. We then use this theoretical framework to explicitly derive the optimum workload distribution that minimizes latency. The paper then uses a similar approach to derive an explicit expression for the share of workload that should be allocated to each edge server that minimizes the system's additional energy consumption per task. The analytical approach we develop has a low computational cost and provides detailed insight into the fraction of tasks that are allocated to the different servers to achieve minimum latency and minimum energy consumption.

1.1. The Main Results Presented in This Paper

After the review of related work on the design of task-dispatching algorithms that optimize edge performance discussed in Section 1.2, the architecture of an edge system that includes a decision platform (DP) that dispatches incoming external tasks to a set of n servers is presented in Section 2. Then, the notation and symbols used in the paper are summarized in Section 2.1. All the proofs related to the theoretical developments in the paper are presented in detail in separate appendix sections that are clearly linked to the sections where the results are presented.

A novel mathematical model of an edge system composed of the DP that sends tasks to n servers is presented in Section 3. The Key Product Form Result for this model is stated and proved in Theorem 1, and Lemma 1 shows that its solution accounts for the processing of all the tasks that enter the system. Then, in Section 4, we show how the decision variables $C_i,\ 1 \leq i \leq n$, which combine the requests from the n servers with the task assignment decisions that are made by the DP to each server, affect the average latency of externally arriving tasks at the DP.

Then, Section 5 derives the task allocation policy that minimizes the average response time for all tasks being processed at the n servers in the system. Section 6 discusses the power consumption of edge servers based on power measurements that were made on NUCs and other processors, and we derive a policy that depends on the known parameters of each server to share the tasks between servers to guarantee that the average **energy consumption** for incoming tasks at the edge is minimized.

Finally, Section 7 provides conclusions and directions for further work.

1.2. Related Work

There has been considerable work on the design of algorithms for distributed system management and task distribution to reduce response times for tasks and maximize data transfer throughputs [12,13]. Real-time techniques have been developed to this effect [14], and various heuristics have often been tested in simulated environments to balance load and reduce response times [15,16]. Energy consumption has been of increasing concern over the last decade because of the steady increase observed over this period in the power consumption of ICT [5,17,18].

Recent research in this area has been primarily motivated by the need for low-cost distributed systems that offer computation and data-intensive applications close to the network edge to achieve low latency [19] for mobile technologies, the IoT, and smart vehicles [20]. Another motivation is the need for distributed computing facilities that locally serve small-scale applications, such as smart homes [21], and in some recent work [22], a system was considered where tasks that arrive at an edge server are either directly executed there or off-loaded to a different server.

As early as the 1990s, the research community proposed AI-based dynamic network management techniques [23–26] that were later facilitated by the introduction of Software Defined Networks [27] to achieve improvements in network performance and security [28,29]. Attempts have been made to use reinforcement learning or, more broadly, machine learning [30–32] as a tool to reduce latency and achieve power savings for tasks that are sensitive to the "quality of service" [33]. Other work has integrated security needs by managing tasks and flows of data so that insecure servers and networks may be dynamically avoided [34,35]. Market-based bidding techniques and games to design low computational cost algorithms that have been shown to offer fast solutions at low cost during simulations [36,37]. Some practical experiments have tested AI in distributed edge systems using Software Defined Networks to reduce latency and improve power consumption [38]. Since edge systems often fulfill multiple functions and support a variety of users, the resulting optimization problems are often NP-hard, and heuristic approximations are often investigated [39].

2. System Description

We consider an edge distributed computing system composed of a Dispatching Platform (DP) that resides on a separate server with n machines or servers, $S_1, \dots, S_n$, that together form a cluster that is accessible through the Internet. Each S_i receives local tasks to execute, as well as tasks that are allocated to it by the DP. External tasks to be executed by the edge system are received by the DP and assigned to the servers based on requests from the servers.

The base station or external user shown in Figure 1 sends tasks to the DP, where they are stored in an input queue as they wait for task requests from the n edge servers.

- When any S_i completes the current task that it is executing, it makes a task request from the DP with probability $1 \leq p_i \leq 1$. If the DP task input queue is empty, then the request is simply rejected by the DP. If the DP task input queue contains at least one task, then the DP assigns the task to S_i with probability $0 \leq a_i \leq 1$.
- Thus, when S_i terminates an ongoing task, a task from the incoming pool is dispatched by the DP to S_i with probability $C_i = p_i a_i$, **provided that the input queue at the DP is not empty**. If the DP queue is empty, obviously, no task can be sent. This is equivalent to assuming that when a server S_i informs the DP that it has terminated a task, then the DP allocates a task to S_i with probability $0 \leq C_i \leq 1$ if the DP has a task waiting at its input. If there are no tasks waiting at the DP, then the request from S_i is rejected.
- Note that task endings at the different servers occur asynchronously with each other, and the decision of the DP is simply to send or not to send a new task to S_i.
- Thus, each server has a queue of tasks, some of which have been sent by the DP and others are local tasks that it receives and executes.

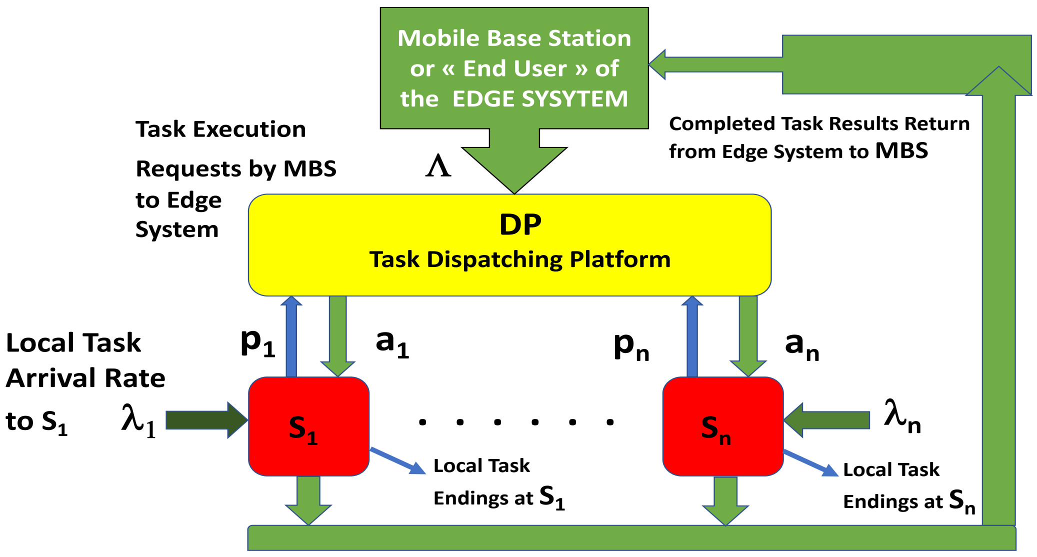
Mobile Base Station or « End User » of the EDGE SYSYTEM
Task Execution Requests by MBS to Edge System
Λ
Completed Task Results Return from Edge System to MBS
DP
Task Dispatching Platform
Local Task Arrival Rate to S_1
λ_1
p_1
a_1
p_n
a_n
S_1
S_n
λ_n
Local Task Endings at S_1
Local Task Endings at S_n

Figure 1. Architecture of an edge system that allocates incoming tasks to a set of locally connected servers for edge computing [40]. It is composed of a Dispatching Platform (DP) that dynamically exploits the n distinct servers' available capacity to allocate tasks to minimize average task delay or to minimize total power consumption. Each server has its own incoming local flow of tasks, and each server requests and receives tasks from the DP.

External tasks arrive at the DP at a rate $\Lambda > 0$ (tasks per second), while each S_i receives "locally generated tasks", e.g., from its local owner or user or as part of its operating system at the following rate:

$$\lambda_i \geq 0,\ \lambda = \sum_{i=1}^{n} \lambda_i\ . \qquad (1)$$

The average execution time of each task at S_i is denoted by μ_i^{-1}.

The DP's objective is to minimize the total average waiting time at the DP and the average response time at all the n servers. However, it also aims to reduce the overall energy consumption of the system. On the other hand, each S_i must execute all the tasks it has received locally, as well as those that it has requested from the DP and that the DP has allocated to it. The S_i may need to generate income from the external tasks it receives from the DP. On the other hand, it also needs to provide low latency (i.e., low response time) for all the tasks it receives. The DP, as well as all S_i, also aims to keep the overall average energy consumption as low as possible because of the cost of the energy and to achieve greater sustainability.

2.1. Summary of Notation and Symbols and Abbreviations

In this sub-section, we present and define all the **symbols that are used throughout this paper**.

- $t \geq 0$ is the real-valued time variable.
- DP is the task dispatching platform that transfers tasks from the end users to the servers.
- S_i denotes a server that receives tasks assigned by the DP, as well as "locally generated tasks", e.g., from its local owner or user or as part of its operating system.
- $\Lambda > 0$ is the rate of arrival of external tasks to the DP.
- λ_i is the rate of arrival of locally generated tasks to S_i.
- $\mu_i > 0$ is the average service rate for tasks at the server S_i. Thus, the average service time per task at S_i is $\frac{1}{\mu_i}$.
- We define $\rho_i = \frac{\lambda_i}{\mu_i}$, $\lambda = \sum_{i=1}^{n} \lambda_i$, and $\mu = \sum_{i=1}^{n} \mu_i$.

- p_i, $0 \le p_i \le 1$, is the probability that, when S_i completes the current task that it is executing, it requests to receive a task from the DP.
- a_i, $0 \le a_i \le 1$, is the probability that the DP accepts S_i's request when the DP's input queue is non-empty.
- $C_i = p_i \dot{a}_i$ is the probability that when S_i asks for a new task from the DP, it receives it provided that a new task is available at the DP.
- $y(t) \ge 0$ is the non-negative integer-valued length of the queue of externally arriving tasks waiting at the Dispatching Platform (DP) at time t.
- $y_i(t) \ge 0$ is the integer-valued total number (queue length) of all the tasks that are in the queue at S_i at time t.
- k is a particular value of $y(t)$.
- k_i is a particular value of $y_i(t)$, and we define the vectors as follows:

$$\begin{aligned} &Y(t) = (y(t), y_1(t), \ldots, y_n(t)), \\ &K = (k, k_1, \ldots, k_n) \; . \end{aligned}$$

- The following vectors are related to $K = (k, k_1, \ldots, k_n)$, *where* $k \ge 0$, $k_i \ge 0$:

$$\begin{aligned} &K^{-0} = (k-1, k_1, \ldots, k_n) \; if \; k > 0, \\ &K^{+0} = (k+1, k_1, \ldots, k_n), \\ &K^{-i} = (k, k_1, \ldots, k_i - 1, \ldots, k_n) \; if \; k_i > 0, \\ &K^{+i} = (k, k_1, \ldots, k_i + 1, \ldots, k_n) \; . \end{aligned} \quad (2)$$

- Φ_i is the fraction of external user tasks that the DP allocates to S_i.
- Φ_i^+ is the fraction of external user tasks that the DP allocates to S_i to minimize the average task response time of the edge system.
- Φ_i^* is the fraction of external user tasks that the DP allocates to S_i to minimize the average energy consumption per external task assigned to the edge system.
- $X_i = \lambda_i + \Phi_i \Lambda$ is the total arrival rate of tasks to server S_i, i.e., the load of S_i.
- X_{i1} is the upper bound for the linear approximation of the power consumption of S_i, and $X_{i1} < \mu_i$
- $q_i = \frac{\lambda_i + \Phi_i \Lambda}{\mu_i}$ is the utilization rate of server S_i. If $q_i < 0$, it can be interpreted as the probability that S_i is busy processing tasks.
- R_{DP} is the average response time at the DP for externally arriving tasks.
- R_S is the average response time of all tasks at the n servers.
- π_{i0} is the power consumption of server S_i when the server is idle, i.e., when $X_i = 0$.
- π_{iM} is the maximum power consumption of server S_i. It is attained when X_i is just under the value μ_i.
- $\alpha_i > 0$ is the approximate linear increase in power consumption of S_i as a function of the load X_i.
- $\pi_i(X_i) = \pi_{i0} + \alpha_i X_i$ is the approximate power consumption of S_i when its load is X_i, for $X_i < \mu_i$.
- π_i' is the derivative of $\pi_i(X_i)$ with respect to Φ_i.
- π_i'' is the second derivative of $\pi_i(X_i)$ with respect to Φ_i.
- E is the average **energy consumption** of the externally arriving tasks that are assigned by the DP to the different servers, and $E = \sum_{i=1}^{n} \Phi_i \pi_i(X_i) \mu_i^{-1}$.

3. Analytical Solution for the Dispatching Platform (DP) and Its *n* Servers

In this section, we construct a G-Network with triggered customer movement [41,42], where the service times at all S_i are mutually independent and exponentially distributed random variables, with parameter μ_i for S_i, and the interarrival times of external tasks to the DP is a Poisson process of rate Λ. The arrivals of local tasks at each S_i constitute a mutually independent Poisson process with rate λ_i and are independent of all the service times at the servers. Thus, in a small time interval of length Δt, an external task arrival occurs to the DP with probability $\Lambda\Delta t + o(\Delta t)$, a local task arrives to any server S_i with probability $\lambda_i \Delta t + o(\Delta t)$, and provided that there is a local task at S_i (i.e. $k_i > 0$), a local task ends its service at S_i with probability $\mu_i \Delta t + o(\Delta t)$. Here, $o(\Delta t)$ represents a function that tends to zero with Δt, i.e., $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$.

Also, when a service completes at S_i, the server requests to receive a new task from the DP with probability p_i, which is allocated instantaneously with probability a_i if $k > 0$ or refused with probability $(1 - a_i)$ or accepted with probability p_i and not allocated when $k = 0$. Thus, the following state transitions occur:

- $K \to K^{+0}$ with probability $\Lambda\Delta t + o(\Delta t)$.
- $K \to K^{+i}$ with probability $\lambda_i \Delta t + o(\Delta t)$.
- $K^{+0} \to K$ with probability $\mu_i C_i \Delta t + o(\Delta t)$ when $k_i > 0$ (a task at S_i departs but is immediately replaced by a task from the DP).
- $K^{+i} \to K$, with probability $\mu_i C_i \Delta t + o(\Delta t)$ when $k = 0$ (a task at S_i departs; the request for a new task is made but the DP queue is empty (i.e., $k = 0$ and, therefore, the DP has no tasks to send to S_i).
- $K^{+i} \to K$ with probability $\mu_i(1 - C_i)\Delta t + o(\Delta t)$ obtained from

$$[\mu_i(1 - p_i) + \mu_i p_i (1 - a_i)]\Delta t + o(\Delta t) = \mu_i(1 - C_i)\Delta t + o(\Delta t) \tag{3}$$

  independently of the value of k or k_i; note that these values refer to the quantities in the vector $K = (k, k_1, \ldots, k_n)$.
- $K \to K$, with probability $1 - (\Lambda + \lambda_i + \mu_i 1[k_i > 0])\Delta t + o(\delta t)$.

Then, the probability $p(K, t) = Prob[Y(t) = K]$ satisfies the following system (4) of Chapman–Kolmogorov differential-difference equations:

$$\begin{aligned}\frac{dp(K,t)}{dt} &= -p(K,t)\big[\Lambda + \sum_{i=1}^{n}(\mu_i 1[k_i > 0] + \lambda_i)\big] + \Lambda p(K^{-0},t)1[k > 0] \\ &\quad + \sum_{i=1}^{n}\big[\lambda_i p(K^{-i},t)1[k_i > 0] + \mu_i C_i p(K^{+0},t)1[k_i > 0] \\ &\quad + \mu_i C_i p(K^{+i},t)1[k = 0] + \mu_i(1 - C_i)p(K^{+i},t)\big].\end{aligned} \tag{4}$$

We now state the following result, which we use throughout this paper. **The proof of Theorem 1 is detailed in Appendix A .**

Theorem 1 (Key Product Form Result). *Assume that the arrival processes whose rates are* Λ, $\lambda_1, \ldots, \lambda_n$ *are all independent Poisson processes and that the service rates* μ_i, $1 \le i \le n$, *are parameters of independent exponentially distributed random variables, which are also independent of the inter-arrival times. Then, if the system of simultaneous non-linear equations*

$$q = \frac{\Lambda}{\sum_{i=1}^{n} q_i \mu_i C_i},\ \ q_i = \frac{\lambda_i + q q_i \mu_i C_i}{\mu_i} = \frac{\rho_i}{1 - qC_i},\ 1 \le i \le n\,, \tag{5}$$

has a solution that satisfies $0 < q < 1$, $0 < q_i < 1$, ***then this solution is unique***, *and*

$$\begin{aligned}&\lim_{\mathbf{t}\to\infty} \mathbf{Prob}\big[\mathbf{x(t) = k, x_1(t) = k_1, \ldots, x_n(t) = k_n}\big] \\ &\qquad = \mathbf{q^k(1 - q)\prod_{i=1}^{n} q_i^{k_i}(1 - q_i)},\end{aligned} \tag{6}$$

where

$$\mathbf{q} = \lim_{\mathbf{t}\to\infty} \mathbf{Prob}\big[\mathbf{x}(\mathbf{t}) > \mathbf{0}\big], \ \ \mathbf{q_i} = \lim_{\mathbf{t}\to\infty} \mathbf{Prob}\big[\mathbf{x_i}(\mathbf{t}) > \mathbf{0}\big]. \tag{7}$$

Note: *The denominator of the expression for q in (5) represents the fact that each server S_i will notify the DP with probability p_i when S_i's ongoing job ends, that it is ready to receive a task from the DP, and that the DP will respond by sending a task to S_i with probability a_i so that $C_i = p_i \cdot a_i$. The rate at which such requests arrive to the DP from S_i is, therefore, $q_i\mu_i p_i$, and the rate at which the DP sends tasks to S_i is $q_i\mu_i C_i$. Note that both of the equations in (5) are **non-linear**, contrary to those of an ordinary "Jackson" (open) or "Gordon–Newell" (closed) product-form queueing network [43,44].*

Corollary 1. *From (6), it is easy to show that when $q < 1$, the average total number of jobs at steady state N_{DP} in the input queue to the DP is*

$$N_{DP} = \frac{q}{1-q}\,, \tag{8}$$

and the average total number of jobs at steady state N_i that are in the input queue of S_i is

$$N_i = \frac{q_i}{1-q_i}. \tag{9}$$

The expression for q_i in (5) has the intuitive property that we now prove; namely, when the stationary solution exists, the total incoming flow of jobs to the DP and the servers S_i is identical to the outgoing flow of jobs whose service ends at the n servers, which we use in the proof of Theorem 1 given in Appendix A.

Lemma 1. *Let us denote*

$$\lambda = \sum_{i=1}^{n} \lambda_i\,. \tag{10}$$

Then, if $0 < q_i < 1,\ 0 < q < 1$, it follows that

$$\sum_{i=1}^{n} q_i\mu_i = \Lambda + \lambda\,. \tag{11}$$

Remark 1. *The expression (11) is an intuitive "flow conservation" identity at steady state for a* stable system, *which states that all the work that arrives at the DP or that arrives locally to the n servers is eventually processed by one of the n servers.*

Proof of Lemma 1. As a consequence of the expressions for q and q_i in (5), we can write

$$\sum_{i=1}^{n} q_i\mu_i \ = \ \sum_{i=1}^{n} \lambda_i[1 + \sum_{l=1}^{\infty}(qC_i)^l],$$

and using the expression for q in (5), we obtain

$$\sum_{i=1}^{n} q_i\mu_i \ = \ \lambda + \frac{\Lambda}{\sum_{j=1}^{n} q_j\mu_j C_j}.\sum_{i=1}^{n}\frac{\lambda_i}{1-qC_i} = \lambda + \frac{\Lambda}{\sum_{j=1}^{n} q_j\mu_j C_j}.\sum_{j=1}^{n} q_j\mu_j C_j = \lambda + \Lambda, \tag{12}$$

which completes the proof. □

Corollary of Lemma 1. *Since we assume that* $0 < q_i < 1,\ 1 \le i \le n$, the following holds:

$$\textit{Denoting } \rho_i = \frac{\lambda_i}{\mu_i}, \textit{ we have} : \rho_i < 1 - qC_i, \textit{ and hence } C_i < \frac{1-\rho_i}{q}. \tag{13}$$

4. Minimizing the Average Response Time or Average Delay at the DP

The well-known "Little's Formula" [45] can be used to compute the average response time of tasks entering through the DP and of tasks entering the edge system composed of n servers. Here, Λ is the total arrival rate of externally arriving tasks to the DP and $qq_i\mu_iC_i$ is the arrival rate of tasks from the DP to server S_i.

Since Λ is the total arrival rate of such tasks, if R_{DP} denotes the average response time of tasks at the DP before they are assigned to a server, by Little's Formula and Equation (8) in Corollary 1, we have

$$R_{DP} \quad = \quad \frac{N_{DP}}{\Lambda} = \frac{1}{\Lambda}\frac{q}{1-q}, \tag{14}$$

and we would like to know how we should choose C_i, $i = 1, \ldots, n$, to minimize R_{DP}. To this effect, the following result is needed:

Theorem 2. *Let $0 < q_i < 1$, and denote $D_i = \frac{dq}{dC_i}$, $d_{ij} = \frac{dD_i}{dC_j}$. It follows that $D_i < 0$, $d_{ij} < 0$, and $d_{ii} > 0$ for $i\,, j \ = 1, \ldots n$, $j \neq i$.*

The proof of Theorem 2 is given in Appendix B.

Using (14), we can derive

$$\frac{dR_{DP}}{dC_i} = \frac{1}{\Lambda}\frac{D_i}{1-q}, \ \frac{d^2R_{DP}}{dC_i^2} = \frac{1}{\Lambda}\frac{d_{ii}(1-q)+D_i^2}{(1-q)^2}. \tag{15}$$

Then, also using Theorem 2, we have $\frac{dR_{DP}}{dC_i} < 0$ and $\frac{d^2R_{DP}}{dC_i^2} > 0$ for $i = 1, \ldots, n$.

Theorem 3. *Using (14), (15), and Theorem 2, it follows that for fixed Λ, the average response time R_{DP} for a task that arrives from the MBS or an external user to the DP, until it is assigned to one of the server input queues, is minimized with respect to $0 \leq C_i \leq 1$ by taking the largest possible value of C_i, which is $C_i = 1$. When all the C_i, $1 \leq i \leq n$, are set to $C_i = 1$, then R_{DP} attains its minimum value wth respect to the vector $C = (C_1, \ldots, C_n)$.*

5. Minimizing the Average Response Time R_S at the Edge Servers

Different edge servers have different task processing rates μ_i and different local task arrival rates λ_i. Therefore, it is worth understanding how the DP should share the tasks that it receives among the edge servers to achieve a minimum average response time R_S for **all the tasks**, both those that arrive locally to each server and those that are assigned by the DP. Let Φ_i denote the proportion of incoming external tasks that the DP assigns to server S_i:

$$\Phi_i = \frac{q_i\mu_iC_i}{\sum_{j=1}^n q_j\mu_jC_j}, \quad \sum_{j=1}^n \Phi_j = 1, \tag{16}$$

so that the total arrival rate of tasks arriving to reach S_i is $\lambda_i + \Lambda\Phi_i$. As a result, when $q < 1, q_i < 1$, $i = 1, \ldots, n$, in steady state, the average number of tasks N_S at the n servers can be obtained from (6) in Theorem 1 as

$$N_S = \sum_{i=1}^n N_i = \sum_{i=1}^n \frac{q_i}{1-q_i}, \ \textit{where } q_i = \frac{\lambda_i + \Lambda\Phi_i}{\mu_i}\ , \tag{17}$$

and by Little's Theorem, we have

$$R_S = \frac{1}{\Lambda+\lambda}\sum_{i=1}^n \frac{q_i}{1-q_i} = \frac{1}{\Lambda+\lambda}\sum_{i=1}^n \frac{\lambda_i+\Lambda\Phi_i}{\mu_i-\lambda_i-\Lambda\Phi_i}, \ \textit{where } \lambda = \sum_{i=1}^n \lambda_i\,. \tag{18}$$

We can now state the following result, **whose proof is given in Appendix C**.

Theorem 4. *Let $0 \leq q < 1$, $0 \leq q_j < 1$ for $1 \leq j \leq n$. Then, the average response time at steady state for all tasks that are processed by the n servers, denoted by R_S,* ***attains its global minimum with respect to the vector $\Phi = (\Phi_1, \ldots, \Phi_n)$ when Φ_j is equal to Φ_j^*****:*

$$\begin{aligned}\Phi_j^* &= \frac{\mu_j - \lambda_j}{\Lambda} - \frac{\mu - \Lambda - \lambda}{\Lambda} \frac{\sqrt{\frac{\mu_j}{\mu_1}}}{[\sum_{i=1}^n \sqrt{\frac{\mu_i}{\mu_1}}]},\ 1 \leq j \leq n,\ where\ \mu = \sum_{j=1}^n \mu_j\,,\\ &= \frac{\sqrt{\frac{\mu_j}{\mu_1}}}{[\sum_{i=1}^n \sqrt{\frac{\mu_i}{\mu_1}}]} + \frac{1}{\Lambda}[\mu_j - \lambda_j - (\mu - \lambda)\frac{\sqrt{\frac{\mu_j}{\mu_1}}}{[\sum_{i=1}^n \sqrt{\frac{\mu_i}{\mu_1}}]}]\,,\ 1 \leq j \leq n\,. \end{aligned} \tag{19}$$

Communication Overhead and Computational Cost. *From (19), we see that the terms*

$$\mu\ and\ \frac{\sqrt{\frac{\mu_j}{\mu_1}}}{[\sum_{i=1}^n \sqrt{\frac{\mu_i}{\mu_1}}]}, \tag{20}$$

can be computed in advance once and for all for a given set of n servers since they only depend on the server speed parameters μ_i, $i = 1, \ldots, n$, and do not need to be re-computed for each decision. Λ is known by the DP, which locally monitors the external arrival rate of tasks, and no communication is needed to update Λ. The parameters λ_j must be updated in (19) and should be sent by each S_j to the DP (where the task assignment decision is taken) each time λ_j changes. This boils down to a periodic communication overhead of, at most, a total of n packets that are sent from the servers to the DP. From a computational standpoint, obtaining (19) only requires four additions and subtractions and two multiplications for each of the n values Φ_j^*.

Corollary 2. *The minimum value of R_S, denoted R_S^* is*

$$R_S^* = \frac{1}{\Lambda + \lambda} \sum_{j=1}^n \frac{\lambda_j + \Lambda\Phi_j^*}{\mu_j - \lambda_j - \Lambda\Phi_j^*} = \frac{1}{\Lambda + \lambda} \sum_{j=1}^n \frac{\mu_j}{\mu_j - \sqrt{\frac{\mu_j}{\mu_1}}\lambda_j - \Lambda\Phi_j^*}. \tag{21}$$

Corollary 3. *In many cases of interest, an edge system is composed* ***of the DP and n identical servers S_i, which, in general, have different local loads*** *λ_i so that we have $\mu_i = \mu$, $1 \leq i \leq n$. In this case,* $\mathbf{R_S}$ *is minimized when*

$$\Phi_i^* = \Phi_1^* + \frac{\lambda_1 - \lambda_i}{\Lambda},\ 2 \leq i \leq n,\ \ \Phi_1^* = \frac{1}{n}[1 + \frac{\sum_{i=2}^n (\lambda_i - \lambda_1)}{\Lambda}]\,. \tag{22}$$

6. Minimizing Energy Consumption

An important system performance metric of interest is the energy consumption of the system. As an example, the measured power and energy consumption characteristics of an Intel NUC processor [46] that is widely used in edge systems are shown in Figure 2 based on accurate measurements that were reported in [47].

Let us note from (11) and (12) that Λ is the total arrival rate of external tasks to the DP; these are, in turn, assigned by the DP to the n edge servers. Also, we define $X_i = \lambda_i + \Lambda\Phi_i$, where (as previously in this paper) λ_i is the local arrival rate of tasks to S_i and Φ_i is the fraction of externally arriving tasks that are allocated by the DP to S_i.

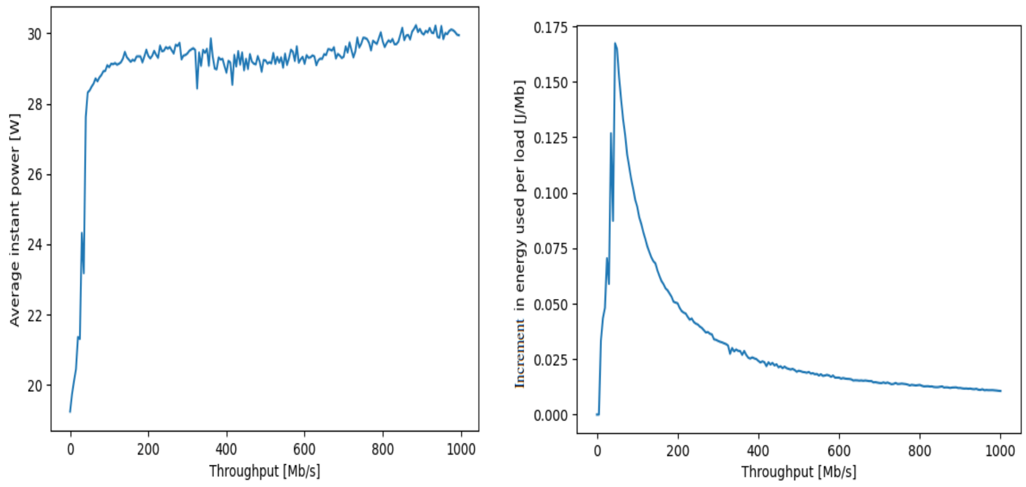

Figure 2. The curve on the left shows the power consumption that was measured on an NUC versus its overall arrival rate of workload. There is a substantial power consumption of close to 63% of its maximum value when the NUC is idle. We observe that the power consumption attains its maximum value of 30 Watts as the workload increases. The curve on the right shows the corresponding energy consumption per arriving request in Joules as a function of the load.

The curve on the left in Figure 2 shows the rise in the power consumption as a function of its load, expressed as the arrival rate of workload to the NUC, starting from a value of roughly 19 Watts when the NUC is idle and attaining a maximum value of approximately 30 Watts when the NUC is fully loaded. The curve on the right in Figure 2 shows the energy consumption in Joules per arriving request as a function of the total arrival rate of tasks X_i to server S_i.

Indeed, the curve on the left-hand-side of Figure 2 and the different measurement curves shown in Figure 3 also suggest the following representation for the power consumption $\pi_i(X_i)$ of server S_i (23), where $X_i = \lambda_i + \Lambda\Phi_i$, rising from the power consumption π_{i0} when S_i is idle, up to its maximum power consumption denoted by π_{iM}. Thus, these measurement results indicate that the power versus workload characteristics of a server may be represented by a piece-wise linear approximation consisting of a straight line from $X_i = 0$ to $X_i = X_{i1}$ with a positive slope and a second flat (nearly zero slope) straight line from X_{i1} to higher values of X_i. Also, X_{i1} is smaller than the maximum processing or service rate μ_i of server i. We, therefore, use this observation to express the approximation for $0 \leq X_i \leq X_{i1}$ with $\pi_i(X_{i1}) = \pi_{iM}$ as

$$\begin{aligned} \pi_i(X_i) &= \pi_{i0},\ if\ \ X_i = 0, \\ &= \pi_{i0} + \alpha_i X_i\ ,\ if\ \ 0 \leq X_i \leq X_{i1} < \mu_i\ , \end{aligned} \tag{23}$$

where $\alpha_i > 0$ is a positive constant that depends on the specific server being considered. We can then define the first and second derivatives of $\pi_i(X_i)$ with respect to Φ_i:

$$\pi_i' = \frac{d\pi_i(X_i)}{d\Phi_i},\ \pi_i'' = \frac{d^2\pi_i(X_i)}{d\Phi_i^2}\ . \tag{24}$$

When $i \neq 1$, we have, for $X_i < \mu_i$,

$$\pi_i' = \alpha_i\Lambda,\ \pi_i'' = 0,\ \ for\ \ \alpha_i > 0,\ when\ 0 \leq X_i < X_{i1}\ . \tag{25}$$

Also, since $\Phi_1 = 1 - \sum_{i=2}^{n} \Phi_i$, we have $\frac{d\Phi_1}{d\Phi_i} = -1$ for $i \neq 1$. Thus, the first and second derivatives of $\pi_1(X_1)$ with respect to Φ_i for $i \neq 1$ are

$$\frac{d\pi_1(X_1)}{d\Phi_i} = -\alpha_1 \Lambda, \; for\; \alpha_1 > 0, \; \frac{d^2\pi_1(X_1)}{d\Phi_i^2} = 0, \; for\; 0 \leq X_1 < X_{11}. \tag{26}$$

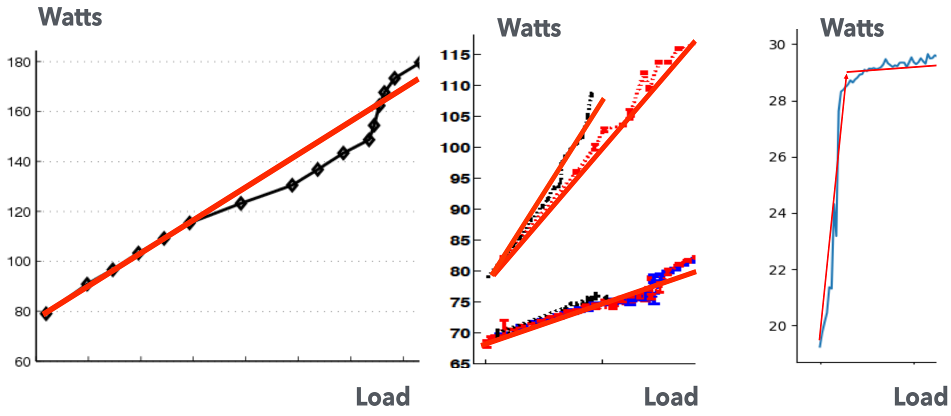

Figure 3. We illustrate the measured characteristics of the power consumption $\Pi_i(X_i)$ along the y-axis in Watts, versus the load X_i along the x-axis in tasks/sec for several different servers, showing the approximately linear increase in power consumption at some rate $\alpha_i > 0$, which depends on the characteristics of the different processors, between the zero load level (no task arrivals and the server is idle), which corresponds to π_{i0}, up to close to the maximum value of the power consumption that we denote by π_{iM}. Note that the value X_{1i} cannot exceed the maximum processing rate of jobs μ_i of S_i. The linear characteristic is displayed as a straight red line on top of the measured data that are also shown in the figure. The rightmost curve refers to the NUC whose characteristics are discussed in Figure 2.

Allocating Incoming Tasks to Minimize the Average Additional Energy Consumed by the Servers

If the DP sends an externally arriving task to server S_i, we know that the task waits for some time, and then it will be processed during μ_i^{-1} time units on average. If the power consumption of S_i is π_i and Φ_i is the probability that the DP has chosen to send the task to S_i, then the energy that is consumed by the task is simply $\pi_i \times \mu_i^{-1}$.

Therefore, the expected average energy consumption E for executing a task sent from the DP to the edge system composed of n servers is

$$E = \sum_{i=1}^{n} [\, \Phi_i \times \frac{\pi_i(X_i)}{\mu_i}]. \tag{27}$$

This leads us directly to the following result, **whose proof is given in Appendix D.**

Theorem 5. *Assuming the power consumption characteristic given in (23), the proportion of incoming traffic that should be allocated to server S_i to **minimize** E for $j = 2, \ldots, n$ is*

$$\Phi_j^+ = \Phi_1^+ \frac{\alpha_1 \mu_j}{\alpha_j \mu_1} + \frac{1}{2\Lambda\alpha_j}\left[\pi_{10}\frac{\mu_j}{\mu_1} - \pi_{j0}\right], \tag{28}$$

where

$$\Phi_1^+ = \frac{1 + \frac{1}{2\Lambda}\sum_{i=2}^{n}[\frac{\pi_{i0}}{\alpha_i} - \frac{\pi_{10}}{\mu_1}\frac{\mu_i}{\alpha_i}]}{1 + \frac{\alpha_1}{\mu_1}\sum_{i=2}^{n}\frac{\mu_i}{\alpha_i}}. \tag{29}$$

As would be expected, when all the servers are identical with $\pi_{i0} = \pi_{i1}$, $\alpha_i = \alpha_1$, $\mu_i = \mu_1$ for $i = 2, \ldots, n$, we have $\Phi_1^+ = \frac{1}{n}$, and $\Phi_j^+ = \Phi_1^+$, $2 \leq j \leq n$.

Communication Overhead and Computational Overhead. *Since the parameters α_j, μ_j, π_{i0} are fixed and can be known in advance for the servers S_j, $j = 1, \dots, n$, the terms $\sum_{i=2}^{n}[\frac{\pi_{i0}}{\alpha_i} - \frac{\pi_{10}}{\mu_1}\frac{\mu_i}{\alpha_i}]$, $1 + \frac{\alpha_1}{\mu_1}\sum_{i=2}^{n}\frac{\mu_i}{\alpha_i}$, $\frac{\alpha_1\mu_j}{\alpha_j\mu_1}$, and $\frac{1}{2\alpha_j}$ $[\pi_{10}\frac{\mu_j}{\mu_1} - \pi_{j0}]$ can be computed just one time in advance for $j = 2, \dots, n$. The only parameter in (28) and (29) that must be measured is Λ; it is measured directly by the DP, which uses it to compute the values of Φ_j that minimize E. Therefore, there is no communication overhead involved in choosing the fraction of externally arriving tasks assigned to each server to minimize the additional average energy consumption E. Considering the computational overhead, we note that the computation of Φ_i^+ involves an additional addition and two divisions. The computation of each of the remaining Φ_j^+ involves one additional multiplication, one division, and one addition. Thus, we see that the number of arithmetic operations needed to compute all of the n values of Φ_j^+ is 3n for each new value of Λ.*

7. Conclusions

Edge computing systems, composed of clusters of processors, are particularly important for supporting the low latency, high throughput, and low power consumption needs of mobile base stations and other communication systems. Their aim is to provide crucial low latency and sustainable low energy consuming services for the Internet of Things and support the transition of communications to 5G and 6th generation (6G) mobile networks. Thus, considerable work has been devoted to the design of different types of algorithms for configuring them, dynamically or statically, to optimize the allocation of tasks to edge system servers.

Much prior work has used machine learning, including reinforcement learning, nonlinear optimization methods, and market-based mechanisms, and some of these methods have been tested in experimental environments. Though this work has been extremely useful in generating experience about the manner in which edge systems can be implemented, it comes at the cost of extensive simulations and time-consuming real-system experimentations. Furthermore, the machine learning-based approaches, such as that in our earlier work [10,47], do not provide insight into the fraction of tasks that should be allocated to different servers to achieve optimality.

Thus, in the present work, we address the edge computing design process through an analytical model that results in explicit formulas for optimal task allocation, minimal task latency, and minimal energy consumption of the system as a whole. We show that this approach leads to simple formulas that provide the optimum share of externally arriving tasks that should be assigned to each edge server. We also observe that these formulas are computationally very simple and that they lead to very low communication overhead. In future work, we plan to prioritize the execution of locally generated tasks and remote tasks and include the effect of different types of tasks being executed in the system.

We also plan to implement the proposed algorithms in an experimental test bed and compare various machine learning-based algorithms and other simple heuristics (such as greedy algorithms) to see how close they can get to achieving the optimum performance obtained via the analytical approach.

Funding: This research was funded by the European Union's Horizon Europe research and innovation programme, DOSS Project under Grant Agreement No. 101120270, and by the UKRI Project No. 10034722.

Data Availability Statement: The data presented in this study are available on request from the author.

Acknowledgments: The author would like to thank the editors and anonymous reviewers for their valuable comments and suggestions.

Conflicts of Interest: The author declares no conflicts of interest.

Appendix A

Proof of Theorem 1 (Key Product Form Result). For the Equation (4) at steady state, we set $\frac{dp(K,t)}{dt} = 0$ and drop the dependency on t to write

$$\begin{aligned}
&p(K)\big[\Lambda + \sum_{i=1}^{n}(\mu_i 1[k_i > 0] + \lambda_i)\big] \\
&= \Lambda p(K^{-0})1[k > 0] + \sum_{i=1}^{n}\big[\lambda_i p(K^{-i})1[k_i > 0] \\
&+\mu_i C_i p(K^{+0})1[k_i > 0] + \mu_i C_i p(K^{+i})1[k = 0] \\
&+\mu_i(1 - C_i)p(K^{+i})\big].
\end{aligned} \tag{A1}$$

Then, we divide both sides of (A1) by $p(K)$ and substitute the expression from (6), to obtain

$$\begin{aligned}
&\big[\Lambda + \sum_{i=1}^{n}(\mu_i 1[k_i > 0] + \lambda_i)\big] = \frac{\Lambda}{q}1[k > 0] \\
&+\sum_{i=1}^{n}\big[\frac{\lambda_i}{q_i}1[k_i > 0] + \mu_i C_i q 1[k_i > 0] + \mu_i C_i q_i 1[k = 0] \\
&+\mu_i(1 - C_i)q_i\big].
\end{aligned}$$

Now, substituting $\mu_i q_i = \frac{\lambda_i}{1-qC_i}$ from the expression for q_i in (5) and the expression $q = \Lambda \sum_{i=1}^{n} q_i \mu_i C_i$, we have

$$\begin{aligned}
&\big[\Lambda + \sum_{i=1}^{n}(\mu_i 1[k_i > 0] + \lambda_i)\big] = \sum_{i=1}^{n} q_i \mu_i C_i 1[k > 0] \\
&+\sum_{i=1}^{n}\big[\mu_i(1 - qC_i)1[k_i > 0] + \mu_i C_i q 1[k_i > 0] + \mu_i C_i q_i 1[k = 0] \\
&+\mu_i(1 - C_i)q_i\big],
\end{aligned}$$

or canceling identical terms with opposite signs and summing identical terms for $k > 0$ and $k = 0]$, we obtain

$$\begin{aligned}
&\big[\Lambda + \sum_{i=1}^{n}(\mu_i 1[k_i > 0] + \lambda_i)\big] = \sum_{i=1}^{n} q_i \mu_i C_i 1 \\
&+\sum_{i=1}^{n}\big[\mu_i 1[k_i > 0] + \mu_i(1 - C_i)q_i\big].
\end{aligned}$$

Now, canceling identical terms on both sides of the equation and also canceling identical terms with opposite signs on the right-hand side, we are left with

$$\Lambda + \sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} \mu_i q_i.$$

However, by **Lemma 1**, the right-hand side and left-hand side of the above equation are identical; hence, the solution (5) and (6) has now been proved. The **uniqueness** of the solutions of the non-linear Equation (5) follows from the known uniqueness of the stationary solution of the Chapman–Kolmogorov differential-difference Equation (4) [48,49]. This completes the **proof of the Key Product Form (Theorem 1).** □

Appendix B

Proof of Theorem 2. We use (5) to derive

$$
\begin{aligned}
D_i &= -\frac{\Lambda[\sum_{j=1}^n d_{ji}\mu_j C_j + q_i\mu_i]}{[\sum_{j=1}^n q_j\mu_j C_j]^2}, \\
&= -\frac{q^2}{\Lambda}[\sum_{j=1}^n d_{ji}\mu_j C_j + q_i\mu_i], \\
d_{ji} &= \rho_j \frac{D_i C_j + q\mathbf{1}[i=j]}{[1-qC_j]^2}, && \text{(A2)} \\
&= \frac{q_j^2}{\rho_j}[D_i C_j + q\mathbf{1}[i=j]]. && \text{(A3)}
\end{aligned}
$$

As a consequence, we can write

$$
\begin{aligned}
D_i &= -\frac{q^2}{\Lambda}[\sum_{j=1}^n \frac{q_j^2}{\rho_j} D_i \mu_j C_j^2 + \frac{q_i^2}{\rho_i} q\mu_i C_i + q_i\mu_i], \\
&= -q^2 \frac{q_i\mu_i[1+\frac{q_i}{\rho_i}qC_i]}{\Lambda + q^2\sum_{j=1}^n \frac{q_j^2}{\rho_j}\mu_j C_j^2} \\
&= -q\frac{q_i\mu_i[1+\frac{q_i}{\rho_i}qC_i]}{\sum_{j=1}^n q_j\mu_j C_j[1+q\frac{q_j}{\rho_j}C_j]}, \\
&= -q\frac{\frac{\lambda_i}{(1-qC_i)^2}}{\sum_{j=1}^n \frac{\lambda_j C_j}{(1-qC_j)^2}}. && \text{(A4)}
\end{aligned}
$$

Thus, (A4) tells us that if $q > 0$ and all the $q_i > 0$, then all $D_i < 0$.

Now, substituting (A4) back into (A3), we have

$$
\begin{aligned}
d_{ji} &= q[\frac{q_j^2}{\rho_j}\mathbf{1}[i=j] - \frac{q_j^2}{\rho_j}\frac{\frac{\lambda_i C_j}{(1-qC_i)^2}}{\sum_{l=1}^n \frac{\lambda_l C_l}{(1-qC_l)^2}}], \\
&= q[\frac{q_j^2}{\rho_j}\mathbf{1}[i=j] - \frac{q_i^2\mu_i}{\rho_i}\frac{\frac{q_j^2 C_j}{\rho_j}}{\sum_{l=1}^n \frac{q_l^2\mu_l C_l}{\rho_l}}], \\
&= q\frac{q_i^2}{\rho_i}[\mathbf{1}[i=j] - \mu_i\frac{\frac{q_j^2 C_j}{\rho_j}}{\sum_{l=1}^n \frac{q_l^2\mu_l C_l}{\rho_l}}]. && \text{(A5)}
\end{aligned}
$$

Since the first term (which is non-negative) in (A5) vanishes when $i \neq j$, we can see that $d_{ji} < 0$ for $i \neq j$.

The last part of the proof must establish that $d_{ii} > 0$. Using (A5), we write

$$
d_{ii} = q\frac{q_i^2}{\rho_i}[1 - \frac{\frac{q_i^2\mu_i C_i}{\rho_j}}{\sum_{l=1}^n \frac{q_l^2\mu_l C_l}{\rho_l}}],
$$

so that $d_{ii} > 0$ is obvious as long as $n > 1$, $0 < q_l < 1$ and all $C_l > 0$. **Hence, under these conditions, we have** $d_{ii} > 0$. This completes the proof of Theorem 2. □

Appendix C

Proof of Theorem 3. We start from (16) and (18) to write

$$R_S \quad = \quad \frac{1}{\Lambda+\lambda}\sum_{i=1}^{n}\frac{\lambda_j+\Lambda\Phi_i}{\mu_i-\lambda_i-\Lambda\Phi_i},\ with\ \Phi_1 = 1-\sum_{i=2}^{n}\Phi_i\,, \tag{A6}$$

so that for $2 \le i \le n$ we have $\frac{d\Phi_1}{d\Phi_i} = -1$ and

$$\begin{aligned}
\frac{dR_S}{d\Phi_i} &= \frac{\Lambda(\mu_i-\lambda_i-\Lambda\Phi_i)+\Lambda(\lambda_i+\Lambda\Phi_i)}{(\mu_i-\lambda_i-\Lambda\Phi_i)^2} \\
&\quad \frac{1}{\Lambda+\lambda}[\frac{\Lambda(\mu_1-\lambda_1-\Lambda\Phi_1)+\Lambda(\lambda_1+\Lambda\Phi_1)}{(\mu_1-\lambda_1-\Lambda\Phi_1)^2}]\,, \\
&= \frac{\Lambda}{\Lambda+\lambda}[\frac{\mu_i}{(\mu_i-\lambda_i-\Lambda\Phi_i)^2}-\frac{\mu_1}{(\mu_1-\lambda_1-\Lambda\Phi_1)^2}]\,, \qquad \text{(A7)} \\
\frac{d^2R_S}{d\Phi_i^2} &= \frac{\Lambda^2}{\Lambda+\lambda}[\frac{\mu_i}{(\mu_i-\lambda_i-\Lambda\Phi_i)^3}+\frac{\mu_1}{(\mu_1-\lambda_1-\Lambda\Phi_1)^3}]. . \qquad \text{(A8)}
\end{aligned}$$

Since $q_i < 1$ for all $1 \le i \le n$, it follows from (A8) that $\frac{d^2R_S}{d\Phi_i^2} > 0$. Therefore, the minimum of $\mathbf{R_S}$ with respect to Φ_i , $i = 1,\ldots,n$, is obtained from (A7) when

$$\frac{dR_S}{d\Phi_i} = 0,\ or\ \ (\mu_1-\lambda_1-\Lambda\Phi_1^*)\sqrt{\frac{\mu_i}{\mu_1}} = \mu_i-\lambda_i-\Lambda\Phi_i^*\ . \tag{A9}$$

Using $\Phi_1^* = 1-\sum_{i=2}^{n}\Phi_i^*$ and summing both sides of (A9) over $2 \le i \le n$, we have

$$\begin{aligned}
&(\mu_1-\lambda_1-\Lambda\Phi_1^*)\sum_{i=2}^{n}\sqrt{\frac{\mu_i}{\mu_1}} = \mu-\mu_1-\lambda+\lambda_1-\Lambda+\Lambda\Phi_1^*\ ,\ \ or \\
&\Lambda\Phi_1^*[1+\sum_{i=2}^{n}\sqrt{\frac{\mu_i}{\mu_1}}] = (\mu_1-\lambda_1)[1+\sum_{i=2}^{n}\sqrt{\frac{\mu_i}{\mu_1}}]+(\mu-\Lambda-\lambda),\ or \\
\Lambda\Phi_1^* \quad &= \quad \mu_1-\lambda_1-\frac{\mu-\Lambda-\lambda}{1+\sum_{i=2}^{n}\sqrt{\frac{\mu_i}{\mu_1}}},\ and\ \Phi_i^* = \frac{\mu_i-\lambda_i}{\Lambda}-\frac{\mu-\Lambda-\lambda}{\Lambda}\frac{\sqrt{\frac{\mu_i}{\mu_1}}}{\sum_{j=1}^{n}\sqrt{\frac{\mu_j}{\mu_1}}}\,, \qquad \text{(A10)}
\end{aligned}$$

and the proof is complete. □

Appendix D

Proof of Theorem 4. Let us use the notation $E_i',\ E_i'',\ and\ \pi_i'\ to\ denote\ \frac{dE}{d\Phi_i},\ \frac{d^2E}{d\Phi_i^2},\ and\ \frac{d\pi_i}{d\Phi_i},$ respectively, $1 \le j \le n$. Using the fact that $\sum_{j=1}^{n}\Phi_j = 1$, we obtain the following expressions for $i \ne 1$:

$$E_i' = \frac{\pi_i}{\mu_i}+\Phi_i\times\frac{\pi_i'}{\mu_i}-\frac{\pi_1}{\mu_1}-\Phi_1\times\frac{\pi_1'}{\mu_1}, \tag{A11}$$

$$E_i'' = \frac{\pi_i'}{\mu_i}+\Phi_i\times\frac{\pi_i''}{\mu_i}+\frac{\pi_i'}{\mu_i}+\frac{\pi_1'}{\mu_1}+\frac{\pi_i'}{\mu_i}+\Phi_1\times\frac{\pi_1''}{\mu_1}\,. \tag{A12}$$

We see easily that $E_i'' > 0$ when $0 \le X_i < X_{i1}$ for $i \ne 1$. Thus, for $i \ne 1$, the value Φ_i^+ of Φ_i that minimizes E is attained by setting $E_i' = 0$ in (A11), leading to

$$\begin{aligned}
\Phi_i^+ \frac{\pi_i'}{\mu_i} &= \Phi_1^+ \frac{\pi_1'}{\mu_1} + \frac{\pi_1}{\mu_1} - \frac{\pi_i}{\mu_i}, \textit{ or} \\
\Phi_i^+ &= \Phi_1 \frac{\alpha_1\mu_i}{\alpha_i\mu_1} + \mu_i \frac{\pi_{10} + \alpha_1\lambda_1 + \alpha_1\Phi_1^+\Lambda}{\alpha_i\Lambda\mu_1} - \frac{\pi_{i0} + \alpha_i\lambda_i + \alpha_i\Phi_i^+\Lambda}{\alpha_i\Lambda}, \\
2\Phi_i^+ &= 2\Phi_1^+ \frac{\mu_i\alpha_1}{\mu_1\alpha_i} + \frac{\lambda_1\mu_i\alpha_1}{\Lambda\mu_1\alpha_i} - \frac{\lambda_i}{\Lambda} + \frac{\frac{\mu_i}{\mu_1}\pi_{10} - \pi_{i0}}{\alpha_i\Lambda}, \textit{ yielding} \\
\Phi_i^+ &= \Phi_1^+ \frac{\mu_i\alpha_1}{\mu_1\alpha_i} + \frac{\frac{\mu_i}{\mu_1}\pi_{10} - \pi_{i0}}{\alpha_i\Lambda}. \qquad \text{(A13)}
\end{aligned}$$

Summing both sides of (A13) from 2 to n, we obtain

$$\begin{aligned}
1 - \Phi_1^+ &= \Phi_1^+ \frac{\alpha_1}{\mu_1} \sum_2^n \frac{\mu_i}{\alpha_i} + \frac{\pi_1}{\Lambda\mu_1} \sum_2^n \frac{\mu_i}{\alpha_i} - \sum_2^n \frac{\pi_i}{\Lambda\alpha_i} \\
&= \Phi_1^+ \frac{\alpha_1}{\mu_1} \sum_2^n \frac{\mu_i}{\alpha_i} + \frac{\pi_1}{\Lambda\mu_1} \sum_2^n \frac{\mu_i}{\alpha_i} - \sum_2^n \frac{\pi_{i0}}{\Lambda\alpha_i} - \sum_2^n \Phi_i^+, \textit{ implying that :} \\
2(1 - \Phi_1^+) &= \Phi_1^+ \frac{\alpha_1}{\mu_1} \sum_2^n \frac{\mu_i}{\alpha_i} + (\frac{\pi_{10}}{\Lambda\mu_1} + \Phi_1^+ \frac{\alpha_1}{\mu_1}) \sum_2^n \frac{\mu_i}{\alpha_i} - \sum_2^n \frac{\pi_{i0}}{\Lambda\alpha_i}, \textit{ or} \\
2(1 - \Phi_1^+) &= 2\Phi_1^+ \frac{\alpha_1}{\mu_1} \sum_2^n \frac{\mu_i}{\alpha_i} + \frac{\pi_{10}}{\Lambda\mu_1} \sum_2^n \frac{\mu_i}{\alpha_i} - \sum_2^n \frac{\pi_{i0}}{\Lambda\alpha_i}, \textit{ that yields :}
\end{aligned}$$

$$\Phi_1^+ = \frac{1 + \frac{1}{2\Lambda} \sum_2^n [\frac{\pi_{i0}}{\alpha_i} - \frac{\pi_{10}}{\mu_1}\frac{\mu_i}{\alpha_i}]}{1 + \frac{\alpha_1}{\mu_1} \sum_2^n \frac{\mu_i}{\alpha_i}}. \qquad \text{(A14)}$$

Finally, (A13) and (A14) provide us with the expression

$$\begin{aligned}
\Phi_i^+ &= \Phi_1^+ \frac{\mu_i\alpha_1}{\mu_1\alpha_i} + \frac{\pi_{10}\mu_i}{\Lambda\alpha_i\mu_1} + \Phi_1^+ \frac{\alpha_1\mu_i}{\alpha_i\mu_1} - \frac{\pi_{i0}}{\Lambda\alpha_i} - \Phi_i^+, \textit{ or} \\
&= \Phi_1^+ \frac{\alpha_1\mu_i}{\alpha_i\mu_1} + \frac{1}{2\Lambda\alpha_i} [\pi_{10} \frac{\mu_i}{\mu_1} - \pi_{i0}]. \qquad \text{(A15)}
\end{aligned}$$

□

References

1. Juniper-Networks. Expel Complexity with a Self-Driving Network: Soon, Your Network Will Adaptively Meet Your Business Goals All by Itself. 2020. Available online: https://www.juniper.net/us/en/dm/the-selfdriving-network/ (accessed on 15 October 2024).
2. Apostolos, J. Improving Networks with Artificial Intelligence. 2019. Available online: https://blogs.cisco.com/networking/improving-networks-with-ai (accessed on 15 October 2024).
3. Kompany, R. *Huawei's 'Autonomous Driving' Mobile Networks Strategy Aims to Increase Automation and Reduce Costs*; Knowledge Centre: Analysis Mason Ltd.: London, UK, 2018.
4. Weiss, P. Making the ICT Sector Energy Efficient: The Information and Communication Technology Sector Is a Major Energy Consumer, But It Also Offers the Potential for Savings...If Used Properly. Let's Work Smarter. 2022. Available online: https://www.theparliamentmagazine.eu/news/article/energy-efficient-and-energy-smart (accessed on 15 October 2024).
5. Gelenbe, E. Electricity Consumption by ICT: Facts, Trends, and Measurements. *Ubiquity* **2023**, *2023*, 1–15. [CrossRef]
6. Ishtiaq, M.; Saeed, N.; Khan, M.A. Edge Computing in IoT: A 6G Perspective. *arXiv* **2022**, arXiv:2111.08943. http://arxiv.org/abs/2111.08943.
7. Al-Ansi, A.; Al-Ansi, A.M.; Muthanna, A.; Elgendy, I.A.; Koucheryavy, A. Survey on Intelligence Edge Computing in 6G: Characteristics, Challenges, Potential Use Cases, and Market Drivers. *Future Internet* **2021**, *13*, 118. [CrossRef]

8. Nguyen, T.A.; Thang, N.K.; Trystram, D. One gradient Frank-Wolfe for decentralized online convex and submodular optimization. In Proceedings of the ACML 2022—14th Asian Conference in Machine Learning, Hyderabad, India, 12–14 December 2022; pp. 1–33.
9. Sadatdiynov, K.; Cui, L.; Zhang, L.; Huang, J.Z.; Salloum, S.; Mahmud, M.S. A review of optimization methods for computation offloading in edge computing networks. *Digit. Commun. Netw.* **2023**, *9*, 450–461. 10.1016/j.dcan.2022.03.003 [CrossRef]
10. Fröhlich, P.; Gelenbe, E.; Nowak, M. Reinforcement Learning and Energy-Aware Routing. In Proceedings of the 4th FlexNets Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility, New York, NY, USA, 26–27 August 2021; FlexNets '21, pp. 26–31. [CrossRef]
11. Safri, H.; Kandi, M.M.; Miloudi, Y.; Bortolaso, C.; Trystram, D.; Desprez, F. Towards Developing a Global Federated Learning Platform for IoT. In Proceedings of the 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), Bologna, Italy, 10–13 July 2022; pp. 1312–1315. [CrossRef]
12. Kim, C.; Kameda, H. An algorithm for optimal static load balancing in distributed computer systems. *IEEE Trans. Comput.* **1992**, *41*, 381–384.
13. Topcuoglu, H.; Hariri, S.; Wu, M.Y. Performance-effective and low-complexity task scheduling for the Bera erogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **2002**, *13*, 260–274. [CrossRef]
14. Zhu, X.; Qin, X.; Qiu, M. Qos-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters. *IEEE Trans. Comput.* **2011**, *60*, 800–812.
15. Tian, W.; Zhao, Y.; Zhong, Y.; Xu, M.; Jing, C. A dynamic and integrated load-balancing scheduling algorithm for cloud datacenters. In Proceedings of the 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, Beijing, China, 15–17 September 2011; pp. 311–315.
16. Zhang, Z.; Zhang, X. A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. In Proceedings of the 2nd International Conference on Mechatronics and Automation, Wuhan, China, 30–31 May 2010; Volume 2, pp. 240–243.
17. Gelenbe, E.; Morfopoulou, C. A framework for energy-aware routing in packet networks. *The Computer Journal* **2011**, *54*, 850–859. [CrossRef]
18. Gelenbe, E.; Mahmoodi, T. Energy-aware routing in the cognitive packet network. In Proceedings of Energy 2011: The First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies, IARIA, Venice, Italy, 22–27 May 2011; pp. 7–11. Available online: https://www.researchgate.net/publication/289245202_Energy-Aware_Routing_in_the_Cognitive_Packet_Network (accessed on 15 October 2024).
19. Sakr, A.; Schuster, R. Edge Resource Allocation Based on End-to-End Latency. In Proceedings of the HotEdge'20, USENIX Association, Online, 25–26 June 2020. Available online: https://www.usenix.org/system/files/hotedge20_poster_sakr_0.pdf (accessed on 15 October 2024).
20. Sarah, A.; Nencioni, G.; Khan, M.M.I. Resource Allocation in Multi-access Edge Computing for 5G-and-beyond networks. *Comput. Netw.* **2023**, *227*, 109720. 10.1016/j.comnet.2023.109720 [CrossRef]
21. Liu, H.; Li, S.; Sun, W. Resource Allocation for Edge Computing without Using Cloud Center in Smart Home Environment: A Pricing Approach. *Sensors* **2020**, *20*, 6545. [CrossRef]
22. Zheng, K.; Jiang, G.; Liu, X.; Chi, K.; Yao, X.; Liu, J. DRL-Based Offloading for Computation Delay Minimization in Wireless-Powered Multi-Access Edge Computing. *IEEE Trans. Commun.* **2023**, *71*, 1755–1770. [CrossRef]
23. Boyan, J.A.; Littman, M.L. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. In *Advances in Neural Information Processing Systems 6, Proceedings of the 7th NIPS Conference, Denver, CO, USA, 29 November–2 December 1993*; Cowan, J.D., Tesauro, G., Alspector, J., Eds.; Morgan Kaufmann: Burlington, MA, USA, 1993; pp. 671–678.
24. Tennenhouse, D.L.; Wetherall, D.J. Towards an active network architecture. *Comput. Commun. Rev.* **1996**, *26*, 5–18. [CrossRef]
25. Tsarouchis, C.; Denazis, S.; Kitahara, C.; Vivero, J.; Salamanca, E.; Magana, E.; Galis, A.; Manas, J.L.; Carlinet, L.; Mathieu, B.; et al. A policy-based management architecture for active and programmable networks. *IEEE Netw.* **2003**, *17*, 22–28. [CrossRef]
26. Gelenbe, E.; Xu, Z.; Seref, E. Cognitive Packet Networks. In Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence, ICTAI '99, Chicago, IL, USA, 8–10 November 1999; IEEE Computer Society: New York, NY, USA, 1999; pp. 47–54. [CrossRef]
27. Masoudi, R.; Ghaffari, A. Software defined networks: A survey. *J. Netw. Comput. Appl.* **2016**, *67*, 1–25. [CrossRef]
28. Tuncer, D.; Charalambides, M.; Clayman, S.; Pavlou, G. On the Placement of Management and Control Functionality in Software Defined Networks. In Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM), Barcelona, Spain, 9–13 November 2015; pp. 360–365. [CrossRef]
29. Montazerolghaem, A. Software-defined load-balanced data center: Design, implementation and performance analysis. *Clust. Comput.* **2021**, *24*, 591–610. [CrossRef]

30. Liu, X.; Qin, Z.; Gao, Y. Resource Allocation for Edge Computing in IoT Networks via Reinforcement Learning. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6. [CrossRef]
31. Wang, J.; Zhao, L.; Liu, J.; Kato, N. Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach. *IEEE Trans. Emerg. Top. Comput.* **2021**, *9*, 1529–1541. [CrossRef]
32. Huang, J.; Wan, J.; Lv, B.; Ye, Q.; Chen, Y. Joint Computation Offloading and Resource Allocation for Edge-Cloud Collaboration in Internet of Vehicles via Deep Reinforcement Learning. *IEEE Syst. J.* **2023**, *17*, 2500–2511. [CrossRef]
33. You, C.; Huang, K.; Chae, H.; Kim, B.H. Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 1397–1411. [CrossRef]
34. Domanska, J.; Gelenbe, E.; Czachorski, T.; Drosou, A.; Tzovaras, D. Research and Innovation Action for the Security of the Internet of Things: The SerIoT Project. In *Recent Cybersecurity Research in Europe, Proceedings of the 2018 ISCIS Security Workshop, London, UK, 26–27 February 2018*; Lecture Notes CCIS No. 821; Springer: Berlin/Heidelberg, Germany, 2018; Volume 821.
35. Gelenbe, E.; Domanska, J.; Fröhlich, P.; Nowak, M.P.; Nowak, S. Self-Aware Networks That Optimize Security, QoS, and Energy. *Proc. IEEE* **2020**, *108*, 1150–1167. [CrossRef]
36. Rublein, C.; Mehmeti, F.; Towers, M.; Stein, S.; Porta, T.L. Online resource allocation in edge computing using distributed bidding approaches. In Proceedings of the 2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS), Denver, CO, USA, 4–7 October 2021.
37. Nguyen, D.; Le, L.; Bhargava, V. Price-Based Resource Allocation for Edge Computing: A Market Equilibrium Approach. *IEEE Trans. Cloud Comput.* **2021**, *9*, 302–317. [CrossRef]
38. Zhao, Z.; Schiller, E.; Kalogeiton, E.; Braun, T.; Stiller, B.; Garip, M.T.; Joy, J.; Gerla, M.; Akhtar, N.; Matta, I. Autonomic Communications in Software-Driven Networks. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 2431–2445. [CrossRef]
39. Ben-Ameur, A.; Araldo, A.; Chahed, T. Multiple Resource Allocation in Multi-Tenant Edge Computing via Sub-modular Optimization. *arXiv* **2023**, arXiv:2302.09888. http://arxiv.org/abs/2302.09888.
40. Hamilton, E. What Is Edge Computing? Definition, Examples and Use Cases Explained in 2025. Available online: https://www.cloudwards.net/what-is-edge-computing/(accessed on 15 October 2024).
41. Gelenbe, E. G-networks with signals and batch removal. *Probab. Eng. Inform. Sci.* **1993**, *7*, 335–342. [CrossRef]
42. Gelenbe, E. G-networks with instantaneous customer movement. *J. Appl. Probab.* **1993**, *30*, 742–748. [CrossRef]
43. Gelenbe, E.; Mitrani, I. *Analysis and Synthesis of Computer Systems*, 2nd ed.; World Scientific: Singapore, 2010.
44. Ross, S.M. *Introduction to Probability Models*, 11th ed.; Academic Press: Cambridge, MA, USA, 2014; Chapter 4.2.
45. Sigman, K. *Stationary Marked Point Processes: An Intuitive Approach*; Chapman and Hall: New York, NY, USA; London, UK; CRC Press: Boca Raton, FL, USA, 1995.
46. Intel. NUC— Small Form Factor Mini PC. 2021. 2021. Available online: https://en.wikipedia.org/wiki/Next-Unit-of-Computing (accessed on 23 March 2021).
47. Fröhlich, P.; Gelenbe, E.; Fiołka, J.; Checinski, J.; Nowak, M.; Filus, Z. Smart SDN Management of Fog Services to Optimize QoS and Energy. *Sensors* **2021**, *21*, 3105. [CrossRef]
48. Feller, W. *An Introduction to Probability Theory and Its Applications*, 3rd ed.; John Wiley & Sons: Hoboken, NJ, USA, 1968; Volume I.
49. Feller, W. *An Introduction to Probability Theory and Its Applications*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 1971; Volume II.