future internet

MDPI

Article

PECSA: Practical Edge Computing Service Architecture Applicable to Adaptive IoT-Based Applications

Jianhua Liu * and Zibo Wu

School of Avionics and Electronics Engineering, Civil Aviation Flight University of China, Guanghan 618307, China; wuzibo@cafuc.edu.cn
* Correspondence: jianhuacafuc13@cafuc.edu.cn

Abstract: The cloud-based Internet of Things (IoT-Cloud) combines the advantages of the IoT and cloud computing, which not only expands the scope of cloud computing but also enhances the data processing capability of the IoT. Users always seek affordable and efficient services, which can be completed by the cooperation of all available network resources, such as edge computing nodes. However, current solutions exhibit significant security and efficiency problems that must be solved. Insider attacks could degrade the performance of the IoT-Cloud due to its natural environment and inherent open construction. Unfortunately, traditional security approaches cannot defend against these attacks effectively. In this paper, a novel practical edge computing service architecture (PECSA), which integrates a trust management methodology with dynamic cost evaluation schemes, is proposed to address these problems. In the architecture, the edge network devices and edge platform cooperate to achieve a shorter response time and/or less economic costs, as well as to enhance the effectiveness of the trust management methodology, respectively. To achieve faster responses for IoT-based requirements, all the edge computing devices and cloud resources cooperate in a reasonable way by evaluating computational cost and runtime resource capacity in the edge networks. Moreover, when cooperated with the edge platform, the edge networks compute trust values of linked nodes and find the best collaborative approach for each user to meet various service requirements. Experimental results demonstrate the efficiency and the security of the proposed architecture.

Keywords: edge computing; cloud; Internet of Things (IoT); efficiency; trust

Citation: Liu, J.; Wu, Z. PECSA: Practical Edge Computing Service Architecture Applicable to Adaptive IoT-Based Applications. *Future Internet* **2021**, *13*, 294. https://doi.org/10.3390/fi13110294

Academic Editor: Paolo Bellavista

Received: 25 October 2021
Accepted: 17 November 2021
Published: 19 November 2021

1. Introduction

The Internet of Things (IoT) is usually composed of a large number of spatially distributed physical devices, vehicles and other items that are embedded as wired or wireless-connected nodes in the Internet to improve humans' quality of life. The interconnected nodes of these items are expected to provide services of data transmission and computation with reduced response time and economic costs. However, many IoT devices are subject to restrictions, such as limited storage and computation capacity, and imbalanced distribution of the capacity among these devices, which can reduce the service performance of the IoT. Cloud computing can enhance the IoT's capability in terms of storage, computation and management of various resources [1–3]. The cloud is able to provide users with many extended services based on its powerful computing and storage capacity. However, the cloud is far away from users of the IoT; therefore, it usually causes large transmission delays. In order to improve the response delay of the requirements, edge computing (EC) is proposed to share the tasks in the cloud. The computing and storage resources of edge computing are composed of many routers, gateways, edge servers, etc. They are deployed at the edge of the Cloud and in close proximity to IoT end nodes at various locations, such as metropolitan centers, malls, cellular base stations, or even WiFi access points [4]. The service provider of the IoT-Cloud can greatly reduce resource consumption by moving some computing tasks from the Cloud to the edges, in terms of

data transmission and data integration. Furthermore, users can obtain shorter responses from the edges than from the Cloud. Edge computing nodes are aware of users' location by analyzing the data received from IoT devices. With edge networks, data can be transmitted between the user and EC nodes without the participation of the core network [5]. Since the IoT nodes and edge nodes move randomly, tasks need to be dynamically assigned to the EC nodes that are adjacent to the users. To minimize the service delay, many solutions tasks should be offloaded to different EC nodes that are adjacent to the users [6–10]. It is inefficient for IoT applications to solve these requirements by simply allocating tasks to available resources; further, to constantly improve hardware configurations is impractical. In addition, a high-configured computing platform is usually associated with more costs. Recently, a lot of studies have paid attention to offloading strategies for high service quality and/or low costs [11–15]. In the literature, there is a common assumption that tasks are only assigned to a single 4G or 5G network. However, for the edge computing network that integrates both 4G and 5G devices, such operation may be not suitable because the quality of the service depends on a single network.

An important part of service costs is energy consumption, regardless of network structure. To minimize the service costs, most of the literature focuses on ways to minimize energy consumption [4,9,15–17]. Unfortunately, in the literature, researchers often separate operating costs from user prices. The authors in [4] investigate both running time and total costs. However, their optimization expression considers the weighting of cost and latency but ignores the requirements of the users who need minimum service price and of those who need a minimum service latency. Furthermore, many data generated on the IoT require security protection [10,18–20]. For instance, the health data of patients need to be transmitted and processed under privacy protection. For all the aforementioned works, the authors do not take into account the costs of security measures, such as encryption and digital signatures. How to design an efficient cooperation service architecture to provide quick-response service by integrating the resources of 4G and 5G devices is still an open problem. Furthermore, due to the openness of network structure in IoT and edge networks, the security operational costs must be evaluated.

To address the above -mentioned problems, we propose a practical edge service architecture including three cost evaluation schemes and a novel trust management mechanism for adaptive IoT-based applications. The proposed service architecture aims to meet the adaptive requirements of IoT-based applications, including minimum time delay, minimum service price, and minimum price within a given time. To be specific, our task offloading scheme includes two parts: (1) Which devices are reliable? That is, it is necessary to decide which devices are suited for task processing according to the available resources and historical behaviors. (2) How can we assign tasks according to the adaptive needs of users (i.e., a resource allocation problem)? How can we then transform this problem into three sub-problems: how can we assign tasks within the minimum service time? How can we assign tasks with the minimum price? Finally, how can we assign tasks with the minimum price within a given time? We transformed these problems into linear programming problems, which can be efficiently solved by the interior point method. In summary, the scientific contributions of the paper are as follows:

(1) We propose a practical edge computing service architecture that integrates a trust management methodology with dynamic cost evaluation schemes. To better allocate online resources and meet the demands of IoT-based applications, all the service requirements are divided into three categories: (1) requirements that need to be handled within the shortest time, (2) requirements that need to be completed using the minimum price, and (3) requirements that need to be handled using the minimum price within a given time. Users are allowed to choose one of the three service types according to the specific situations, while the providers are able to provide various services.

(2) Our architecture is robust regardless of the relationship between data volume and computation amount. A scale factor is used to construct the relationship. Our estima-

tion schemes change adaptively with the change in coefficients; thus, the robustness of our architecture is not be undermined by task changes.

(3) The available devices are filtered before task allocation based on our trust management scheme. The service provider selects the nodes with a high trust value for data processing, and the nodes with a low trust value are rejected. The convergence time is very small, and the trust bias is acceptable. Moreover, the trust managements in the IoT and the edge networks are distinguished. The former focuses on the legality of behavior, while the latter focuses on the availability of resources. Moreover, the edge platform can monitor the edge network status to dynamically adjust the resource allocation strategies.

This paper is organized as follows: Section 2 provides related works. Section 3 presents the basic concept of a novel architecture. Section 4 focuses on the design details of the architecture. Section 5 presents the experiments results and analyses. We discuss some practical issues in Section 6 and conclude this paper in Section 7.

2. Related Work

The concept of edge computing was presented to offload tasks from the cloud to the network edge to enhance the performance of mobile devices [11,13]. While many works have paid attention to service architectures for edge computing [12,15–17,21,22], the resource allocation issue remains a critical challenge.

In order to overcome this challenge, Wang et al. [4] proposed a mobility-agnostic online resource allocation that takes task reconfiguration as the main means to reduce service costs. For task scheduling and resource allocation, an intelligent offloading system for vehicular edge computing [14] was constructed by leveraging deep reinforcement learning. The procedure is initiated by the requests of vehicle users, without any regulatory process for these users; therefore, the robustness of this structure is poor. In order to solve allocation optimization problems, a coalition-structure's generation method [23] was proposed, which introduces the concept of bargaining set and removes the impossible coalition-structures by judging the no-bargain coalition to narrow the strategic space; however, the assumption that any member can only participate in one task narrows its application. In [15], the capabilities of edge computing and 5G are used to solve the challenges in the development of an efficient energy cloud system. To save the battery life of user's equipment, Chen et al. [24] proposed a non-linear program to offload tasks on the edge cloud or process them locally; they claimed that their scheme could reduce the system's energy consumption. To maximize the quality of experience (QoE) of more users in the IoT, Shah-Mansouri et al. [25] proposed a near-optimal resource allocation mechanism based on the Nash equilibrium, which can provide an upper bound for the price of anarchy. To reduce overall power consumption in the IoT-Cloud, Barcelo et al. [26] formulated the service distribution problem (SDP) in IoT-Cloud networks (IoT-CDSP) as a min-cost mixed-cast flow problem and provided a solution for this problem in multiple smart environments, e.g., 4G Link, WiFi Link, and Zigbee Link. To improve the quality of service of multiuser, ultradense edge server scenarios, Guo et al. [27] proposed a two-tier game-theoretic greedy offloading scheme to utilize the computation resources among multiple edge servers. Premsankar et al. [28] evaluated the response delay in 3D applications and showed that edge computing can be helpful for satisfying the latency requirements of IoT applications. However, task offloading to the edge cloud via cellular networks would generate insider-attack issues.

To address this challenge, one intuitive idea is to include trust relationships between individual nodes in the IoT-Cloud. To facilitate the implementation of this idea, various trust mechanisms, which quantify trust relationships according to different applications' security requirements, have been proposed and integrated into IoT services. Focusing on social attributes, Chen et al. [29] proposed a trust management protocol for IoT systems that can select trust feedback from IoT nodes sharing similar social interests through a filtering technique. To minimize the convergence time and trust bias of trust evaluation,

an adaptive filtering method was proposed, by which each node can adapt its weight parameters for combining direct and indirect trust. However, this mechanism pays little attention to physical attributes. To ensure the scalability of the trust management scheme and resist second-hand trust attacks, Alshehri et al. [30] presented a clustering-driven trust management methodology for the IoT. It develops methods to counter bad-mouthing attacks by identifying trust outliers from all the values. However, this mechanism requires greater energy consumption [12]. Considering the limited bandwidth and power of IoT nodes, Duan et al. [31] proposed an energy-aware trust derivation scheme that is claimed to be able to reduce the overhead of the trust derivation process by a game theoretic approach. The scheme only provides partial security. In order to meet repeated or similar service requirements, Wang et al. [12] established service templates in the cloud and on the edge platform to improve the efficiency of service response. They also integrated a trust evaluation mechanism into the IoT-Cloud for security considerations. The mechanism does not carry out normalization for the coefficient when calculating the direct trust; thus, it is difficult to accurately measure the level of security by the trust values.

However, most of the existing works were performed based on the benefits of service providers and/or the limitations of the mobile devices, while the various service requirements of users have been ignored. Taking into account the limitations of the existing literature, in the paper, we propose a novel service architecture based on users' requirements and the processing capabilities to help obtain a better service for IoT-based applications. The service requirements of users are divided into three categories: minimum service time, minimum price costs, and minimum price costs, within a given time. In fine-grained resource management, trust management is used to enhance the robustness and security of our service architecture.

3. Preliminary

3.1. Trust Evaluation Mechanism

In general, trust management in any communication system can be executed as a procedure with three sequential phases [32]: phase 1, direct trust establishment; phase 2, indirect trust generation; and phase 3, trust-based operations. In phase 3, direct and indirect trust are utilized to support all operations, such as reliable data fusion and mining, information security enhancement, and service assurance [33–36].

The effectiveness of trust management comes from observing the behavior of a large amount of historical behavioral data collected during a recent observation period; thus, the construction of trust values must be application-oriented. For example, if a trust value is obtained by observing the data-forwarding behavior, it is not reasonable to use it to the judge computing power. An acceptable trust scheme comes from interaction behaviors between among in a specific application context.

3.2. Novel Service Architecture

Figure 1 shows the framework of an edge computing service for IoT-based applications that includes three layers, i.e., the IoT layer, the edge network layer, and the cloud layer. The IoT layer consists of users who need to submit computing tasks. The edge network layer mainly corresponds to edge clouds deployed near the IoT users. The cloud layer comprises the data processing center deployed far from the users. The cloud and the edge networks accept service requests from users.

IoT applications usually collect a lot of raw data and initiate various service requirements. Trust management, in the IoT layer, is mainly used to monitor the historical behavior of users and distinguish malicious users. All requirements of legitimate users are processed by edge networks and/or the cloud, and the results should be returned as soon as possible. The cloud is responsible for trust management, task allocation, data processing, and edge resource management. For reasons of cost control and response time, many tasks are delegated to the edge cloud.

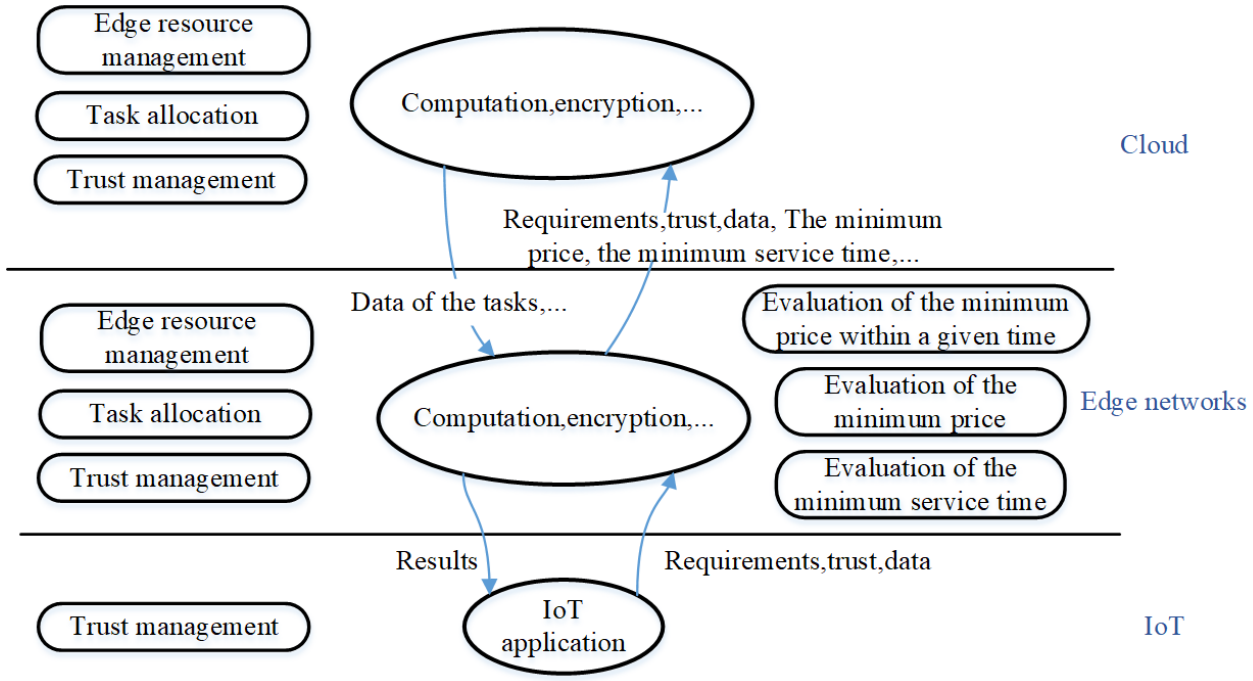

Figure 1. Edge computing service architecture for IoT-based applications.

Edge networks consist of six primary functions:

(1) Trust management: unlike IoT nodes, the trust values of the edge nodes are focused on the available resources.
(2) Edge resource management: in edge networks, all the available resources are managed by edge servers. The trust value, runtime resources, and tasks of every device in the edge cloud are reported to the servers periodically.
(3) Evaluation of the minimum service time: for time-sensitive users, their service requests need to be processed as soon as possible, regardless of the price costs.
(4) Evaluation of the minimum service price: many users want to obtain the results of their requirements with a minimum price, although this service type results in longer wait times.
(5) Evaluation of the minimum price within a given time: users want to achieve a tradeoff between service time and price.
(6) Task allocation: based on the above functions, each task is divided into several parts and sent to the appropriate edge nodes.

The cloud mainly performs the following functions:

(1) Trust management: the cloud develops trust management strategies and methods for calculating trust values in edge networks and the IoT, respectively.
(2) Edge resource management: the cloud cooperates with edge servers to manage the resources and coordinate resources among different edge servers.
(3) Task allocation: the cloud participates in task allocation and data processing if the edge servers cannot meet stringent user demands.

Computation costs and transportation costs are the two most important factors for a task allocation scheme. Generally, all the data transmitted to the edge nodes are used for some computation. Thus, we make an assumption about the relationship between data size and computation amount as follows.

Assumption 1. *In an edge computation, the computation amount is proportional to the size of the data transported from the users' side to the edge nodes.*

The assumption is used in our model construction and performance evaluation. This assumption is rational since the data transmitted from the user end to the edge network usually need to be processed—data irrelevant to calculation is not transferred for efficiency reasons. The scale factor between data size and computation amount is set as l.

4. Materials and Methods

In this section, we first introduce the framework of the trust evaluation mechanism. Then, we give three basic service types and the associated resource allocation models.

4.1. Trust Evaluation Mechanism

4.1.1. Trust Evaluation Mechanism in the IoT

The major functions of the nodes in the IoT include data acquisition, data forwarding, and request generation. Trust evaluation models should be established based on their functions. Thus, evidence used to compute the direct trust $Tu(i,j)$ may include the device's endurance capacity, packet loss rate, bandwidth, transmission delay, device activity, data correctness rate, and overall success rate of historical events. To record historical events, each node maintains a trust table that stores some information about the adjacent nodes and the trust states, as shown in Table 1. A white list and a black list are used to record the number of positive and negative historical events in the table. $e^w_{(i,j)}$ and $e^b_{(i,j)}$ denote the number of positive activities and negative activities of node v_i towards node v_j, respectively. A single interactive activity of v_i toward v_j can be seen as an outcome of tossing a coin. When there is a positive activity, $e^w_{(i,j)}$ adds 1; otherwise, $e^b_{(i,j)}$ adds 1. Thus, $e^w_{(i,j)}/e^b_{(i,j)}$ can be seen as an outcome of a Bernoulli trial with the probability-of-success parameter $\theta_{i,j}$ following a Beta distribution $Beta(\alpha_{i,j}, \beta_{i,j})$. Considering an exponential decay [29], the hyper parameters $\alpha_{i,j}, \beta_{i,j}$ are given as follows:

$$\begin{aligned} \alpha(i,j) &= e^{-\varphi\Delta t} \cdot \alpha(i,j)' + e^w_{(i,j)} \\ \beta(i,j) &= e^{-\varphi\Delta t} \cdot \beta(i,j)' + e^b_{(i,j)} \end{aligned} \tag{1}$$

where Δt is the trust update interval and $\alpha(i,j)'$ and $\beta(i,j)'$ are the hyper parameters in the last update interval. According to [29], the decay factor φ is a small positive number. We adopt the Bayesian framwork [29] as the underlying model to evaluate the direct trust $Tu(i,j)$; here, $Tu(i,j)$ denotes the trust value of node v_j from the perspective of node v_i.

$$Tu(i,j) = \frac{\alpha(i,j)}{\alpha(i,j) + \beta(i,j)} \tag{2}$$

For the sustainable development of IoT-based business, the initial value of $\alpha(i,j)'$ and $\beta(i,j)'$ are suggested to be set as 1 and 0, respectively. The update interval is a time slice, which can be set as several seconds or tens of seconds. When a new update interval arrives, each node will update its trust values. It is easy to obtain that $Tu(i,j)$ is a value in the interval $[0,1]$. We set a threshold Td_{it} to separate the outliers from honest nodes. If a node is honest, its trust value is located in the interval $[Td_{it}, 1]$. Otherwise, it is located in $[0, Td_{it})$. The value of the observation window and Td_{it} are set according to the monitoring sensitivity.

Table 1. Trust table of node v_i.

Adjacent Nodes	Trust Value	Location List	White List	Black List	Time
node v_1	0.7	GPS position 1	$e^w_{(i,1)}$	$e^b_{(i,1)}$	1
node v_2	0.8	GPS position 2	$e^w_{(i,2)}$	$e^b_{(i,2)}$	2
node v_3	0.9	GPS position 3	$e^w_{(i,3)}$	$e^b_{(i,3)}$	3

The level of trust value is helpful for the node in the IoT to reveal the outliers and establish a set of credible adjacent nodes and reliable routes for data transmission. If $Tu(i,j)$ is located in $[Td_{it}, 1]$, node j is selected in the route list of node i, otherwise it is moved out of the route list of node i. For the sustainable development of IoT-based business, the initial trust value of new entrants is suggested to be set in the interval $[Td_{it}, 1]$.

4.1.2. Trust Evaluation Mechanism in Edge Networks

The major functions of the nodes in edge networks include computing, data storage, and data forwarding. Many nodes in edge networks are wired devices with unlimited endurance capacity. Thus, for these nodes, the evidence used to compute the direct trust value $Tu(i,j)$ of the nodes should include the calculation capacity, storage space, response speed, and overall success rate of historical events. $Tu(i,j)$ is a key parameter associated with the task distribution scheme. As the value of the $Tu(i,j)$ of an edge node increases, its data processing capacity becomes sharper. Therefore, a greater workload is assigned to reduce service time. The value of the $Tu(i,j)$ of an edge node changes dynamically. For a busy edge network, the running capacity of data processing of each node is unpredictable. For example, it may have a high CPU rate after a period with low CPU usage, low CPU rates after high utilization, high physic memory utilization, and so on. Similar to the Td_{it}, a threshold Td_{ed} is set to separate the outliers from honest nodes in the edge network. If a node is honest, its trust value is located in the interval $[Td_{ed}, 1]$. Otherwise, it is located in $[0, Td_{ed})$. Therefore, the trust value changes with the running capacity of the edge nodes.

4.1.3. Comprehensive Trust Computation

The trust values established in the IoT and edge networks are utilized to support service operations. Generally, the service provider selects an edge node with a high trust value for data processing, while the nodes with low trust values are rejected. Thus, a comprehensive trust of each node should be established for the classification of honest nodes and malicious nodes. Let us suppose that there is a q direct trust value about node j submitted to the server; the mean value of all the direct trust values is as follows:

$$mt(j) = \frac{\sum_{i=1}^{q} Tu(i,j)}{q}, \tag{3}$$

while the comprehensive trust $T_C(j)$ of a node j is given as

$$T_C(j) = \sum_{i=1}^{q} \frac{e^{-(Tu(i,j)-mt(j))^2} \cdot Tu(i,j)}{q \sum_{i=1}^{q} e^{-(Tu(i,j)-mt(j))^2}}, \tag{4}$$

The comprehensive trust $T_C(j)$ changes with the real-time performance of the edge nodes. Furthermore, the changes lead to offloading workload from the nodes with lower trust values to the nodes with higher trust values. When the comprehensive trust $T_C(j)$ of an edge node is less than a threshold Tu_e, its workload is reallocated to other edge nodes for security and efficiency. The value of Tu_e is set according to the whole edge network's state. If the trust values of most edge nodes are increasing, the value of Tu_e should be increased; otherwise, it should be decreased. When the $T_C(j)$ of a node in the IoT is less than a threshold Tu_i, the node is blacklisted and its service requirements are rejected. The trust estimation process is presented in Algorithm 1.

Algorithm 1: Trust estimation algorithm.

Input:
NT:node type// node in IoT or node in the edge network;
Td_{it},Tu_e, Tu_i:thresholds used in the algorithm;
N:number of currently available nodes in the edge network;
Output:
$Tu(i,j)$,$T_C(j)$,N
Begin
01: each node calculates $Tu(i,j)$ using (2) and sends it to the associated server;
02: **if** NT=1 //the node is in IoT
03: {**if** $Tu(i,j) \leq Td_{it}$
04: moves node j out of the route list of node i;
05: **else**
06: moves node j into the route list of node i;
07: the server calculates $T_C(j)$ using (4);
08: **if** $T_C(j) \leq Tu_i$
09: blacklists node j; }
10: **else if** NT=2//the node is in the edge network
11: {the server calculates $T_C(j)$ using (4);
12: **if** $T_C(j) \leq Tu_e$
13: **if** node j is on the available list
14: $\{N = N - 1$;
15: moves node j out of the available list;}
16: **else**
17: **if** node j is on the available list
18: $N = N$;
19: **else**
20: moves node j into the available list;
21: $N = N + 1$;}
end

4.2. Three Basic Service Types and the Associated Cost Evaluation Schemes

The local edge devices or edge servers first analyze the requirements submitted by adjacent users, such as the acceptable price and time of the services, service types, amount of computing, and amount of data transmission, as well as which kind of software resources and hardware resources would be used. If all the requirements can be matched, the requirements can be further processed; otherwise, they are refused and the program is terminated. How to better meet users' needs is one of the most important issues for any edge computing provider. Edge computing service providers (ECSPs) compete against each other to attract and serve the demands generated by the customers. The competition arises from the fact that the users (the individuals or organizations) who generate demand for services in the market rationally choose the ECSP that offers good quality of services at a lower price. Service time and service price are the two factors users are concerned about the most. Based on these two factors, users' requirements can be divided into three categories: (1) requirements that need to be completed as soon as possible, regardless of service price; (2) requirements that need to be completed at the lowest price, regardless of service time; and (3) requirements that need to be completed based on a trade-off between service price and service time, i.e., the minimum price within a given time. Any requirement is analyzed on the edge platform and some important information is extracted. According to the extraction, the requirements are classified into the three types: (1) requirements that need to be completed as soon as possible, (2) requirements that need to be completed at the lowest price, and (3) requirements that need to be completed with the minimum price within a given time.

To enhance the efficiency of execution of the process, devices with more computing power are recommended. For example, devices with higher CPU frequency and larger memory are recommended to be the first choice. Mass data transmission occupies much bandwidth and leads to a long service time. If mass data transmission is involved in the service, it is very important to consider the time cost of data transmission and computing power of devices together. In some special cases, it is required to build a new computing platform for special services. The software and hardware preparation demand a certain amount of time. Thus, establishing an excellent resource allocation strategy is actually equivalent to finding a trade-off between various factors, including computing power, cost of data transmission, cost of software and hardware preparation, security, and requirements of users. The security requirements include confidentiality, integrity, non-repudiation, etc. The security requirements should be considered at the beginning of the service. For example, the edge platform should encrypt sensitive messages for confidentiality. To achieve rapid response, it is advised to estimate the lower bound or upper bound of the calculation quantity and data size.

Information on available software and hardware resources of every edge platform, such as the IP address of the devices, physical memory, usage rate of CPU, and service software, should be updated by the cloud and the edge servers with a certain time interval. In each time interval, every device reports to the associated edge server of its runtime resources. The cloud collects the available resource information from all the edge servers. This method not only reduces the data transfer volume but also improves the efficiency of resources distribution.

Table 2 shows the correspondent relationship between words and abbreviations.

Table 2. List of main notations.

Symbol	Meaning
R	requirements of user
S	available services of service provider
R_T	time requirement of user
R_S	security requirement of user
R_P	price requirement of user
S_T	available service time of service provider
S_S	available security service of service provider
S_P	available service price of service provider
E_i^{max}	total battery energy of node i
$T_{transmission}$	data transmission time
$T_{security}$	security arrangement time, such as time of encryption, signature, etc.
$T_{calculation}$	calculation time

The requirement of user R consists of R_S, R_T, and R_P. The available services of service provider S consist of S_T, S_S, and S_P. Table 2 shows the correspondent relationship between symbols and their meaning.

$$R = R_T \cap R_S \cap R_P \tag{5}$$

$$S = S_T \cap S_S \cap S_P \tag{6}$$

Service S matches requirements R, and this is denoted as $S \succeq R$, that is to say, the following expressions are valid:

$$\begin{cases} R_T \geq S_T \\ R_P \geq S_P \\ R_S \subseteq S_S \end{cases} \tag{7}$$

The basic needs of security include confidentiality, integrity, availability, controllability, and auditability. For simplification, we only consider three security attributes, i.e., confidentiality, integrity, and non-repudiation, for S_S and R_S; thus, S_S and R_S are

both subsets of set $\{\text{Confidentiality, Integrity, and Non} - \text{repudiation}\}$. Confidentiality can be guaranteed by data encryption. Integrity and Non-repudiation can be ensured by a digital signature. The edge network's security capacity is required to cover the security demand of users. Let us suppose that R_S is a set of security requirements of a user; then, S_S must contain all the elements in R_S to satisfy the security requirements. For example, if $R_S = \{Confidentiality, Integrity\}$, then S_S can be $\{Confidentiality, Integrity\}$ or $\{Confidentiality, Integrity, and\ Non - repudiation\}$. It should be noted that the validity judgement of the formula $R_S \subseteq S_S$ should be ahead of the estimation of S_T and S_P. When the formula $R_S \subseteq S_S$ is not valid, the service process is stopped. When $R_S \subseteq S_S$ is valid, the estimation of S_T and S_P is initiated. Below, we introduce three evaluation schemes for resource allocation.

4.2.1. The Evaluation of the Minimum Service Time

The total service time S_T in the edge network can be generally captured by the following formula:

$$S_T = T_{security} + T_{transmission} + T_{calculation} \tag{8}$$

In fact, $T_{transmission}$, $T_{security}$, and $T_{calculation}$ can overlap for some services; thus, the formula above is a loose estimate of S_T. A loose estimate gives service provider a loose amount of time to complete tasks, contributing to a smaller probability to exceed the user's time limit than a tight estimate. The security measures' execution efficiency is decided by the equipment capability. Considering a widely used encryption algorithm; advanced encryption standard (AES); and a mainstream digital signature algorithm, SHA-256, their performing times can be seen, approximately, as a positive proportion of the data volumes of plain texts [37,38], respectively. The performing time of security measures is calculated using the following formula:

$$T_{security} = t_s \cdot cp_i \cdot da_i \tag{9}$$

where t_s is the complexity coefficient of the security algorithms, such as encryption algorithm and signature algorithm. We denote the CPU computing capability of the ith edge node as cp_i, $(i = 1, 2, \cdots, m.)$. da_i denotes the data volume of ith subtask. To reduce service time and improve resource utilization of the IoT-Cloud, a task can be divided into multiple subtasks and assigned to several devices. Furthermore,subtasks require different execution times for their calculation. Thus, one must establish a resource allocation scheme prior to evaluating the value of $T_{transmission} + T_{calculation}$. One of the most important goals of a resource allocation scheme is to obtain a minimum value of $T_{transmission} + T_{calculation}$.

Let us assume that there are N edge computing nodes available and m edge nodes are involved in a service, that is to say, one task of the service is broken down into m small subtasks. For simplicity, let us assume that the ith subtask is processed by the ith edge node. The total data volume of a task demand to be transmitted from the user end to the edge networks is denoted by tv_t. We denote the calculation amount of the ith subtask as ca_i. Thus, the time for computing the ith subtask by edge node i is:

$$t_i^E = \frac{ca_i}{cp_i} \tag{10}$$

According to circuit theories [24], the CPU power consumption of the ith edge node is calculated as

$$p_i^E = \rho_i ca_i \tag{11}$$

where ρ_i is the power coefficient of energy consumed per CPU cycle.

Similar to [24], our primary concern is upload time, while downlink delay is ignored since the data size after task processing is generally much smaller than before processing. Let v_i denote the uplink data rate. Then, we can obtain that the time to transmit the ith

subtask is $t_i = da_i/v_i$. Formally, the task offloading problem based on minimum service time can be formulated as follows:

$$\min S_T = \sum_{i=1}^{m} (t_s \cdot cp_i \cdot da_i + da_i/v_i + ca_i/cp_i) \tag{12}$$

subject to

$$\begin{cases} \Sigma_{i=1}^{m} da_i = tv_t, \\ \rho_i ca_i \leq E_i^{max} \quad (1 \leq i \leq m), \\ da_i \geq 0 \end{cases} \tag{13}$$

Note that the objective function S_T is linear and the constraints are all linear. As a result, $\min S_T$ can be optimally solved by any solver for linear programming problems.

4.2.2. The Minimum Price Evaluation

Generally, a service provider needs to pay for equipment leasing, broadband connection, and information security measures during the business operation period. Thus, the total service price S_P can be generally computed by the service provider using the following formula:

$$S_p = P_{security} + P_{transmission} + P_{calculation} \tag{14}$$

where $P_{security}$ is the price of security measures, e.g., encryption and digital signature. We denote the unit price of node i on a security operation as p_s. Thus, $P_{security}$ can be calculated as follows:

$$P_{security} = \sum_{i=1}^{m} p_{s_i} \cdot da_i \tag{15}$$

$P_{transmission}$ is the price of data transmission. In order to attract users, broadband billing models are varied in various countries. The three most popular means are flat rate billing, billing by flow, and billing by time. If the flat rate billing is adopted, the broadband Internet fee is a constant. The transmission price of the 4G network can be calculated as follows:

$$P_{transmission-1_{4G}} = p_{cs4}, \tag{16}$$

The transmission price of the 5G network can be denoted as follows:

$$P_{transmission-1_{5G}} = p_{cs5}, \tag{17}$$

where p_{cs4}/p_{cs5} denotes the monthly fee, which is a constant. When the flow charge model is adopted, the broadband price is proportional to the data volume. Moreover, the unit price in the 4G network, which we denote by p_{4f}, is assumed to be different from that in the 5G network, denoted by p_{5f}. The data amounts transmitted by the 4G link and 5G link are denoted by da_{4G} and da_{5G}, respectively. Then, the transmission price can be calculated as follows:

$$P_{transmission-2} = p_{4f} \cdot da_{4G} + p_{5f} \cdot da_{5G}, \tag{18}$$

We ignore the price evaluation of 3G and 2G because of their low rate and low market competitiveness compared with 4G and 5G. Without loss of generality, we assume that the data associated with the ith subtask, with $i \in \{1, 2, \cdots, F\}$, is transmitted using the 4G link and that the data associated with the ith subtask, with $i \in \{F+1, F+2, \cdots, m\}$, is transmitted using the 5G link. We can obtain that

$$P_{transmission-2} = p_{4f} \cdot \sum_{i=1}^{F} da_i + p_{5f} \cdot \sum_{i=F+1}^{m} da_i \tag{19}$$

The price evaluation of the mixture of the flow charge model and time charge model is easy to obtain by adding the two estimation formulas above; therefore, we omit it for

brevity. The above estimation formulas of the transmission price can be merged into a single formula:

$$\begin{aligned} P_{transmission} &= \gamma \cdot da_{4G} + \delta \cdot da_{5G} + \epsilon \\ &= \gamma \cdot \sum_{i=1}^{F} da_i + \delta \cdot \sum_{i=F+1}^{m} da_i + \epsilon \end{aligned} \tag{20}$$

where $\gamma, \delta, \epsilon \geq 0$ are constant coefficients determined by the billing models above. $(\gamma, \delta, \epsilon) = (0, 0, p_{cs4}/p_{cs5})$ if the flat rate billing model is adopted, or $(\gamma, \delta, \epsilon) = (p_{4f}, p_{5f}, 0)$ if the flow charge model is adopted. We denote the unit calculation price of the ith edge node by pc_i, $i \in \{1, 2, \cdots, m\}$. The calculation price of the ith subtask can be computed as $pc_i \cdot ca_i/cp_i$. Two tasks with the same data size often require different calculation times; therefore, the calculation costs of the two tasks are quite different. In order to better estimate the various costs of edge computing services, such as computation costs, transportation costs, and storage overhead, we must carefully analyze the relationship between the amount of computation and the amount of data. Referring to Assumption 1, we assume that the computation amount is proportional to the data size transported from the users' side to the edge nodes and the scale factor is denoted by l. The scale factor l varies with the task. The total calculation price of the task can be computed by

$$\begin{aligned} P_{calculation} &= \sum_{i=1}^{m} \frac{pc_i \cdot ca_i}{cp_i} \\ &= \sum_{i=1}^{m} \frac{l \cdot pc_i \cdot da_i}{cp_i} \end{aligned} \tag{21}$$

Thus, S_P can be calculated as follows:

$$\begin{aligned} S_P = &\sum_{i=1}^{m} p_{s_i} \cdot da_i + \gamma \cdot \sum_{i=1}^{F} da_i + \delta \cdot \sum_{i=F+1}^{m} da_i + \epsilon \\ &+ \sum_{i=1}^{m} \frac{l \cdot pc_i \cdot da_i}{cp_i} \end{aligned} \tag{22}$$

In order to obtain the optimal solution of S_P, we need to compute the minimum value of the following expression:

$$\begin{aligned} \min S_P &= \gamma \cdot \sum_{i=1}^{F} da_i + \delta \cdot \sum_{i=F+1}^{m} da_i + \sum_{i=1}^{m} \frac{l \cdot pc_i \cdot da_i}{cp_i} \\ &= \sum_{i=1}^{F} (p_{s_i} + \gamma + \frac{l \cdot pc_i}{cp_i}) \cdot da_i + \sum_{i=F+1}^{m} (p_{s_i} + \delta + \frac{l \cdot pc_i}{cp_i}) \\ &\quad \cdot da_i + \epsilon \end{aligned} \tag{23}$$

subject to

$$\begin{cases} \Sigma_{i=1}^{m} da_i = tv_t, \\ \rho_i ca_i \leq E_i^{max} (1 \leq i \leq m), \\ da_i \geq 0 \end{cases} \tag{24}$$

Similar to the solution of $\min S_T$, S_P can be also be solved by any solver for linear programming problems.

4.2.3. Minimum Price Evaluation with a Given Time

In certain situations, users want to enjoy services at the lowest price within a given time. Let us denote the given time by T_G. To successfully proceed with the service process, T_G must be no less than $\min \hat{S}_T$ computed by the Formula (13) above. Combining all the

aforementioned models, the minimum price estimation problem within a given time can be further formulated with the following linear program:

$$\begin{aligned} \min S_P^* = & \sum_{i=1}^{F}(p_{s_i} + \gamma + \frac{l \cdot pc_i}{cp_i}) \cdot da_i + \sum_{i=F+1}^{m}(p_{s_i} + \delta \\ & + \frac{l \cdot pc_i}{cp_i}) \cdot da_i + \epsilon \end{aligned} \tag{25}$$

s.t.

$$\begin{cases} \sum_{i=1}^{m}\{t_s \cdot cp_i \cdot da_i + da_i/v_i + ca_i/cp_i\} \leq T_G, \\ \sum_{i=1}^{m} da_i = tv_t, \\ \rho_i ca_i \leq E_i^{max}(1 \leq i \leq m), \\ da_i \geq 0(1 \leq i \leq m) \end{cases} \tag{26}$$

This multivariate linear optimization problem can be solved by any solver for linear programming problems. The service estimation and resource allocation algorithm is presented in Algorithm 2. All services are divided into three types: Type 1, services completed within the shortest time; Type 2, services completed with the lowest price; and Type 3, services completed with minimum price estimation within a given time. Since the number of edge nodes used for completing a service is unpredictable, to complete the service by using a minimum amount of equipment, we set the initial value of m as 1 and the step length as 1 in Algorithm 2. These settings can be changed according to concrete scenarios, e.g., we can use dichotomy to quickly search for an appropriate value of m. Trust management can effectively distinguish malicious nodes from normal nodes through the observation of historical behavior. The IoT nodes with low trust values are removed from the routing table; that is to say, the IoT device without security functions is not used as a transmission relay node for IoT applications with security requirements, such as encryption and digital signature. Furthermore, both the total service time S_T and the total service price S_P include the costs of security operations (refer to Formulas (12), (24) and (25)). Thus, trust management can help users obtain information security services at a lower price and less delay.

4.3. Time Complexity

It is convenient to check that the PECSA algorithms can be finished in polynomial time since they only rely on solving a series of linear programming problems with polynomial time complexity. There are many solution approaches available for linear programming, from which we choose the interior point method due to its practical performance. In general, the interior point method converges in time $O(m^3)$ [39], where m is the total number of available edge nodes in each observation window.

Algorithm 2: Service estimation and resource allocation algorithm.

Input:
R_T,R_P,R_S: requirements of user
tv_t: the total data volume of a task
cp_i: the computing power of the ith edge node
v_i: data transmission rate from user to the ith edge node
l: the ratio of the data volume associated to calculation amount
γ, δ, ϵ: coefficients determined by billing models
p_s: the unit price of a digital signature
$pc_i (i = 1, 2, \cdots, m)$: unit calculation price of the ith edge node
ST: the service type
Output:
S_T: available service time
S_P: available service price
$da_1, da_2, \cdots, da_m$: the task allocation quota
01: **for** $m = 1; m \leq N; m++$ {
02: **if** ST=1
03: {compute the minimum service time S_T;
04: **if** $S_T \leq R_T$
05: {compute $da_1, da_2, \cdots, da_m$;
06: compute S_P ;
07: **break**; }
08: **else**
09: **continue**;}
10: **else if** ST=2
11: { compute the minimum service price S_P ;
12: **if** $S_P \leq R_P$
13: { compute $da_i (i = 1, 2, \cdots, m)$;compute S_T ;
14: **break**; }
15: **else**
16: **continue**;}
17: **else if** ST=3
18: {compute the minimum service price within a given time S_P^*;
19: compute $da_i (i = 1, 2, \cdots, m)$ related with S_P^* }}

5. Performance Evaluation

In this section, we provide a simulation experiment concerning the trust evaluation mechanism, the service time, and the service costs for edge computing. The experimental results are divided into four parts: (1) we study the impact of trust value on the performance of task offloading; (2) we investigate the impact of the task computation amount on the service time and price; (3) we study the impact of task data size on the service quantities in terms of task duration and price cost; and (4) we compare the proposed PECSA with several task offloading schemes.

5.1. Experiment Setup

The simulation platform is Matlab 2020a. The experimental network settings include 20 edge nodes and 200 IoT nodes, where 10 edge nodes and 100 IoT nodes are 4G-enabled devices and the others are 5G-enabled devices. Users and edge nodes move randomly at a speed of 20 m/s. The 20 edge nodes are divided into five groups with four nodes in each group, denoted as GA, GB, GC, GD, and GE. The computing power and price of each group are inconsistent; please refer to Table 3. Each device in the IoT and edge networks has a trust value T in [0, 1]. When the trust value decreases by 0.1, the computing power also decreases by 0.1. With reference to the charging standard of China unicom, one of the biggest network service providers in China, p_{cs4}/p_{cs4} are set as 80/500. The security prices are usually invisible to users; in fact, they are measured by the amount of data stored in

the cloud or edge cloud. With reference to the charging standard of Alibaba Cloud, one of the biggest cloud service providers, p_{4f}/p_{5f} are set as 0.002/0.003. The main parameters used in the simulations are listed in Table 3.

Table 3. Experimental parameters.

Parameter	Value
Data transmission rate of 5G, v_i	1024 Mbps
Data transmission rate of 4G, v_i	72 Mbps
Computing capability of nodes in GA, cp_i	0.08 ms/10,000 multiplications
Computing capability of nodes in GB, cp_i	0.12 ms/10,000 multiplications
Computing capability of nodes in GC, cp_i	0.15 ms/10,000 multiplications
Computing capability of nodes in GD, cp_i	0.20 ms/10,000 multiplications
Computing capability of nodes in GE, cp_i	0.25 ms/10,000 multiplications
Calculation price of nodes in GA, pc_i	0.06/10,000 multiplications
Calculation price of nodes in GB	0.03/10,000 multiplications
Calculation price of nodes in GC	0.012/10,000 multiplications
Calculation price of nodes in GD	0.0054/10,000 multiplications
Calculation price of nodes in GE	0.0036/10,000 multiplications
Security price, p_{s_i}	0.08/1 MB
Computing energy consume in edge cloud, ρ	90 W/Gigacycles
The total battery capacity, E_i^{max}	1000 J

5.2. Comparative Analysis

For trust evaluation, we consider the situation in which the malicious data device behaves abnormally and generates incorrect trust values to cause the user to make incorrect decisions. Figures 2 and 3 show the comprehensive trust evaluation results of a good node and a malicious node, randomly picked, respectively. *pm* denotes the proportion of malicious nodes. The convergence time is very small, about 50 s for both figures, where the update interval is set as 5 s. We see that trust convergence behavior is observed for both figures. The trust bias is acceptable since the trust value of the good node is between 0.7 and 0.8 and that of the malicious node is between 0.45 and 0.55. In a complex communication environment, good nodes and malicious nodes could be distinguished by their trust value.

The service time for a requirement is determined by the amount of computation and transmission. Two tasks that transmit the same amount of information often require different calculation amounts. The product factor l is used to describe the relationship between the data volume and calculation amount of a task. Our proposed architecture can be applied to fields such as artificial intelligence, image recognition, coding, and so on. In these fields, various iterative algorithms are usually used to complete specific work. The iterative algorithm can be implemented by a for-loop statement. The calculation times of each cycle are the same, so the value of l can be determined by the number of iterations. For example, in [40], BP-decoding algorithms achieved good results in 20 iterations. If the amount of calculation per iteration is 50, then l is 1000. In [41], the RANSAC algorithm is used for image matching. The upper limit of the number of iterations is 400. Let us suppose that one iteration requires 250 operations; then, l is 100,000. If the number of iterations of the improved RANSAC algorithm is 40, then l is 10,000. In fact, the value of l depends on the number of iterations of a specific algorithm. To illustrate the efficiency of our proposed scheme, here we set l = 1000, 2000, 5000, 10,000, 50,000, and 100,000, respectively. In Figures 4–7, the data volumes of the task are set to 5 GB. Usually, the price is a critical factor for a user when purchasing cloud services. Four charging models are considered in our architecture: flat rate billing of the 4G network, flat rate billing of the 5G network, and billing by flow.

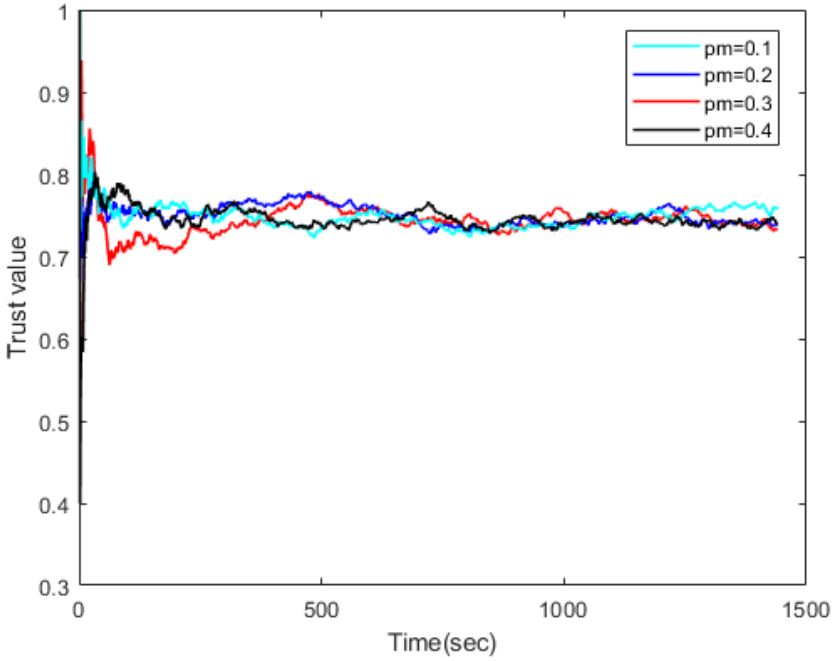

Figure 2. Trust value of a good node under adaptive control with *pm* ranging from 0.1 to 0.4.

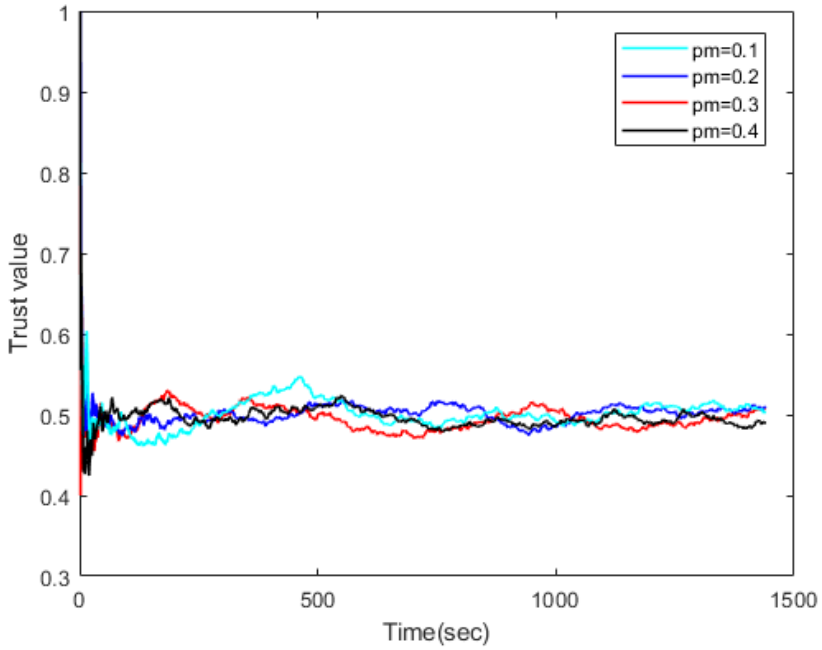

Figure 3. Trust value of a malicious node under adaptive control with *pm* ranging from 0.1 to 0.4.

Figure 4 shows that the time cost of a task increases with the increase in the coefficient l. The abscissa is not uniformly distributed; the delay at l = 10,000 is about twice the delay at l = 5000, and the delay at l = 50,000 is about five times the delay at l = 10,000. In general, cost and delay are approximately linear. This is because, when l takes a larger value, the delay of data calculation is much greater than that of data transmission. Therefore, the increase in total delay is mainly caused by the increased amount of calculations. Figure 5 shows that the price costs of the four models increase with the increase in l. The price costs of the two flat rate billing models include not only the cost of the task but also the monthly fee; therefore, they are more than the costs of billing by flow. Note that the price at l = 10,000 is about twice the price at l = 5000, but the price at l = 50,000 is only about twice the price at

l = 10,000. This is because the price of data transmission and security operations remain unchanged, and they account for a large proportion of the total price.

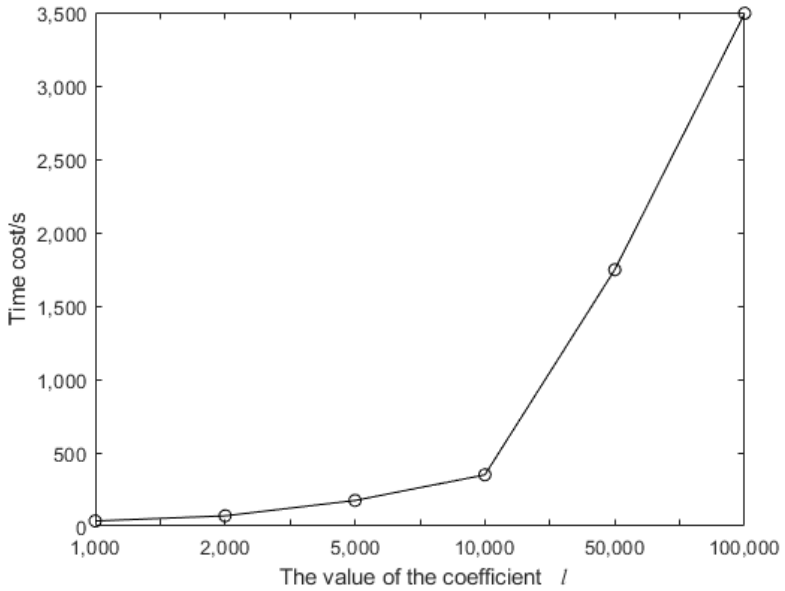

Figure 4. Service time based on different values of l.

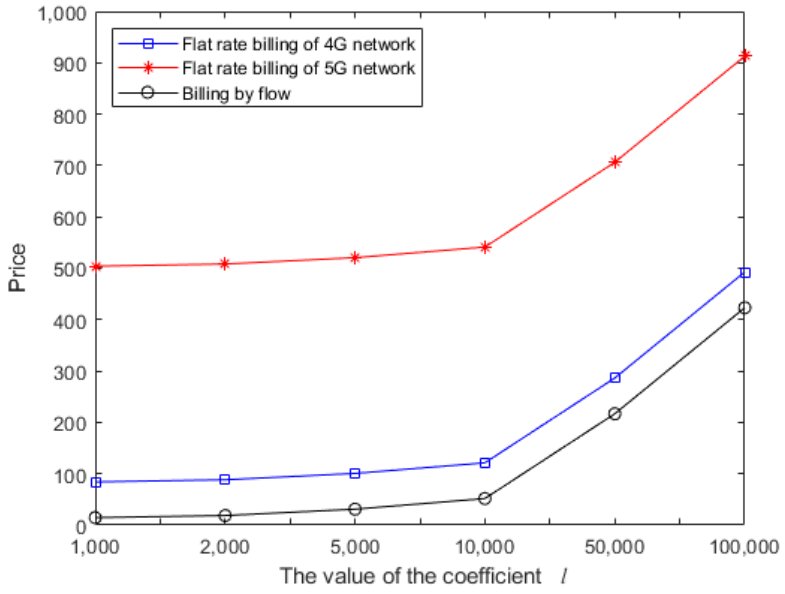

Figure 5. Service price based on different values of l and network charging models.

Figure 6 shows the impact of trust values on time costs. When the trust value is equal to 1, the edge nodes have the best performance and the fastest computing speed. When the trust value is less than 0.7, the time cost increases significantly. When the trust value is 0.6, the time cost is almost three times as long as that when the trust value is 1. There is a small increase in the price costs with the decrease in trust value, in Figure 7, which is due to edge devices with the lowest price costs; however, low trust values cannot continue to provide services. The performance degradation is very severe if the average trust value is less than 0.6 and the probability is very small in practice; thus, we ignore the situation. By comparing Figures 6 and 7, we can see that lower trust value brings a slight rise in price cost but also a significant increase in time cost. In other words, users pay almost the same price to buy a lower quality service. Therefore, a low trust value is unfavorable to users.

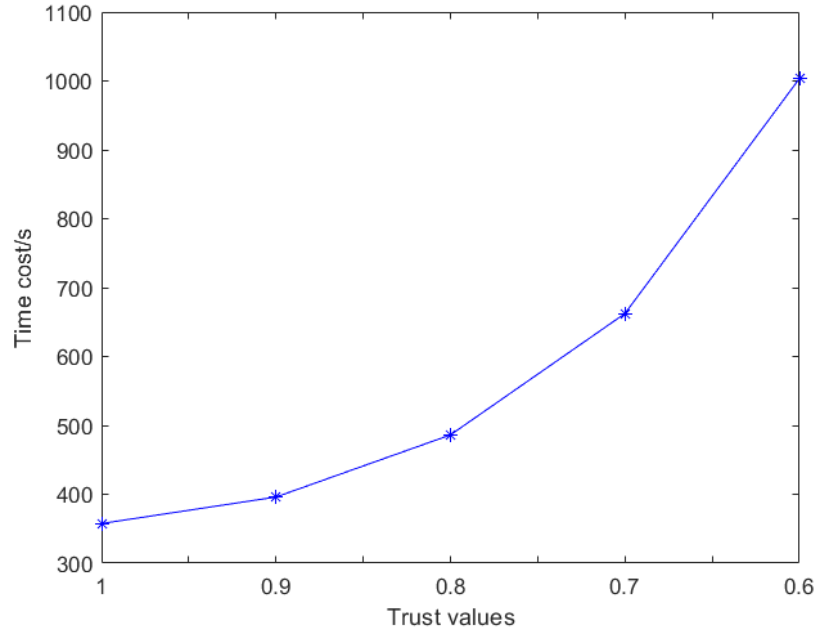

Figure 6. Service time based on different trust values.

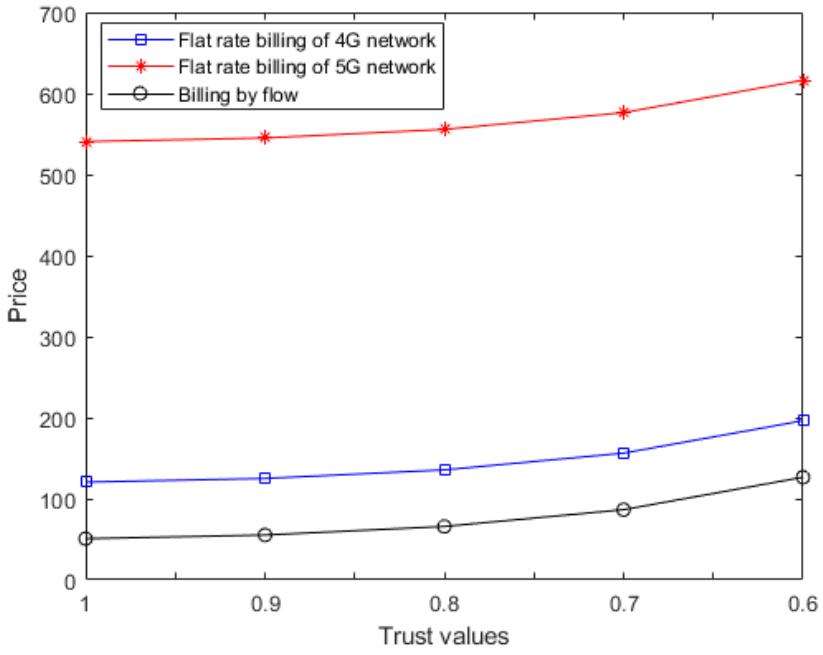

Figure 7. Service price based on different trust values.

In fact, many users want to achieve a trade-off between price and time costs. The tasks are required to be fulfilled at the lowest price within a given period of time. Figure 8 shows that the price costs decrease with the increase in the time limit. Thus, it is feasible to wait longer and pay a lower price. For tasks with large amounts of computation, e.g., l = 50,000, or l = 100,000, adding a small service time leads to a significant price drop. When the given service time is up to $\Delta + 500$ s, the downward trend becomes very small. For less computationally intensive tasks, e.g., $l = 1000, l = 2000, l = 5000$, or l = 10,000, the effect of relaxing the time limit to reduce the price cost is not obvious.

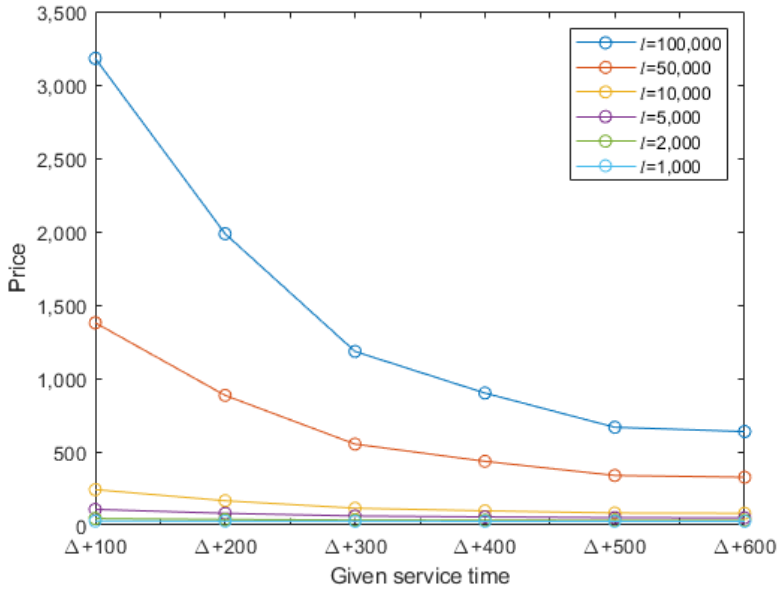

Figure 8. Service price based on different values of l and time limitations: $\triangle + 100$ s, $\triangle + 200$ s, $\triangle + 300$ s, $\triangle + 400$ s, $\triangle + 500$ s, $\triangle + 600$ s, and $\triangle + 100$ s, where $\triangle$ denotes the minimum service time.

We compare the service price and time costs of PECSA to those of the MOERA [4] and to the online greedy scheme under different workloads. The experimental results are shown in Figures 9 and 10, respectively. Note that, with the increase in ε, the empirical competitive ratio of MOERA varies over a very small range. Thus, the parameter ε of MOERA is set to be 5 in our experiment. Figures 9 and 10 show that our scheme has obvious advantages in time cost and price cost.

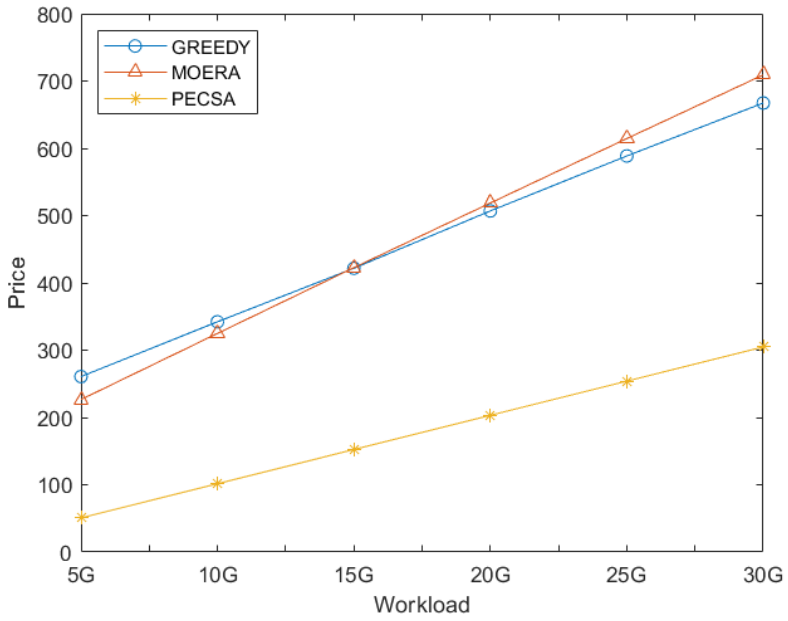

Figure 9. Service price comparison.

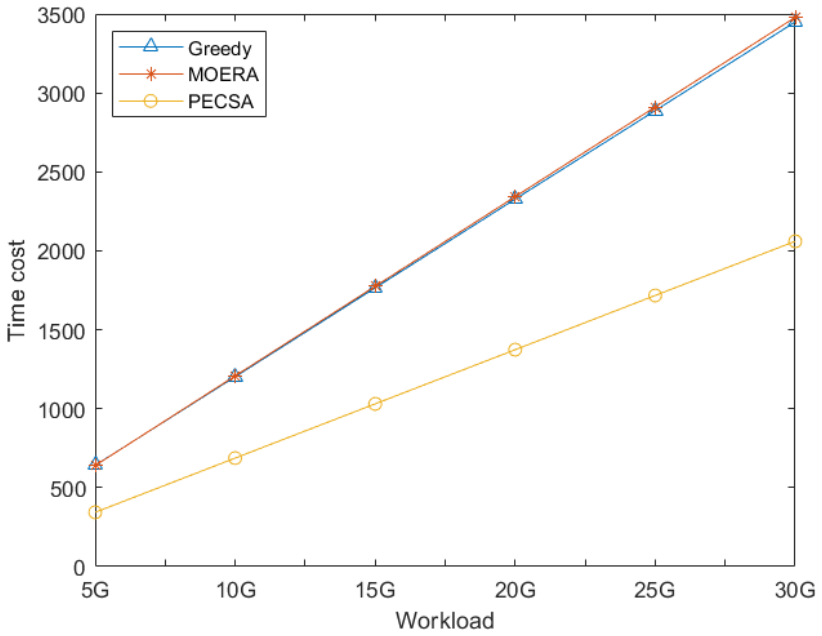

Figure 10. Service price comparison.

6. Discussions

We now discuss some practical issues in implementing the proposed algorithm in real systems. One important issue is the threshold selection in Section 4.1. In real scenarios, threshold selection is usually conducted depending on the features of a real environment. In this work, we acknowledge that 0.6 is a suitable threshold for our settings and experimental environment. However, there is a need to explore and choose a specific threshold in other environments. Due to network congestion and other problems, the performance of IoT nodes in busy times is weaker than that in idle times. The threshold for busy hours is recommended to be lower than that for idle hours. Another important issue is how to determine the relationship between data quantity and calculation quantity, that is to say, how to set the value of l. For the algorithm with uncertain iteration times, a slice of the total data can be taken out for processing to obtain the specific value of the scale coefficient. For an algorithm with a known number of iterations, the scale coefficient can be determined by multiplying the number of iterations by the amount of calculation per iteration.

7. Conclusions

In this paper, we divide the requirements of IoT-based applications into three categories: security requirements, price requirements, and time requirements. To meet these requirements, we first propose an edge computing service architecture integrated with a trust management methodology. Then, we propose three cost estimation algorithms to evaluate the minimum service time, the minimum service price, and the minimum service price with a given time. The experimental results show that our proposed architecture can greatly improve the service efficiency of IoT-based applications and meet various service requirements adaptively. In order to perfect the architecture, interesting directions for future work should seek to solve other issues through this architecture, such as studying the problem of queuing of tasks, and the dynamics on the addition and removal of edge nodes and network congestion.

Author Contributions: Conceptualization, J.L. and Z.W.; data curation, J.L.; formal analysis, Z.W.; funding acquisition, J.L.; methodology, J.L. and Z.W.; software, Z.W.; supervision, J.L.; writing—original draft, J.L. and Z.W.; writing—review and editing, Z.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the first-class undergraduate program "Double Ten Thousand Plan" under grant 14002600100019J111, in part by the Civil Aviation Education Talent project under grant E2021013, and by the industry, university and research cooperation project of the Ministry under grant 202101311032, and by the 2020 Central University Education Teaching Reform Special Fund under grant E2020017.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Shen, S.; Han, Y.; Wang, X.; Wang, Y. Computation Offloading with Multiple Agents in Edge-Computing–Supported IoT. *ACM Trans. Sens. Netw.* **2020**, *16*, 1–27. [CrossRef]
2. Salaht, F.; Desprez, F.; Lebre, A. An Overview of Service Placement Problem in Fog and Edge Computing. *ACM Comput. Surv.* **2020**, *53*, 1–35. [CrossRef]
3. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
4. Wang, L.; Jiao, L.; Li, J.; Gedeon, J.; Mühlhäuser, M. MOERA: Mobility-Agnostic Online Resource Allocation for Edge Computing. *IEEE Trans. Mob. Comput.* **2019**, *18*, 1843–1856. [CrossRef]
5. Liu, Y.; Peng, M.; Shou, G.; Chen, Y.; Chen, S. Toward Edge Intelligence: Multiaccess Edge Computing for 5G and Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 6722–6747. [CrossRef]
6. Sladana, J.; Gyorgy, D. Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing. *IEEE/ACM Trans. Netw.* **2020**, *28*, 667–680. [CrossRef]
7. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Turorials* **2017**, *19*, 1628–1656. [CrossRef]
8. Chen, X.; Jiao, L.; Li, W. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE-ACM Trans. Netw.* **2016**, *24*, 2827–2840. [CrossRef]
9. Mao, Y.; Zhang, J.; Letaief, K. Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3590–3605. [CrossRef]
10. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile Edge Computing: A Survey. *IEEE Internet Things J.* **2018**, *5*, 450–460. [CrossRef]
11. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The casefor VM-based cloudlets in mobile computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]
12. Wang, T.; Zhang, G.; Liu, A.; Bhuiyan, A.; Jin, Q. A Secure IoT Service Architecture with an Efficient Balance Dynamics Based on Cloud and Edge Computing. *IEEE Internet Thing J.* **2019**, *6*, 4831–4843. [CrossRef]
13. Satyanarayanan, M.; Simoens, P.; Xiao, Y.; Pillai, P.; Chen, Z.; Ha, K.; Hu, W.; Amos, N. Edge analytics in the internet of things. *IEEE Pervasive Comput.* **2015**, *14*, 24–31. [CrossRef]
14. Ning, Z.; Dong, P.; Wang, X.; Rodrigues, J.; Xia, F. Deep Reinforcement Learning for Vehicular Edge Computing: An Intelligent Offloading System. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–24. [CrossRef]
15. Jararweh, Y. Enabling efficient and secure energy cloud using edge computing and 5G. *J. Parallel Distrib. Comput.* **2020**, *145*, 42–49. [CrossRef]
16. Jia, G.; Han, G.; Du, J.; Chan, S. A Maximum Cache Value Policy in Hybrid Memory-Based Edge Computing for Mobile Devices. *IEEE Internet Things J.* **2019**, *6*, 4401–4410. [CrossRef]
17. Wang, F.; Xu, J.; Wang, X.; Cui, S. Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 1784–1797. [CrossRef]
18. Yi, S.; Qin, Z.; Li, Q. Security and privacy issues of fog computing: A survey. In Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications, Qufu, China, 10–12 August 2015; pp. 685–695.
19. Stojmenovic, I.; Wen, S.; Huang, X.; Luan, H. An overview of fog computing and its security issues. In *Concurrency and Computation: Practice and Experience*; Wiley: New York, NY, USA, 2015; Volume 28.
20. Li, H.; Shou, G.; Hu, Y.; Guo, Z. Mobile edge computing: Progress and challenges. In Proceedings of the 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), Oxford, UK, 29 March–1 April 2016; pp. 83–84.
21. Jeong, S.; Simeone, O.; Kang, J. Mobile Edge Computing via a UAV-Mounted Cloudlet: Optimization of Bit Allocation and Path Planning. *IEEE Trans. Veh. Technol.* **2018**, *67*, 2049–2063. [CrossRef]
22. Bi, S.; Zhang, Y. Computation Rate Maximization for Wireless Powered Mobile-Edge Computing With Binary Computation Offloading.*IEEE Trans. Wirel. Commun.* **2018**, *17*, 4177–4190. [CrossRef]

23. Zhang, K.; Hua, Y.; Tian, F. A coalition-structure's generation method for solving cooperative computing problems in edge computing environments. *Inf. Sci.* **2020**, *536*, 372–390. [CrossRef]
24. Chen, M.; Hao, Y. Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 587–597. [CrossRef]
25. Shah, H.; Wong, V. Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game. *IEEE Internet Things J.* **2018**, *5*, 3246–3257. [CrossRef]
26. Barcelo, M.; Correa, A.; Llorca, J.; Tulino, A.M.; Vicario, J.L.; Morell, A. IoT-cloud service optimization in next generation smart environments. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 4077–4090. [CrossRef]
27. Guo, H.; Liu, J.; Zhang, J.; Sun, W.; Kato, N. Mobile-Edge Computation Offloading for Ultradense IoT Networks. *IEEE Internet Things J.* **2018**, *5*, 4977–4988. [CrossRef]
28. Premsankar, G.; Di Francesco, M.; Taleb, T. Edge Computing for the Internet of Things: A Case Study. *IEEE Internet Things J.* **2018**, *5*, 1275–1284. [CrossRef]
29. Chen, I.-R.; Guo, J.; Bao, F. Trust management for SOA-based IoT and its application to service composition. *IEEE Trans. Serv. Comput.* **2017**, *9*, 482–495. [CrossRef]
30. Alshehri, M.D.; Hussain, F.K.; Hussain, O.K.; Omar, K. Clustering-Driven Intelligent Trust Management Methodology for the Internet of Things (CITM-IoT). *Mob. Netw. Appl.* **2018**, *23*, 419–431. [CrossRef]
31. Duan, J.; Gao, D.; Yang, D.; Foh, C.H.; Chen, H.-H. An energy aware trust derivation scheme with game theoretic approach in wireless sensor networks for IoT applications. *IEEE Internet Things J.* **2014**, *1*, 58–69. [CrossRef]
32. Zhang, C.; Zhu, X.; Song, Y.; Fang, Y. A Formal Study of Trust-Based Routing in Wireless Ad Hoc Networks. In Proceedings of the INFOCOM'10: 29th Conference on Information Communications, San Diego, CA, USA, 14–19 March 2010; pp. 2838–2846.
33. Suryani, V.; Sulistyo, S.; Widyawan, W. Internet of Things (IoT) framework for granting trust among objects. *J. Inf. Process. Syst.* **2017**, *13*, 1613–1627.
34. Liu, X.; Liu, Y.; Liu, A.; Yang, L.T. Defending ON–OFF attacks using light probing messages in smart sensors for industrial communication systems. *IEEE Trans. Ind. Informat.* **2018**, *14*, 3801–3811. [CrossRef]
35. Meng, W.; Li, W.; Su, C.; Zhou, J.; Lu, R. Enhancing trust management for wireless intrusion detection via traffic sampling in the era of big data. *IEEE Access* **2018**, *6*, 7234–7243. [CrossRef]
36. Zhu, C.; Leung, V.C.; Yang, L.T.; Shu, L.; Rodrigues, J.J.; Li, X. Trust assistance in sensor-cloud. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015; pp. 342–347.
37. Patil, P.; Narayankar, P.; Narayan, D.G.; Meena, S.M. A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish. In Proceedings of the 1st International Conference on Information Security and Privacy 2015, Nagpur, India, 11–12 December 2015; pp. 617–624.
38. Christoph, D.; Maria, E.; Florian, M. Analysis of SHA-512/224 and SHA-512/256. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, 29 November–3 December 2015; pp. 1–30.
39. Khan, M.S. Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture. Ph.D. Thesis, University of Victoria, Victoria, BC, Canada, 1998.
40. Zhao, M.; Xu, B. Serial Decoding Algorithm with Continuous Backtracking for LDPC Convolutional Codes. *Wirel. Pers. Commun.* **2019**, *4*, 1–11. [CrossRef]
41. Zhu, W.; Sun, W.; Wang, Y.; Liu, S.; Xu, K. An Improved RANSAC Algorithm Based on Similar Structure Constraints. In Proceedings of the 2016 International Comference on Robots and Intelligent System, Zhangjiajie, China, 27–28 August 2016; pp. 94–100.