ORIGINAL RESEARCH
published: 28 April 2022
doi: 10.3389/frsip.2022.788943

A Robust Security Task Offloading in Industrial IoT-Enabled Distributed Multi-Access Edge Computing

Eric Gyamfi and Anca Jurcut*

School of Computer Science, University College Dublin, Dublin, Ireland

The rapid increase in the Industrial Internet of Things (IIoT) use cases plays a significant role in Industry 4.0 development. However, IIoT systems face resource constraints problems and are vulnerable to cyberattacks due to their inability to implement existing sophisticated security systems. One way of alleviating these resource constraints is to utilize multi-access edge computing (MEC) to provide computational resources at the network edge to execute the security applications. To provide resilient security for IIoT using MEC, the offloading latency, synchronization time, and turnaround time must be optimized to provide real-time attack detection. Hence, this paper provides a novel adaptive machine learning–based security (MLS) task offloading (ASTO) mechanism to ensure that the connectivity between the MEC server and IIoT is secured and guaranteed. We explored the trade-off between the limited computing capacity and high cloud computing latency to propose an ASTO, where MEC and IIoT can collaborate to provide optimized MLS to protect the network. In the proposed system, we converted the MLS task offloading and synchronization problem into an equivalent mathematical model, which can be solved by applying Markov transition probability and clock offset estimation using maximum likelihood. Our extensive simulations show that the proposed algorithm provides robust security for the IIoT network with low latency, synchronization accuracy, and energy efficiency.

Keywords: internet of things, iot-edge, task offloading, multi-access edge computing, time-synchronization, latency, security task offloading

OPEN ACCESS

Edited by:
Jihong Park,
Deakin University, Australia

Reviewed by:
Jun Li,
Guangzhou University, China
Xuelin Cao,
Singapore University of Technology and Design, Singapore

****Correspondence:***
Eric Gyamfi
eric.gyamfi@ucdconnect.ie

Specialty section:
This article was submitted to Signal Processing for Communications, a section of the journal Frontiers in Signal Processing

***Received:** 04 October 2021*
***Accepted:** 03 February 2022*
***Published:** 28 April 2022*

Citation:
Gyamfi E and Jurcut A (2022) A Robust Security Task Offloading in Industrial IoT-Enabled Distributed Multi-Access Edge Computing. Front. Sig. Proc. 2:788943. doi: 10.3389/frsip.2022.788943

INTRODUCTION

Over the decade, high latency, throughput, high cost of setting up cloud infrastructure, and energy optimization have been the principal influence of the IIoT developers to find an alternative to the cloud computing services. Cloud computing consolidates computing, storage, and network management functions in a centralized manner. Hence, there is a need for a modern computing paradigm that will support the IIoT system, which alleviates the challenges mentioned above. Another contributing factor for a change in cloud computing services is the IIoT devices' ability to get real-time responses. Industrial and manufacturing segments have embraced IoT technologies, known as the Industrial Internet of Things (IIoT). This concept of IIoT, which also referred to as industry 4.0., enhances factory and industrial transformation. The IIoT transforms traditional and linear manufacturing supply chains into dynamic, interconnected systems, also known as the digital supply network (DSN). As key enablers of DSNs, IIoT technologies change the way products are made and delivered. IIoT makes factories more efficient, ensuring better safety for human operators

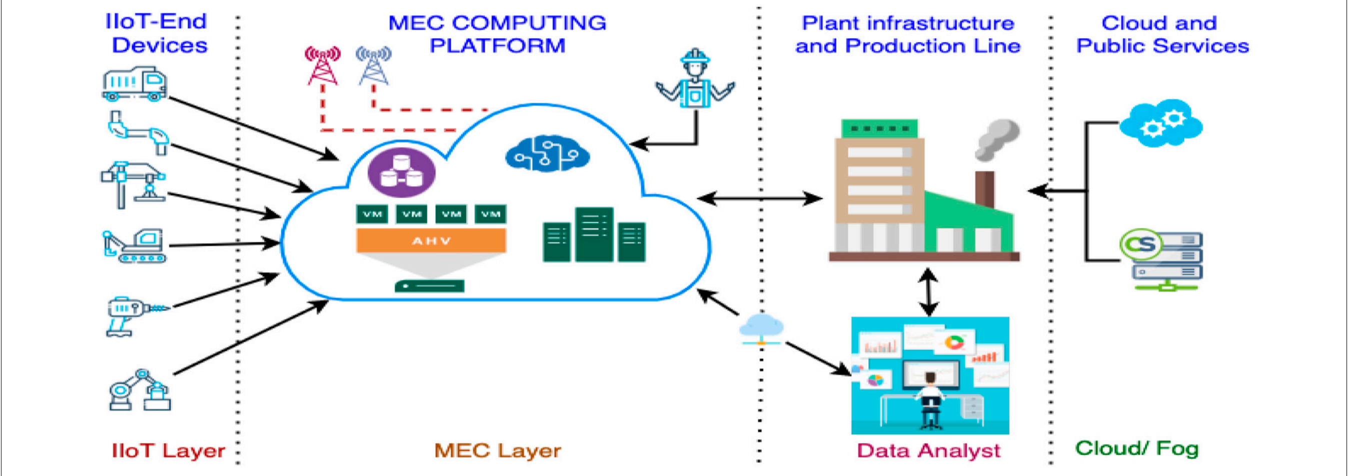

FIGURE 1 | MEC server connected IIoT architecture.

and, in some cases, saving millions of dollars. However, the development of this emerging phenomenal (IIoT) has faced substantial security challenges (Bakhshi et al., 2018). Security is a fundamental concern for the safe and reliable operation of IIoT devices. According to research conducted by the University of Portsmouth (Finnerty et al., 2018) in 2018, 32% of small businesses, 60% of medium companies, and 69% of large-scale businesses in the United Kingdom experienced IIoT security breaches. IIoT security breach had increased to 69% across industries in 2019. In the work of Irdeto-Media-Team (2019), it was forecasted in 2019 that eight out of 10 businesses encountered cyberattacks on their IIoT devices. The increasing number of cyberattacks is critical because organizations use IIoT to capture vital information to make life-dependent decisions. IIoT devices also serve as a platform for cyberattackers to initiate threats against other connected devices in the same network. Such attacks are successful because the IIoT devices are resource-constrained and cannot run the existing security software. Hence, deploying sophisticated security applications at the edge of the IIoT network utilizing multi-access edge computing (MEC) becomes the ideal solution (**Figure 1**). Edge computing is implemented based on a network virtualized platform. Specifically, *network functions virtualization* (NFV) enhances an edge device to supply computing services to numerous connected IIoT devices by creating multiple *virtual machines* (VMs) (Mao et al., 2017) to perform different tasks spontaneously or operate different network functions.

Paper Motivation

Because IIoT plays a critical role in the modern industrial revolution and livelihood, creating a resilient security system to protect IIoT devices is paramount (Wurm et al., 2016). However, the IIoT has resource-constraint problems, and they are deployed in hostile distributed environments, which expose their network interfaces to external access and cyberattacks. The issues above can be mitigated by placing a MEC server at the edge of the IIoT network, which hosts a sophisticated security system. The proposed method demonstrates the deployment of an online machine learning–based security (MLS) system to the MEC server while offloading the MLS task from the IIoT devices at a timely interval to perform deep intrusion analysis and provide adequate security for the connected IIoT devices. An online machine learning algorithm is employed in our proposed method to enhance the security model ability to continuously learn new attacks and normal data to prevent it from been obsolete. The offloading process between the IIoT and the MEC server must be optimized to achieve real-time security response. Because of the number of IIoT devices that share the MEC server resources in the network topology, three common problems arise (Akherfi et al., 2018).

- Network congestion
- Energy wastage
- The inability of the MEC server to identify high priority tasks.

This paper presents a novel ASTO that uses a minimum resource to offload computational MLS from the IIoT devices (nodes or the sources) to a proximal MEC server (vertices). The MEC server receives the IIoT end device request and creates a schedule for offloading MLS task based on the availability of resources, congestion rate (latency on the network), and priority of the task. Each IIoT in the queue receives an update timestamp from the MEC server to enhance time synchronization and energy consumption optimization. We also modeled the network between the IIoT devices and the MEC servers as a probabilistic direct acyclic graph. Moreover, we applied the Markov queuing process to optimize the congestion and latency rate during the MLS task offloading. Our contributions in this paper are summarized as follows:

- We propose a novel energy-efficient ASTO to produce reliable security in a distributed large-scale IIoT network.

- Node pair selection algorithms to determine the next available MEC server to handle the incoming high-intensity security computation are also applied.
- We derive an efficient algorithm for clock offset that employs a two-way synchronization exchange model for the MLS task offloading process.
- Finally, the proposed technique is validated by offloading a MLS task to a proximal MEC server in a distributed IIoT network.

Outline

The rest of this paper is organized as follows: **Section 2** contains the related works. **Section 3** presents the mathematical models of the task offload process. **Section 4** provides a detailed presentation of the proposed ASTO algorithm. **Section 5** describes the experimental setup and performance evaluations, and **Section 6** concludes our findings.

RELATED WORKS

Recently, many researchers have proposed algorithms for computational task offloading in MEC. Sun and Ansari (Sun and Ansari, 2017) solved the latency problem in task offloading by formulating a mathematical model for the task offloading to minimize the average response time for IoT devices and presented a method to solve it efficiently. The resources available in MEC compared to centralized cloud computing have motivated many researchers to jointly address the resource allocation and task offloading problem in the quest to utilize the available resources (Tanaka et al., 2018). Chen et al. (2019) tackled the energy efficiency challenges in task offloading by applying stochastic optimization techniques to transform the original stochastic problem (energy wasted) into a deterministic optimization problem and proposed an energy-efficient dynamic offloading algorithm called EEDOA. Xu et al. (2019) used a blockchain-enabled computation offloading method, called BeCome, in their research. Blockchain technology is applied in edge computing to secure data integrity. Using a genetic sorting algorithm, they adopted strategies to balance resource allocation among the connected IoT devices. Wu et al. (2020) solved the trade-off overheads of limited computing capacity and high latency while ensuring data integrity during the offloading process in IoT. The authors considered a blockchain scenario where edge computing and cloud computing can collaborate to secure the task offloading process. Chen et al. (2018) and Hsu et al. (2019) explored the novel perspective of resource efficiency and created an efficient computation offloading mechanism, which consists of a delay-aware task graph partition algorithm. They used an optimal VM selection approach to reduce IoT and edge resource occupancy while satisfying its QoS requirement. Moreover, Ansere et al. (2019) focused on using time synchronization in large-scale VANETs and proposed an adaptive beacon time synchronization (ABTS) algorithm to intensify timing message synchronization. Their ABTS algorithm selects the best time synchronization pairs to decrease the number of timing messages transmitted. A distributed time synchronization approach was proposed by Nasrallah et al. (2016) for the two-way timing message synchronization exchange system in inter-cluster and intra-cluster nodes communication. Their architecture allows the automatic clustering of nodes and selection of node heads to maximize energy efficiency. Yang et al. (2021, 2022) proposed machine learning–based IoT to MEC offloading. The authors demonstrated the need to offload computational intensive to the MEC server due to resource-constraint problems on the IoT devices. Their proposed frameworks were backed with experimentation, which proved promising. Sun et al. (2019) also proposed an intelligent computing architecture for the IIoT, including cooperative edge and cloud computing. An AI-enhanced offloading framework for service accuracy maximization is provided based on the proposed computational architecture, including service correctness as a new parameter and latency, and intelligently distributes traffic to MEC servers. In all the above research works, the authors provided the various ways of offloading data to MEC server. However, our paper adds to knowledge by proposing a novel method of providing security for the resource constraint IIoT by offloading the computational logic (MLS) to the MEC while enhancing the task offloading processes using the Markov transition.

Tange et al. (2020) and Sadeghi et al. (2015) reviewed most of the security challenges in IIoT and elaborated on the need to use MEC and Fog to design security for the IIoT devices. They demonstrated that IIoT devices generate, process, and exchange vast amounts of security-critical and privacy-sensitive data, which draws the attention of attackers.

SYSTEM MODEL AND PROBLEM FORMULATION

This section details the system design in detail, including the IIoT model, MEC model, energy-efficient control, latency in IIoT to MEC offloading, and MLS to be offloaded to the MEC server.

Industrial Internet of Things to Multi-Access Edge Computing Network Model

Consider a practical industrial context consisting of *nth* number of IIoT devices μ_n, such that each IIoT device can connect to one of the *kth* number of the proximal distributed MEC servers Φ_k, through a wireless connection with distance δ at the edge of the network. We also assumed that each IIoT device connects to a selected MEC server in an orthogonal link, whereas the IIoT devices share security information through context-awareness. Let ϑ represent the network traffic density for the interconnected IIoT devices and the MEC servers. By applying Poisson distribution for a large-scale network, the probability of selecting the closest MEC server Φ_k with a wireless link x_n can be expressed as follows:

$$F(\Phi_k, x_n) = \frac{(\chi)^{\Phi}(e^{-\chi})}{\Phi_k!}, \Phi_k \geq 0 \tag{1}$$

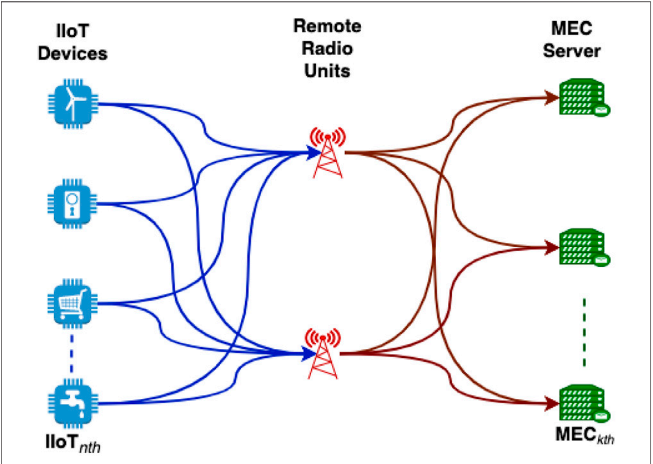

FIGURE 2 | IIoT to MEC DAG representation.

where $\chi = \vartheta x$ signifies the Poisson distribution parameter for a random bounded section of the distributed network. We obtain the probability that the distance between the connected IIoT device and a MEC is minimal than δ. Hence, the likelihood that at least one IIoT connects to a MEC server within δ is expressed as follows:

$$\rho_c\{\delta \le x_n\} = F(x_n) = 1 - e^{-\vartheta x_n} \tag{2}$$

Considering that the IIoT devices connects to a MEC server in an industrial settings, we define a variable $\mathfrak{R}_k$ to determine whether the MEC servers Φ_k are deployed at location $\mathfrak{L}$; then, we can express the variable as follows:

$$\mathfrak{R}_k = \begin{cases} 1 & \text{if MEC server } \Phi_k \text{ is deployed at location } \mathfrak{L}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

However, let the probability that there is no accessible MEC server available for the IIoT device be represented as ρ_o at a distance δ such that MLS task offloading can take place is given by

$$\rho_o = F(0, x_n) = e^{-\vartheta x_n} \tag{4}$$

The state of ρ_o could occur when all the MEC servers are engaged with an equally important task to execute. However, the IIoT device will move into a waiting state and continue to retry till a connection is established to a MEC server. This state is considered in the MLS task offload model in the following subsection.

Computational Machine Learning–Based Security Task Offloading Model

In a MEC-IIoT environment, each IIoT device executes MLS locally or remotely on the MEC servers. We represent each MLS task as $\varphi[\gamma, \tau, \kappa]$ where γ denotes the input size (in bit), τ denotes completion deadline (in seconds), and κ is the computational workload intensity (in CPU cycles per bit). Moreover, the MLS algorithms are implemented on various MEC servers based on a virtualized platform (Bing et al., 2019; Doan et al., 2019) that leverages the current improvements in NFV, information-centric, and software-defined network. During task offloading processes, the IIoT devices could be destructed by network attacks, such as denial-of-service (DoS) network flooding, increasing energy consumption and latency. In our proposed model, the MLS detects network attacks, thereby mitigating all attacks to enhance the optimal offloading process. All our assumptions are based on these virtualization technologies. The MLS begins its process on the IIoT devices and performs deep intrusion analysis, which requires high computation resources on the MEC server. During the MLS task offloading process, energy and latency must be paramount. We represent the task offloading algorithms with the principle of the task call graph. **Figure 2** shows the directed acyclic graph (DAG) of a computational offloading.

For a computational MLS task $\varphi[\gamma, \tau, \kappa]$ with CPU clock speed *fm*, the total energy that is required to execute the task on the IIoT is derived as follows:

$$E_m = z\gamma\kappa f_m^2 \tag{5}$$

and the base execution latency task of $\varphi[\gamma, \tau, \kappa]$ can be calculated with the equation

$$T_m = \frac{\gamma\kappa}{f_m} \tag{6}$$

where $z = 1$ denotes the coefficient of proportionality of the energy consumed per CPU cycle on the IIoT device. If the wireless bandwidth required during task offloading from each IIoT devices to the proximal MEC is $\mathbb{H}_m$, ϖ_m is the latency on the wireless network during the task offloading, and $\mathbb{A}_m$ is the size of offloaded task, then we compute the wireless communication overheads ν_m as follows:

$$\nu_m = \varpi_m + \frac{\mathbb{A}_m}{\mathbb{H}_m} \tag{7}$$

According to **Eqs 5–7**, we can then compute the computational overhead in terms of offloading latency and energy as follows:

$$C_m = \alpha_m^t T_m + \alpha_m^e E_m + \nu_m, \tag{8}$$

where α_m^t and α_m^e represent the weighting parameters of computational time and energy for IIoT device, respectively. The α_m^t and α_m^e are expressed mathematically as follows:

$$\alpha_m^t, \alpha_m^e \in [0, 1], \alpha_m^t + \alpha_m^e = 1. \tag{9}$$

Suppose each MLS task must be offloaded from the IIoT to the MEC server at the edge of the network, then the probability of losing a task during the offloading process is ρ. If a task is destroyed on any wireless networks, then it must be re-offloaded across a different route in the distributed network. The probability that the MLS task is offloaded successfully is denoted by $(1 - \rho)^r$, where r is the total number of remote radio units (RRU) available and $(1 - \rho)$ is the probability of successful offloaded task in d consecutive independent trials.

An IIoT composed of d MLS tasks is much less likely to offload efficiently on the first trial. In such a scenario, the d tasks must be offloaded over the RRU network until it is received successfully or a total of rd successful transmissions. The probability of such an instance is expressed as $(1-\rho)^{rd}$. Let ω be the random variable representing the number of times the MLS is offloaded over the selected wireless network path, the average number of offloaded tasks through the shortest path, $E(\omega)$, and $q_{r,d}$ denote the probability that the task is offloaded successfully in d trials. The $q_{r,d}$ can be expressed mathematically as follows:

$$r, d = (1-\rho)^{rd}$$

$$q_{r,d} = \left(1 - \sum_{j=1}^{d-1} q_{r,j}\right)(1-\rho)^{rd} \tag{10}$$

All the previous attempts must have failed for the re-offloading to be accomplished on the dth try. For the possibility that the offloading succeeded for some $j < d$ is $\sum_{j=1}^{d-1} q_{r,j}$, the prospect of failing is $1 - \sum_{j=1}^{d-1} q_{r,j}$. Finally, the dth attempt must succeed to obtain $(1-\rho)^r$. However, using non-recursive computation, **Eq. 8** can be expressed as follows:

$$\begin{aligned}
q_{r,d} &= \left(1 - \sum\nolimits_{j=1}^{d-1} q_{r,j}\right)(1-p)^r \\
q_{r,d} &= (1-\rho)^r - (1-\rho)^r \sum\nolimits_{j=1}^{d-1} q_r, \\
q_{r,d} &= q_{r,d-1} - (1-\rho)^r q_{r,d-1} \\
q_{r,d} &= \rho_{d-1}\left(1 - (1-\rho)^r\right) \\
q_{r,d} &= q_{r,r}\left(1 - (1-\rho)^r\right)^{d-1} \\
q_{r,d} &= (1-\rho)^r \left(1 - (1-\rho)^r\right)^{d-1}
\end{aligned} \tag{11}$$

Then, the average number of transmissions required to offload a single MLS task to the MEC is given by

$$E(\omega) = \sum_{d=1}^{\infty} d(1-\rho)^r \left(1-(1-\rho)^r\right)^{d-1} = \frac{1}{(1-\rho)^r} \tag{12}$$

We can also estimate the energy saved after offloading the MLS task completely to the selected MEC server as follows:

$$E^{sv} = \sum_{d=1}^{k} \left(\delta_d E_d^{sv}\right) \tag{13}$$

where E_d^{sv} is the dth energy saved. If C_m represent the computational overhead in terms of offloading latency and energy, then, using **Eqs 4**, **6**, E_d^{sv} for offloading MLS task to a MEC server can be calculated with the equation

$$E_d^{sv} = E_m - C_m(E(\omega)) \tag{14}$$

Industrial Internet of Things to Multi-Access Edge Computing Network Connectivity Analysis

We determined the shortest route from an IIoT device (source node) to a selected MEC server by analyzing a collection of available wireless links with a minimum distance and congestion (minimum latency). In the proposed model, the IIoT device denotes the source node, and the target vertex is the MEC server. We apply the shortest route analysis (SRA) approach to solve a minimization problem to create an optimization technique for the offloading process. Our adopted SRA uses the greedy search algorithm to find the optimal solution.

Let S denote a set of IIoT devices (nodes) connected in a network whose final shortest route to the MEC server is determined, μ_n is the source node, and Φ_k is any of the MEC server nodes connected to μ_n. ϑ is the cost of weight or the network traffic on the link between two nodes, and then, ϑ, in our scenario, can be defined as $\vartheta = [N_T, \delta]$, where N_T is the offloaded tasks between two nodes, and δ is the distance between the nodes with a priority queue Q of

$$Q = \Phi_k - S_k \tag{15}$$

We can represent our network topology as a directed graph G and find the shortest distance between the IIoT device and MEC server using Algorithm 1.

Algorithm 1. SRA(G, $\boldsymbol{\vartheta}$, S)

```
1  INITIALIZE SOURCE NODE $(G, S)$
2  $S \leftarrow \emptyset$
3  $Q \leftarrow G.\Phi_k$
4  while $Q \neq \emptyset$ do
5      $\mu_n = min(Q)$
6      $S = S \cup \{\mu_n\}$
7      for each vertices $\Phi_k \in G.Adj[\mu_n]$ do
8          $RELAX(\mu_n, \Phi_k, \vartheta)$
9          if $\delta[\mu_n] + \vartheta[\mu_n, \Phi_k] < \delta[k]$ then
10             $\delta[\Phi_k] = \delta[\mu_n] + \vartheta[\mu_n, \Phi_k]$
```

Adaptive Time Synchronization

We made the following assumptions during our design process to help the IIoT device synchronize with the MEC server during the offloading process.

- Each IIoT device is considered a node that offloads its task and receives responses from the proximal MEC server.
- Each IIoT device maintains its time synchronization information.
- Each IIoT device uses its address (mac address) as an ID during the synchronization process to avoid collisions.
- It is assumed that the network can consist of more than one MEC server with multiple functionalities.

The following two major steps were considered during the ATS processes:

Two-Way Time Synchronization Between Multi-Access Edge Computing Server and Industrial Internet of Things Device

In this section, the two-way timing synchronization and update mechanism for the queuing process required for task offloading are determined. The timing process requires minimal complexity

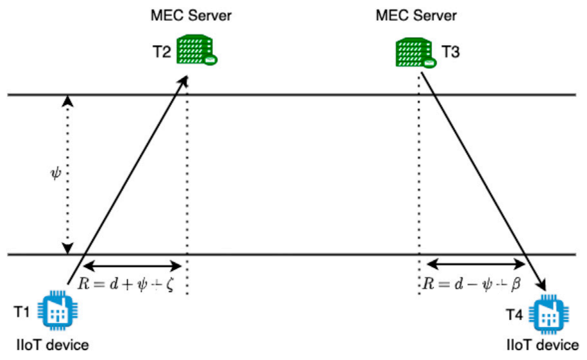

FIGURE 3 | ATS time Synchronization model.

due to the resource constraint problems faced by the IIoT device. The proposed algorithm is self-configured, which allows each IIoT device to record, update, and store its timestamps with its local clock. This method requires three phases: *discovery phase, synchronization*, and *the network evaluation phase*. The following brief description summarizes the ATS;

1. Discovery phase: This is when the IIoT device identifies the proximal MEC server in the network.
2. Synchronization: The ATS establishes a link between the selected MEC server and requests time for offloading based on the priority and available jobs.
3. Evaluation phase: The MEC server re-examines the available jobs in the process queue and network congestion to update the various IIoT devices.

In the ATS, the time updates received by the IIoT device determine when to initiate the task offloading process to minimize energy consumption.

Figure 3 notes a two-way IIoT-MEC-IIoT time exchange handshake between an IIoT device and a MEC server. The timestamps T_1, T_2, T_3, and T_4 are obtained from the duration of the kth offloading request based on the local clocks of IIoT and MEC server, respectively. During the request process, T_1 and T_4 signify the local clock time captured by the IIoT device, whereas T_2 and T_3 denote local clock time captured by MEC server. First, the IIoT sends an offloading request to the MEC server with its current timestamp T_1 after obtaining the shortest path with the SRA. The proximal MEC server records and keeps its new time T_2 at the reception of the request. At time T_3, the MEC server sends a synchronization message to the IIoT device including T_2 and T_3 and a timestamp T_4 for the offloading to initiate. The IIoT device has a new set of timestamps T_1, T_2, T_3, and T_4 rounds of message exchanges. On the basis of the pairwise synchronization model in **Figure 3**, we represent the clock offset in the following equations:

$$T_2 = T_1 + d + \psi + \zeta \tag{16}$$

$$T_4 = T_3 + d - \psi + \beta \tag{17}$$

ψ is the clock offset of source IIoT device, d is the propagation delay assuming symmetric direction is employed, and ζ and β are random variables. The presence of clock skew causes the drifting of clock offset in the IIoT end device and the MEC server. The ATS improves the clock skew to secure offloading reliability and optimize energy consumption in synchronization processes. The clock offset (ψ) and propagation delay d can be estimated as follows:

$$\psi = \frac{[(T_2 - T_1) - (T_4 - T_3)]}{2} \tag{18}$$

$$d = \frac{[(T_2 - T_1) + (T_4 - T_3)]}{2} \tag{19}$$

We considered the errors as a Gaussian probability density function to reduce synchronization errors. The IIoT devices' clock offsets are changed regularly to ensure that the clock and timing synchronization is reliable. The timing synchronization ensures that the IIoT device offloading its MLS task maintains functional network connectivity.

ATS Clock Offset Estimation Using Maximum Likelihood Technique

To retain the MLS task offloading process described in **Figure 3**, we assume that the clock skew is absent by this period. The clock offset for maximum likelihood (ML) and CRLB across linked IIoT devices is calculated using the two-way timing data exchange paradigm. Because of the task offloading latency over the network, the random variables ζ and β are assumed to be autonomous and arbitrarily distributed, for the corresponding mean ν and variance ξ^2 as $C \sim (\mathfrak{G}, \xi^2)$. Hence, the likelihood function is realized in $\mathfrak{I} = T_2 - T_1 = d + \phi + \zeta$ and $\mathbb{H} = T_4 - T_3 = d - \phi + \beta$ observations. Mathematically, the ML is computed as follows:

$$\begin{aligned} L\left(\nu, \xi^2; x_1, \ldots, x_1\right) &= \sum_{j=1}^{n} f_x\left(x_1; \nu, \xi^2\right) \\ &= \sum_{j=1}^{n} \left(2\pi\xi^2\right)^{-1/2} \exp\left\{-\frac{1}{2}\frac{\left(x_j - \nu\right)^2}{\left(\xi^2\right)}\right\} \\ &= \left(2\pi\xi^2\right)^{-n/2} \exp\left\{-\frac{1}{2\xi^2}\sum_{j=1}^{n}\left(x_j - \nu\right)^2\right\} \end{aligned} \tag{20}$$

Applying **Eq. 17** to estimate ML expression

$$\begin{aligned} L\left(\phi, \nu, \xi^2\right) = \left(2\pi\xi^2\right)^N \exp\Bigg\{-\frac{1}{2\pi\xi^2}\Bigg[\sum_{i=1}^{N}\left(\mathfrak{I}_i - (d + \phi + \nu)^2\right. \\ \left. + \sum_{i=1}^{N}\left(\mathbb{H}_i - (d + \phi + \nu)^2\right.\Bigg]\Bigg\} \end{aligned} \tag{21}$$

By differentiating the log-likelihood role in **Eq. 18**, we arrive at

$$\hat{\phi} = \text{argmax}_\phi\left[InL(\phi)\right] = \frac{\sum_{i=1}^{N}\left[\mathfrak{I}_i - \mathbb{H}_i\right]}{2N} = \frac{\bar{\mathfrak{I}} - \bar{\mathbb{H}}}{2} \tag{22}$$

Therefore, ML of clock offset is estimated by

$$\frac{\partial^2 InL(\phi)}{\partial\phi^2} = -\frac{2N}{\xi^2}\mathbb{H}ar\left(\hat{\phi}\right) \geq -E\left[\frac{\partial^2 InL(\phi)}{\partial\phi^2}\right]^{-1} = \frac{\xi^2}{2N} \tag{23}$$

$$\hat{\phi} = \arg\max_{\phi}[InL(\phi)] = \frac{\sum_{i=1}^{N}[\mathfrak{I}_i - \mathbb{H}_i]}{2N} = \frac{\bar{\mathfrak{I}} - \bar{\mathbb{H}}}{2} \tag{24}$$

where N designates the number of observations; $\bar{\mathfrak{I}}$ and $\bar{\mathbb{H}}$ denote the mean sampling of observations $\mathfrak{I}$ and $\mathbb{H}$.

Cramer–Rao Lower Bound Technique

The deterministic parameter of the approximate estimate in variance is the foundation of CRLB. It is mostly useful in practice due to its easy implementation. Assume that Equation **(15)** fulfills a regularity requirement in CRLB that exists in a given estimation is 0 and that the CRLB result is obtained by differentiating Equation (15) as follows:

$$\frac{\partial^2 InL(\phi)}{\partial\phi^2} = -\frac{2N}{\xi^2} \tag{25}$$

Then, the CRLB for clock offset is given by

$$\mathbb{H}ar(\hat{\phi}) \geq -E\left[\frac{\partial^2 InL(\phi)}{\partial\phi^2}\right]^{-1} = \frac{\xi^2}{2N} \tag{26}$$

Industrial Internet of Things to Multi-Access Edge Computing Pair Selection and Scheduling Using Markov Transition

Assume that there is c number of MEC servers at the network edge, and the various IIoT devices preparing to offload their data are organized in a queue according to a Poisson process with rate $\lambda > 0$. The inter-arrival times of the various IIoT devices are then independent and identically exponentially distributed with λ. The service times are also independent and identically exponentially distributed with rate $v > 0$. We apply First-In First-Out principle and leave the queue $\mathbf{Q}$ capacity to infinity. This information indicates that the process $\mathbb{X} = \{\mathbb{X}_t, t \geq 0\}$, where $\mathbb{X}_t$ represent the number of IIoT end devices in the queue at time t, is a homogeneous Markov chain on state space $\mathbb{S} = \mathbb{N}$. The continuous-time Markov transition employs the birth-and-death process, where **birth** increase the number of IIoT end devices in the offloading queue (state variables) by one and **death** decrease the state by one. When birth transpires, the number of IIoT devices in the queue advances from state n to $n + 1$. When a death occurs, the queue goes from state n to state $n - 1$. The Markov chain state diagram is represented in **Figure 4**.

For any $n \geq 0$, we represent λ_n by the transition rate from state n to state $n + 1$ and for any $n \geq 1$, we denote by v_n the transition rate from state n to state $n - 1$. We assume that $v_n > 0$ for any $n \geq 1$ and that $v_n > 0$ for any $n \geq 1$. The IIoT end device MLS task offloading process generates $\mathbf{Q}$ of chain represented as follows:

$$\mathbf{Q} = \begin{pmatrix} -\lambda_0 & \lambda_0 & 0 & 0 & 0 & \ldots, \\ v_1 & -(\lambda_1 + v_1) & \lambda_1 & 0 & 0 & \ldots, \\ 0 & v_2 & -(\lambda_2 + v_2) & \lambda_2 & 0 & \ldots, \\ 0 & 0 & v_3 & -(\lambda_3 + v_3) & \lambda_3 & \ldots, \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix} \tag{27}$$

The discrete-time Markov chain embedded during the transition period of $\mathbb{X}$ consists of the parameters $\mathbf{p}_n$ and $\mathbf{q}_n$, for $n \geq 1$ is expressed mathematically as follows:

$$\mathbf{p}_n = \frac{\lambda_n}{\lambda_n + v_n} \quad \text{and} \quad \mathbf{q}_n = \frac{v_n}{\lambda_n + v_n} \tag{28}$$

However, we introduce a quantity η_n defined as follows:

$$\eta_0 = 1 \text{ and } \eta_n = \frac{v_1 \ldots, v_n}{\lambda_1 \ldots \lambda_n} \tag{29}$$

We then compute the sums A as follows:

$$A = \sum_{n=0}^{\infty} \eta_n \tag{30}$$

The transition process converges to equilibrium with the expected time, $w_n = \mathbb{E}\{\tau_{\mathbb{X}}(0) \mid \mathbb{X}_0 = n\}$, of the first passage to state 0, starting from state n. The first passage time $\tau_{\mathbb{X}}(0)$ to state 0 is defined as follows:

$$\tau_{\mathbb{X}}(0) = \inf\{t \geq T_1 \mid \mathbb{X}_t = 0\} \tag{31}$$

where T_1 is the first instant of jump of chain $\mathbb{X}$. The transient process with respect to $\mathbb{X}$ can be represented using the annotations

$$\mathbb{X} \text{ is transient} \Leftrightarrow A < \infty,$$

or

$$\mathbb{X} \text{ is recurrent} \Leftrightarrow A = \infty$$

To define the positive recurrence of chain $\mathbb{X}$, it is no longer adequate in the discrete state to examine the invariant probability, and chain $\mathbb{X}$ is non-explosive, which is expressed as follows:

$$v_n = 1 - f_{n,0}^{\mathbb{X}} = \begin{cases} \frac{1}{A}\sum_{j=0}^{n-1}\eta_j & \text{if } A < \infty \\ 0 & \text{if } A = \infty \end{cases} \tag{32}$$

where $f_{n,0}^{\mathbb{X}}$ is the probability, starting from n that the first return to state n occurs at $\mathbb{X}$. If $n \geq 0$, applying **Equation (30)** to η_n will produce

$$\eta_n = \begin{cases} \frac{v^n n!}{\lambda^n} & \text{if } n \leq c - 1 \\ \frac{v^i c^{n-c} c!}{\lambda^n} & \text{if } n \geq c. \end{cases} \tag{33}$$

However, the quantity A defined in **Equation (32)** is given by

$$A = \sum_{n=0}^{c-1} \frac{v^n n!}{\lambda^n} + \frac{c!}{c^c}\sum_{n=c}^{\infty}\left(\frac{cv}{\lambda}\right)^n \tag{34}$$

where $A < \infty \Leftrightarrow \lambda > cv$. The above relations indicate that chain $\mathbb{X}$ is non-explosive and uniform.

Estimating the Task Offloading Error

In this paper, we considered two types of errors that are likely to occur.

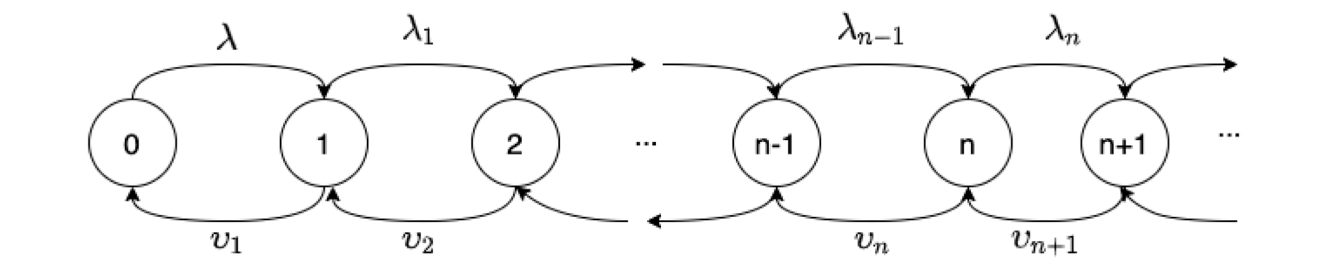

FIGURE 4 | Continuous-time birth-and-death process state diagram.

Maximum Location Error

The proposed algorithm requires the IIoT device to find the shortest route to the MEC server. Hence, the location of the IIoT devices and the MEC server in the network is important. Therefore, minimizing location error M_E is determined to measure the connected IIoT device's estimated location (a_2, b_2) and the actual location (a_1, b_1). The average location error A_{VE} is given by

$$A_{VE} = \sum_{k=1}^{N} \frac{\sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2}}{N} \tag{35}$$

We then compute the maximum location error M_E achieved by applying the localization approach, which is calculated as follows:

$$M_E = \max\sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2} \tag{36}$$

Offloading Failure Probability

During the MLS task offloading processes, wireless and network channel fluctuation causes fading and shadow effects. The effects cause an offloading failure, which is a problem. Assuming that η is the block error rate, νm denotes transport block size for offloading the MLS task to the MEC server, and then, the error probability of offloading is given by

$$P_m^{err}(\gamma) = 1 - (1 - \eta)^{\frac{\gamma\tau\kappa}{\nu_m}} \tag{37}$$

The offloading failure probability is given by

$$P^f(\gamma) = 1 - \prod_{m=1}^{N}\left[1 - P_m^{err}(\gamma)\right] \tag{38}$$

Combining the two errors will give the total error rate T^{err} that is likely to occur in the offloading process.

$$T^{err} = \frac{1}{P^f(\gamma) + M_E} \tag{39}$$

PROPOSED ASTO ALGORITHM

This section analyzes the proposed algorithms to enhance the MLS task offloading among the connected IIoT devices and the MEC server. The ASTO algorithm ensures that the IIoT device under attack can offload their MLS tasks to the selected MEC server and receive the attack analysis results in time. The whole detection process must complete in real time to prevent the IIoT device from experiencing downtime. The IIoT must first establish the shortest route to the proximal MEC server in the network hierarchy (Han et al., 2018). Assuming that there are different MEC servers available at the edge of the network such as $\{\mathcal{M}_1, \mathcal{M}_2 \ldots \mathcal{M}_k\}$ and that the source node connects to different routes to the MEC server, then the IIoT selects the optimal path that maximizes latency and the synchronization errors. This approach aims to reduce the time required by the IIoT end device to thoroughly perform attack detection and minimize energy consumption while preserving synchronization accuracy. The proposed ASTO algorithm is divided into two: the machine learning–based security system and the adaptive task offload.

Machine Learning–Based Security System

In this subsection, we demonstrate how the online machine learning method was used to design network attack detection system deployed to the MEC server. All the network traffic captured (MLS) by the IIoT devices are offloaded to MEC server using the proposed ASTO algorithm. Several machine learning methods exist in literature, but, to prove our concept, a stochastic gradient descent (SGD) was adopted to design an online network attack detection system. We adopted SGD due to the dynamic (frequent change in environmental data and network traffic) nature of the data offloaded from the IIoT devices. Moreover, cyberattackers keep changing their approaches. It is difficult to rely on classical machine learning methods to design a network attack detection system. Hence, using online SGD allows the model to learn new attacks and system operational data to prevent the model from becoming obsolete. Unlike batch gradient descent, which calculates the gradient using the whole dataset, SGD, also known as incremental gradient descent, iterates over a single randomly selected training sample to identify minimums or maximums. To achieve accurate results with SGD, the data sample should be in a stochastic order by shuffling the training set for every epoch. First, we define a cost function for determining the weights of $i - th$ observation in the training dataset for an adaptive linear neuron as follows:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \varpi\left(\mathbf{w}^T\mathbf{x}\right)^{(i)}\right)^2 \tag{40}$$

where $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_i] \in \mathbb{R}$ is least squares with features from the training data, $y = [y_1, y_2, \ldots, y_i] \in \mathbb{R}$ denote the observations, $\mathbf{w} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_i]$ are the weight vectors, and ϖ represent the an activation function. If a defined negative gradient weight change $\Delta\mathbf{w}$ is multiply by a learning rate $\mathbf{r}$, then the cost function can be expressed as follows:

$$\Delta\mathbf{w} = -\mathbf{r}\nabla J = \mathbf{r}\sum_{i=1}^{N}\left(y^{(i)} - \varpi\left(\mathbf{w}^T\mathbf{x}\right)^{(i)}\right)\mathbf{x}^{(i)} \tag{41}$$

In batch gradient descent, the gradient is computed from the whole training set to minimize a cost function. If we have a large dataset with millions of data points, then performing batch gradient descent may be extremely expensive because we must assess the whole training dataset once we go one step closer to the global minimum. Rather than updating the weights based on the total cumulative errors across all samples $\mathbf{x}^{(i)}$ through the $\Delta\mathbf{w}$ described above, we may utilize the following update in the SGD method:

$$\Delta\mathbf{w} = -\mathbf{r}\nabla J = \mathbf{r}\left(y^{(i)} - \varpi\left(\mathbf{w}^T\mathbf{x}\right)^{(i)}\right)\mathbf{x}^{(i)} \tag{42}$$

However, we update the weights incrementally when new data arrive from IIoT devices.

Adaptive Security Task Offloading

Assuming the T_{sync} is the average synchronization time (in seconds) of the MEC server connected with S_i IIoT devices, $\mathbb{X}_t$ denotes the various of IIoT in the queue, and T_{resp} is the average response time of the MEC server.

Algorithm 2. Proposed Adaptive Time Synchronization

```
1  INITIALIZE $\{S, T_{sync}, \mathbb{X}_t\}$
2  SET $\{\mathcal{M}_1, \mathcal{M}_2 \ldots \mathcal{M}_n\}$
3  SET $i \leftarrow 1$
4  while $i <= n$ do
5      Calculate: $T_{resp(i)} = \frac{\mu_i}{\mathbb{X}_t} - T_{sync}$
6      $\mathcal{M}_i \leftarrow T_{resp(i)}$
7      Increment $i$ $(i++)$
8  Select $sc \leftarrow min[n-1, n, \ldots n+1]$
9  Find $\delta[v]$ From $\mu_i$ to the $\mathcal{M}_{sc}$ (Algorithm 1)
10 Apply equation (22)
11 Synchronize offloading time $T_{off}$ with Source $T_{sn}$
12 if $T_{local}$ != $T_{sn}$ then
13     if $T_{sn}$ != $T_{off}$ then
14         Update $T_{sn} \leftarrow T_{off}$
15     Retry Offloading
16 else
17     Apply equation (8) and (10)
18     Perform Attack detection with SGD
```

The shortest route between the selected MEC server and the source node is determined based on Algorithm 2. The MEC server creates an offloading task schedule and communicates the result to the IIoT devices. If any of the MEC servers experience changes or delays, then the new offloading time is synchronized with the IIoT devices to minimized latency, energy consumption, and offloading errors. The task offloading process initializes after the aforementioned processes. Finally, the selected MEC server applies the online SGD to perform attack detection and return the security responses to the corresponding IIoT devices in the network to implement the right security policies on the network, where T_{off} is the scheduled offloading time by the MEC server, T_{sn} is the synchronized time on the IIoT end device, and T_{local} is the local time of the IIoT end device.

PERFORMANCE EVALUATION AND RESULTS ANALYSIS

In this section, we present the experimental implementation of our proposed system in a controlled environment using MATLAB Simulink library. Our design mimics a real-world IIoT scenario.

Testbed Preparation

The setup consists of three IIoT devices connected in a mesh topology to two remote radio head made up of wireless routers. The IIoT devices connect to the router *via* its wireless links. The MEC layer consists of three MEC servers connected in a mesh topology *via* a border router. In each of the experiments, the network traffics *w* on the various links are varied. **Figure 5** shows a sample of the SRA algorithm finding the shortest route from the IIoT device to the MEC server based on *w* on the network link. Node 1, Node 2, and Node 3 represent our IIoT end devices, Node 4 and Node 5 form the RHH that connects the three IIoT devices in a mesh topology, Node 6 and Node 7 are the border routers that connect the MEC servers (Node 8, Node 9, and Node 10) at the edge of the network. The red path indicates the selected optimal path with minimum network congestion from a source node 1 (IIoT end device) to a MEC server (Node 10). The values assigned to the links between the connected nodes show the network traffic *w*.

The MEC servers contain the trained MLS model for network attack analysis, the synchronization algorithm, and the Markov chain queuing algorithm. Because the IIoT devices are resource constraints, we reduced the computational stress on these nodes. We deployed a simple model consisting of a network packet capturing, a table for the queuing, synchronization time parameter, and task offloading system. The IIoT devices sampled different sizes of the test set and submit it to the MEC servers. In each sample, MLS model performs network attack analysis and send responds to the IIoT device that offloaded the task.

Dataset Used

There are no approved public datasets for network attack detection created in the context of IIoT devices to MEC networks during our studies, as far as we know. However, we created and tested the MLS model's performance using a publicly available DoS dataset known as the CICDDoS2019 intrusion datasets created by the Canadian Institute for Cybersecurity (Sharafaldin et al., 2019). CICDDoS2019 is a collection of benign and up-to-date common DDoS

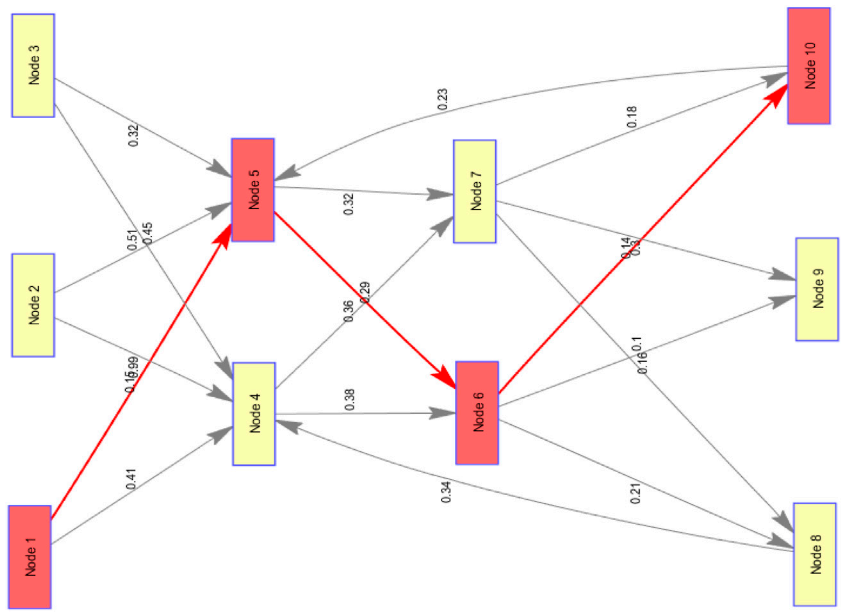

FIGURE 5 | Sra algorithm response and network connection.

TABLE 1 | SGD MLS model on the MEC server.

Performance of the MLS model. Accuracy: 0.99			
	Precision	Recall	F1-Score
Attacks detection rate	0.98	0.99	0.99
Normal detection rate	0.99	0.99	0.99
Macro avg	0.99	0.99	0.99
Weighted avg	0.99	0.99	0.99

TABLE 2 | MLS model response for 10 samples offloaded to the MEC server.

Performance of 10 samples					
Experiment	**Data records**	**Precision**	**Recall**	**F1-score**	**Accuracy**
EXP 1	1,000	0.89	0.87	0.87	0.87
EXP 2	2,491	0.81	0.75	0.73	0.75
EXP 3	3,985	0.99	0.99	0.99	0.99
EXP 4	4,981	0.99	0.99	0.99	0.99
EXP 5	7,471	0.99	0.99	0.99	0.99
EXP 6	9,962	0.99	0.99	0.99	0.99
EXP 7	12,452	0.99	0.99	0.99	0.99
EXP 8	14,942	0.99	0.99	0.99	0.99
EXP 9	17,433	0.99	0.99	0.99	0.99
EXP 10	19,923	0.99	0.99	0.99	0.99

attacks that closely mimics real-world data. It also provides the results of a network traffic analysis with labeled flows based on the time stamp, source and destination IPs, source and destination ports, protocols, and attack using CICFlowMeter-V3. PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, SYN, NTP, DNS, and SNMP are among the contemporary reflecting DDoS network attacks included in the dataset. The CICDDoS2019 dataset suites our experiment due to its modern attacks, structure, and network architecture used during its creation.

Numerical Results

In the experiment, we repeated our algorithms multiple times in a loop (30 iterations) and calculated an average value from the results obtained. **Table 1** shows the response from the MLS model, which was trained and deployed to the MEC servers.

The experiment was repeated for 10 different samples of the test dataset, and **Table 2** shows the results obtained.

Figure 6 compares the prediction of different the machine learning models deployed to the MEC server. **Figure 6A** provides the total time (turnaround time) required to complete for each IIoT device to offload the sampled test data to the MEC server for network attack analysis. **Figure 6B** shows the average time taken for the MEC server to perform attack analysis based on the size of test sampled dataset received.

In **Figure 7**, we increased the number of connected IIoT devices and measured the latency and the energy consumed when our proposed system was applied. **Figure 7C** and **Figure 7D** show respectfully the latency and the energy consumed (in kilojoules).

RESULTS DISCUSSION

From **Figure 6**, when the load of the sampled test set is increased, the time required by the IIoT to perform a network attack also increases. The average accuracy of the SGD model is 99.9%,

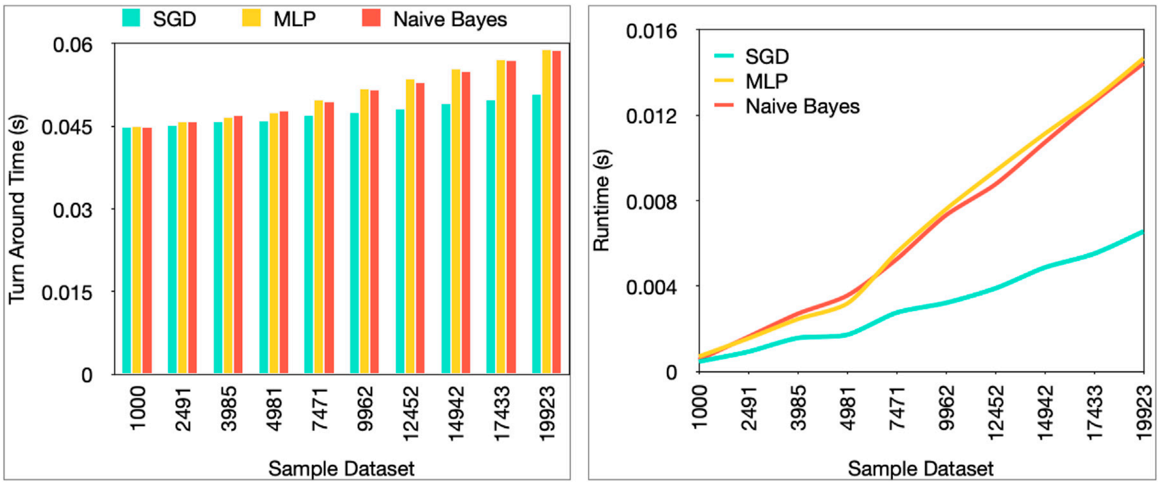

FIGURE 6 | Comparing different machine learning models.

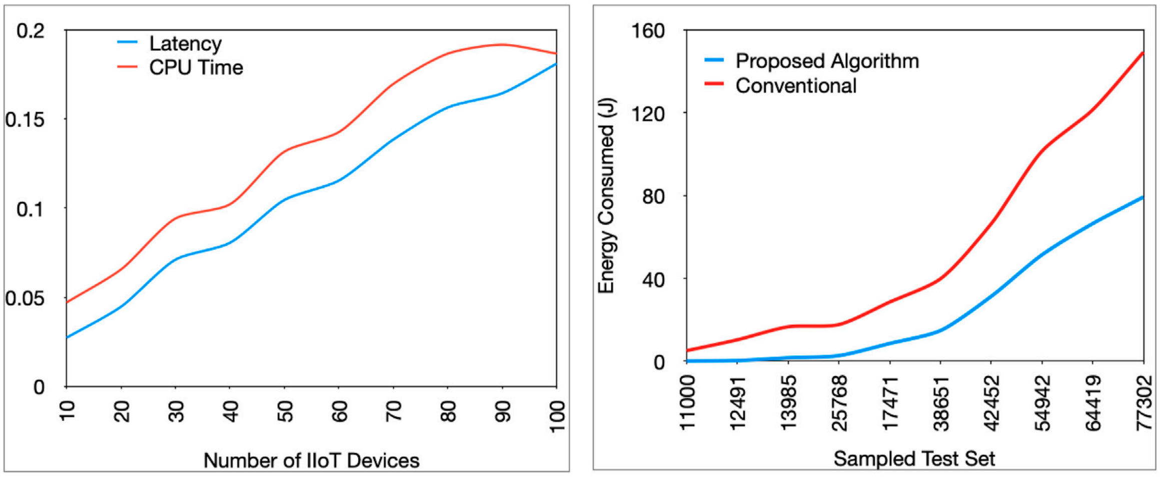

FIGURE 7 | Latency and energy consumed.

multilayer perceptron also produced 99.62%, and the Naive Bayes also produced the lowest with 57.78%. However, in all cases, the SGD outperformed the other machine learning models in execution time and performance (accuracy). It is evident that, as the number of connected IIoT devices requesting to offload their MLS task increases, the energy required to offload their task also increases. The conventional method also performed better than the conventional method. **Figure 7C** shows that, when the number of IIoT devices increases, the MEC server requires more time to synchronize and offload the MLS task. Energy consumed increases with the increasing number of IIoT devices. Hence, increasing the number of IIoT devices will require more energy. The latency also increases rapidly, which indicate that special bandwidth and network channel must be allocated for such a security system. We believe that our proposed method improves the task offloading, but applying other techniques such as machine learning, specifically reinforcement learning methods, will help optimize the security system. Moreover, attacks such as network flooding targeting the IIoT device could hinder the offloading of the MLS task to the IIoT devices before the MEC server performs the deep intrusion detection. Such cases may require exceptional circumstances to be created in the proposed model to resolve them.

CONCLUSION

This paper proposed a novel adaptive time synchronization MLS to provide security for the IIoT utilizing MEC. By employing the ATSO, all connected IIoT end devices can synchronize with the MEC server to enhance performance requirements in latency, MLS task offloading, and energy consumption. We proposed node pair selection algorithms for IIoT devices to synchronize with the proximal MEC servers. ATS significantly increased the task offloading accuracy and reduced the energy consumed by the various IIoT devices. The Markov chain contributed to the queuing process of the MLS task offloading by the various IIoT devices. However, the adopted SGD online learning model also outperformed the other machine learning models. The proposed ATSO system is scalable with the IIoT to MEC mesh topology from the experimental results. Future works will explore different online machine learning methods to improve intrusion detection on the MEC server and deep reinforcement learning to create routing algorithms to optimize the task offloading process.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found at https://www.unb.ca/cic/datasets/ddos-2019.html.

AUTHOR CONTRIBUTIONS

EG prepared the paper while AJ also reviewed and provided extensive ideas.

FUNDING

The paper publication will be paid for by the School of Computer Science, University College Dublin.

REFERENCES

Akherfi, K., Gerndt, M., and Harroud, H. (2018). Mobile Cloud Computing for Computation Offloading: Issues and Challenges. *Appl. Comput. Inform.* 14, 1–16. doi:10.1016/j.aci.2016.11.002

Ansere, J. A., Han, G., and Wang, H. (2019). A Novel Reliable Adaptive beacon Time Synchronization Algorithm for Large-Scale Vehicular Ad Hoc Networks. *IEEE Trans. Veh. Technol.* 68, 11565–11576. doi:10.1109/tvt.2019.2946225

Bakhshi, Z., Balador, A., and Mustafa, J. (2018). "Industrial Iot Security Threats and Concerns by Considering cisco and Microsoft Iot Reference Models," in Proceedings of the 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Barcelona, SpainApril 2018 (Piscataway, New Jersey, United States: IEEE), 173–178. doi:10.1109/wcncw.2018.8368997

Bing, L., Yunyong, Z., and Lei, X. (2019). An Mec and Nfv Integrated Network Architecture. *ZTE Commun.* 15, 19–25. doi:10.3969/j.issn.1673-5188.2017.02.003

Chen, X., Shi, Q., Yang, L., and Xu, J. (2018). Thriftyedge: Resource-Efficient Edge Computing for Intelligent Iot Applications. *IEEE Netw.* 32, 61–65. doi:10.1109/mnet.2018.1700145

Chen, Y., Zhang, N., Zhang, Y., Chen, X., Wu, W., and Shen, X. (2021). Energy Efficient Dynamic Offloading in mobile Edge Computing for Internet of Things. *IEEE Trans. Cloud Comput.* 9, 1050–1060. doi:10.1109/TCC.2019.2898657

Doan, T. V., Nguyen, G. T., Salah, H., Pandi, S., Jarschel, M., Pries, R., et al. (2019). "Containers vs Virtual Machines: Choosing the Right Virtualization Technology for mobile Edge Cloud," in Proceedings of the 2019 IEEE 2nd 5G World Forum (5GWF), Dresden, Germany, 30 September-2 October 2019 (Piscataway, New Jersey, United States: IEEE), 46–52. doi:10.1109/5gwf.2019.8911715

Finnerty, K., Motha, H., Shah, J., White, Y., Button, M., and Wang, V. (2018). *Cyber Security Breaches Survey 2018: Statistical Release*. Portsmouth, England: University of Portsmouth.

Han, G., Yang, X., Liu, L., Chan, S., and Zhang, W. (2018). A Coverage-Aware Hierarchical Charging Algorithm in Wireless Rechargeable Sensor Networks. *IEEE Netw.* 33, 201–207. doi:10.1109/MNET.2018.1800197

Hsu, C.-W., Hsu, Y.-L., and Wei, H.-Y. (2019). "Energy-efficient and Reliable Mec Offloading for Heterogeneous Industrial Iot Networks," in Proceedings of the 2019 European Conference on Networks and Communications (EuCNC), Valencia, Spain, June 2019 (Piscataway, New Jersey, United States: IEEE), 384–388. doi:10.1109/eucnc.2019.8802020

Irdeto-Media-Team (2019). New 2019 Global Survey: Iot-Focused Cyberattacks Are the New normal - Irdeto. Available at: https://irdeto.com/news/new-2019-global-survey-iot-focused-cyberattacks-are-the-new-normal/(Accessed on 03 06, 2020).

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A Survey on mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutorials* 19, 2322–2358. doi:10.1109/comst.2017.2745201

Nasrallah, Y. Y., Al-Anbagi, I., and Mouftah, H. T. (2016). "Distributed Time Synchronization Mechanism for Large-Scale Vehicular Networks," in Proceedings of the 2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT), Cairo, EgyptApril 2016 (Piscataway, New Jersey, United States: IEEE), 1–6. doi:10.1109/mownet.2016.7496600

Sadeghi, A.-R., Wachsmann, C., and Waidner, M. (2015). "Security and Privacy Challenges in Industrial Internet of Things," in Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, June 2015 (Piscataway, New Jersey, United States: IEEE), 1–6. doi:10.1145/2744769.2747942

Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. (2019). "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," in Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, October 2018 (Piscataway, New Jersey, United States: IEEE), 1–8. doi:10.1109/ccst.2019.8888419

Sun, W., Liu, J., and Yue, Y. (2019). Ai-enhanced Offloading in Edge Computing: When Machine Learning Meets Industrial Iot. *IEEE Netw.* 33, 68–74. doi:10.1109/mnet.001.1800510

Sun, X., and Ansari, N. (2017). Latency Aware Workload Offloading in the Cloudlet Network. *IEEE Commun. Lett.* 21, 1481–1484. doi:10.1109/lcomm.2017.2690678

Tanaka, H., Yoshida, M., Mori, K., and Takahashi, N. (2018). Multi-access Edge Computing: A Survey. *J. Inf. Process.* 26, 87–97. doi:10.2197/ipsjjip.26.87

Tange, K., De Donno, M., Fafoutis, X., and Dragoni, N. (2020). A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities. *IEEE Commun. Surv. Tutorials* 22, 2489–2520. doi:10.1109/comst.2020.3011208

Wu, H., Wolter, K., Jiao, P., Deng, Y., Zhao, Y., and Xu, M. (2021). Eedto: An Energy-Efficient Dynamic Task Offloading Algorithm for Blockchain-Enabled Iot-Edge-Cloud Orchestrated Computing. *IEEE Internet Things J.* 8, 2163–2176. doi:10.1109/JIOT.2020.3033521

Wurm, J., Hoang, K., Arias, O., Sadeghi, A.-R., and Jin, Y. (2016). "Security Analysis on Consumer and Industrial Iot Devices," in Proceedings of the 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macao, China, January 2016 (Piscataway, New Jersey, United States: IEEE), 519–524. doi:10.1109/aspdac.2016.7428064

Xu, X., Zhang, X., Gao, H., Xue, Y., Qi, L., and Dou, W. (2019). Become: Blockchain-Enabled Computation Offloading for Iot in mobile Edge Computing. *IEEE Trans. Ind. Inform.* 16, 4187–4195.

Yang, B., Cao, X., Bassey, J., Li, X., and Qian, L. (2021). Computation Offloading in Multi-Access Edge Computing: A Multi-Task Learning Approach. *IEEE Trans. Mobile Comput.* 20, 2745–2762. doi:10.1109/TMC.2020.2990630

Yang, B., Fagbohungbe, O., Cao, X., Yuen, C., Qian, L., Niyato, D., et al. (2022). A Joint Energy and Latency Framework for Transfer Learning over 5g Industrial Edge Networks. *IEEE Trans. Ind. Inf.* 18, 531–541. doi:10.1109/TII.2021.3075444

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.