An Edge Computing Framework for Real-Time Monitoring in Smart Grid

Yutao Huang[1], Yuhe Lu[1], Feng Wang[2], Xiaoyi Fan[3,1], Jiangchuan Liu[1], Victor C. M. Leung[3]
[1]School of Computing Science, Simon Fraser University, Canada
[2]Department of Computer and Information Science, The University of Mississippi, USA
[3]Electrical and Computer Engineering, The University of British Columbia, Canada

***Abstract*—Due to the ever-growing demands in modern cities, unreliable and inefficient power transportation becomes one critical issue in nowadays power grid. This makes power grid monitoring one of the key modules in power grid system and play an important role in preventing severe safety accidents. However, the traditional manual inspection cannot efficiently achieve this goal due to its low efficiency and high cost. Smart grid as a new generation of the power grid, sheds new light to construct an intelligent, reliable and efficient power grid with advanced information technology. In smart grid, automated monitoring can be realized by applying advanced deep learning algorithms on powerful cloud computing platform together with such IoT (Internet of Things) devices as smart cameras. The performance of cloud monitoring, however, can still be unsatisfactory since a large amount of data transmission over the Internet will lead to high delay and low frame rate. In this paper, we note that the edge computing paradigm can well complement the cloud and significantly reduce the delay to improve the overall performance. To this end, we propose an edge computing framework for real-time monitoring, which moves the computation away from the centralized cloud to the near-device edge servers. To maximize the benefits, we formulate a scheduling problem to further optimize the framework and propose an efficient heuristic algorithm based on the simulated annealing strategy. Both real-world experiments and simulation results show that our framework can increase the monitoring frame rate up to 10 times and reduce the detection delay up to 85% comparing to the cloud monitoring solution.**

***Index Terms*—Edge Computing; Smart Grid; Deep Learning.**

I. Introduction

The electric power grid has become an important infrastructure in daily life, delivering electricity from large power stations to customers. Though it has existed for a long period of time, the current power grid is almost built and operated following the theories and technologies 100 years ago [1], , which have been shown less capable to fulfill the requirements in modern society. For example, one of the important components in power grid system, the regular monitoring on high voltage power line, still adopts the traditional manual monitoring scheme, causing a number of problems such as low efficiency, hard to be checked during off-work period and unable to provide real-time monitoring. As a result, the electricity system frequently suffers from blackouts and grid failures in the past decades, leading to security risks and inconvenience issues, as well as huge economic losses for both power providers and consumers.

In recent years, the concept of smart grid has been proposed to resolve the issues related to a variety of operational and energy problems in the electrical power grid. By combining the advanced technologies like Internet of Things (IoT) [1] and Artificial Intelligence (AI) [2], the smart grid can create an automated energy delivery network which is more secure, reliable and intelligent than the current grid. In smart grid, the regular monitoring is expected to be performed with all kinds of IoT devices and processing servers instead of manpower, where IoT devices such as sensors and cameras will collect and upload the real-time videos and other information about the power line to the processing servers. Such information will then be automatically processed by the processing servers running deep learning algorithms to detect potential threats and, if necessary, trigger appropriate actuations to achieve timely and intelligent monitoring with automatic threat identification.

Since deep learning algorithms are extremely data intensive, computation intensive and hardware-dependent, the processing servers of smart grid are expected to be equipped with abundant computation resources. This makes cloud computing be widely proposed as a natural choice to host such servers [3]. However, transferring large volume of data into the cloud will push significant pressure to the network and generate huge communication costs. In addition, from power providers perspective, moving data to the remote cloud may also incur privacy concerns. Moreover, the latency in the network can become a severe performance bottleneck due to the latency sensitivity of real-time monitoring.

Recently, the concept of edge computing has been proposed as a complement of cloud computing, attracting great interests from both academia and industry. In contrast to cloud, edge usually refers to a geographical concept which is in close proximity to the end devices in the network. By pushing applications, data and services away from centralized cloud to edge servers, the computing paradigm will be extended to an edge-cloud collaborative computing, which has shown outstanding performance on communication latency and traffic reduction [4], and ease the privacy concerns of users as well.

In this paper, we for the first time introduce edge computing to the smart grid scenario and propose a five-layer edge computing framework to achieve high performance real-time monitoring for smart grid, where by moving deep learning algorithms to edge servers, the monitoring performance can be greatly improved in terms of detection latency, frame rate, etc. To maximize the potential benefits, we further formulate a scheduling problem based on the proposed framework to better coordinate the affiliations between the smart grid monitoring devices and edge servers. We also propose a heuristic algorithm using simulated annealing strategy to solve the problem efficiently. Both our real-world experiments and extensive simulations have demonstrated that compared to cloud based smart grid monitoring, our solution can achieve about 10 times increase in frame rate and up to 85% reduction in detection latency. To our best knowledge, this is the first effort to propose a holistic solution to apply edge computing framework into smart grid to achieve real-time monitoring with automatic threat identification.

The remainder of the paper is organized as follows. In section II, we discuss the background and motivation for our work. In section III, we propose our edge computing framework for real-time monitoring and further compare it with cloud monitoring and traditional monitoring schemes. We formulate a scheduling problem to further optimize the proposed edge computing framework and present a heuristic algorithm in section IV. The performance evaluations by real-world experiments and extensive simulations are presented in section V. Section VI will summarize the related work and we conclude of our work in section VII.

II. Background and Motivation

In this section, we briefly discuss the background of power grid and describes cloud computing and edge computing which motivates our work."

A. From Traditional Power Grid to Smart Grid

The job of traditional power grid is generally to deliver the electricity power from a few number of power generator to a large number of customers. Yet given the ever-growing demands in modern cities, current grid becomes less capable to fulfill the requirements on energy efficiency and reliability. The smart grid, by adding a information flow along with the traditional electricity flow, allows a two-way flow to construct an automated and distributed energy delivery network.

The smart grid takes control over every single event occurred anywhere in the power grid by utilizing the modern information technologies. The working scope of a smart grid generally covers aspects including power generation, transmission, distribution and consumption, and each aspect may have close relationships with other aspects in the grid. The self-monitoring system in the smart grid will report the current grid status to the central server. Once it finds a failure event such as voltage transformer failure, the smart grid will automatically changes the power flow and calls the self-recover service. Also, it may influence the current local power price from consumers perspective. All these aspects are automatically executed and controlled by the strategies set in the smart grid.

From systems view, the smart grid mainly includes three systems: infrastructure system, management system and protection system. The infrastructure system mainly focuses on energy, information and communication infrastructure in the smart grid environment. Its jobs contains energy generation, delivery and consumption, metering and monitoring, communications between components of the smart grid, etc. The management system is a centralized system which dispatches and coordinates each component of the smart grid to collaboratively work for a specific job. According to different management objective, the management system will dispatch different strategies to utilize the resource in an efficient and economic way. The protection system in a smart grid works automatically to maintain the whole smart grid systems reliability and privacy, not only addressing the issue caused by user mistakes, equipment failures and natural disasters, but also being able to make responses to deliberate man-made attacks such as spies and terrorists.

In recent years, deep learning techniques rise in machine learning fields with the convolutional neural networks (CNN), as one of the typical deep and feed-forward neural networks, gaining huge success in many areas such as speech recognition, image recognition and natural language processing. It has also attracted wide attentions from industry, such as Google, Microsoft and Nvidia and they all develop applications with deep learning techniques such as Google Translate and Microsoft Xiaoice. Deep learning also can be widely applied under smart grid environment. By introducing well-trained model, the important module in a smart grid such as threat detection, malicious attack identification and intelligent electricity power control [5].

B. From Cloud Computing to Edge Computing

Deep learning tasks are computation-intensive tasks and usually involve great amount of training data, which usually put strict requirements on the hardware. One recent trend is using GPU to replace CPU as the deep learning computation hardware. With thousands of computational cores, GPUs can greatly outperform CPUs in matrix calculations which are massively involved in deep learning algorithms. Since powerful GPU resource is expensive, cloud providers such as Amazon have developed GPU cloud platform to provide users large GPU computation resources which are flexible for allocation. However, the latency to cloud is often high, making it not suitable for latency-sensitive applications.

Recently, edge computing has been proposed as a complement of cloud computing, where edge is defined as the network topology at close proximity to end devices which can be accessed by Radio Access Networks (RAN), WLAN, ethernet and other network connections. Comparing to cloud servers, edge servers usually have relatively smaller scale of moderate powerful computation resources, which can play a critical role in aggregating, preprocessing while the distributed data are forwarded to central servers in data centers. By moving part of the computation and storage resources from the cloud to the network edge, it has been shown that great network performance gain will be obtained in edge computing applications such as edge computation offloading [6] [7] [8] [9], edge caching [10] [11] and edge resource allocation [12]. Recent research on moving deep learning applications to edge has also demonstrated to successfully reduce the communica-

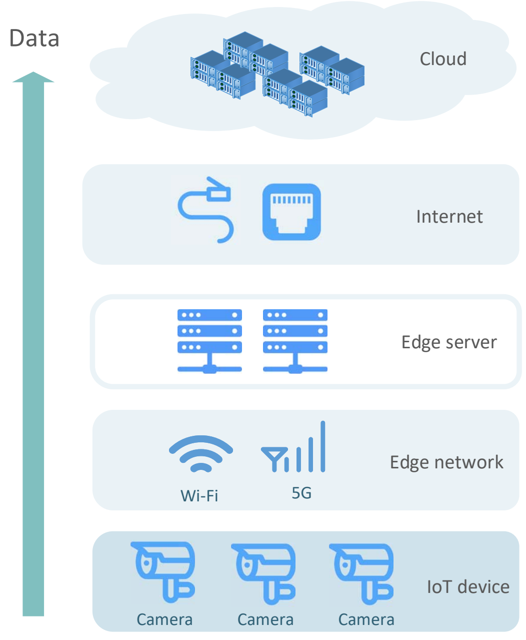


Fig. 1: An edge computing framework for real-time monitoring in smart grid

tion cost [13], inference latency and energy consumption [14]. This motivates us to investigate the opportunity to introduce edge computing to the smart grid scenario and propose a edge computing framework to achieve high performance real-time monitoring for smart grid.

III. Introducing Edge to Real-time Monitoring for Smart Grid

In this section, we present our framework that introduces edge computing to smart grid real-time monitoring. We will first describe the details of the framework and then make a comparison with cloud and traditional monitoring schemes.

A. Real-time Edge Monitoring Framework

The power line monitoring is one of the important work in power grids. The common solution in traditional grid is to delegate professional staff to inspect whether the voltage power lines are damaged or not, while in smart grid, all monitoring tasks are replaced by IoT devices attached to the grid. The centralized cloud will then gather such information as pictures or videos captured by IoT devices through the Internet and process with deep learning algorithms to detect the potential threats.

We propose a novel framework which introduces edge computing into smart grid, as illustrated in Fig. 1. Our framework contains five main layers: IoT device layer, edge network layer, edge server layer, Internet layer and cloud server layer. The monitoring procedure will go through these five layers from the bottom to the top.

IoT device layer: The IoT device layer consists of such IoT devices as smart cameras that are used to monitor the smart grid. These devices are not only responsible for capturing the high resolution pictures or video streams, but also installed with a communication module and local storage to assist the transmission of the captured information to the edge server.

Edge network layer: This layer connects IoT devices to edge servers. For connectivities, since each edge server may take charge of multiple IoT devices that may be attached to the grid arbitrarily, wireless connections are often preferred than wired connections due to its convenience. Also, the transmissions of images and videos require high data rate for the connection. This makes Wi-Fi and 5G networks are better choices over ZigBee and Bluetooth.

Edge server layer: One key module in this layer is the threat identification module, which executes the inference algorithm based on some trained deep learning models. Once it finds a threat in the received picture or videos, such as people, a car or a bird getting close to the power line, the module will upload the monitoring data with warning information to the cloud. Based on the edge network connectivity type, edge servers can be deployed at Wi-Fi access points or 5G base stations or both. These servers should be equipped with certain high performance GPU resources to support the execution of the threat identification module.

Internet layer: This layer connects edge servers to the cloud server. Different from edge network layer, wired connections are often used here as backbones in the public network domain.

Cloud layer: This layer is the central controller of the monitoring system. It will receive warning information from all the edge servers and according to the smart grid's strategies to react to those warnings. For instance, if the cloud consistently receives warnings from one specific area, then the smart grid will recognize that there are potential threats in the area and further checking and restoring process will be executed.

B. Comparison Between Edge, Cloud and Traditional Monitoring Approaches

Traditional monitoring approaches adopt manual inspection, which involves enormous human labors and cannot satisfy the real-time requirements , not to mention the introduced high employment cost. Even in developing countries, the monetary cost for one climb on the electric tower can take USD $100. Comparing to new monitoring schemes using advanced information technologies, the traditional approaches should not be adopted under the smart grid environment.

Cloud monitoring system, in contrast, is intelligent, automated and able to achieve real-time monitoring. Also, the management cost is significantly reduced comparing to traditional monitoring. However, one critical issue is about the network performance. The traffic created in the Internet is enormous because all the picture and video information captured by every single IoT device is required to uploaded to the cloud through

	Edge monitoring	Cloud monitoring	Traditional monitoring
Manual or automated	Automated	Automated	Manual
Monitoring frequency	Real-time	Real-time	Long-term periodic inspection
Threat detection	Deep learning based approaches	Deep learning based approaches	Manual inspection
Server hardware	Moderate storage and computation resources with one or several GPUs	Large storage and computation resources with powerful GPU clusters	No server required
Server location	Co-locate with wireless access points, e.g., Wi-Fi routers and 5G base stations	Set at specific places where the cloud service providers choose, with the size of several stadiums	No server required
Network latency	Less than tens of milliseconds	Usually between 50 to 500 ms	No network
Network traffic	Small since only warning information will be transmitted through the Internet and this occurs in low frequency.	Large because of all the picture and video information will be transmitted through the Internet.	No network
Data privacy	More secure. Most of the data stored at IoT devices and edge servers without go through the Internet.	Captured pictures and videos may leak on the Internet.	No data leak concern
Reliability	High reliability with distributed located edge servers. Single edge server down won't affect the whole monitoring system.	Low reliability. Centralized control module will cause termination of the whole system once there are severe issues with the cloud.	Low reliability since its inspection cycle is extremely long comparing with automated real-time monitoring.
Price	Moderate. Most cost is the one-time charge for buying and setting those IoT devices and edge servers.	High. Cloud monitoring will require huge network bandwidth to serve for large-scale data transmission, which will be charged a lot from Internet Service Provider (ISP).	Extremely high. Each time of the maintenance requires high-quality staff and the labor cost is relatively high.

TABLE I: The comparison between edge, cloud and traditional monitoring approaches

the Internet. This results in large network bandwidth resource is occupied and network is likely to be congested. Also, the high network latency will increase the threat detection latency and degrade the detection performance.

Edge monitoring system, on the other hand, can well complement the cloud system. Since edge servers are distributedly located near Wi-Fi routers and/or 5G base stations, each edge server only needs to take charge of limited number of IoT devices and will less likely be overloaded. The network traffic generated over the Internet is also greatly reduced since the data will only be transmitted when the edge server finds a potential threat and this usually happens with only a small probability. Also, the edge network latency is largely smaller than the Internet latency. This makes the real-time threat detection become more feasible. Besides these advantages, edge monitoring excels cloud monitoring from the perspectives of data privacy and system reliability. We summarize the comparison between edge, cloud and traditional monitoring approaches in TABLE I.

IV. Edge Scheduling Optimization

As edge servers are distributedly located, an IoT device may be eligible to connect to multiple edge servers and each feasible connection will possess different network bandwidth. Thus it is important to appropriately arrange the connections to obtain the optimal performance. In this section, we will formulate this as a scheduling problem and further propose an algorithm to optimize it.

A. Problem Formulation

In the edge monitoring system, there can be hundreds of IoT devices and a few number of edge servers. Though these edge servers and IoT devices are located in fixed place, the connections between IoT devices and edge servers are still flexible and they are influencing the monitoring performance of the overall system. One important performance metric is the effective frame rate. The effective frame rate is defined as the picture frames the whole system can capture and process in the threat detection module within one second. The higher effective frame rate the system owns, the better monitoring quality the system can achieve. Also the delay is another important metric we need to consider. We hope that the potential threat in one picture frame should be detected with low delay after this frame being captured by the device. Based on these two optimization objectives, we mathematically formulate a scheduling problem for our edge monitoring system.

Suppose there are n IoT devices, m edge servers and a centralized cloud in the smart grid. We use set C to represent the set of IoT devices and c_i to represent the each IoT device where $1 \leq i \leq n$. Also, we use set E to represent the set of edge servers and e_j to represent each edge server where $1 \leq j \leq m$.

We first introduce the constants involved in our problem formulation. For simplicity, we consider each IoT device as the same type of smart camera and each edge server possesses the same computation resource. For each IoT device c_i, each picture it captures is in the same format with the size of s. The maximum frame rate for each IoT device to capture is f_u. Since the probability to be detected with threats among different locations varies, we define p_i as the probability to be detected with threats in one picture captured by device c_i. These probabilities can be counted from real-world data. For each edge server, we define the edge processing rate as v_e, which refers to the number of picture frames can be processed on each edge server within 1 second. For the cloud, the processing rate is denoted as v_c. For the connections, the uplink bandwidth from each IoT device c_i to the cloud is denoted as bc_i, and the uplink bandwidth from each edge server e_j to the cloud is denoted as be_j. The uplink bandwidth between each IoT device and each edge server will change due to the geographical distance. We define for the connection between device c_i and edge server e_j, the uplink bandwidth is $b_{i,j}$.

Then we introduce the variables. We define the set $X = \{x_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m\}$ to indicate the connection between each IoT device and edge server. If c_i is chosen to connect to e_j, $x_{i,j}$ will be set 1 otherwise 0. Since each IoT device can only connect to one edge server, therefore

$$\sum_{j=1}^{m} x_{i,j} = 1, \forall i \tag{1}$$

$$x_{i,j} \in \{0, 1\}, \forall i, j \tag{2}$$

For the IoT device c_i, the effective frame rate which is the actual number of picture frames being processed within 1 second is denoted as f_i. f_i cannot be greater than the number of frames which the edge network can transmit during 1 second, and also cannot be greater than the maximum frame rate for an IoT device. It should satisfy the minimum frame rate requirement, so that

$$x_{i,j} * f_i \leq \min(\frac{b_{i,j}}{s}, f_u), \forall i, j \tag{3}$$

Also, for one specific edge server e_j, the sum of frames it receives in 1 second should not be greater than the number of frames it can process, thus

$$\sum_{i=1}^{n} x_{i,j} * f_i \leq v_e, \forall j \tag{4}$$

Till now, we are able to calculate some important metrics. We define the effective frame rate for edge monitoring as F_e. It can be calculated as the sum of each device's effective frame rate:

$$F_e = \sum_{i=1}^{n} f_i \tag{5}$$

Also, according to equation (1)(2)(3)(4), F_e can be transformed to:

$$F_e = \sum_{j=1}^{m} \min(\sum_{i=1}^{n} x_{i,j} * \min(\frac{b_{i,j}}{s}, f_u), v_e) \tag{6}$$

For cloud monitoring, the effective frame rate F_c is constrained by either the uplink bandwidth from IoT device to cloud, or the cloud's processing rate.

$$F_c = \min(\frac{bc_i * n}{s}, v_c) \tag{7}$$

We are able to calculate the average detection delay D_e for one picture frame for edge monitoring as well. D_e can be divided into three parts: the uploading time from the IoT device to the edge server, the edge processing time, and the potential uploading time from the edge server to the cloud when this picture frame is detected with threats. The mathematical expression for D_e is shown as below:

$$D_e = \frac{\sum_{i=1}^{n} x_{i,j} * (\frac{s}{b_{i,j}} + \frac{1}{v_e} + \frac{s}{be_j} * p_i)}{n} \tag{8}$$

For comparison, the average detection delay for cloud monitoring D_c is calculated as below:

$$D_c = \frac{\sum_{i=1}^{n} \frac{s}{bc_i} + \frac{1}{v_c}}{n} \tag{9}$$

For better system performance, our two optimization objectives are increasing the effective frame rate and reducing the average detection delay respectively. Since the cloud monitoring's effective frame rate and average detection delay are fixed values, we transform the first objective into reducing effective frame rate ratio between cloud monitoring and edge monitoring, and transform the second objective into reducing average detection delay ratio between edge monitoring and cloud monitoring. To unify these two objectives, we set coefficients before the two ratios and the finalized optimization objective O is to minimize the sum of the weighted ratios. O can be defined as the function of scheduling X as below:

$$O(X) = A * \frac{F_c}{F_e} + B * \frac{D_e}{D_c} \tag{10}$$

And the coefficients A and B should follow

$$A + B = 1, A, B \geq 0 \tag{11}$$

Thus our problem can be generalized as:

$$\arg\min_{X} O(X) \tag{12}$$

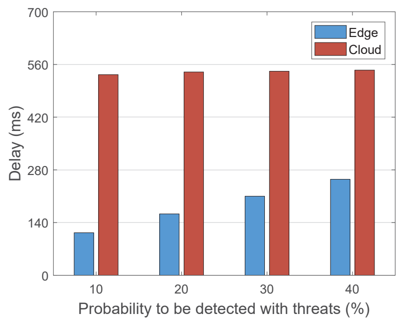
subject to the constraints in equations (1), (2), (6) to (11).

B. A Simulated Annealing Scheduling Solution

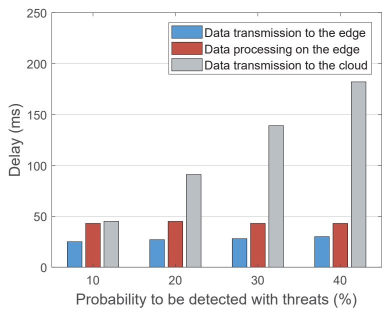
Our optimization goal is to solve a NP-hard scheduling problem. In this subsection, we give a heuristic scheduling algorithm with simulated annealing strategies.

Simulated annealing is a probabilistic strategy for approximating the optimum solution of a given function. Based on an initial solution, simulated annealing will continuously revise it and try to approach the optimum solution. It has the similar idea like greedy search to accept the move which turns to a better return of the given function, meanwhile, it also accepts the move which gets a worse return under the dynamic probability, since this move may create space for future moves which can result in finding the global optimum solution. Each time it accepts a worse move, the probability to accept the next worse move will decrease.

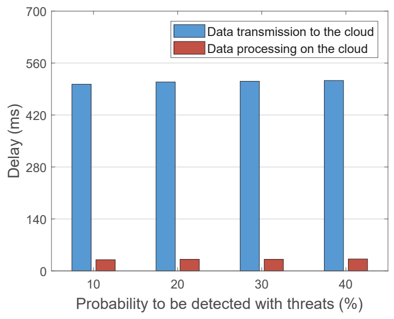
The algorithm's pseduo code is shown as Algorithm 1. This algorithm can be divided into two parts. The first part is using greedy search to find an initial scheduling. We first sort the set of IoT devices C following a non-increasing order according to its probability to detect with threats pi. Then we arrange each IoT device to connect to one edge server which has the largest uplink edge network bandwidth.



(a) Detection delay comparison

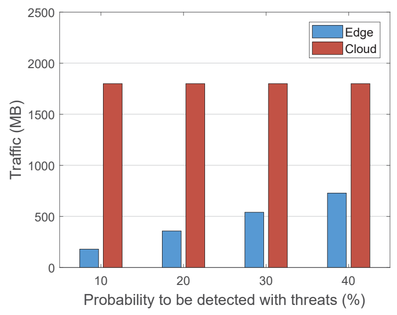


(b) Detection delay of edge monitoring

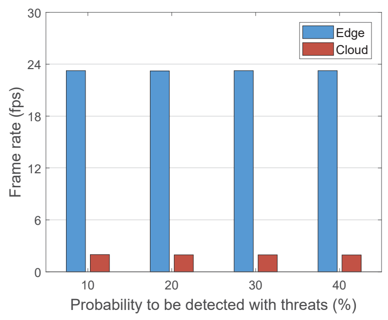


(c) Detection delay of cloud monitoring

Fig. 2: Real-world experiments on delay



(a) Traffic comparison



(b) Frame rate comparison

Fig. 3: Real-world experiments on traffic and frame rate

The second part is to use the simulated annealing strategy to optimize our current scheduling. The simulated annealing rescheduling process will last for a fixed rounds, and in each round every IoT device will own the chance to reconnect to other edge servers. For the reconnection, the IoT device will reconnect to anyone of all the edge servers. If current scheduling scheme can achieve a better return which means the calculated value of O is smaller than previous scheduling, we consider it is a better move and always accept it. Otherwise, we consider it as a worse move. The algorithm will accept this worse move according to the current acceptance probability.

Besides the parameters involved in the problem definition in Section IV, we also define rl as the number of rounds which the simulated annealing rescheduling process will last, and pc, pd as the two parameters to control the acceptance probability for worse moves.

V. System Evaluation

In this section, we will first evaluate our edge monitor system's performance from real-world experiments. Since the experiment resource is limited, we further design a simulation test to show our system's execution efficiency involving with multiple edge servers and multiple smart cameras.

Parameters	Values
s	0.9 MB
f_u	60 fps
v_e	115
v_c	1550
be_j	17.76 Mbps
bc_i	15.76 Mbps
$b_{i,j}$	10 - 320 Mbps or no connection
pc	0.5
pd	0.05

TABLE II: The values of simulation parameters

A. Real-world experiment

The first part is the real-world experiment. We made a prototype following the proposed edge monitoring framework in Section III and deploy it in real-world experiments. The experiments use 1 smart camera, 1 edge server and 1 cloud platform which has a powerful GPU cluster. The type of the smart camera is D-Link DCS-936L HD Wi-Fi camera . The edge server is a Dell server (OPTIPLEX 7010) and equipped with an Intel Core i7-3770 3.4 GHz quad core CPU, 16 GB 1333 MHz DDR3 RAM, and an NVIDIA GeForce GTX 1080 Ti GPU. The cloud platform is the Google Cloud Platform with the instance with 13 GB RAM and a NVIDIA Telsa K80 GPU. The deep learning application for threat detection uses

Algorithm 1 Simulated Annealing Rearranging Algorithm

```
Input:
    $X$, $b_{i,j}, \forall i, j$;
    optimization objective function $O(X)$;
    simulated annealing parameters $rl$, $pc$, $pd$.
Output:
    best scheduling scheme produced by simulated annealing
    strategy $X_{best}$
 1: Set all $x_{i,j} = 0, \forall 1 \le i \le n, 1 \le j \le m$
 2: Sort the IoT device set $C = \{c_i | 1 \le i \le n\}$ in a non-
    increasing order according to the value of $p_i$
 3: for $i$ from 1 to $n$ do
 4:   for $j$ from 1 to $m$ do
 5:     Find the index $j0$ where $\{b_{i,j0} \ge b_{i,j} | \forall 1 \le j \le m\}$;
 6:     $x_{i,j0} \leftarrow 1$;
 7:   end for
 8: end for
 9: $X_{best}, X_{current} \leftarrow X$;
10: $obj_{best} \leftarrow O(X_{best})$;
11: for $r$ from 1 to $rl$ do
12:   Initialize visited edge server set as $visited \leftarrow \{\}$;
13:   for $k$ from 1 to $n$ do do
14:     $i \leftarrow$ a random number from 1 to $n$ but not in $visited$;
15:     Set $visited \leftarrow visited \bigcup i$;
16:     for $j$ from 1 to $m$ do
17:       Set $x_{i,j} \leftarrow 1$;
18:       Calculate objective function difference
          $\delta \leftarrow O(X) - O(X_{current})$;
19:       if $\delta < 0$ then
20:         Update $X_{best}$, $obj_{best}$;
21:         Accept this move by setting $X_{current} \leftarrow X$;
22:       else
23:         if $e^{pc} >$ a random number between $(0, 1)$ then
24:           Accept this move by setting $X_{current} \leftarrow X$;
25:           Adjust parameter $pc \leftarrow pd * pc$;
26:         end if
27:       end if
28:       Set $x_{i,j} \leftarrow 0$;
29:     end for
30:   end for
31: end for
32: return $X_{best}$;
```

singleshot detection [15].

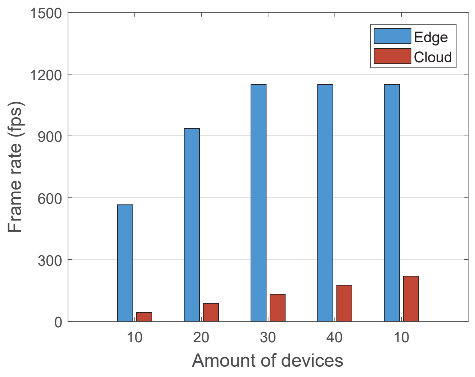
The dataset are the pictures captured by the camera beside the high voltage power line. We prepared 4 sets of pictures. Each set of picture contains 2,000 pictures and there are 10%, 20%, 30% and 40% of these pictures will be detected with threats for each set respectively.

We first evaluate the detection latency for each picture frame. See Fig. 2. The first picture presents the delay comparison between edge monitoring and cloud monitoring. The cloud monitoring's delay from the picture being sent by the IoT device to the cloud finished the detection processing is about 533 to 545 ms per picture frame. Our edge monitoring's delay is ranging from 113 to 255 ms. Comparing to cloud monitoring, the delay is largely reduced by 53% - 79%. Though with the increasing on the probability to be detected with threats, the edge monitoring's traffic will increase as well. The threat will be detected beside a high voltage power line in the real-world environment is extremely low and usually not beyond 10%. So this can prove the excellent efficiency on detection delay of edge monitoring.

We further investigate the construction of the delay. Fig. 2(b)(c) shows the construction of edge monitoring delay and cloud monitoring delay respectively. We can see that the biggest difference for edge computing and cloud computing is the data transmission time between IoT device and the processing server. In cloud monitoring system, the pictures captured by the IoT device will be directly sent to the cloud over the Internet, this process will cost more than 500 ms. While in our edge monitoring system, the pictures are sent to the near-end edge servers, which has superior network condition and larger bandwidth and the transmission only takes about 30 ms to finish. This is the most significant advantage of edge monitoring.

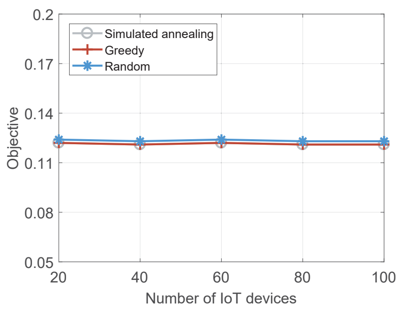
Then we evaluate the traffic produced on the Internet. See Fig. 3(a), the traffic produced by cloud monitoring is totaly 1.8 GB, while the traffic produced by edge monitoring is less than 750 MB. Though the traffic will increase with the increasing on probability to be detected with threats, since more and more processed pictures with detection information will be transmitted to the cloud. The size of traffic produced by edge monitoring is still far less than the cloud monitoring. We also evaluate the effective frame rate and the result is shown as Fig. 3(b). It proves that the frame rate has no relation with the probability to be detected with threats. And for our edge monitoring, the frame rate is more than 10 times higher than the cloud monitoring.

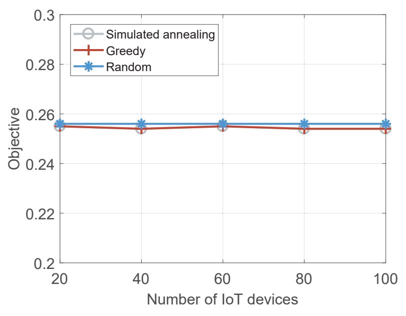
From the above real-world evaluation result, our edge monitoring framework significantly outperforms cloud monitoring in detection delay, traffic and frame rate. In the next



(a) Frame rate

Fig. 4: Simulation evaluation (part 1)


(a) Delay (p_i = 5%)


(b) Delay (p_i = 20%)

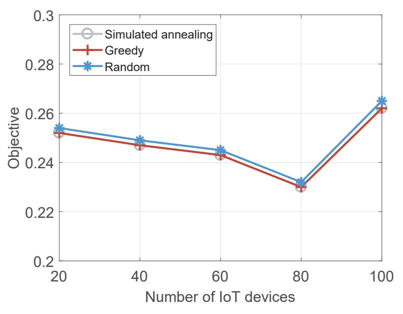
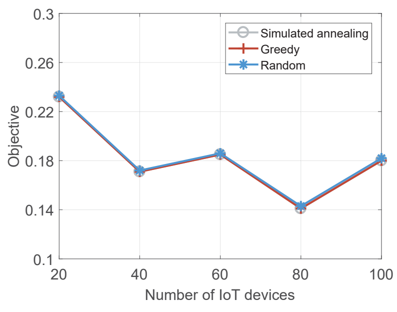
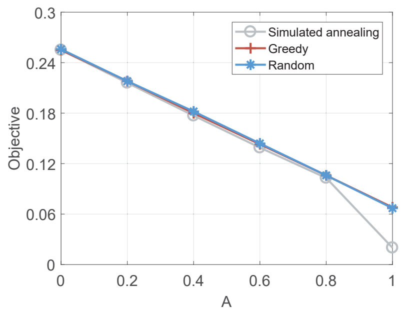

(c) Delay (normal distributed probabilities)

Fig. 5: Simulation evaluation (part 2)


(a) Delay (power-law distributed probabilities)


(b) Objective comparison (20 edge servers)

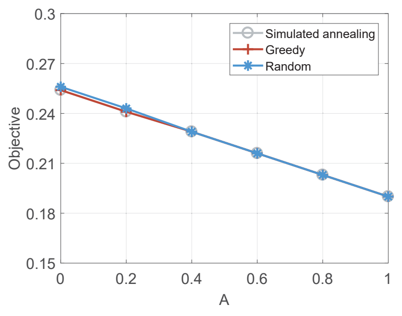

(c) Objective comparison (100 edge servers)

Fig. 6: Simulation evaluation (part 3)

subsection, we will consider the case of multiple IoT devices and multiple edge servers.

B. Simulation evaluation

In this subsection, we will give the simulation evaluation of our edge monitoring system. We evaluate the edge monitoring performance using the solution provided in section IV. B. Concerning the consistency with real-world environment, our simulation uses the data collected from real-world experiments, see TABLE II. The parameters defined in the table is the same as the parameters defined in section IV. In our simulation evaluation, the edge monitoring system will contain a large number of IoT devices, totally 10 edge servers and 1 cloud server. Each edge server is a GPU cluster which has the computation resources 5 times of the single NVIDIA GTX 1080 Ti GPU. The cloud server is a more powerful GPU cluster which has the computation resources 50 times of the single NVIDIA Telsa K80 GPU. For simplicity, we consider each of the IoT device will have the same uplink bandwidth connecting to the cloud. Also, each edge server owns the same uplink bandwidth of the connection to the cloud. The bandwidths from each IoT device to the cloud are different. For each IoT device, we confine that it can get access to at least 1 edge servers. The bandwidth between each IoT device and each edge server is a random value between 10 - 320 Mbps, or 0 Mbps if they are not connectable.

We will make comparison of performance under different conditions, i.e., the different number of IoT devices, the different value of coefficient A and B and the different distribution of probability to be detected with threats. Lower objective value we get, the better performance our edge monitoring system will own comparing to the cloud monitoring. We also introduce the random scheduling as the baseline, which has the strategy to randomly choose to connect for each IoT device to one of the accessible edge servers.

We first evaluate the effective frame rate, see Fig.4. To measure the effective frame rate, we can set the coefficient $A = 0, B = 1$ in equation (10) which is the formulated optimization objective in our problem formulation. Since $B = 0$ here, then our optimization objective will not be related with the p_i. Starting from 20 IoT devices, our algorithm will give the scheduling with $O = 0.061$, which means the effective frame rate of cloud monitoring is only 6% of our edge monitoring system has. With the increasing on number of IoT devices, objective O is increasing as well. This phenomenon doesn't mean edge monitoring's performance is being worse, actually comparing to cloud monitoring, the increasing frame rate of the edge monitoring is even larger. Since O is a ratio, the base of the numerator (the frame rate of edge monitoring) is so large that even a little increasing on cloud monitoring will lead to the reduction of the calculated objective O. Also we noticed that, our simulated annealing algorithm performs better than other algorithms when the number of IoT devices is small. However, the performance of each algorithm become similar when the number of IoT devices is large. This is caused by the bottleneck of limited computation performance of the edge server. Because the number of edge server is fixed, there exists a maximum effective frame rate for edge computing. When the amount of IoT devices reaches a specific number, then the only solution to increase effective frame rate is to enlarge the computation resource on each edge server or increase the amount of accessible edge servers.

We then evaluate the detection delay. By setting $A = 1, B = 0$ in equation (10), we are able to calculate the average detection delay for each picture frame. Fig.5(a)(b) presents the results being evaluated under similar environments except the probability to be detected with threats. The probability for each IoT device is all set to 5% in Fig.5(a) and 20% in Fig.5(b). Similar as the effective frame rate evaluation above, our edge monitoring system can finish the detection using only 12% - 25% time of what the cloud monitoring requires. When concerning about the algorithms, our simulated annealing algorithm has the same performance with greedy algorithm. It is because the greedy algorithm always looks for the maximum network bandwidth for edge network connection and this is the only variant in measuring the detection delay. So that the greedy algorithm will always achieve the optimal solution, as well as our algorithm since it is an enhancement of greedy strategy. With the increasing of IoT devices, there is no significant changes in the objective function value. These proves the scalability of the edge monitoring system. Comparing with these two graphs, we can find the objective function value is increasing when p_i increases. This is the natural phenomenon since more and more pictures will be uploaded to the cloud and increase the average detection delay.

Also, we conduct another two experiments where the probabilities to be detected with threats for each IoT device will follow two specific distributions, i.e., normal distribution and power-law distribution. See Fig.5(c) and Fig.6(a), in each of the graphs, there is no evident trend of what our objective function value will change with the increasing on amount of IoT devices. This phenomenon shows that the probability is not the determining factors in edge monitoring, comparing to the connection of edge network.

We finally make two simulations focusing on the coefficients of our formulated problem, see Fig.6(b)(c). The x-axis means the value of A. We can find that with the increasing on the value of A, the objective function will get smaller values. This shows that our system is more efficient in increasing effective frame rate comparing to reducing detection delay.

VI. Related work

The smart grid was first proposed in 2005. It enables a new era for the electrical power grid system to be constructed with more reliable and efficient services combing with information technologies. Key researches on smart grid is to investigate on all kinds of IoT devices and sensors and build the communication network between these devices. Yun *et al.* [16] proposes the architecture of smart grid and listed several key technologies of IoT will be involved. Gungor *et al.* [17] investigates in the challenges and opportunities of wireless sensor networks under the smart grid environment and conducts experiments on the statistical characterization study under different electric-power-system environments. Sagiroglu *et al.* [18] presents their vision on dealing with issues featuring big data and smart grid.

Edge computing enables the computation performed on the near-end and deal with the high-latency issue of cloud computing. Zhu *et al.* [19] investigates the advantages of mobile edge computing proposed a content optimization infrastructure. Karim *et al.* [8] proposed FemtoClouds to enable multiple mobile devices to share the computation resources and configure a coordinated edge-cloud collaborative computing service, by leveraging mobile devices to provide cloud services at the edge. Dawei *et al.* [20] proposed an adaptive mobile object recognition framework called DeepCham, which solves the issue of recognition accuracy degradation due to context variations caused by different locations and time, by collaboratively training a domain-aware adaption model together with a domain-constrained deep model with the introducing of the intermediate edge master. Chen *et al.* [9] focused on the distributed computational offloading problem and proposed a offloading model for mobile edge computing. He *et al.* [21] proposed heuristic strategies to protect the user's location privacy by mimicking the user's mobility.

VII. Conclusion

In this paper, we introduce the edge computing into the smart grid monitoring system. Edge monitoring has the huge advantage in low latency and is able to largely improve the monitoring quality comparing to cloud monitoring. We also formulate a scheduling problem of optimizing the performance of our edge monitoring system. We further make a prototype of the edge monitoring system and conduct real-world experiments and simulations to prove its efficiency. Our edge monitoring framework has been accepted by the State Grid Corporation of China, and we are deploying over 1,000 devices and edge servers in the power system of Liaoning Province, China.

Acknowledgment

This research is supported by an Industrial Canada Technology Demonstration Program (TDP) grant, an NSERC Discovery Grant, and a MITACS grant.

References

[1] S. E. Collier, "Ten steps to a smarter grid," in *Rural Electric Power Conference, 2009. REPC'09. IEEE.* IEEE, 2009, pp. B2–B2.

[2] Y. He, G. J. Mendis, and J. Wei, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2505–2516, 2017.

[3] M.-P. Hosseini, H. Soltanian-Zadeh, K. Elisevich, and D. Pompili, "Cloud-based deep learning of big eeg data for epileptic seizure prediction," in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*. IEEE, 2016, pp. 1151–1155.

[4] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proceedings of 10th International Conference on Intelligent Systems and Control (ISCO)*. IEEE, 2016.

[5] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustainable Energy, Grids and Networks*, vol. 6, pp. 91–99, 2016.

[6] H. Tan, Z. Han, X.-Y. Li, and F. C. Lau, "Online job dispatching and scheduling in edge-clouds," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

[7] T. Li, C. S. Magurawalage, K. Wang, K. Xu, K. Yang, and H. Wang, "On efficient offloading control in cloud radio access network with mobile edge computing," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 2258–2263.

[8] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*. IEEE, 2015, pp. 9–16.

[9] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[10] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Cachier: Edge-caching for recognition applications," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 276–286.

[11] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, "Fair caching algorithms for peer data sharing in pervasive edge computing environments," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 605–614.

[12] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 1281–1290.

[13] S. Teerapittayanon, B. McDanel, and H. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proceedings of 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[14] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of 21st International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, 2017, pp. 615–629.

[15] S. Leonardo, "Multibox single shot detector (ssd)," https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/single-shot-detectors/ssd.html.

[16] M. Yun and B. Yuxin, "Research on the architecture and key technology of internet of things (iot) applied on smart grid," in *Advances in Energy Engineering (ICAEE), 2010 International Conference on*. IEEE, 2010, pp. 69–72.

[17] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE transactions on industrial electronics*, vol. 57, no. 10, pp. 3557–3564, 2010.

[18] S. Sagiroglu, R. Terzi, Y. Canbay, and I. Colak, "Big data issues in smart grid systems," in *Renewable Energy Research and Applications (ICRERA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1007–1012.

[19] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, "Improving web sites performance using edge servers in fog computing architecture," in *Proceedings of 7th International Symposium onService Oriented System Engineering (SOSE)*. IEEE, 2013, pp. 320–323.

[20] D. Li, T. Salonidis, N. V. Desai, and M. C. Chuah, "Deepcham: Collaborative edge-mediated adaptive deep learning for mobile object recognition," in *Edge Computing (SEC), IEEE/ACM Symposium on*. IEEE, 2016, pp. 64–76.

[21] T. He, E. N. Ciftcioglu, S. Wang, and K. S. Chan, "Location privacy in mobile edge clouds: A chaff-based approach," *IEEE Journal on Selected Areas in Communications*, 2017.