IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 2020, date of current version Oct. 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.DOI

Optimization-Based Offloading and Routing Strategies for Sensor-Enabled Video Surveillance Networks

FRANK YEONG-SUNG LIN[1], **CHUN-MING CHIU**[2], **CHIU-HAN HSIAO**[3], **(MEMBER, IEEE), YEAN-FU WEN**[4], **(Member, IEEE)**

[1]Department of Information Management, National Taiwan University, Taipei, Taiwan (R.O.C.)
[2]Convergence Services Laboratory, Chunghwa Telecom Laboratories, Taoyuan, Taiwan (R.O.C.)
[3]Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan (R.O.C.)
[4]Graduate Institute of Information Management, National Taipei University, New Taipei City, Taiwan (R.O.C.)

This work is supported in part by Ministry of Science and Technology (MOST), Taiwan, under Grant Number MOST 107-2221-E-305-003-MY2.

Corresponding author: Yean-Fu Wen (e-mail: yeanfu@mail.ntpu.edu.tw).

ABSTRACT For the Internet of things, having sensors in devices used for video surveillance services, such as cameras, is crucial. The advancement of edge computing technology has enabled high computing capacity and the handling of massive data sets. The concept of cloudlets is employed in edge computing for in-network processing, especially for large-size multimedia data processing. Cloudlets are essential for services with high computing costs. Contrary to traditional cloud computing, data can be offloaded to in-network devices and core clouds, thereby improving the quality of service and enhancing resource utilization. However, the trade-off between network transmissions and nodal processes with delay-aware multimedia traffic has been demonstrated to be an NP-complete problem. The problem is presented as a mathematical formula to maximize the minimal delay gap between the tolerable event delay, sending time, and processing time. The problem is subject to in-network processing node assignment, routing paths, transmission capacities, computing capacities, and the effective service period. The Lagrangian approach was employed to evaluate the method proposed in this study; a near-optimal solution was obtained, and several experiments were performed to demonstrate that the proposed method outperforms existing methods.

INDEX TERMS Edge Computing, Offloading, Quality of Service, Routing, Video Surveillance.

I. INTRODUCTION

The most crucial breakthrough in the Internet of things (IoT) field has been the increased computing capability of sensors. Previously, sensors could only complete tasks with simple data processing or return raw data to the sink node [1], whereas they are now capable of performing highly complex tasks, such as handling complex and massive data processes, because of novel artificial intelligence (AI) chips [2], [3]. Despite assistance from AI chips, IoT devices remain reliant on machines with high-power computing capacities, such as cloudlet or core cloud capability, for the processing of high quality multimedia data [4].

The traditional cloud computing structure is insufficient for handling IoT applications because of the limited computing power of IoT devices and the transmission capacity and round-trip time delay constraints of these services. The concept of edge computing was proposed to overcome these limitations. The main idea of edge computing is to process the data through last-mile services [2], [5]–[8]. For IoT applications, multimedia data are offloaded to the edge cloudlet or fog-computing hosts to ensure high computation capacity [2], [5], [6], [9]. This method reduces the time required for transmission and processing and the workload of the central servers.

Several IoT applications–such as the smart city, health care, and campus safety–involve video surveillance, which is employed to detect incidents of interest [4], [8], [10]–[13]. The main concept of video surveillance is to apply data analytic methods to real-time data, including videos obtained using IoT cameras distributed across a city or campus [14]. Guards can then inspect the city or campus for safety and make decisions using the patterns extracted from the data. This paper discusses the use of video for detecting dangerous events. Internet protocol (IP) cameras equipped with AI chips were used to detect incidents involving, for instance, dangerous individuals, fire, and violence [15]. Situations involving the foregoing may be dangerous for individuals or threaten property if they are not promptly detected and prevented or quickly stopped. Therefore, the present study employed tolerable delay as a metric to ensure that events were detected within a given time.

Real-time facial recognition has been demonstrated to be a practical and effective technology with high recognition accuracy [16], [17]. An IP camera records images at a rate of 30 frames per second in most usage configurations. The highest human movement speed is approximately 10 m/s. We assume that the effective range of an IP camera is 30 m. Therefore, the IP camera captures enough frames to enable analysis of when a person moves through the zone covered by the IP camera. The face is captured using several cameras and highly accurate facial recognition algorithms. The danger that the person presents can be determined by analyzing the video recordings.

A common practice for detecting fire using IoT systems is to employ smoke sensors. However, smoke detectors only work well in enclosed spaces, such as classrooms, shops, and factory workshops [15], [18]. Smoke detectors are not effective at detecting fire with high accuracy in open spaces, such as parking lots and parks, because the smoke must accumulate to a sufficient level before an alert can be issued. Some researchers have proposed algorithms with high accuracy and a low false alarm rate [19] that are purely based on computer vision. Therefore, IP cameras are more suitable in open areas than smoke detectors are.

The bag-of-words model is employed in the field of image processing and tends to be efficient at performing highly complex computer vision tasks. The concept of the bag-of-words model originates from the domain of information retrieval [20], [21]. A document is represented by a bag-of-words, with the order of words not considered. The document is thus transformed into a high-dimensional vector and can be used in further information retrieval applications, such as document classification. In the computer vision field, this idea is used to extract critical features (the ’words’). Each image is then transformed into a high-dimensional vector to represent a feature [20]. The vectors are used for motion classification. In this study, we classified frames in which violence was captured and demonstrate the reasonableness and appropriateness of using IP cameras for violence detection.

IP cameras are deployed near devices with a continuous electric power supply. For example, traffic lights and streetlights are often selected as locations for IP camera installation because they enable continuous operation of the cameras. Planners must be familiar with the landscape to ensure that the locations of the IP cameras do not result in blind spots. An IP camera can offload its computation task onto neighboring IP cameras that can perform detection of a type of incident. However, IP cameras can become overloaded with computation from other cameras if an appropriate offloading scheduling assignment scheme is not implemented. Although the compression algorithm can reduce the amount of data, media data must be processed by the cloudlet when all IP cameras offload their data. Therefore, a long computation delay occurs, and the transmissions between IP cameras and cloudlets are congested. High transmission latency occurs when all videos are sent to the core cloud for processing. An in-network offloading algorithm is required to solve this problem.

Currently, IP cameras are principally used for recording videos that are then sent to a security office where a security guard assesses the threat. The present study investigated in-network processing through both edge computing and offloading [5], [9] along the routing path to the security office. The selected processing functions were predefined and installed on the responding nodes to match the required functions before routing to the next processing node. We identified a near-optimal solution that provides stable and reliable transmission and processing. In the proposed system, the IP cameras determine the target, and the hosts and cameras cooperate to manage the computation and link capacity constraints. Accordingly, the objective of this study was to maximize the minimal delay gap between the tolerable delay of a detected event and the actual event processing time to enhance performance.

The main contributions of this study are as follows:

1) Development of in-network multimedia processing for operations with IoT devices, edge computing, and a cloud network.

2) Model of the offloading and routing assignment problem using linear and nonlinear mathematical planning formulations. The objective was to maximize the minimal gap between the processing and transmission tolerable delay for various types of events.

3) Proposal of an in-network processing node to match the processing of the type of event within a tolerable delay.

4) Evaluation of the proposed method through comparison with existing methods.

In the next section, we discuss and compare the main idea with existing approaches. Section III describes the research problem. Section IV details the proposed method. Section V presents the simulation results and discusses related issues. Finally, in Section VI, the conclusions are presented.

II. RELATED WORK

An IP camera typically detects several types of events. Three tasks–facial recognition, fire detection, and violence detection–were considered in this study. When implementing facial recognition, the first step is to determine whether each frame contains a face. The frame information and the background are then removed by selecting a rectangular region of interest from which to extract the face. These rectangular images are then inputted to a trained classifier to identify the face [22].

The procedure for violence detection is as follows. In the training phase, we employ motion scale-invariant feature transform (MoSIFT) to detect and describe features. A

MoSIFT comprises the normal SIFT descriptor and a histogram of optical flow. The features obtained using MoSIFT are then processed using k-means clustering. The center of each cluster is selected as a 'word' in the bag-of-words model. A histogram of word occurrence is then generated by changing the image into descriptors and assigning them to the closest visual words. The final step is classifying these high-dimensional vectors in which each entry represents a visual word; this classification is achieved using a support vector machine [23]. During testing, frames are classified as either containing or not containing violence.

In fire detection, regions in a video that contain flame movement are identified. The regions in which motion has been detected are passed to the fire color classifier. The dynamic textures models are constructed for video sequences containing specific stationarity. We then use the nearest neighbor classifier to determine whether the region is on fire [19]. Other procedures, such as sensor network-based fire detection [15], are also used in this work.

Numerous papers have discussed IoT applications and constraints in light of the properties of various IoT architectures. Sensors and smart devices with data-gathering abilities are battery powered. The protocols for these low-power devices are usually supported with low bandwidth. Bluetooth low energy (BLE) and ZigBee are commonly used protocols in the IoT, and they connect edge hosts with gateways. The main purpose of these protocols is to increase the life cycle of battery-operated devices and manage the constraint of low bandwidth shared with other devices [24]. The throughput of BLE and ZigBee is 270 and 150 kbps, respectively, which are low compared with Wi-Fi and Ethernet networks.

Research on video surveillance has mainly discussed how the accuracy of event detection or object tracking can be improved [10], [25], [26]. Because of the considerable volume of data involved in video streaming, a video must be preprocessed before being transmitted to a cloudlet and subjected to advanced computing tasks [5]. The three main preprocessing methods currently used are decreasing the frame rate, discarding frames, and decreasing the resolution [11], [27]. These methods are set in advance and are not adjustable during the execution period. Although these methods reduce the amount of data, some can cause data loss that reduces the accuracy of event detection or object tracking.

In-network processing had been studied in sensor networks as an approach to reduce the amount of transmitted data that consequently also reduces the transmission delay. Although edge computing supports local processes to avoid routing to far cloud servers, most studies have discussed the processing range between edge servers and end-users [2], [7], [25] or processing when data are fully sent to the core cloud for processing [28]. Few studies, such as [6], have addressed data analytics with collaborative edge and cloud computing.

Traditional routing methods have been discussed and designed for transmission from source to destination using shortest-path algorithms, such as Dijkstra's algorithm and the Bellman-Ford routing algorithm [29]. The main idea of these methods is to identify a routing path with a quality-of-service (QoS) metric, such as transmission bandwidth or delay [30]. However, nodal delay and segment routing [31] are not included as strategies for in-network processing applications, such as monitoring applications of wireless sensor networks (WSNs) [32].

Segment routing strategies, such as multiprotocol label switching (MPLS), are beneficial to in-network processing. The main idea of the MPLS routing technique is to transmit data from one node to the middle node but not to the destination [33]. This technique has been adopted in telecommunications networks to reduce routing table search costs and accelerate traffic flows. The minimal computation times attained through node selection and min-max link utilization using flow assignment problems are addressed in [32]. Segment routing methods address the routing problem from the perspective of network access and IP layers. However, the present study aimed to address routing and nodal processes using cross-layer methods (covering network to application layers). The main idea of the segment routing method was adopted in this work, and a newly designed nodal selection approach with a delay-aware routing strategy was also adopted.

Zhang, Lei, and Zhang (2020) proposed a multipath multimedia segment routing method to provide the required bandwidth and end-to-end transmission delay for real-time interactive multimedia [34]. This work addresses routing and nodal process problems as well as diverse data aggregation, offloading, and in-network process strategies to reduce the amount of data transmitted along the routing path to the destination. Although the main idea of wireless multimedia sensor networks is to reduce the amount of multimedia data transmitted through the intranode fusion process [35], the specific functions of selecting and routing data to a specific node was not included in their study. From the event-driven routing viewpoint, reducing the transmission and nodal processing time is a major problem [1], [11], [32], [36]. The reverse multicast routing method has been studied for WSNs that concentrate on simple data collection and data aggregation. Studies have investigated the collection of event data by sensor nodes, summarized the collected data, and forwarded the data to the sink node [1], [37]. The present work extends the use of WSNs for complex problems that in-network processes require for specific functions, computing power, storage, and support services. The routing discovery constructs the spanning tree with Prim's method [29]. The present work extends the reverse multicast routing strategy for multimedia processes and specific functional devices. Both the routing pathway and the status of functional device selection are balanced to maximize the gap between transmission/processing and tolerable delays.

Several routing methods adopt the number of hops, bandwidth, traffic load, and delay as link metrics. The number of hops does not reflect the bandwidth and aggregated flow. The bandwidth does not reflect the traffic load. Although the traffic load reflects the transmission link status, the nodal status is not reflected [30]. Therefore, herein, both transmission delay and nodal processing time were employed as combination metrics for routing to identify the near-optimal routing path and processing node.

Table 1 summarizes the comparison of routing strategies from the various factors. These factors were correlated with this work, but the strategies and feasibility did not match the problem studied. Therefore, we modified and extended these methods to suit the objective of the research problem. Shortest-path routing methods are based on the idea of reducing the transmission and processing delay. The segment routing method must be extended to suit the required functional node. This work adopts reverse multicast on data aggregated in the middle nodes, but we cannot only select the closest node. We considered the link delay, nodal delay, traffic load of links and nodes, and the functions of in-network processing nodes. This work combines these factors and develops a set of algorithms to achieve the research goal of this work.

III. PROBLEM DESCRIPTION

Here, we consider an actual case of a video surveillance system in a parking lot (see "The Bank of New Hampshire Pavilion in New England," https://www.videosurveillance.com/blog/applications/event-surveillance/new_case_study_highlights_crowd_management.asp). The monitoring of potential accidents and damage to vehicles, theft, or vandalism in large parking lots requires a powerful high-definition (HD) video surveillance system. The detected event must be processed within a tolerable delay. Several cameras are installed to capture HD videos from various angles. The videos are then subjected to in-network processing to filter and eliminate redundant transmission to the cloudlet hosts. Then, the initial image analysis results are sent to the security guard. The processed videos are also sent to a core cloud for facial and behavioral recognition to be recorded in facial and behavioral databases.

Accordingly, the system model is a hierarchical structure with three levels: IoT device, cloudlet, and core cloud levels. At the IoT device level, a circle represents an IP camera, and some IP cameras are associated with a Wi-Fi local area network, as illustrated in FIGURE 1. Each link represents two nodes associated with a network, which transmits data through a wireless mesh network or a wired network. Most of the wireless links transmit data through the independent channel because the number of channels suffices. Cloudlets are associated with some IP cameras on the IoT device level through Ethernet or 4G/5G networks. IP cameras connected with cloudlets through Ethernet are spatially close to buildings. At the core cloud level, the core cloud is connected to cloudlets through optical fibers. The system automatically updates end-users with information regarding the location at which a certain type of event has been detected.

This study assumed that an IP camera is triggered by one type of sensing device. The IP camera must detect one type of event in our model. The software-defined network (SDN) assumes that inclusion in the network relies on network statuses being periodically collected for routing decisions. We also assumed that background loading for the IP camera must be computed, and thus, the IP cameras tend to offload tasks to other devices, cloudlets, or cloud hosts. The computing capacity and detection functions of IoT devices, cloudlets, and the core cloud hosts are varied because of the properties of the hardware.

TABLE 1. COMPARISON OF ROUTING STRATEGIES

Strategic	Description	Literature
Shortest path routing	Identify a path with a QoS metric using an algorithm. Metric selection is crucial for determining how the current node routes to the next middle node.	[29], [30]
Segment routing	Identify a subpath from one node to the middle node, and then combine segment paths into a complete path.	[31], [33]
Multipath routing	Divide the data into several sections, and then transmit them to the destination over multiple paths to reduce the transmission delay.	[34]
Data aggregation	Collect event-driven data using several devices and process the collected data in-network to reduce the amount of data that must be transmitted.	[1], [35]
Offloading	Offloading is a type of aggregation of similar data. It requires advanced interactions with other data, such as a facial database. Advanced computing, storage, and service resources are required to support data processing.	[38]
Routing metrics	The numbers of hops, bandwidth, traffic load, and delay are used as metrics. The metric selected depends on the goal of transmission.	[30]

The IP cameras in the IoT system record video if they are triggered by the embedded sensors. For instance, if smoke is detected, the IP camera is triggered, the video is tagged with the type of suspected event, and the video data are offloaded for further image processing and application. We designed an algorithm to prevent the overloading of IP cameras and transmissions because of the various computational capabilities of IP cameras, cloudlets, and the core cloud. Overloading contributes to high computation and transmission times, resulting in an exceedance of the tolerable delay. Therefore, the goal of this study was to maximize the minimum delay gap between the tolerable delay of the detected event and the actual transmission and processing time.

A. NETWORK MODEL

V refers to the set of IoT devices, cloudlet hosts, and core cloud hosts. L is a set of links, including routers, switches, and access points, used to associate the IoT devices. The set of hosts and devices V and links L form a graph $G(V, L)$. FIGURE 1 illustrates the network architecture. A set of events U is recorded by a camera device, with $u \in U$ being an individual event. Another device, a cloudlet or the core cloud

$v \in V$, is assigned to execute in-network processing. The given parameters and decision variables are listed in Tables 2 and 3, respectively. A trade-off exists between data processing and transmission time. If an event video is sent to the core cloud, the processing time is short, but the transmission time is long. If an event can be processed by another device or an edge computing host, the processing time is long but the transmission time is short. This study aimed to reduce the response time for various types of detection events.

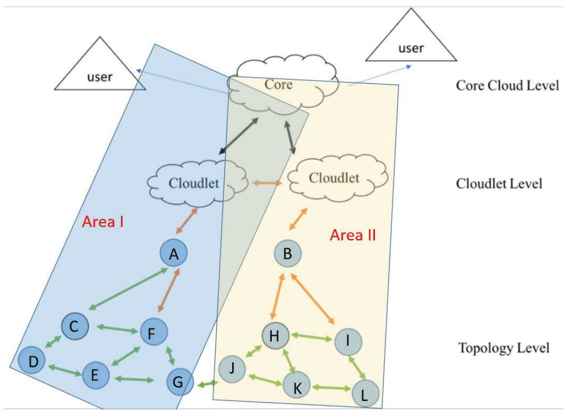

FIGURE 1. Network architecture.

TABLE 2. GIVEN PARAMETERS

Symbol	Description
V	Set of all nodes, where $v \in V$.
U	Set of IP cameras that are triggered by events, where $u \in U$ and U is a subset of V.
I	Index set of all events that the IoT network can detect, where $I_u \in I$ and $u \in U$.
I_u	Index set of all events that node u is able to detect, where $u \in U$.
h_{vi}	Indicator function, which is 1 if node v can detect event i and 0 otherwise, where $i \in I_u$, $u \in U$, and $v \in V$.
p_{uv}	Index set of all paths for nodes u and v, where $u \in U$ and $v \in V$.
$\delta_{p(m,n)}$	Indicator function, which is 1 if candidate link (m,n) is on path p and 0 otherwise, where $(m,n) \in L$, $p \in P_{uv}$, $u \in U$, and $v \in V$.
Q_v	Index set of all paths between node $v \in V$ and the end user.
$\delta_{q(m,n)}$	Indicator function, which is 1 if candidate link (m,n) is on path q and 0 otherwise, where $(m,n) \in L$, $q \in Q_v$, and $v \in V$.
β_u	Computation cost of data sent from node $u \in U$.
γ_u	Transmission cost of data sent from node $u \in U$.
ω_{uv}	Transmission cost of the result packet sent from node v, where $u \in U$ and $v \in V$.
L	Index set of all candidate links, where $(m,n) \in L$.
C_v	Computation capacity of node v, where $v \in V$.
$C_{(m,n)}$	Link capacity of (m,n), where $(m,n) \in L$.
M	A large arbitrary positive number, which can help prevent multiplication of the decision variable.
T_u	Tolerable delay of node u, where $u \in U$.

TABLE 3. DECISION PARAMETERS

Symbol	Description
α_{uvi}	1 if event i is offloaded from node u to node v and 0 otherwise, where $u \in U$, $v \in V$, and $i \in I_u$.
x_p	1 if path p is selected for transmission and 0 otherwise, where $p \in P_{uv}$, $u \in U$, and $v \in V$.
$y_{u(m,n)}$	1 if node u transmits through candidate link (m,n) and 0 otherwise, where $u \in U$ and $(m,n) \in L$.
x_q	1 if path q is selected for transmission and 0 otherwise, where $q \in Q_v$ and $v \in V$.
g_v	The total aggregate computation cost of node $v \in V$.
$f_{(m,n)}$	The total aggregate transmission volume on link (m,n), where $(m,n) \in L$.
d_v	The computation delay of node v, where $v \in V$.
$d_{(m,n)}$	The transmission delay of link (m,n), where $(m,n) \in L$.
$\chi_{u(m,n)}$	The transmission time for each selected link (m,n) from node u to node v, which is always larger or equal to zero, where $u \in U$, and $(m,n) \in L$.
$\lambda_{u(m,n)}$	The transmission time of the result from node u to the user, which is always larger or equal to zero, where $u \in U$ and $(m,n) \in L$.
ψ_v	The true computation delay of node u, where $u \in U$.
d_u	The total delay of node u, where $u \in U$.

The objective function (IP) is used to maximize the minimum delay gap, which is calculated using (15). The main idea of function (IP) is to consider various types of events that require various delay ranges. Achieving the minimum delay for emergency events may affect the required delay for emergency data. Therefore, the objective function aims to achieve the required delay for all events and balance loads through maximizing the minimal delay gap d, which is determined in Constraint (15).

$$\max d \tag{IP}$$

subject to the following constraints.

1) Offloading Assignment Constraints:

Constraint (1) states that function i can be performed by node v to determine whether the required functions can be satisfied by node u. Once all functions are satisfied, the decision variable α_{uvi} is set to 1; otherwise, the requested job of node u should not be performed by node v.

$$\alpha_{uvi} \leq h_{vi}, \forall v \in V, u \in U, i \in I_u \tag{1}$$

Once a job has been requested by node u, Constraint (2) states that all functions i will be processed by one or fewer nodes v. The job cannot be handled by neighboring nodes, but it is finally handled by the core cloud, which ensures that all jobs can be processed in the network.

$$\sum_{v \in V} \alpha_{uvi} \leq 1, \forall u \in U, i \in I_u \tag{2}$$

Constraint (3) states that a path must exist once the function i is offloaded from node u to node v. Not all paths between nodes u and v must be listed. This is achieved using a greed-based routing algorithm.

$$\alpha_{uvi} \leq \sum_{p \in P_{uv}} x_p, \forall v \in V, u \in U, i \in I_u \tag{3}$$

If a link (m, n) is on the selected path p from nodes u to v, the link is marked with the decision variable $y_u(m, n)$, which can be used to assess the number of paths passing through the link, to calculate aggregated traffic load and transmission delay. Constraint (4) states that if a link (m, n) is on a path p, the link is used by a node u.

$$\sum_{p \in P_{uv}} x_p \delta_{p(m,n)} \leq y_{u(m,n)}, \quad \forall (m, n) \in L, u \in U, v \in V \tag{4}$$

2) Return Result Path Constraints:

Once the process requirement is offloaded to node u, a path routes to end-user v must be identified. Constraint (5) states that a path must exist for node v to send the result to an end-user.

$$\alpha_{uvi} \leq \sum_{q \in Q_v} x_q, \forall v \in V, u \in U, i \in I_u \tag{5}$$

Once path q is determined from nodes u to v, the set of links on path q should be recorded. Decision variable $y_{v(m,n)}$ is used to mark whether a link (m, n) is used. Constraint (6) states that if the link (m, n) is on path q, the link is used by node v.

$$\sum_{q \in Q_v} x_q \delta_{q(m,n)} \leq y_{v(m,n)}, \forall (m, n) \in L, v \in V \tag{6}$$

3) Aggregate Flow Constraints:

After node u offloads the process to node v, the aggregated load is calculated and Constraint (7) constrains the aggregate computation on node v. The "less than and equal" symbols is used in this equation because the Lagrangian approach [39] is adopted to solve the minimum problem.

$$\sum_{u \in U} \alpha_{uvi} \beta_u \leq g_v, \forall v \in V, i \in I_u \tag{7}$$

Constraint (8) constrains the aggregate flow on the link (m, n) that is the sum of all paths transmitted through the link (m, n), including the source node that offloads the traffic load to in-network processing nodes and the offloading nodes sending the processed results to the end-user.

$$\sum_{u \in U} y_{u(m,n)} \gamma_u + \sum_{v \in V} y_{v(m,n)} \omega_v \leq f_{(m,n)}, \quad \forall (m, n) \in L \tag{8}$$

4) Delay Constraints:

Constraint (9) constrains the computation delay of node v. The processing delay is calculated based on the $M/M/1$ queuing model because herein only one process unit is used for the in-network process host.

$$\frac{1}{C_v - g_v} \leq d_v, \forall v \in V \tag{9}$$

Constraint (10) constrains the transmission delay of the link (m, n). The delay is calculated using an $M/M/1$ queuing model because a single process is used to forward the data for each device. The traffic load used in this system is aggregated because the outbound traffic load is difficult to collect. The delay value is stored in the decision variable $d_{(m,n)}$.

$$\frac{1}{C_{(m,n)} - f_{(m,n)}} \leq d_{(m,n)}, \forall (m, n) \in L \tag{10}$$

Constraint (11) constrains the end-to-end transmission delay from node u to node v. According to this constraint, the transmission delay must be considered when is set to 1. If $y_{u(m,n)}$ is set to 0, M is a sufficiently large positive number for $d_{(m,n)}$ - M to be less than zero. However, the decision variable $\chi_{u(m,n)}$ is ultimately set to 0 because of the physical meaning assigned to it.

$$d_{(m,n)} - (1 - y_{u(m,n)})M \leq \chi_{u(m,n)}, \quad \forall (m, n) \in L, u \in U \tag{11}$$

Constraint (12) states that the transmission delay must be considered when both α_{uvi} and $y_{u(m,n)}$ are set to 1. M is a sufficiently large positive number under conditions that $d_{(m,n)}$ - $2M$ is less than zero. However, the decision variable $\lambda_{u(m,n)}$ is ultimately set to 0 because of the physical meaning assigned to it.

$$d_{(m,n)} - (2 - \alpha_{uvi} - y_{v(m,n)})M \leq \lambda_{u(m,n)}, \quad \forall (m, n) \in L, v \in V, u \in U, i \in I_u \tag{12}$$

Constraint (13) states that the computation delay must be considered when α_{uvi} is set to 1. If α_{uvi} is set to 0, M is a sufficiently large positive number for d_v - M to be less than zero. However, the decision variable ψ_u is ultimately set to 0 because of the physical meaning assigned to it.

$$d_v - (1 - \alpha_{uvi})M \leq \psi_u, \quad \forall v \in V, u \in U \tag{13}$$

5) End-to-end Delay Constraints:

According to the link and nodal delay, Constraint (14) constrains the end-to-end delay with all links and nodes used for the given detected node u. At least two paths and one in-network node are used to process an event. The total delay is recorded on the decision variable u.

$$\sum_{(m,n)\in L} \chi_{u(m,n)} + \lambda_{u(m,n)} + \psi_u \leq d_u, \quad \forall u \in U \tag{14}$$

The aim is to minimize the maximal delay gap between the tolerable delay and the actual transmission delay to achieve weighted fairness and satisfy the required delay for each detected event. Constraint (15) states that the delay must be maximized and the results d are maximized by the objective function (IP). The division of the gap by T_u does not affect the results. The normalization delay gap results are calculated to ensure the ratios are set in [0,1] to fairly compare the QoS.

$$d \leq \frac{(T_u - d_u)}{T_u}, \forall u \in U \tag{15}$$

IV. SOLUTION APPROACH

Lagrangian relaxation (LR) can be used to solve constrained optimization problems, such as integer programming, linear programming with a combinatorial objective function, and nonlinear programming. The main concept of LR is relaxing complicated constraints of primal problems and moving them into objective functions. A corresponding Lagrangian multiplier exists for each relaxed constraint. The primal problem is then transformed into an LR problem, and the LR problem is divided into several subproblems [39], as illustrated in the Appendix. An optimization-based algorithm is employed to solve each subproblem.

Relaxing the complicated constraints reduces the complexity and difficulty of primal problems. For minimization problems, the optimal value of the LR problem is a lower bound of the primal problem. The subgradient method is used to adjust the Lagrangian multiplier iteration by iteration to reduce the gap between the primal problem and the LR problem; this is referred to as a Lagrangian dual problem.

To apply the LR method and decompose the problem into subproblems, the maximization problem must be transformed into a minimization problem. The objective function (IP) is transformed into

$$\min -d \tag{IP'}$$

The constraints to form the following LR problem are relaxed through the introduction of the Lagrangian nonnegative multiplier vectors $\mu^1_{uvi}, ..., \mu^{13}_u$.

Objective function:

$$\begin{aligned}
&Z_{LR}(\mu^1_{uvi}, \mu^2_{(m,n)uv}, \mu^3_{uvi}, \mu^4_{(m,n)v}, \mu^5_{vi}, \mu^6_{(m,n)},\\
&\mu^7_v, \mu^8_{(m,n)}, \mu^9_{(m,n)u}, \mu^{10}_{(m,n)uvi}, \mu^{11}_{uvi}, \mu^{12}_u, \mu^{13}_u) =\\
&min - d\\
&+ \sum_{u\in U}\sum_{v\in V}\sum_{i\in I_u} \mu^1_{uvi}[\alpha_{uvi} - \sum_{p\in P_{uv}} x_p]\\
&+ \sum_{(m,n)\in L}\sum_{u\in U}\sum_{v\in V} \mu^2_{(m,n)uv}[\sum_{p\in P_{uv}} x_p\delta_{p(m,n)} - y_{u(m,n)}]\\
&+ \sum_{u\in U}\sum_{v\in V}\sum_{i\in I_u} \mu^3_{uvi}[\alpha_{uvi} - \sum_{q\in Q_v} x_q]\\
&+ \sum_{(m,n)\in L}\sum_{v\in V} \mu^4_{(m,n)v}[\sum_{q\in Q_v} x_q\delta_{q(m,n)} - y_{v(m,n)}]\\
&+ \sum_{v\in V}\sum_{i\in I_u} \mu^5_{vi}[\sum_{u\in U} \alpha_{uvi}\beta_u - g_v]\\
&+ \sum_{(m,n)\in L} \mu^6_{(m,n)}[\sum_{u\in U} y_{u(m,n)}\gamma_u + \sum_{v\in V} y_{v(m,n)}\omega_v - f_{(m,n)}]\\
&+ \sum_{v\in V} \mu^7_v[\frac{1}{C_v - g_v} - d_v]\\
&+ \sum_{(m,n)\in L} \mu^8_{(m,n)}[\frac{1}{C_{(m,n)} - f_{(m,n)}} - d_{(m,n)}]\\
&+ \sum_{(m,n)\in L}\sum_{u\in U} \mu^9_{(m,n)u}[d_{(m,n)} - (1 - y_{u(m,n)})M - \mathcal{X}_{u(m,n)}]\\
&+ \sum_{(m,n)\in L}\sum_{u\in U}\sum_{v\in V}\sum_{i\in I_u} \mu^{10}_{(m,n)uvi}\\
&[d_{(m,n)} - (2 - \alpha_{uvi} - y_{v(m,n)})M - \lambda_{u(m,n)}]\\
&+ \sum_{u\in U}\sum_{v\in V}\sum_{i\in I_u} \mu^{11}_{uvi}[d_v - (1 - \alpha_{uvi})M - \psi_u]\\
&+ \sum_{u\in U} \mu^{12}_u[\sum_{(m,n)\in L} \mathcal{X}_{u(m,n)} + \lambda_{u(m,n)} + \psi_u - d_u]\\
&+ \sum_{u\in U} \mu^{13}_u[d - \frac{(T_u - d_u)}{T_u}]
\end{aligned} \tag{LR}$$

subject to (1) and (2).

The LR problem can be decomposed into 14 subproblems, and each subproblem can be solved optimally, as detailed in Appendix A. Once these subproblems have been solved, the useful information in the corresponding multipliers can be utilized. We used $\mu^5_{vi}, \mu^6_{(m,n)}, \mu^7_v$, and $\mu^8_{(m,n)}$ to design our algorithm to be able to obtain feasible solutions to the primal problem.

Step 1: Use Dijkstra's algorithm to determine the total transmission delay to each node. The link cost is $\frac{\mu^8_{(m,n)}}{C_{(m,n)} - \mu^6_{(m,n)} f_{(m,n)}}$.

Step 2: Sort the nodes with an appropriate function type that can process the task. The weight to be sorted is the total transmission delay to that node plus the computation delay. $\frac{\mu^7_v}{C_v - \mu^5_v g_v}$.

Step 3: Use Dijkstra's algorithm [29] to determine the total transmission delay to the core cloud. The link cost is $\frac{\mu^8_{(m,n)}}{C_{(m,n)}-\mu^6_{(m,n)}f_{(m,n)}}$.

We designed and implemented two algorithms for comparison. The First-Fit algorithm is used to offload the tasks generated by the source node to the first node that can perform the function type of that task.

Step 1: Use Dijkstra's algorithm to determine the shortest path to each node for a device, which detects an event for processing.

Step 2: Offload the task to the first node that can perform the function type of the task.

Step 3: Use Dijkstra's algorithm to calculate the shortest path to the core cloud.

Step 4: Send the result to the core cloud.

The All-Core algorithm is employed to offload all the tasks to the core cloud.

Step 1: Use Dijkstra's algorithm to identify the shortest path to the core cloud.

Step 2: Offload the task to the core cloud.

Theorem 1: *The LR algorithm can be evaluated using* $O(|\Gamma||L||T||V|^2)$.

Proof: Suppose that the time complexity of the event process is $O(|\Gamma|)$, which is dependent on the complexity of a detected event. Finding the shortest path with a centralized shortest-path algorithm for each pair [29] requires $O(|V|^2)$ to run with $|T|$, yielding $O(|\Gamma||L||T||V|^2)$, where $|L|$ is the number of affected links for calculating the coefficient of Lagrangian multipliers. Therefore, the time complexity in the least favorable situation is $O(|\Gamma||L||T||V|^2)$ as determined by the LR-based algorithm.

V. PERFORMANCE EVALUATION

We designed a series of experiments with various parameters to evaluate the solution quality of our primal feasible solution algorithm. Furthermore, we employed two algorithms, the All-Core and First-Fit algorithms, to assess the performance of our algorithm.

A. EXPERIMENTAL ENVIRONMENT

The computational experiment was performed in Python, and we used a Linux server as our running platform. Table 4 displays the experimental parameters. A series of experiments were performed with various parameters to observe the trend and performance of the LR-based algorithm proposed in this paper. We used two performance metrics to evaluate the solution quality, the 'Gap' and 'Improvement Ratio'. These metrics are defined as follows:

$$Gap = \frac{|Primal - LB|}{|Primal|} \times 100\% \tag{16}$$

$$ImprovementRatio = \frac{Algorithm_i - Algorithm_j}{Algorithm_j} \times 100\% \tag{17}$$

The experiment results, illustrated in FIGURE 2, indicate the effect of the number of source nodes on the objective value obtained using each algorithm. When the number of source nodes increased, the objective value decreased. The primal feasible solution algorithm maintained the objective value at approximately 0.904 as the number of source nodes increased. The All-Core algorithm displayed a more favorable performance than did the First-Fit algorithm. The objective value was affected by the number of source nodes in both algorithms. The Gap was maintained at approximately 9.6% when the number of source nodes increased. The highest Improvement Ratio for the All-Core algorithm was 9.95%, whereas it was 22.83% for the First-Fit algorithm.

TABLE 4. EXPERIMENTAL ENVIRONMENT AND PARAMETERS

Parameter	Description
Number of IP cameras	50 120
Number of Source Nodes	10 80
Number of Functions	2 9
Number of Cloudlets	2
Number of Core cloud data center	1
Bandwidth size (Mb/s)	100
Computation Cost of Low Resolution Video	1–10 ms
Computation Cost of High Resolution Video	200–700 ms
Transmission cost of Low Resolution Video	1–10 ms
Transmission cost of High Resolution Video	200–700 ms

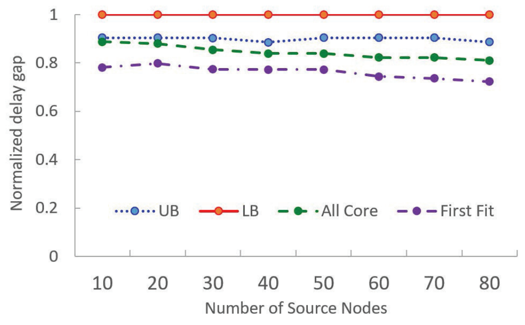

FIGURE 2. Normalized delay gap effect on the number of source nodes.

This study also determined the effect of the number of total nodes on the objective value obtained using each algorithm, as displayed in FIGURE 3. When the number of total nodes increased, the objective value was determined to increase slightly. The primal feasible solution algorithm maintained the objective value at approximately 0.86 as the number of total nodes increased. The All-Core algorithm exhibited superior performance to the First-Fit algorithm, especially for more than 90 nodes. Gap was maintained at approximately 12% as the number of source nodes increased. The highest Improvement Ratios achieved using the All-Core and First-Fit algorithms were 14.77% and 23.54%, respectively.

We investigated the effect of the size of data on the objective value obtained using each algorithm. The results are displayed in FIGURE 4. When the size of the data increased,

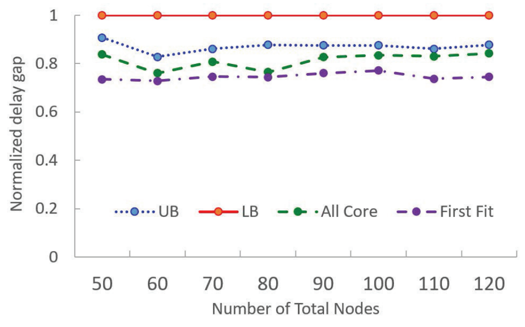

FIGURE 3. Normalized delay gap effect on the number of total nodes.

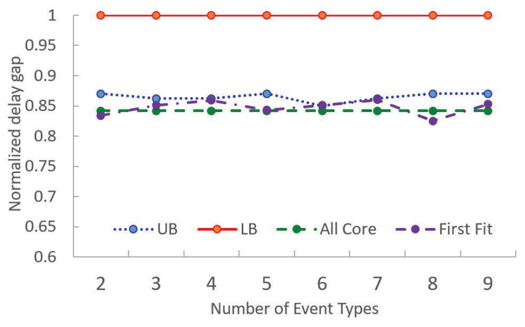

FIGURE 5. Normalized delay gap effect on the event types.

the objective value decreased. The primal feasible solution algorithm maintained the objective value at approximately 0.88 as the quantity of data increased. The objective values obtained using the All-Core and First-Fit algorithms both decreased considerably when the number of source nodes was larger than 50 and 60, respectively. Gap was maintained at approximately 12% as the number of source nodes increased. The highest Improvement Ratios obtained using the All-Core and First-Fit algorithms were 23.56% and 32.37%, respectively.

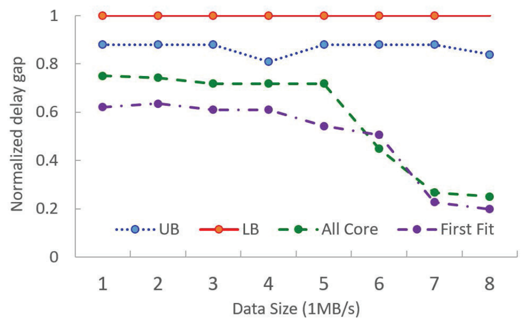

FIGURE 4. Normalized delay gap effect on the data size.

Furthermore, we investigated the effect of the number of functions on the objective value obtained using each algorithm. The results are illustrated in FIGURE 5. The primal feasible solution algorithm maintained the objective value at approximately 0.87 as the number of functions increased. The All-Core algorithm maintained the objective value at 0.842. However, the objective value obtained using the First-Fit algorithm was strongly affected by the number of functions. Gap was maintained at approximately 13% as the number of functions increased. The highest Improvement Ratios obtained using the All-Core and First-Fit algorithms were 3.33% and 5.45%, respectively.

B. DISCUSSION

This study aimed to apply its results to the real world, such as video surveillance monitor for in parking lots, which was built-up by the Bank of New Hampshire Pavilion in New England. However, the LR-based algorithm proposed herein requires a relatively long calculation time to obtain the optimal solution. Several advanced methods can be studied to address this problem. To use this method in real-time situations, we could record the computational result in the form of a phone book. When an IP camera is triggered, it could search the phone book with the information observed from the SDN controllers, which periodically collect the network and device traffic load, service status, storage, and host status, to implement a routing and offloading strategy. The second method would be to continually execute the converged LR-based algorithms with the update status to reduce the execution time because observing the results for each iteration and outputting the results at any iteration is a feature of the LR approach.

The SDN controller had been integrated with network management tools to collect the network, buffer, storage, CPU, and memory status of IoT devices (e.g., IP cameras) and cloudlet hosts. These data are used by network management to monitor the status of devices and cloudlet hosts so that real-time information is adopted in the proposed method to identify a feasible solution and periodically update the decision variables to support event-driven monitor applications in in-network processing within a tolerable delay.

The network architecture used to process event data with cloudlets forms a cluster-based network, as illustrated in FIGURE 1. The formation is event-centric, with shortest-path routing used to reduce the transmission time; hence, the range is narrowed to a cluster with a subset of IoT devices and cloudlet hosts. The network calculation range and the time used to collect hosts and network statuses are then reduced. Therefore, the decision variables, illustrated in Table 3, are periodically updated on each device, with the proposed LR-based method calculated using the cloudlet hosts. Any device can decide, based on calculation results, to complete an in-network multimedia processing operation when an event occurs.

The LR-based method solves a complex problem by using the principle of 'divide and conquer' [39]. In this method, the original (IP) problem is divided into 14 subproblems, as shown in Appendix A. Each subproblem can be divided into small independent subproblems, enabling a series of decentralized algorithms for these subproblems to be partially offloaded on various hosts, thereby accelerating the processing. The nodes of Subproblems 1, 6, 8, 12, and 13 are divided into |V| independent subproblems, which can be processed and distributed on each node. The results are then sent to the SDN controller. The linking and routing of Subproblems 2, 3, 4, 5, 7 9, 10, 11, and 14 require the statuses of all links. To avoid duplicate transmissions among the nodes, the calculations are performed by the SDN controller and cloudlet hosts. Although the overhead exchange message might be large, achieving the objective of this work to support QoS multimedia processing is more important. The overhead can be controlled by handling the event data within a given segment area. For example, Area I in FIGURE 1 is set as a solution range. Through these techniques, the proposed algorithm can be used in real-time applications with IP camera devices to obtain near-optimal solutions.

When an IP camera offloads various functions, it transmits a duplicate video. When it offloads the functions to the core cloud or a cloudlet, the video only requires one transmission. The problem can be viewed as a network planning problem and solved near-optimally.

The transmission delay is higher when the volume of data is larger. Recording video in high resolution also increases the computational delay. All-Core algorithms transmit tasks to the core cloud. The high computation power of the core cloud can result in a relatively low computational delay. However, the video must be transmitted through numerous links, causing the total transmission delay to be extremely high. Furthermore, the links between the IP cameras and cloudlets, and the links between the cloudlets and the core cloud, bottleneck. The First-Fit algorithm offloads the task generated by the source node to the nearest node that can perform the function type of the task. This algorithm can reduce the transmission delay because the video is transformed into a message that tells the end-user whether an IP camera detects a certain event. The size of the message is relatively small compared with that of a video. However, given the computational capacity of IP cameras, the computational delay is high, and the number of IP cameras that can perform a certain type of task may be small. Therefore, the IP cameras may have to perform numerous tasks, resulting in high computational delay.

VI. CONCLUSION

Routing and offloading in video surveillance that combine a core cloud, cloudlets, and IP cameras still face several problems. This study considered several elements, including the computational capacity of each device and host, the transmission capacity of each link, and, most crucially, the tolerable delay of each type of event. An LR-based approach was employed to solve these complex problems. A series of experiments using different parameters demonstrated that the proposed method outperformed the All-Core and First-Fit algorithms. This study modeled the entire system within an extremely small time. A long-term study on the applications in video surveillance is intended for future work.

REFERENCES

[1] Frank Y. S. Lin and Y. F. Wen, "Multi-sink Data Aggregation Routing and Scheduling with Dynamic Radii in WSNs," *IEEE Communications Letters*, vol. 10, no. 10, pp. 692–694, October 2006.

[2] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge Computing Framework for Cooperative Video Processing in Multimedia IoT Systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, October 2018.

[3] M. Razian, M. Fathian, and R. Buyya, "ARC: Anomaly-aware Robust Cloud-integrated IoT Service Composition Based on Uncertainty in Advertised Quality of Service Values," *Journal of Systems and Software*, vol. 164, p. 110557, February 2020.

[4] L. Kong and R. Dai, "Object-detection-Based Video Compression for Wireless Surveillance Systems," *IEEE MultiMedia*, vol. 24, no. 2, pp. 76–85, May 2017.

[5] H. Guo, J. Liu, and H. Qin, "Collaborative Mobile Edge Computation Offloading for IoT over Fiber-wireless Networks," *IEEE Network*, vol. 32, no. 1, pp. 66–71, January 2018.

[6] S. K. Sharma and X. Wang, "Live Data Analytics With Collaborative Edge and Cloud Processing in Wireless IoT Networks," *IEEE Access*, vol. 5, pp. 4621–4635, March 2017.

[7] S. Taherizadeh, A. C. Jones, I. Taylor, Z. Zhao, and V. Stankovski, "Monitoring Self-adaptive Applications within Edge Computing Frameworks: A State-of-the-art Review," *Journal of Systems and Software*, vol. 136, pp. 19–38, February 2018.

[8] J. Chen, K. Li, Q. Deng, K. Li, and P. S. Yu, "Distributed Deep Learning Model for Intelligent Video Surveillance Systems with Edge Computing," *IEEE Transactions on Industrial Informatics, Early Access*, pp. 1–1, April 2019.

[9] E. Eriksson, G. Dan, and V. Fodor, "Coordinating Distributed Algorithms for Feature Extraction Offloading in Multi-Camera Visual Sensor Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3288–3299, November 2018.

[10] G. L. Foresti, "Object Recognition and Tracking for Remote Video Surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 7, pp. 1045–1062, October 1999.

[11] L. Hu and Q. Ni, "IoT-driven Automated Object Detection Algorithm for Urban Surveillance Systems in Smart Cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 747–754, April 2018.

[12] Y. Ye, S. Ci, A. K. Katsaggelos, Y. Liu, and Y. Qian, "Wireless Video Surveillance: A Survey," *IEEE Access*, vol. 1, pp. 646–660, September 2013.

[13] F. Lian, Y. Lin, C. Kuo, and J. Jean, "Voting-based Motion Estimation for Real-time Video Transmission in Networked Mobile Camera Systems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 172–180, July 2013.

[14] G. Gualdi, A. Prati, and R. Cucchiara, "Video Streaming for Mobile Video Surveillance," *IEEE Transactions on Multimedia*, vol. 10, no. 6, pp. 1142–1154, October 2008.

[15] S. Bhattacharjee, P. Roy, S. Ghosh, S. Misra, and M. S. Obaidat, "Wireless Sensor Network-based Fire Detection, Alarming, Monitoring and Prevention System for Bord-and-Pillar Coal Mines," *Journal of Systems and Software*, vol. 85, no. 3, pp. 571–581, March 2012.

[16] K. Zhang, Y. Huang, Y. Du, and L. Wang, "Facial Expression Recognition Based on Deep Evolutional Spatial-temporal Networks," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4193–4203, September 2017.

[17] S. L. Happy and A. Routray, "Automatic Facial Expression Recognition Using Features of Salient Facial Patches," *IEEE Transactions on Affective Computing*, vol. 6, no. 1, pp. 1–12, December 2015.

[18] K. Zhang, K. Yang, S. Li, D. Jing, and H. B. Chen, "ANN-based Outlier Detection for Wireless Sensor Networks in Smart Buildings," *IEEE Access*, vol. 7, pp. 95 987–95 997, July 2019.

[19] N. True, "Computer Vision Based Fire Detection," https://pdfs.semanticscholar.org/46f6/63bd1d32cdbec9b7ba206d471f94b9bbc03f.pdf.

[20] J. Zeng, M. Liu, X. Fu, R. Gu, and L. Leng, "Curvature Bag of Words Model for Shape Recognition," *IEEE Access*, vol. 7, pp. 57 163–57 171, April 2019.

[21] F. Zeng, Y. Ji, and M. D. Levine, "Contextual Bag-of-words for Robust Visual Tracking," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1433–1447, March 2018.

[22] A. N. F. Ahmad and Z. Ahmed, "Image-based Face Detection and Recognition: "State of the Art"," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 6, pp. 169–172, February 2013.

[23] E. Bermejo, O. Deniz, G. Bueno, and R. Sukthankar, "Violence Detection in Video Using Computer Vision Techniques," in *14th International Conference on Computer Analysis of Images and Patterns*, August 2011, pp. 332–339.

[24] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation Offloading and Resource Allocation for Low-power IoT Edge Devices," in *IEEE World Forum on Internet of Things (WF-IoT)*, December 2016, pp. 7–12.

[25] N. Chen, Y. Chen, Y. You, H. Ling, P. Liang, and R. Zimmermann, "Dynamic Urban Surveillance Video Stream Processing Using Fog Computing," in *IEEE Second International Conference on Multimedia Big Data (BigMM)*, April 2016, pp. 105–112.

[26] N. Chen, Y. Chen, E. Blasch, H. Ling, Y. You, and X. Ye, "Enabling Smart Urban Surveillance at The Edge," in *IEEE International Conference on Smart Cloud (SmartCloud)*, November 2017, pp. 109–119.

[27] P. Gorur and B. Amrutur, "Skip Decision and Reference Frame Selection for Low-complexity H.264/AVC Surveillance Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 7, pp. 1156–1169, July 2014.

[28] S. Javaid, H. Afzal, M. Babar, F. Arif, Z. Tan, and M. A. Jan, "ARCA-IoT: An Attack-resilient Cloud-assisted IoT System," *IEEE Access*, vol. 7, pp. 19 616–19 630, February 2019.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press., 2001.

[30] Y. F. Wen and J. C. Shen, "Load-balancing Metrics: Comparison for Infrastructure-based Wireless Networks," *Computers & Electrical Engineering*, vol. 40, no. 2, pp. 730–753, February 2014.

[31] T. Settawatcharawanit, Y. H. Chiang, V. Suppakitpaisarn, and Y. Ji, "A Computation-efficient Approach for Segment Routing Traffic Engineering," *IEEE Access*, vol. 7, pp. 160 408–160 417, November 2019.

[32] P. Guo, X. Liu, J. Cao, and S. Tang, "Lossless In-network Processing and Its Routing Design in Wireless Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 10, pp. 6528–6542, October 2017.

[33] P. Zhang, Y. Gang, X. Huang, S. Zeng, and K. Xie, "Bandwidth Allocation With Utility Maximization in the Hybrid Segment Routing Network," *IEEE Access*, vol. 7, pp. 85 253–85 261, June 2019.

[34] W. Zhang, W. Lei, and S. Zhang, "A Multipath Transport Scheme for Real-Time Multimedia Services Based on Software-Defined Networking and Segment Routing," *IEEE Access*, vol. 8, pp. 93 962–93 977, May 2020.

[35] A. Yazici, M. Koyuncu, S. A. Sert, and T. Yilmaz, "A Fusion-based Framework for Wireless Multimedia Sensor Networks in Surveillance Applications," *IEEE Access*, vol. 7, pp. 88 418–88 434, July 2019.

[36] Z. Sun, L. Wei, C. Xu, T. Wang, Y. Nie, X. Xing, and J. Lu, "An Energy-Efficient Cross-layer-sensing Clustering Method Based on Intelligent Fog Computing in WSNs," *IEEE Access*, vol. 7, pp. 144 165–144 177, October 2019.

[37] D. Westhoff, J. Girao, and M. Acharya, "Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution, and Routing Adaptation," *IEEE Transactions on Mobile Computing*, vol. 5, no. 10, pp. 1417–1431, October 2006.

[38] J. Tang, R. Yu, S. Liu, and J. L. Gaudiot, "A Container Based Edge Offloading Framework for Autonomous Driving," *IEEE Access*, vol. 8, pp. 33 713–33 726, February 2020.

[39] M. L. Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science*, vol. 50, no. 12, pp. 1861–1871, December 2004.

.

APPENDIX A

The solutions of 14 subproblems separate from the LR problem are represented in this section. Each subproblem must be optimally solved. The summation of these objective results is the LB value, which is the final result of the dual problem.

A. SUBPROBLEM 1 (RELATED TO DECISION VARIABLE α_{uvi}):

Objective function:

$$\min \sum_{i\in I_u}\sum_{u\in U}\sum_{v\in V}\Big(\mu^1_{uvi}+\mu^3_{uvi}+\mu^5_{vi}\beta_u$$

$$-\Big(\sum_{(m,n)\in L}\mu^{10}_{(m,n)uvi}+\mu^{11}_{uvi}\Big)M\Big)\alpha_{uvi}$$

subject to (1) and (2).

Subproblem 1 can be divided into $|U||V|$ independent subproblems. Each node u that needs to offload an event to another node is recorded. Both the index and coefficient of the candidate node v, which can handle the event, are calculated to determine the minimum value of $\mu^1_{uvi}+\mu^3_{uvi}+\mu^5_{vi}\beta_u-\big(\sum_{(m,n)\in L}\mu^{10}_{(m,n)uvi}+\mu^{11}_{uvi}\big)M$. For the candidate that can process the event, α_{uvi} is simply set to 0. To determine the optimal value, because an event can be handled by only one candidate node, all candidate nodes that have been recorded are tested, and the candidate with the minimum α_{uvi} is selected. The pseudocode of the algorithm is as follows:

```
1:  For each node $u$:
2:   For the event $i$:
3:    For each node $v$:
4:     If $h_{iv}$ = 1:
5:      Calculate the coefficient of $\alpha_{uvi}$->a_cof[v]
6:     Else:
7:      a_cof[v] = infinity
8:      Sort a_cof
9:      Select the minimum value of a_coef;
10:     record the node and the value
11:   For each node $v$:
12:    If $v$ == the node with the minimum value of a_coef
       < 0:
13:     Set $\alpha_{uvi}$ to 1
14:    Else:
15:     Set $\alpha_{uvi}$ to 0
```

B. SUBPROBLEM 2 (RELATED TO DECISION VARIABLE x_p):

Objective function:

$$\min \sum_{u\in U}\sum_{v\in V}\sum_{p\in P_{uv}}\Big(\sum_{(m,n)\in L}\mu^2_{(m,n)uv}\delta_{p(m,n)}-\sum_{i\in I_u}\mu^1_{uvi}\Big)x_p$$

subject to:

$$\sum_{u\in U}\sum_{v\in V}\sum_{p\in P_{uv}}x_p\le 1.$$

Subproblem 2 can be divided into $|U||V|$ subproblems. For each node u that must offload a function to another node v, the shortest path is computed using the Dijkstra's algorithm for all paths with link costs $\mu^2_{(m,n)uv}$. The pseudocode of the algorithm is as follows:

C. SUBPROBLEM 3 (RELATED TO DECISION VARIABLE $y_{u(m,n)}$):

Objective function:

$$\min \sum_{u\in U}\sum_{(m,n)\in L}\Big(\sum_{v\in V}(-\mu^2_{(m,n)uv})-\mu^9_{(m,n)u}M$$

```
1: For each node $u$:
2:   For each node $v$:
3:     Find the shortest path $x_p$ by using Dijkstra's algorithm.
4:     Calculate the coefficient of $x_p$.
5:   Select the path $p$ with the minimum aggregated link coefficient along path plus $\sum_{i\in I_u} \mu^1_{uvi}$.
```

$$+\mu^6_{(m,n)}\gamma_u\Big) y_{u(m,n)}$$

subject to

$$y_{u(m,n)} \le 1.$$

Subproblem 3 can be divided into $|V||L|$ independent subproblems. The decision variable $y_{u(m,n)}$ is set to 1 if the coefficient is less than 0 in each subproblem; otherwise, $y_{u(m,n)}$ is set to 0. All objective values of all divided subproblems are then aggregated. The pseudocode of the algorithm as follows:

```
1: For each node $u$:
2:   For each link $(m,n)$:
3:     Calculate the coefficient of $y_{u(m,n)}$ -> y_cof[(m,n)]
4:     If y_cof[(m, n)] < 0:
5:       Set $y_{u(m,n)}$ to 1
6:     Else:
7:       Set $y_{u(m,n)}$ to 0
```

D. SUBPROBLEM 4 (RELATED TO DECISION VARIABLE x_q):

Objective function:

$$\min \sum_{v\in V}\sum_{q\in Q_v}\Big(-\sum_{i\in I_u}\sum_{u\in U}\mu^3_{uvi}+\sum_{(m,n)\in L}\mu^4_{(m,n)v}\delta_{q(m,n)}\Big)x_q$$

subject to

$$\sum_{v\in V}\sum_{q\in Q_v} x_q \le 1.$$

This subproblem can be divided into $|V|$ independent subproblems. For each node u that must offload a function to another node v, the shortest path is calculated using the Dijkstra's algorithm for all paths to all candidate nodes with link costs $\mu^4_{(m,n)v}$. The path with the aggregated link coefficient $\mu^4_{(m,n)v}$ along path q minus $\sum_{u\in U}\sum_{i\in I_u}\mu^3_{uvi}$ is selected. The pseudocode of the algorithm is as follows:

```
1: For each node $u$:
2:   For each node $v$:
3:     Find the shortest path $x_q$ by using the Dijkstra's algorithm.
4:     Calculate the coefficient of $x_q$
5:   Select the path with the minimum aggregated link coefficient $\mu^4_{(m,n)v}$ along path $q$ minus $\sum_{u\in U}\sum_{i\in I_u}\mu^3_{uvi}$.
```

E. SUBPROBLEM 5 (RELATED TO DECISION VARIABLE $y_{v(m,n)}$):

Objective function:

$$\min \sum_{(m,n)\in L}\sum_{v\in V}\Big(\mu^6_{(m,n)}\omega_{uv}-\sum_{u\in U}\sum_{i\in I_u}\mu^{10}_{(m,n)uvi}M$$
$$-\mu^4_{(m,n)v}\Big)y_{v(m,n)}$$

subject to

$$y_{v(m,n)} \le 1.$$

Subproblem 5 can be divided into $|V||L|$ independent subproblems. The decision variable $y_{v(m,n)}$ is set to 1 if the coefficient $\mu^6_{(m,n)}\omega_{uv}-\sum_{u\in U}\sum_{i\in I_u}\mu^{10}_{(m,n)uvi}M-\mu^4_{(m,n)v}$ is less than 0 in each subproblem; otherwise, $y_{v(m,n)}$ is set to 0. Accordingly, the objective value of each subproblem is minimized so that the objective function of the subproblem 5 is minimized. The pseudocode of the algorithm is as follows:

```
1: For each node $v$:
2:   For each link $(m,n)$:
3:     Calculate the coefficient $\mu^6_{(m,n)}\omega_{uv}-\sum_{u\in U}\sum_{i\in I_u}\mu^{10}_{(m,n)uvi}M-\mu^4_{(m,n)v}$ of $y_{v(m,n)}$ -> y_cof[m,n]
4:     If y_cof[m,n] < 0:
5:       Set $y_{v(m,n)}$ to 1
6:     Else:
7:       Set $y_{v(m,n)}$ to 0
```

F. SUBPROBLEM 6 (RELATED TO DECISION VARIABLE g_v):

Objective function:

$$\min \sum_{v\in V}\Big(\mu^7_v\frac{1}{C_v-g_v}-\sum_{i\in I_u}\mu^5_{vi}g_v\Big)$$

subject to

$$\frac{1}{C_v-g_v}\le \max T_u.$$

Subproblem 6 can be divided into $|V|$ independent subproblems. By applying the derivative of $f(g_v)=\mu^7_v/(C_v-g_v)-\mu^5_v g_v$, the optimal value of g_v $(0\le g_v)$ can be obtained. The decision variable g_v is then set to $C_v-\mu^5_v/\mu^7_v$, 0, or $(1-\max T_u\times C_v)/\max T_u$, which can lead to the minimum value. The pseudocode is as follows:

G. SUBPROBLEM 7 (RELATED TO DECISION VARIABLE $f_{(m,n)}$):

Objective function:

$$\min \sum_{(m,n)\in L}\Big(\mu^8_{(m,n)}\frac{1}{C_{(m,n)}-f_{(m,n)}}-\mu^6_{(m,n)}f_{(m,n)}\Big)$$

```
1: For each node $v$:
2:   $f(g_v) = \frac{\mu_v^7}{C_v - g_v} - \mu_v^5 g_v$
3:   Calculate $f'(g_v) = 0$
4:   $\mu_v^7 (C_v - g_v)^{-2} - \mu_v^5 = 0$
5:   $g_v = C_v - (\mu_v^5/\mu_v^7)^2$
```

subject to

$$\frac{1}{C_{(m,n)} - f_{(m,n)}} \le \max T_u.$$

Subproblem 7 can be divided into $|L|$ subproblems. By applying the derivative of $h(f_{(m,n)}) = \mu^8_{(m,n)} \frac{1}{C_{(m,n)} - f_{(m,n)}} - \mu^6_{(m,n)} f_{(m,n)}$, the optimal value of $f_{(m,n)}$ can be obtained. The decision variable $f_{(m,n)}$ is then set to 0, $C_{(m,n)} - (\mu^6_{(m,n)} - \mu^8_{(m,n)})^2$, or $(1 - \max T_u \times C_{(m,n)}) / \max T_u$, which can lead to the minimum value. The pseudocode is as follows:

```
1: For each link $(m,n)$:
2:   $h(f_{(m,n)}) = \mu^8_{(m,n)} \frac{1}{C_{(m,n)} - f_{(m,n)}} - \mu^6_{(m,n)} f_{(m,n)}$
3:   Calculate $h'(f_{(m,n)}) = 0$
4:   $\mu^8_{(m,n)} (C_{(m,n)} - f_{(m,n)})^{-2} - \mu^6_{(m,n)} = 0$
5:   $f_{(m,n)} = C_{(m,n)} - (\mu^6_{(m,n)}/\mu^8_{(m,n)})^2$
```

H. SUBPROBLEM 8 (RELATED TO DECISION VARIABLE d_v):

Objective function:

$$\min \sum_{v \in L} \Big(-\mu_v^7 + \sum_{i \in I_u} \sum_{u \in U} \mu^{11}_{uvi} \Big) d_v$$

subject to

$$0 \le d_v \le \max T_u.$$

Subproblem 8 can be divided into $|V|$ independent subproblems. For each node v that must offload data to another node, the process delay is calculated to the candidate nodes with coefficient $-\mu_v^7 + \sum_{i \in I_u} \sum_{u \in U} \mu^{11}_{uvi}$. If the coefficient value is less than 0, the decision variable d_v is set to maximum T_u, otherwise, d_v = 0. Accordingly, the objective value of each subproblem is minimized. The pseudocode is as follows:

```
1: For each node $v$:
2:   Calculate the coefficient of $d_v$
3:   If the coefficient of $d_v < 0$:
4:     Set $d_v$ to maximize $T_u$
5:   Else:
6:     Set $d_v$ to 0
```

I. SUBPROBLEM 9 (RELATED TO DECISION VARIABLE $d_{(m,n)}$):

Objective function:

$$\min \sum_{(m,n) \in L} \Big(\sum_{u \in U} \mu^9_{(m,n)u} + \sum_{i \in I_u} \sum_{v \in V} \mu^{10}_{(m,n)uvi} - \mu^8_{(m,n)} \Big) d_{(m,n)}$$

subject to

$$0 \le d_{(m,n)} \le \max T_u.$$

Subproblem 9 can be divided into $|L|$ subproblems. The transmission time required to transmit data from nodes m to n for each link (m,n) is calculated using the link coefficient. The objective value is minimized such that the maximum possible value is set if the coefficient is less than 0, and otherwise, the decision variable $d_{(m,n)}$ is set to 0. The pseudocode of the algorithm is as follows:

```
1: For each link $(m,n)$:
2:   Calculate the coefficient of $d_{(m,n)}$
3:   If the coefficient of $d_{(m,n)} < 0$:
4:     Set $d_{(m,n)}$ to maximize $T_u$
5:   Else:
6:     Set $d_{(m,n)}$ to 0
```

J. SUBPROBLEM 10 (RELATED TO DECISION VARIABLE $\mathcal{X}_{u(m,n)}$):

Objective function:

$$\min \sum_{(m,n) \in L} \sum_{u \in U} (\mu_u^{12} - \mu^9_{(m,n)u}) \mathcal{X}_{u(m,n)}$$

subject to

$$0 \le \mathcal{X}_{u(m,n)} \le T_u.$$

Subproblem 10 can be divided into $|V||L|$ subproblems. The time required to transmit data from node u to node v for each link (m,n) is calculated using the coefficient $\mu_u^{12} - \mu^9_{(m,n)u}$. The decision variable value is set to T_u if the coefficient is less than 0, otherwise, the decision variable $\mathcal{X}_{u(m,n)}$ is set to 0. Then, the minimum aggregation time from node u to offloading node v is calculated using a greedy routing method, such as Dijkstra's algorithm. The decision variable $\mathcal{X}_{u(m,n)}$ is set along the path to observe the minimum objective value for this subproblem. The pseudocode of the algorithm is as follows:

K. SUBPROBLEM 11 (RELATED TO DECISION VARIABLE $\lambda_{u(m,n)}$):

Objective function:

$$\min \sum_{(m,n) \in L} \sum_{u \in U} \Big(\mu_u^{12} - \sum_{i \in I_u} \sum_{v \in V} \mu^{10}_{(m,n)uvi} \Big) \lambda_{u(m,n)}$$

subject to

$$0 \le \lambda_{u(m,n)} \le T_u.$$

```
1: For each node $u$:
2:   For each link $(m,n)$:
3:     Calculate the coefficient $\mu_u^{12} - \mu_{(m,n)u}^{9}$ of $\mathcal{X}_{u(m,n)}$
4:     If the coefficient of $\mathcal{X}_{u(m,n)} < 0$:
5:       Set $\mathcal{X}_{u(m,n)}$ to $T_u$
6:     Else:
7:       Set $\mathcal{X}_{u(m,n)}$ to 0
```

Subproblem 11 can be divided into $|V||L|$ subproblems. The time requires to transmit data from nodes m to n for each link (m,n) is calculated using the link coefficient $\mu_u^{12} - \sum_{i\in I_u}\sum_{v\in V}\mu_{(m,n)uvi}^{10}$. The decision variable value is set to T_u if the coefficient is less than 0, otherwise, the decision variable $\lambda_{u(m,n)}$ is set to 0. Then, the minimum aggregation time from node u to end user is calculated using a greedy routing method, such as Dijkstra's algorithm. The decision variable $\lambda_{u(m,n)}$ is set along the path to observe the minimum objective value for this subproblem. The pseudocode of the algorithm is as follows:

```
1: For each node $u$:
2:   For each link $(m,n)$:
3:     Calculate the coefficient of $\lambda_{u(m,n)}$
4:     If the coefficient $\mu_u^{12} - \sum_{i\in I_u}\sum_{v\in V}\mu_{(m,n)uvi}^{10}$ of $\lambda_{u(m,n)}$
   $< 0$:
5:       Set $\lambda_{u(m,n)}$ to $T_u$
6:     Else:
7:       Set $\lambda_{u(m,n)}$ to 0
```

L. SUBPROBLEM 12 (RELATED TO DECISION VARIABLE ψ_u):

Objective function:

$$\min \sum_{u\in U}\Big(\mu_u^{12} - \sum_{i\in I_u}\sum_{v\in V}\mu_{uvi}^{11}\Big)\psi_u$$

subject to

$$0 \le \psi_u \le T_u.$$

Subproblem 12 can be divided into $|V|$ subproblems. The time required for node u to process data is calculated using the coefficient $\mu_u^{12} - \sum_{i\in I_u}\sum_{v\in V}\mu_{uvi}^{11}$. The objective value is minimized if the coefficient is less than 0 and the maximum possible value is set, otherwise, the decision variable ψ_u is set to 0. The pseudocode of the algorithm is as follows:

```
1: For each node u:
2:   Calculate the coefficient of $\psi_u$
3:   If the coefficient $\mu_u^{12} - \sum_{i\in I_u}\sum_{v\in V}\mu_{uvi}^{11}$ of $\psi_u < 0$:
4:     Set $\psi_u$ to $T_u$
5:   Else:
6:     Set $\psi_u$ to 0
```

M. SUBPROBLEM 13 (RELATED TO DECISION VARIABLE d_u):

Objective function:

$$\min \sum_{u\in U}\Big(\frac{\mu_u^{13}}{T_u} - \mu_u^{12}\Big)d_u$$

subject to

$$0 \le d_u \le T_u.$$

Subproblem 13 can be divided into $|V|$ subproblems. The process time for variable d_u requires handling data on node u. The computation delay is calculated using coefficient $\mu_u^{13}/T_u - \mu_u^{12}$. The objective value of this subproblem is minimized if the coefficient is less than 0 and the maximum possible value is set; otherwise, the decision variable d_u is set to 0. The pseudocode of the algorithm is as follows:

```
1: For each node $u$:
2:   Calculate the coefficient $\mu_u^{13}/T_u - \mu_u^{12}$ of $d_u$
3:   If the coefficient of $d_u < 0$:
4:     Set $d_u$ to $T_u$
5:   Else:
6:     Set $d_u$ to 0
```

N. SUBPROBLEM 14 (RELATED TO DECISION VARIABLE d):

Objective function:

$$\min \Big(\sum_{u\in U}\mu_u^{13} - 1\Big)d$$

subject to

$$0 \le d \le 1.$$

If the coefficient $\sum_{u\in U}\mu_u^{13} - 1$ is less than 0, d is set to the maximum value, otherwise, it is set to 0. Accordingly, the maximum objective value for this subproblem and dual problem can be observed.

```
1: Calculate the coefficient $\sum_{u\in U}\mu_u^{13} - 1$ of $d$.
2: If the coefficient of d is less than zero:
3:   Set d to 1
4: Else:
5:   Set $d$ to 0.
```