Adaptive Offloading for Time-Critical Tasks in Heterogeneous Internet of Vehicles

Chunhui Liu, Kai Liu, *Senior Member, IEEE*, Songtao Guo, *Senior Member, IEEE*, Ruitao Xie, Victor C. S. Lee, *Member, IEEE*, and Sang H. Son, *Life Fellow, IEEE*

***Abstract*—With the recent development of wireless communication, sensing, and computing technologies, Internet of Vehicles (IoV) has attracted great attention in both academia and industry. Nevertheless, it is challenging to process time-critical tasks due to unique characteristics of IoV, including heterogeneous computation and communication capacities of network nodes, intermittent wireless connections, unevenly distributed workload, massive data transmission, intensive computation demands, and high mobility of vehicles. In this article, we propose a two-layer vehicular fog computing (VFC) architecture to explore the synergistic effect of the cloud, the static fog, and the mobile fog on processing time-critical tasks in IoV. Then, we give a motivational case study by implementing a prototype of a traffic abnormity detection and warning system, which demonstrates the necessity and urgency of developing adaptive task offloading mechanisms in such a scenario and gives insight into the problem formulation. Furthermore, we formulate the offloading model, aiming at maximizing the completion ratio of time-critical tasks. On this basis, we propose an adaptive task offloading algorithm (ATOA). Specifically, it adaptively categorizes all tasks into four types of pending lists by considering the dynamic requirements and resource constraints, and then tasks in each list will be cooperatively offloaded to different nodes based on their features. Finally, we build the simulation model and give a comprehensive performance evaluation. The results demonstrate the superiority of ATOA.**

***Index Terms*—Adaptive offloading, fog computing, Internet of Vehicles (IoV), time-critical task.**

Manuscript received February 28, 2020; revised May 4, 2020; accepted May 19, 2020. Date of publication May 26, 2020; date of current version September 15, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61872049, Grant 61802263, and Grant 61772432, in part by the Fundamental Research Funds for the Central Universities under Project 2020CDCGJ004, and in part by the Institute for Information and Communications Technology Promotion grant funded by the Korean Government (MSIP, Resilient Cyber-Physical Systems Research) under Grant 2014-0-00065. *(Corresponding author: Kai Liu.)*

Chunhui Liu, Kai Liu, and Songtao Guo are with the Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, and College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: chhliu0302@cqu.edu.cn; liukai0807@cqu.edu.cn; guosongtao@cqu.edu.cn).

Ruitao Xie is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: drtxie@gmail.com).

Victor C. S. Lee is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: csvlee@cityu.edu.hk).

Sang H. Son is with the Department of Information and Communication Engineering, Daegu Gyeongbuk Institute of Science and Technology, Daegu 42988, South Korea (e-mail: son@dgist.ac.kr).

Digital Object Identifier 10.1109/JIOT.2020.2997720

I. Introduction

INTERNET of Vehicles (IoV) is stepping into a new era with recent advances on wireless communications, sensing, and computation technologies. Dedicated short-range communication (DSRC) is regarded as one of the *de facto* communication technologies for IoV [1], which enables vehicles equipped with onboard units (OBUs) to communicate with both vehicles and roadside units (RSUs) via vehicle-to-vehicle (V2V) and vehicle-to-infrastructures (V2I) communications, respectively. Meanwhile, the rapid development of cellular networks and 5G technologies makes the long-term evolution-vehicle (LTE-V) and cellular vehicle-to-everything (C-V2X) become promising technologies to support large-scale, low-latency, and high-reliable communications for IoV [2]. Thus, it is envisioned that a variety of wireless communication interfaces with heterogeneous capacities will coexist in future IoV environments.

Meanwhile, various intelligent transportation systems (ITSs) impose great demands on massive data transmission and intensive task computation with stringent time constraints, such as high-resolution video crowdsourcing [3], traffic congestion detection, and collision warning [4]. Clearly, solely based on traditional cloud computing-based services cannot meet the real-time requirement of such applications due to the competition of limited wireless bandwidth and the high overhead of centralized scheduling.

Many researches have studied on the fog computing paradigm in IoV since vehicles and infrastructures at the edge of the network become more powerful with respect to computation, communication, and storage capabilities. Specifically, different types of vehicular fog computing (VFC) architectures have been designed for supporting low latency and high quality services in IoV [5]–[9]. Due to highly heterogeneous environments in IoV, as well as dynamic requirements of different ITS applications, a lot of studies focused on task offloading [10]–[13], resource management [14], [15], and data scheduling [16]–[19] problems in IoV. Distinguishing from previous studies, this article focuses on the task offloading via different wireless communication interfaces with heterogeneous capacities in IoV, and meanwhile, it incorporates vehicle mobility and time constraint of tasks into consideration.

The main challenges on task offloading in such an environment are summarized as follows. First, it is expected to best exploit the heterogeneous communication capacities in terms of radio coverage and data transmission rate to balance the workload on task offloading. Second, the heterogeneous computing and storage capacities of different nodes as well as the high mobility of vehicles make it nontrivial to select the most appropriate node for task offloading. Last but not least, the time-constraint of tasks and the highly dynamic service environment make it even more challenging on searching optimal solutions.

With above motivations, this article is dedicated to investigating on adaptive offloading for time-critical tasks in IoV. Specifically, a VFC-based architecture is presented to enable the cooperation among the cloud and fog nodes. Then, we implement a system prototype, which gives insight into the problem formulation. Furthermore, a task offloading model is formulated. On this basis, we propose an adaptive task offloading algorithm (ATOA) to maximize the task completion ratio (TCR). The main contributions of this article are outlined as follows.

1) We propose a two-layer VFC architecture, which consists of the cloud nodes, static fog nodes (SFNs), and mobile fog nodes (MFNs). The network nodes have different communication interfaces as well as heterogeneous computation, communication, and storage capacities. On the other hand, vehicles may submit tasks with deadlines, which are offloaded to different network nodes based on a certain scheduling algorithm. Furthermore, we implement a system prototype, which represents a typical time-critical application in IoV. The analysis based on the system prototype further motivates the necessity of adaptive task offloading in such a scenario and gives insight into the problem formulation.
2) We formulate an adaptive task offloading problem. Specifically, we first analyze and model four types of delays, including the transmission delay, waiting delay, computing delay, and task delay. On this basis, the diverse resources requirements, the heterogeneous capabilities, and the high mobility of vehicles are jointly considered to analyze task offloading procedures. Finally, the optimization problem is formulated with the deadline constraints of tasks, aiming at maximizing the TCR.
3) We propose an ATOA, which consists of three components. First, we design a delay-driven classification policy to divide tasks into two categories. Then, we design a resource-driven division policy to construct the four types of pending lists, which form the basis for searching valid candidate offloading nodes. Finally, we propose a deadline-driven offloading policy to collaboratively offload tasks from different lists to appropriate nodes so as to maximize overall TCR.

The remainder of this article is organized as follows. Section II reviews the related work. Section III proposes the system architecture. Section IV gives a motivational case study by implementing a system prototype, and Section V formulates the task offloading problem. Section VI proposes the ATOA. Section VII builds the simulation model and gives performance evaluation. Finally, we conclude this article and discuss future research directions in Section VIII.

II. Related Work

A great number of studies have investigated data dissemination problems in IoV. Considering a roadside-to-vehicle communication system, Liu *et al.* [16] proposed a heuristic scheduling algorithm to enhance the temporal data dissemination performance under different traffic scenarios and application requirements. The work [17] studied on cooperative scheduling for data dissemination via the hybrid of I2V and V2V communications, and proposed an online scheduling algorithm to enhance the performance. Targeting at minimizing both the service delay and the network access cost, Dai *et al.* [18] proposed a coding-assisted multiobject evolutionary algorithm to optimize the data broadcast efficiency and network interface selection. Ali *et al.* [19] investigated the effect of heterogeneous data sizes in network coding and proposed a dynamic threshold-based coding-assisted approach for serving on-demand real-time requests in accessing heterogeneous data items in IoV. Furthermore, based on different data rates supported by DSRC, they proposed an enhanced method named inverse of slack time multiply distance with THRESHOLD (Θ) (ISXD_{Θ}) to reduce system response time.

To support low-latency and high-reliability services, many studies have investigated on new architectures in IoV. Hou *et al.* [5] presented a VFC-based architecture, in which vehicles are considered as mobile infrastructures to improve the computation performance and mitigate the cost of additional infrastructure deployment. Wang *et al.* [6] proposed a novel collaborative vehicular-edge computing (VEC) framework, which supports more scalable vehicular services and applications via both horizontal and vertical collaborations. Ning *et al.* [7] constructed a three-layer VFC model, consisting of the cloud layer, cloudlet layer, and fog layer, to enable distributed real-time traffic management and minimize the response time. Furthermore, they summarized and highlighted research challenges and open issues toward VFC-enabled traffic management. Huang *et al.* [8] discussed the potential benefits, security, and forensic challenges in VFC. Furthermore, a fog-assisted traffic control system has been proposed as a use case, which is designed to deliver benefits, such as reducing road traffic congestion and car accidents, including two subsystems: one responsible for the local area and one responsible for the global area. Liu *et al.* [9] proposed a hierarchical system architecture for synthesizing the paradigms of software-defined networking and fog computing in IoV and exploiting their synergistic effects to support large-scale, real-time, and reliable information services.

There have been many studies on task offloading in IoV. Tang *et al.* [20] studied on the task offloading in the VEC system, where tasks are offloaded to VEC servers. They formulated the delay minimization problem as a temporal–spatial mixed-integer nonlinear programming problem. Furthermore, they divided the problem into two subproblems, including task placement and delayed offloading. Then, they developed a two-stage decision tree algorithm and a dynamic programming algorithm to solve the problems. Wu *et al.* [21] studied on the task offloading scheme in the 802.11p-based VFC system, where tasks are only offloaded to neighboring vehicles. They derived and analyzed the transmission delay and task arrival rate based on the 802.11p standard. Furthermore, they designed a semi-Markov decision process (SMDP) to formulate the task offloading problem and utilized an iteration algorithm to maximize the long-term reward of the system. Zhu *et al.* [22] studied on the latency and quality optimization for task offloading in the VFC system. Specifically, they designed a dynamic task allocation (DTA) solution, called Folo, to optimize service latency and quality loss of results (QLRs) under the application-specific requirements, e.g., communicating and computing demands. Furthermore, they used linear programming-based optimization (LBO) and binary particle swarm optimization (BPSO) to solve it.

Recently, there are a number of studies investigated on task offloading and resource management problems in IoV based on deep learning technologies. Ning *et al.* [23] studied on the energy-efficient computation offloading scheme in a three-layer architecture in IoV, which includes layers of cloudlet, RSUs, and fog nodes. They formulated the energy consumption minimization problem and divided it into two subproblems, including flow redirection for balancing the processing load among fog nodes and offloading decision for minimizing the average energy consumption of task offloading. Furthermore, they constructed an Edmonds–Karp algorithm-based scheme and a double deep *Q*-network (DDQN)-based algorithm to solve the problems. Qiao *et al.* [24] designed a novel edge caching framework based on the cooperation among base station, RSUs, and vehicles, and modeled the cooperative caching problem as a double time-scale Markov decision process (DTS-MDP), where the policy decisions of content placement/updating, vehicle scheduling, and bandwidth allocation occur on different time scale. The objective of cooperative caching is to achieve an optimal tradeoff between the total caching cost and the average content delivery latency. Furthermore, they proposed a DDPG-based cooperative caching scheme to solve the problem. Ning *et al.* [12] integrated a DRL method to solve the optimization of task scheduling and resource allocation in the VFC system. First, they modeled both the communication and computation states by finite Markov chains, and formulated the task scheduling and resource allocation strategy as a joint optimization problem to maximize users' Quality of Experience (QoE). Then, they divided the original problem into two suboptimization problems. In the first stage, they defined a utility function to quantize the level of QoE, and a two-sided matching scheme is proposed with the purpose of maximizing the total utilities. While in the second stage, the decision making of resource allocation is resolved by leveraging a DRL algorithm, whose target is to maximize the cumulative reward through obtaining the optimal policy.

Although there have been extensive studies on data scheduling algorithms, service architectures, task offloading, and resource management policies in IoV, as far as we know, this article makes the first efforts on considering adaptive offloading for time-critical tasks in a heterogeneous vehicular communication environment.

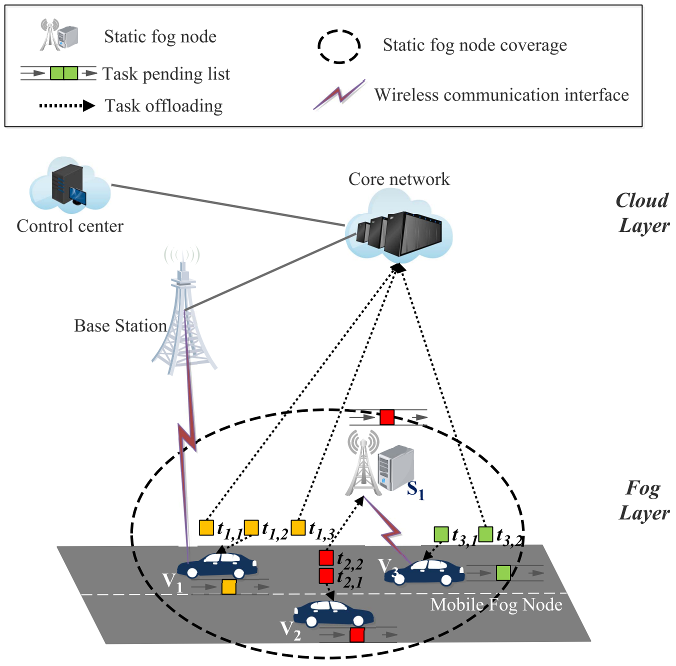


Fig. 1. Adaptive task offloading in a heterogeneous IoV.

III. System Architecture

A. System Overview

As shown in Fig. 1, the system architecture consists of two layers, namely, the cloud layer and the fog layer. In the cloud layer, the remote cloud servers are connected to base stations through the core network, and vehicles can offload tasks to the cloud server via vehicle-to-cloud (V2C) communication. In the fog layer, the static infrastructures, such as RSUs and 5G micro base stations are connected with each other via wired connections, which are regarded as the SFNs. Different SFNs may have different communication, computation, and storage capacities. Vehicles in the coverage of an SFN can offload their tasks to SFN via vehicle-to-infrastructure (V2I) communications. Also, vehicles may execute the tasks locally, which are considered as the MFNs.

In the following, we present an overview of the system's operational flow. First, the network nodes, including SFNs and cloud nodes, update their status to the control center via the core network periodically, including their available communication, computation, and storage resources. Meanwhile, vehicles/MFNs are also required to update their real-time status to the control center via the cellular network, including both physical (e.g., locations, velocities, heading directions, etc.) and cyber (e.g., submitted tasks, available resources, etc.) information. Second, the control center makes offloading decisions based on real-time system workloads and status. In particular, the tasks can be scheduled to be processed locally (i.e., MFNs) or to be offloaded to the SFNs or the cloud node. Finally, the control center notifies the offloading decisions to the corresponding network nodes via the control message. Then, the cloud node, SFNs, and MFNs collaboratively perform the task offloading operations.

As shown in Fig. 1, V_1, V_2, and V_3 are scheduled to process the task locally as MFNs. Outstanding tasks are queued in the task pending list and will be processed based on a first-come–first-served policy. Such a schedule can save the bandwidth for raw data transmission, but it may cause long task waiting delay of processing time due to the limited computation capability of terminals. Also, note that V_2 is scheduled to offload their tasks to S_1. First, the raw data have to be uploaded via V2I communication. Then, after the task is processed, the result should be returned to V_2. To be elaborated below, the vehicle mobility may have significant impacts to the final service performance in such a case. Finally, all vehicles are scheduled to offload other tasks to the cloud via V2C communication. We assume that the cloud server has infinite computation and storage capacity, and hence, the task can be processed immediately as long as data is uploaded. However, note that the competition of limited cellular network bandwidth may result in an excessively long time for raw data transmission.

B. System Characteristics

The presented two-layer VFC architecture is unique with respect to the following characteristics.

1) First, the architecture synthesizes the heterogeneity of both communication interfaces and computation resources in IoV into consideration for task offloading. Specifically, different communication interfaces may have different data transmission rates and radio coverages, which may have great impacts on the delay of raw data transmission. On the other hand, the different computation capacities of different network nodes will affect the task waiting time and processing delay. Therefore, it is critical to strike a balance on task offloading to maximize the efficiency of heterogeneous communication, computation, and storage resources in IoV.
2) In addition, the architecture pays particular attention to the effect of vehicle mobility on task offloading. First, considering vehicles may have short connections with each other due to high mobility, the tasks are only allowed to be processed locally when vehicles act as MFNs. Second, when offloading tasks to SFNs, the vehicles must be within the coverage of SFN during raw data transmission. Otherwise, it is a failed offloading. Finally, even the raw data could be transmitted to the SFN successfully, the vehicle may still leave the coverage of the SFN during task processing. Then, the cloud server will be cooperated to return the result to the corresponding vehicle. Clearly, extra delay cost will be modeled in such a case.
3) Last but not least, this architecture considers the offloading of time-critical tasks. Specifically, tasks are associated with different deadlines based on particular application requirements. Vehicles have to retrieve the task processing results before the stipulated deadline. Otherwise, the task is failed. Therefore, unlike conventional task offloading strategies in IoV, which mainly focused on minimizing average task processing time, the target of this service scenario is to complete as many tasks as possible before their deadlines. Clearly, the distinction of the objective makes the problem unique, and accordingly, novel solutions are expected to be developed.

TABLE I
TASK FEATURES

Vehicle	Task	θ	c	γ	d	φ
V_1	$t_{1,1}$	10	15	2.8	1	3
	$t_{1,2}$	10	10	2	1	
	$t_{1,3}$	10	12	2.7	1	
V_2	$t_{2,1}$	20	15	3	1	1.5
	$t_{2,2}$	10	20	2.7	1	
V_3	$t_{3,1}$	15	12	3	1	4
	$t_{3,2}$	10	10	3	1	

TABLE II
AVAILABLE RESOURCES

Nodes	C	f	R	τ
V_1	20	5	-	-
V_2	20	5	-	-
V_3	20	4	-	-
S_1	30	15	10	-
$Cloud$	-	30	14	8

C. Example

We give an example to further illustrate the presented system as well as to reveal the challenges of adaptive task scheduling. As shown in Fig. 1, there are three vehicles V_1, V_2, and V_3 within the coverage of the SFN S_1, and the tasks submitted by each vehicle are denoted by $t_{1,1}$, $t_{1,2}$, etc. Table I summarizes the task features, including the data size (θ), required computation resources (c), deadline (γ), and the result size (d). Assume that the three vehicles submit their tasks at time t_0. Furthermore, the remaining dwell time (φ) of each vehicle is also stated in Table I, which can be estimated based on existing state prediction technologies [25]. On the other hand, Table II shows the features of heterogeneous network nodes, including the storage capacity (C), available computation resource (f), and transmission rate (R) of different network nodes, as well as the transmission rate (τ) of the core network. Note that the transmission delay from the cloud server to vehicles includes both the core network transmission delay and the cellular network transmission delay, which is computed by $d/R_{\text{Cloud}} + d/\tau$.

Furthermore, due to vehicle mobility, the task offloading could not be completed if the vehicle has left the radio coverage of an SFN during raw data transmission. Therefore, from the statistics shown in Tables I and II, the data transmission time of offloading the task $t_{2,1}$ to S_1 is 2 time unit, while the remaining dwell time of V_2 in S_1 is 1.5 time unit. Thus, the task $t_{2,1}$ cannot be offloaded to S_1. Moreover, even if the raw data can be transmitted to an assigned SFN, vehicles may leave its coverage during the task computation period. In such a case, the result will be forwarded to the cloud node, and then it is delivered to the corresponding vehicle. As noted from this example, when offloading the task $t_{2,2}$ to S_1, its data transmission time and the computing delay are 1 time unit and 1.33 time unit, respectively, while the remaining dwell time of

TABLE III
COMPARISON OF DIFFERENT SCHEDULING STRATEGIES

Strategies	Offloading nodes						Task completion ratio
	V_1	V_2	V_3	S_1	*Cloud*	*Overdue*	
MFN-first	$t_{1,2}$	$t_{2,1}$	$t_{3,1}$	$t_{1,1}$, $t_{3,2}$	$t_{1,3}$	$t_{2,2}$	86%
Clou-first	-	$t_{2,1}$	$t_{3,1}$	$t_{1,2}$	$t_{1,1}$, $t_{1,3}$, $t_{3,2}$	$t_{2,2}$	86%
Optimal	$t_{1,2}$	$t_{2,1}$	$t_{3,1}$	$t_{2,2}$	$t_{1,1}$, $t_{1,3}$, $t_{3,2}$	-	100%

V_2 in S_1 is 1.5 time unit, which implies that V_2 leaves the radio coverage of S_1 before the computation completed. Thus, the result transmission time consists of the transmission time from S_1 to the cloud server (d/τ) and the transmission time from the cloud server to the corresponding vehicle ($d/R_{\text{Cloud}}+d/\tau$), which is $0.125 + 0.196 \approx 0.32$ time unit. Note that if V_2 is still in the coverage of S_1, the result transmission time would be $d/R_{S_1} = 0.1$ time unit. Clearly, the mobility of vehicles may affect system performance if the task could not be appropriately offloaded.

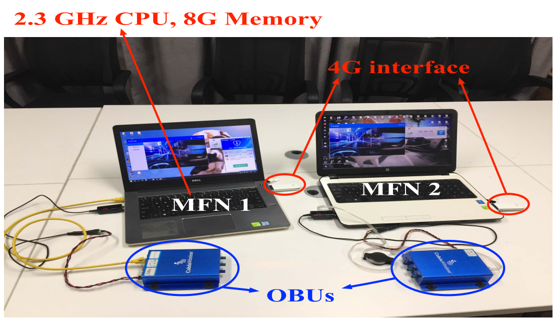
With the above knowledge, we compare three offloading strategies, which are shown in Table III. Specifically, the MFN-first (MF) strategy always prefers to offload the tasks to the local vehicle. First, given task deadlines and computation resource requirements, tasks $t_{1,2}$, $t_{2,1}$, and $t_{3,1}$ will be offloaded to the local MFN since they can be completed before the deadline. Then, $t_{1,1}$ and $t_{3,2}$ are offloaded to S_1, which results that task $t_{1,3}$ has to be offloaded to the cloud. Nevertheless, task $t_{2,2}$ will miss its deadline with such a schedule. Thus, the TCR of MF is 86%. The cloud-first (CF) strategy always prefers to offload the tasks to the cloud node. First, $t_{1,1}$, $t_{1,3}$, and $t_{3,2}$ can be completed in the cloud node. Then, only $t_{1,2}$ is offloaded to S_1 since the others cannot satisfy their deadline requirements. Finally, $t_{2,1}$ and $t_{3,1}$ can be completed in time by processing them locally. However, as observed, $t_{2,2}$ will miss its deadline with such a schedule. Thus, the TCR of CF is 86%. Finally, we may observe that the optimal solution is to offload $t_{1,1}$, $t_{1,3}$, and $t_{3,2}$ to the cloud node, and offload $t_{2,2}$ to S_1. Meanwhile, $t_{1,2}$, $t_{2,1}$, and $t_{3,1}$ are executed in the local MFNs. In such a schedule, all the tasks can be completed before their deadlines, given the 100% completion ratio.

IV. MOTIVATIONAL CASE STUDY

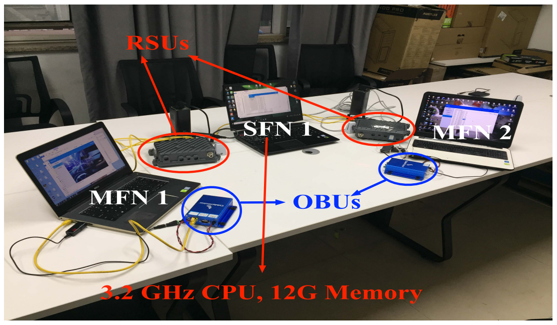
In this section, we implement the prototype of a traffic abnormity detection and warning (TAD&W) system, which represents a typical time-critical task offloading application in IoV. The observed results based on the system prototype further motivate the necessity of the investigated problem and gives insight into the problem formulation.

A. Prototype Implementation

The primary objective of the TAD&W system is to detect traffic abnormities based on real-time video recorded by a vehicle and send warning messages to other vehicles, so that relevant vehicles can respond in time to enhance driving safety and improve traffic efficiency. Clearly, such a system prototype involves massive data transmission (e.g., recorded video) and intensive computation (e.g., abnormity detection). Besides, the tasks have to be completed within a certain time constraint. Otherwise, it may compromise system performance or even cause serious consequences. Therefore, such a system prototype captures key features of our concerned application scenario, and hence it is suitable to be adopted as a motivational case study.



(a)



(b)

Fig. 2. System prototype. (a) Tasks are offloaded to MFN/Cloud. (b) Tasks are offloaded to SFN.

Fig. 2 shows the hardware-in-the-loop system prototype. In particular, Fig. 2(a) implements the system architecture in which tasks can be offloaded to either MFNs or the cloud. The configuration is described as follows. A cloud node is configured with 3.2-GHz CPU and 32-GB memory, and a notebook connected with an OBU is considered as a mobile fog node (MFN_1), which is configured with 2.3-GHz CPU and 8-GB memory. Meanwhile, another mobile fog node (MFN_2) is configured with 2.3-GHz CPU and 8-GB memory. Both MFN_1 and MFN_2 can communicate with the cloud server via the 4G interface, and they can communicate with each other via the DSRC interface. Fig. 2(b) shows the system architecture in which tasks can be offloaded to the SFN, and the configurations are described as follows. Two notebooks connected with two OBUs act as MFN_1 and MFN_2, which are configured with 2.3-GHz CPU and 8-GB memory, respectively. Meanwhile, another notebook connected with two RSUs, act as SFN_1 with the configuration of 3.2-GHz CPU and 12-GB memory. Both MFN_1 and MFN_2 can communicate with SFN_1 via the DSRC interface.

With the above configuration, the system operational flow is described as follows. MFN_1 generates videos with the frame rate of 10 frames/s. The task is to detect the traffic abnormity from the video frames. Once it is detected, the corresponding frame will be extracted and send to the other MFN together with warning messages. To implement the traffic abnormity

TABLE IV
DELAY COMPARISON OF DIFFERENT OFFLOADING POLICIES

Average delay (ms)	*MFN-based*	*SFN-based*	*cloud-based*
T_{comp}	247.67	208.60	184.30
T_{trans}	149.06	252.11	330.24
T_{task}	396.73	460.71	514.54

detection, we adopt the database from [26], in which there are 1750 videos, consisting of 620 positive samples and 1130 negative samples. Then, the MobileNet_SSD model [27] is adopted for training the samples. 903 selected video clips are selected as the training set and 166 clips are adopted as the testing set. The trained model is installed on the cloud node, the SFN, and the MFNs for abnormity detection based on different offloading modes.

Furthermore, we implement three task offloading modes.

1) For cloud-based offloading, MFN_1 transmits its raw data to the cloud node, and the cloud node executes the trained model based on received videos and transmitted the warning message to MFN_2.
2) For SFN-based offloading, the raw data are transmitted to SFN_1, where the abnormity detection is executed and the warning message is transmitted to MFN_2.
3) For MFN-based offloading, MFN_1 executes the abnormity detection locally and sends the warning message to MFN_2.

B. Observations

Table IV shows the average computing delay, transmission delay, and task delay of the three offloading policies. Note that the task delay is the sum of the computing and transmission delay. As noted, MFN-based offloading achieves the lowest average transmission delay of 149.06 ms, which is just around half of that achieved by cloud-based offloading (330.24 ms). On the other hand, cloud-based offloading has the shortest average computing delay of 184.30 ms, while the MFN-based offloading has the longest average computing delay, which is 247.67 ms. Furthermore, for the SFN-based offloading, its computing delay is short than that of MFN-based offloading, but higher than that of cloud-based offloading. Meanwhile, its transmission delay is shorter than that of cloud-based offloading, but higher than that of MFN-based offloading. Although it seems that SFN-based offloading strikes a balance in terms of computing and transmission, as noted from Table IV, actually, MFN-based offloading achieves the shortest task delay in this case, which implies that reducing the transmission delay dominates the system performance in this particular application.

From the above case study, we have two observations. First, simply prefer to offload tasks to MFNs and SFNs may not help to reduce the task delay, since computation overhead may dominate the overall performance. Therefore, it is expected to strike a balance among the cloud, the MFN and the SFN to enhance system performance. Second, although such a prototype has demonstrated the significance of task offloading in the proposed architecture, it has not yet incorporated the dynamic task requirements on communication, computation, and storage. Also, the mobility of MFNs as well as the intermittent wireless connection issues are not considered. Therefore, it is imperative to formulate an adaptive task offloading problem in such an architecture by jointly considering the above-discussed factors.

TABLE V
PRIMARY NOTATIONS

Notation	Summary
System Elements	
M	The set of MFNs
S	The set of SFNs
W_j	The task set of MFN v_j
w_{ji}	The i_{th} task of MFN v_j
Task Features	
θ_{ji}	The raw data size of w_{ji}
c_{ji}	The required computation resource of w_{ji}
d_{ji}	The result size of w_{ji}
γ_{ji}	The deadline of w_{ji}
Resource Characteristics	
f_n	The computing ability of node n
C_n	The storage capacity of node n
B_n	The bandwith of node n
ρ_n	The number of channels of node n
R_n	The data transmission rate of node n
τ	The transmission rate in the core network
Analytic Notations	
$\varphi_{ju}(t)$	The dwell time of v_j in an SFN r_u at time t
$x_{j,i,n}$	A binary variable, indicate whether the task w_{ji} is offloaded from the vehicle v_j to the node n
$t_{j,i,n}^{delay*}$	Delay of offloading task w_{ji} to the node n, where $delay* \in \{trans, data, result, comp, wait, task\}$
$t_{j,i}^{task}$	The service time for task w_{ji}
TCR	Task completion ratio

V. TASKS OFFLOADING MODEL

A. Definitions

The primary notations are summarized in Table V. Denote $M = \{v_1, v_2, \ldots, v_{|M|}\}$, $S = \{r_1, r_2, \ldots, r_{|S|}\}$, and $\{c\}$ as the set of MFNs, SFNs, and cloud node, respectively. Each MFN generates time-critical tasks. The set of tasks of $v_j \in M$ is denoted by $W_j = \{w_{j1}, w_{j2}, \ldots, w_{j|W_j|}\}$. Each task is atomic and associated with a four-tuple $<\theta_{ji}, c_{ji}, d_{ji}, \gamma_{ji}>$, where θ_{ji} means the size of data for task processing (e.g., execution code and input data), c_{ji} means the required computation resource, d_{ji} represents the output data size after being processed, and γ_{ji} represents the deadline. In addition, $\varphi_{ju}(t)$ means the dwell time of the vehicle (v_j) in the radio coverage of the SFN (r_u), which is computed by $\varphi_{ju}(t) = (Dis_{ju}(t)/Vel_j(t))$, where $Dis_{ju}(t)$ is the current distance to the exit of the service region and $Vel_j(t)$ is the current velocity of v_j [17].

Also, we assume that each node $n \in \{M \cup S \cup \{c\}\}$ is characterized by a four-tuple (B_n, ρ_n, C_n, f_n), where B_n means the total wireless bandwidth and ρ_n denotes the number of channels, while f_n represents the computation capability and C_n represents the storage capability. Furthermore, we give a binary variable $x_{j,i,n}$ indicating whether the task w_{ji} is offloaded from the vehicle v_j to the node n.

Then, we model the task processing procedures by defining and analyzing the delay composition. Specifically, *transmission delay* consists of the *data transmission time* and the *result transmission time*. The *data transmission time* is the duration transmitting all data to an offloading node. On the other hand, the *result transmission time* refers to the time consumption of sending results from the offloading node to the corresponding vehicles. The *waiting delay* is the time duration of task pended at the offloading node until it begins to be processed. The *computing delay* is the time for processing the task at the offloading node. Finally, the *task delay* is the duration from task generation to result received. We denote $t_{j,i,n}^{\text{delay}*}$ as different delays of offloading task w_{ji} to offloading node n, where $delay* \in \{trans, data, result, comp, wait, task\}$ and $n \in \{S \cup M \cup \{c\}\}$. With the above notations, the task delay for offloading w_{ji} to the offloading node n is computed by

$$t_{j,i,n}^{\text{task}} = t_{j,i,n}^{\text{data}} + t_{j,i,n}^{\text{wait}} + t_{j,i,n}^{\text{comp}} + t_{j,i,n}^{\text{result}}. \tag{1}$$

B. Analytic Model

With respect to computing delay and waiting delay, first, we consider the case that the task is offloaded locally. That is, given an MFN v_j, considering that tasks are processed one by one in v_j, the computing delay and the waiting delay can be formulated as

$$t_{j,i,v_j}^{\text{comp}} = \frac{c_{ji}}{f_{v_j}} \tag{2}$$

$$t_{j,i,v_j}^{\text{wait}} = \frac{\sum_{k=1}^{i-1} x_{j,k,v_j} \cdot c_{jk}}{f_{v_j}}. \tag{3}$$

Consider the task is offloaded to an SFN r_u, the computing delay and the waiting delay are formulated as

$$t_{j,i,r_u}^{\text{comp}} = \frac{c_{ji}}{f_{r_u}} \tag{4}$$

$$t_{j,i,r_u}^{\text{wait}} = \frac{\sum_{p=1}^{|M|} \sum_{q=1}^{|W_j|} x_{p,q,r_u}^{\text{before}_{ji}} \cdot c_{pq}}{f_{r_u}} \tag{5}$$

where $x_{p,q,r_u}^{\text{before}_{ji}}$ represents the tasks processed before w_{ji} in the pending list.

Considering that all tasks in the cloud node can be processed in parallel, and thus, we have $t_{j,i,c}^{\text{wait}} = 0$ when task w_{ji} is offloaded to the cloud node. Then, the computing delay is formulated as

$$t_{j,i,c}^{\text{comp}} = \frac{c_{ji}}{f_c}. \tag{6}$$

Meanwhile, the transmission delay is another key factor in task offloading. Specifically, when the task w_{ji} is scheduled to be processed locally, the data transmission time $t_{j,i,v_j}^{\text{data}} = 0$.

When offloading tasks to an SFN r_u, we consider that the total spectral bandwidth B_{r_u} is divided into ρ_{r_u} channels, and each task transmission only occupies a single channel. Thus, the maximum of the tasks, which can be offloaded to r_u at the same time, is ρ_{r_u}, and the data transmission rate is denoted by $R_{r_u} = B_{r_u}/\rho_{r_u}$. Accordingly, the data transmission time is formulated as

$$t_{j,i,r_u}^{\text{data}} = \frac{\theta_{ji}}{R_{r_u}}$$
$$\text{s.t. } t_{j,i,r_u}^{\text{data}} \le \varphi_{ju}(t) \tag{7}$$

where the constraint means the consideration of vehicle mobility, which stands that $t_{j,i,r_u}^{\text{data}}$ should be less than v_j's dwell time $\varphi_{ju}(t)$. Otherwise the transmission cannot be completed.

When considering offloading tasks to the cloud node, data are transmitted from v_j to the base-station through the cellular network, and then transmitted to the cloud node via the core network. Thus, the data transmission time is represented by

$$t_{j,i,c}^{\text{data}} = \frac{\theta_{ji}}{R_c} + \frac{\theta_{ji}}{\tau} \tag{8}$$

where τ and R_c represent the transmitting rate in the core network and the cellular network, respectively. Specifically, $R_c = B_c/\rho_c$, where B_c is the total spectral bandwidth of the cellular network and ρ_c is the number of channels.

For the result transmission time $t_{j,i,n}^{\text{result}}$, there are four cases given as follows.

1) The task w_{ji} is processed in the cloud, and results will be sent back via the core network and the cellular network. Thus, the result transmission time is formulated as

$$t_{j,i,c}^{\text{result}} = \frac{d_{ji}}{R_c} + \frac{d_{ji}}{\tau}. \tag{9}$$

2) The task is processed in an SFN r_u, and the vehicle is still within its coverage when the task is completed, so that r_u can transmit the results directly to the vehicle. Then, the result transmission time is formulated as

$$t_{j,i,r_u}^{\text{result}} = \frac{d_{ji}}{R_{r_u}}. \tag{10}$$

3) The task is processed in an SFN r_u, but the vehicle have left its coverage when task is completed. Consequently, the results has to be sent to cloud node by r_u via the core network, and then it is transmitted to the vehicle from the cloud node. Thus, the result transmission time is formulated as

$$t_{j,i,r_u}^{\text{result}} = \frac{d_{ji}}{\tau} + \frac{d_{ji}}{R_c} + \frac{d_{ji}}{\tau}. \tag{11}$$

4) The task is processed locally, the results transmission time is formulated as $t_{j,i,v_j}^{\text{result}} = 0$.

C. Problem Formulation

Given the MFN set M, the SFN set S, and the task set W_j $(1 \le j \le |M|)$ with attributes $<\theta_{ji}, c_{ji}, d_{ji}, \gamma_{ji}>$, vehicles' dwell times $\varphi_{ju}(t)$ $(1 \le j \le |M|, 1 \le u \le |S|)$ and the attributes (B_n, ρ_n, C_n, f_n) of node $n \in \{S \cup M \cup \{c\}\}$, we define the service time for task w_{ji} as $t_{j,i}^{\text{task}}$, which is computed by

$$t_{j,i}^{\text{task}} = x_{j,i,v_j} \cdot t_{j,i,v_j}^{\text{task}} + \sum_{u=1}^{|S|} x_{j,i,r_u} \cdot t_{j,i,r_u}^{\text{task}} + x_{j,i,c} \cdot t_{j,i,c}^{\text{task}}$$

s.t.

$$1 \le j \le |M|, \ 1 \le i \le |W_j|$$
$$C1: \ x_{j,i,v_j}, \ x_{j,i,r_u}, \ x_{j,i,c} \in \{0, 1\}$$
$$C2: \ x_{j,i,v_j} + \sum_{u=1}^{|S|} x_{j,i,r_u} + x_{j,i,c} \le 1$$

$$
\begin{aligned}
C3:&\quad \sum_{j=1}^{|M|}\sum_{i=1}^{|W_j|} x_{j,i,v_j}\cdot\theta_{ji}\le C_{v_j}\\
C4:&\quad \sum_{j=1}^{|M|}\sum_{i=1}^{|W_j|} x_{j,i,r_u}\cdot\theta_{ji}\le C_{r_u}\\
C5:&\quad x_{j,i,r_u}\cdot t^{\text{data}}_{j,i,r_u}\le \varphi_{ju}(t)\\
C6:&\quad \sum_{j=1}^{|M|}\sum_{i=1}^{|W_j|} x_{j,i,r_u}\le \rho_{r_u}\\
C7:&\quad \sum_{j=1}^{|M|}\sum_{i=1}^{|W_j|} x_{j,i,c}\le \rho_{c}.
\end{aligned}
\tag{12}
$$

Constraints (C1) and (C2) imply that each task can be offloaded to at most one node; constraints (C3) and (C4) represent the storage constraints of the MFNs and the SFNs, respectively; constraint (C5) represents that the data have to be uploaded to an SFN before the vehicle leaves its coverage; constraints (C6) and (C7) represent that the number of tasks transmitted at the same time cannot exceed the bandwidth capacities of the SFN and the cloud, respectively.

Given the deadline γ_{ji} of w_{ji}, it can be completed only when $t^{\text{task}}_{j,i}\le\gamma_{ji}$. Therefore, the TCR is defined as the ratio of completed tasks to the total number of tasks, which is computed by

$$
\begin{aligned}
&\text{TCR}=\frac{\sum_{j=1}^{|M|}\sum_{i=1}^{|T_j|}\sum_{n\in\{S\cup M\cup\{c\}\}}x_{j,i,n}}{\sum_{j=1}^{|M|}|W_j|}\\
&\text{s.t.}\\
&\qquad C8:\quad t^{\text{task}}_{j,i}\le\gamma_{ji}.
\end{aligned}
\tag{13}
$$

With the above knowledge, the overall objective is to maximize TCR by adaptively offloading task to appropriate service nodes, which is formulated as follows:

$$
\begin{aligned}
&\underset{\forall x_{j,i,n}}{\text{MAX}}\quad \text{TCR}\\
&\text{s.t.}\quad C1-C8.
\end{aligned}
\tag{14}
$$

VI. Adaptive Task Offloading Algorithm

In this section, we propose an ATOA. The framework of ATOA is shown in Fig. 3. First, a *delay-driven classification policy* is designed to categorize all the pending tasks into two lists, namely, computing delay (t^{comp}) dominated category and transmission delay (t^{trans}) dominated category (Section VI-A). Subsequently, we design a *resource-driven division policy* to construct the four types of pending lists (i.e., cloud-only list, SFN-cloud-mixed list, SFN-MFN-mixed list, and MFN-preferred list) for searching valid candidate offloading nodes of each task (Section VI-B). Finally, we design a *deadline-driven offloading policy* for offloading tasks in each pending list (Section VI-C).

A. Tasks Categorization

Considering the fact that the performance of cloud-based offloading and MFN-based offloading is dominated by the

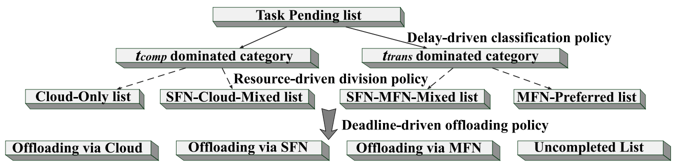


Fig. 3. Algorithm framework.

transmission delay and computing delay, respectively, we classify tasks into two categories, namely, the computing delay (t^{comp}) dominated category (i.e., D^{comp}) and the transmission delay (t^{trans}) dominated category (i.e., D^{trans}). Specifically, the tasks in D^{comp} have intensive computing requirements while the tasks in D^{trans} will transmit volumes of data.

Furthermore, we propose a *delay-driven classification policy* to classify tasks. Clearly, given a task w_{ji}, the task delay of offloading via cloud ($t^{\text{task}}_{j,i,c}$), and computing delay of offloading it to MFN ($t^{\text{comp}}_{j,i,v_j}$) are mainly decided by its input data θ_{ji} and computing requirement c_{ji}. Thus, ignoring the waiting delay in MFN-based offloading, the task delays of two offloading can be compared when w_{ji} is submitted, and it is inclined to be offloaded via MFN when $t^{\text{comp}}_{j,i,v_j}\le t^{\text{task}}_{j,i,c}$. Given the computing abilities of the cloud node (f_c) and v_j (f_{v_j}), the transmission rates of the core network (τ) and the cellular network (R_c), it is formulated as

$$
\frac{c_{ji}}{f_{v_j}}\le\frac{\theta_{ji}}{R_c}+\frac{\theta_{ji}}{\tau}+\frac{c_{ji}}{f_c}+\frac{d_{ji}}{R_c}+\frac{d_{ji}}{\tau}. \tag{15}
$$

In addition, considering that the input data size of w_{ji} should not over the available storage resources of MFN, the categorization constraint is formulated as

$$
c_{ji}\cdot\frac{\tau R_c\left(f_c-f_{v_j}\right)}{(\tau+R_c)f_{v_j}f_c}-d_{ji}\le\theta_{ji}\le C_{v_j}. \tag{16}
$$

According to (16), all tasks can adaptively classified into two categories, i.e., D^{comp} and D^{trans}. Specifically, first, initialize D^{comp} and D^{trans} as an empty set, respectively. Then, traverse the submitted tasks. If the task w_{ji} meet the categorization constraint, insert it to the set of D^{trans} (i.e., $D^{\text{trans}}\leftarrow\{w_{ji}\}$). Otherwise, insert w_{ji} to the set of D^{comp} (i.e., $D^{\text{comp}}\leftarrow\{w_{ji}\}$). Without loss of generality, offloading via cloud is better for tasks in D^{comp} and MFN-based offloading is more suitable for tasks in D^{trans}.

B. List Construction

Based on the above category classification, to facilitate offloading scheduling, we further define the four types of pending lists as follows.

1) *Cloud-Only List (l_{CO}):* Tasks in this list can only be offloaded to the cloud.
2) *SFN-Cloud-Mixed List (l_{SC}):* Tasks in this list can be either offloaded to the cloud or the SFN.
3) *SFN-MFN-Mixed List (l_{SM}):* Tasks in this list can be either offloaded to SFN or MFN.
4) *MFN-Preferred List (l_{MP}):* Tasks in this list can be offloaded to either the MFN or the cloud, and they are preferred to be offloaded to MFN when the waiting delay in MFN is low.

Furthermore, based on both required resources of tasks and available resources of heterogeneous nodes, we design a *resource-driven division policy* to construct the four lists. Specifically, note that the SFN r_u becomes the potential offloading node for vehicle v_j when v_j lies in its radio coverage, we formulate the resource-driven constraint as

$$R_{r_u} \cdot \min\{\varphi_{ju}(t), \gamma_{ji}\} \le \theta_{ji} \le C_{r_u}. \tag{17}$$

Equation (17) implies the data transmission time should be less than both vehicle's dwell time and task's deadline, and the data size also should not over the storage limitation of the SFN. Otherwise task cannot be completed. The steps of the division policy are illustrated as follows. First, initialize four lists as empty sets. Then, traverse the computing delay dominated category (D^{comp}), if the task w_{ji} satisfies the constraint, allocate it to l_{SC} (i.e., $l_{\text{SC}} \leftarrow \{w_{ji}\}$), since it can be prioritized to be processed via corresponding SFN. Otherwise allocate w_{ji} to l_{CO} (i.e., $l_{\text{CO}} \leftarrow \{w_{ji}\}$). Furthermore, traverse the transmission delay dominated category (D^{trans}), allocate task w_{ji} to l_{SM} (i.e., $l_{\text{SM}} \leftarrow \{w_{ji}\}$) when it satisfies the constraint, otherwise allocate it to l_{MP} (i.e., $l_{\text{MP}} \leftarrow \{w_{ji}\}$).

C. Task Offloading

With the above constructed four lists, in this part, we design a *deadline-driven offloading policy* to offload tasks in each list. Detailed procedures are elaborated as follows.

1) *Step 1:* Rearrange tasks in l_{CO} with the ascending order of deadline, traverse the list, and compute the task delay in cloud-based offloading for each task. For task w_{ji}, if the task delay of cloud-based offloading is less than its deadline (i.e., $t^{\text{task}}_{j,i,c} \le \gamma_{ji}$) and the number of the tasks already offloaded via cloud does not over the limitation (i.e., $\sum_{p=1}^{|M|}\sum_{q=1}^{|W_p|} x_{p,q,c} < \rho_c$), then, offload it to the cloud node (i.e., $x_{j,i,c} = 1$). Otherwise, it is uncompleted (i.e., $\sum_{n\in\{S\cup M\cup\{c\}\}} x_{j,i,n} = 0$).
2) *Step 2:* Traverse the list l_{MP} with the ascending order of deadline and compute the computing delay in MFN-based offloading for each task. For each task w_{ji}, if the computing delay of MFN-based offloading over its deadline (i.e., $t^{\text{comp}}_{j,i,v_j} > \gamma_{ji}$), then it is uncompleted. Otherwise, if the task delay of MFN-based offloading is less than its deadline and the total data of the tasks already offloaded to MFN does not over the storage limitation (i.e., $t^{\text{task}}_{j,i,v_j} \le \gamma_{ji}$ and $\sum_{i=1}^{|W_j|} x_{j,i,v_j} \cdot \theta_{ji} < C_{v_j}$), then offload it to the MFN v_j (i.e., $x_{j,i,v_j} = 1$). Otherwise, offload w_{ji} according to step 1.
3) *Step 3:* Clearly, tasks in both l_{SC} and l_{SM} may offloading to the same SFN, and thus we jointly offload them based on a merge sorting method. First, we pick up the task with the lowest deadline in both lists. Subsequently, for this task w_{ji}, if the task delay of SFN-based offloading is less than its deadline, the number of tasks already offloaded in the SFN r_u is less than ρ_{r_u} and all data of them does over the storage limitation (i.e., $t^{\text{task}}_{j,i,r_u} \le \gamma_{ji}$, $\sum_{j=1}^{|M|}\sum_{i=1}^{|W_j|} x_{j,i,r_u} < \rho_{r_u}$ and $\sum_{j=1}^{|M|}\sum_{i=1}^{|W_j|} x_{j,i,r_u} \cdot \theta_{j,i} \le C_{r_u}$), then offload it to the SFN r_u (i.e., $x_{j,i,r_u} = 1$). Otherwise, offload w_{ji} according to step 1 when $w_{ji} \in l_{\text{SC}}$, or offload it according to step 2 when $w_{ji} \in l_{\text{SM}}$.

VII. Numerical Simulations and Analysis

A. Setup

The simulation model is built based on the system architecture proposed in Section III. In the default setting, we extract a 1.5 km ×1.5 km area from the Qingyang District of Chengdu, China, and 180 taxi trajectories are extracted on August 20, 2014. In each time slot, the number of vehicles that submit tasks ranges from 2 to 5. One cloud server and six SFNs are simulated in the concerned scenario, where the cloud node has full coverage of the service area and the fog nodes are randomly deployed. The radio coverages of the SFNs are uniformly generated in the range of [300, 400]. The cloud consists of ten channels with the bandwidth $R_c = 7$, and the transmission rate via the wired network is set to $\tau = 5$. Each SFN consists of five channels with the bandwidth $R_{r_u} = 10$. The computation capacity of the cloud node is set as $f_c = 30$. The storage capacity of SFNs is uniformly generated between [200, 300] and their computation capacity is uniformly generated in the range of [12, 16]. In addition, we set the storage capacity of MFNs in the range of [10, 15] and set their computation capacity in the range of [2, 4].

By default, each vehicle generates [2, 3] tasks in each time slot. For each task, the data size is randomly distributed in the range of [4, 8], the deadline is set to *CurrentTime*+[5, 15] and the required computing resources are set in the range of [6, 14]. Furthermore, the result transmission time from SFNs is set to 0.5 while it is set to 1 if the result was delivered to the vehicle from the cloud node.

To compare performance, we implement three solutions described as follows.

1) *Cloud First and MFN Last (CF&ML):* Given a set of tasks, the offloading priority in descending order is cloud, SFNs, and MFNs.
2) *MFN First and Cloud Last (MF&CL):* Given a set of tasks, the offloading priority in descending order is MFNs, SFNs, and cloud.
3) *Heuristic Greedy Algorithm (Heuristic) [10]:* It schedules the task offloading based on a metric called delay-weight-ratio, which measures both the gained service delay and the effectiveness of available resources in a particular node. Accordingly, a smaller value of this metric implies better profits on resource utilization. Then, the algorithm selects the node with the smallest delay-weight-ratio value for task offloading in each scheduling period.

B. Effect of the Number of Task Generation Vehicles

Fig. 4 shows the TCR of different algorithms under different numbers of task generation vehicles. As noted, with an increasing number of task generation vehicles, the TCR of all the algorithms decreases. Nevertheless, excepting ATOA, the TCR of other algorithms drops significantly with an increasing system workload, which demonstrates the scalability of

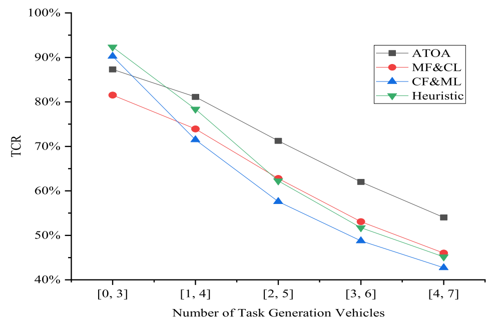


Fig. 4. TCR under different number of task generation vehicles.

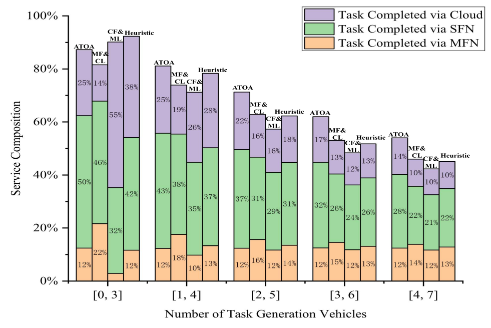


Fig. 5. Offloading composition under different number of task generation vehicles.

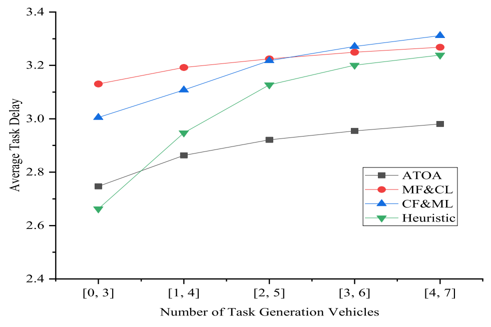


Fig. 6. Average task delay under different number of task generation vehicles.

ATOA. Fig. 5 illustrates the offloading composition of different algorithms. As shown, when the number of task generation vehicle is small, the preference of cloud or MFN offloading would not affect the TCR very much. This is mainly because the system workload is quite low, so that most tasks can be completed in time. However, with an increasing number of vehicles, ATOA shows its advantages on maximizing the TRC by adaptively adjusting the service proportion to different nodes. Fig. 6 shows the average task delay of different algorithms. As noted, although the Heuristic achieves the shortest average task delay in a very low workload environment, its performance drops significantly with an increasing number of task generation vehicles. In contrast, ATOA manages to achieve much better performance than all the other algorithms in a high system workload environment.

C. Effect of the Number of Generated Tasks Per Vehicle

Fig. 7 shows the TCR of different algorithms under different numbers of generated tasks per vehicle, we observe that

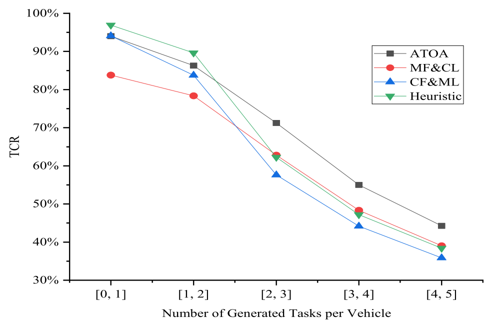


Fig. 7. TCR under different number of generated tasks per vehicle.

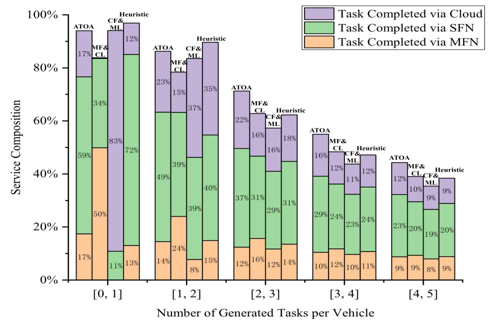


Fig. 8. Offloading composition under different number of generated tasks per vehicle.

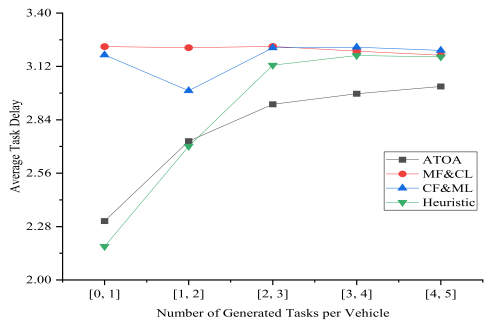


Fig. 9. Average task delay under different number of generated tasks per vehicle.

compared with Fig. 4, although both the figures show the TCR of algorithms under different task numbers, the performance of all the algorithms drops more dramatically in such a case. This is mainly because unlike varying the number of task generation vehicles, where the MFN nodes increase accordingly, this experiment only increases the task number while remaining the same number of MFNs. Fig. 8 gives offloading composition of different algorithms. As noted, except ATOA, the other three solutions cannot effectively exploit the three types of offloading nodes and their TCR drops quickly. Fig. 9 shows the average task delay of different algorithms. Similar to the observations from Fig. 6, although Heuristic achieves the best performance in a very low workload environment, its average task delay increases dramatically with an increasing number of tasks, which demonstrate that excepting ATOA, other algorithms can not adaptively adjust the task offloading with varying number of tasks.

TABLE VI
TRAFFIC CHARACTERISTICS OF DIFFERENT SCENARIOS

Traffic Scenario	Size	Number of Traces	Map	Date	ADT	VDT	ANV
Senario 1	$1.5km * 1.5km$	180	Chengdu	Aug. 20, 2014	251.6(s)	0.04(s)	151.6
Senario 2	$3km * 3km$	393			217(s)	2.9(s)	285.6
Senario 3	$3km * 3km$	403	Beijing	Nov. 13, 2015	214.9(s)	17.6(s)	290

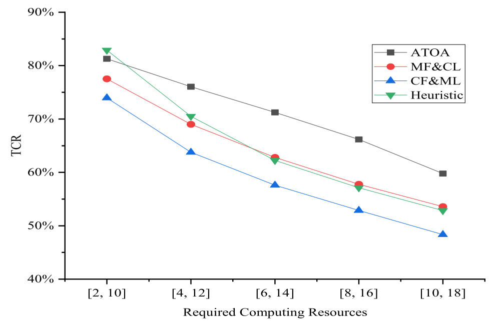


Fig. 10. TCR under different required computing resources.

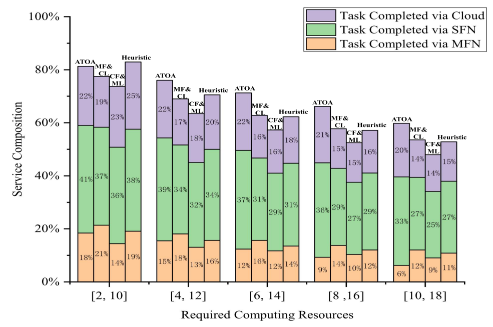


Fig. 11. Offloading composition under different required computing resources.

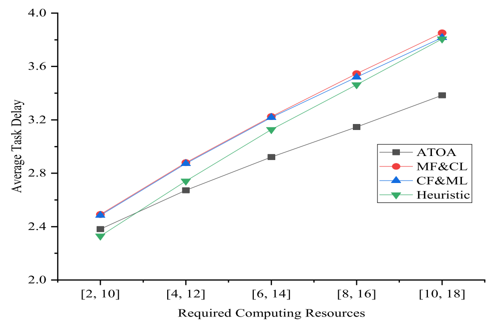


Fig. 12. Average task delay under different required computing resources.

D. Effect of Required Computing Resources

Fig. 10 shows the TCR of different algorithms under different required computing resources. As expected, the TCR of all the algorithms decreases with an increase in required computing resources. Nevertheless, ATOA always manages to maintain a decent performance, especially for serving computation-intensive tasks. Fig. 11 shows offloading composition of different algorithms. As noted, ATOA shows its effectiveness in the adaptive adjustment of task offloading under different task computation requirements. Fig. 12 shows the average task delay of different algorithms. Note that the increasing computation requirements will not only result in a longer computing delay but it may also lead to a longer waiting delay. Therefore, it is even important to strike a balance on offloading tasks to SFNs and the cloud to avoid excessively long waiting delay at SFNs as well as the overwhelming data transmission overhead at the cloud. Clearly, as observed, ATOA achieves the best performance in this regard, since it cannot only serve more tasks than other algorithms but also achieves the lowest average task delay.

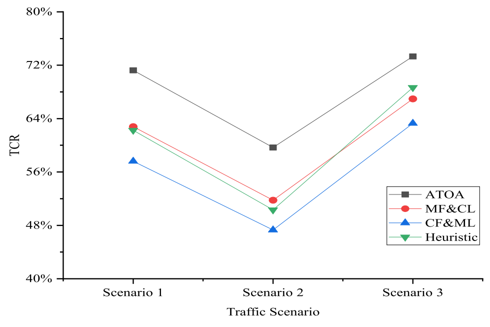


Fig. 13. TCR under different scenarios.

E. Effect of Different Traffic Scenarios

To give a comprehensive performance evaluation, in addition to the default traffic scenario, we selected two more scenarios with different traffic workloads and patterns, which are described as follows: 1) *Scenario 2:* a 3 km × 3 km area of Qingyang District, Chengdu, China, on August 20, 2014 and 2) *Scenario 3:* a 3 km × 3 km area of Haidian District, Beijing, China, on November 13, 2015. Detailed statistics, including the total number of observed vehicle traces, the average dwell time (ADT) of vehicles, the variance of dwell time (VDT), and the average number of vehicles (ANVs) in each second are summarized in Table VI. As noted, Scenarios 1 and 2 are the same area with different sizes, and Scenario 3 represents a totally different traffic pattern in a different city.

Fig. 13 shows the TCR of different algorithms under different scenarios. As analyzed above, the selected three scenarios can represent different traffic scales and different traffic patterns. Therefore, we may safely conclude that ATOA is able to achieve satisfactory performance in a variety of traffic scenarios. Fig. 14 shows the offloading composition of different algorithms. As shown, ATOA always achieves the highest TCR by adaptively allocating tasks to different types of nodes under different traffic scenarios. Fig. 15 shows the average task delay of different algorithms. First, compared with Fig. 13, we note that although the Heuristic achieves slightly lower TRC than MF&CL, it outperforms MF&CL in terms of reducing the average task delay. This makes sense since Heuristic serves

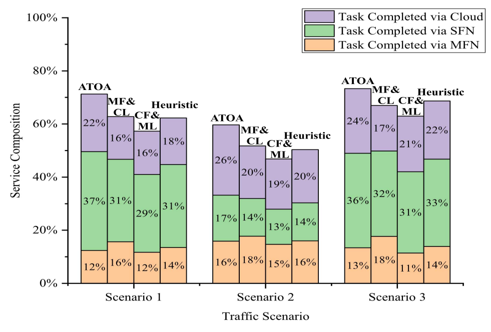


Fig. 14. Offloading composition under different scenarios.

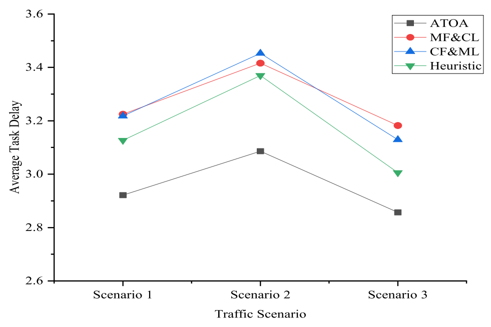


Fig. 15. Average task delay under different scenarios.

fewer tasks than MF&CL, which makes it easier to have better performance for those served tasks. However, we note that ATOA not only serves most tasks as shown in Fig. 13, but also achieves much lower average service delay compared with other algorithms, which further demonstrates that it is nontrivial for ATOA to achieve such performance.

VIII. Conclusion and Future Work

In this article, we first presented a two-layer VFC architecture, where the cloud nodes, SFNs, and MFNs are cooperated to enable time-critical IoV applications. Subsequently, we gave a motivational case study by implementing a prototype of the TAD&W system. The observations motivated the necessity of the problem to be investigated. Furthermore, we formulated an adaptive task offloading problem. Specifically, we modeled the task delay by decomposing it into transmission delay, waiting delay, and computing delay, and analyzed task offloading procedures by jointly considering the diverse transmission, computation, and memory requirements of tasks, the heterogeneous capabilities of different nodes and the mobility of vehicles. Then, we proposed an ATOA. Specifically, we categorized all tasks based on the designed delay-driven classification policy. Then, we constructed the four types of pending lists based on a designed resource-driven division policy. Furthermore, a deadline-driven offloading policy was proposed to collaboratively offload tasks from different lists to appropriate nodes. Finally, we built the simulation model with realistic vehicular trajectories and give a comprehensive performance evaluation. The results demonstrated the effectiveness and superiority of the proposed algorithm under a wide range of scenarios.

In our future work, the prototype system will be further evolved from the current hardware-in-the-loop testing to small scale realistic IoV environments. In addition, more time-critical applications in IoV will be examined with the proposed framework.

References

[1] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proc. IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[2] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart., 2016.

[3] C. Zhu, G. Pastor, Y. Xiao, and A. Ylajaaski, "Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 58–63, Oct. 2018.

[4] X. Xu, K. Liu, K. Xiao, L. Feng, Z. Wu, and S. Guo, "Vehicular fog computing enabled real-time collision warning via trajectory calibration," *Mobile Netw. Appl.*, to be published.

[5] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[6] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Netw.*, vol. 32, no. 5, pp. 112–117, Sep./Oct. 2018.

[7] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 87–93, Feb. 2019.

[8] C. Huang, R. Lu, and K.-K. R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[9] K. Liu, X. Xu, M. Chen, B. Liu, L. Wu, and V. C. S. Lee, "A hierarchical architecture for the future Internet of Vehicles," *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 41–47, Jul. 2019.

[10] J. Wang, K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Trans. Mobile Comput.*, early access, May 27, 2019, doi: 10.1109/TMC.2019.2918773.

[11] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.

[12] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, 2019.

[13] P. Dai *et al.*, "Multi-armed bandit learning for computation-intensive services in MEC-empowered vehicular networks," *IEEE Trans. Veh. Technol.*, early access, Apr. 30, 2020, doi: 10.1109/TVT.2020.2991641.

[14] X. Peng, K. Ota, and M. Dong, "Multi-attribute based double auction towards resource allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3094–3103, Apr. 2020.

[15] P. Dai, K. Liu, X. Wu, Z. Yu, H. Xing, and V. C. S. Lee, "Cooperative temporal data dissemination in SDN-based heterogeneous vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 72–83, Feb. 2019.

[16] K. Liu, V. C. S. Lee, J. K.-Y. Ng, J. Chen, and S. H. Son, "Temporal data dissemination in vehicular cyber–physical systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2419–2431, Dec. 2014.

[17] K. Liu, J. K. Y. Ng, V. C. S. Lee, S. H. Son, and I. Stojmenovic, "Cooperative data scheduling in hybrid vehicular ad hoc networks: VANET as a software defined network," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1759–1773, Jun. 2016.

[18] P. Dai, K. Liu, X. Wu, Y. Liao, V. C. S. Lee, and S. H. Son, "Bandwidth efficiency and service adaptiveness oriented data dissemination in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6585–6598, Jul. 2018.

[19] G. M. N. Ali *et al.*, "Efficient real-time coding-assisted heterogeneous data access in vehicular networks," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3499–3512, Oct. 2018.

[20] D. Tang, X. Zhang, and X. Tao, "Delay-optimal temporal-spatial computation offloading schemes for vehicular edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2019, pp. 1–6.

[21] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay sensitive task offloading in the 802.11 p based vehicular fog computing systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 773–785, Jan. 2020.

[22] C. Zhu *et al.*, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.
[23] Z. Ning *et al.*, "Deep reinforcement learning for intelligent Internet of Vehicles: An energy-efficient computational offloading scheme," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.
[24] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.
[25] Y. Wu, H. Huang, Q. Wu, A. Liu, and T. Wang, "A risk defense method based on microscopic state prediction with partial information observations in social networks," *J. Parallel Distrib. Comput.*, vol. 131, pp. 189–199, Sep. 2019.
[26] F.-H. Chan, Y.-T. Chen, Y. Xiang, and M. Sun, "Anticipating accidents in Dashcam videos," in *Proc. Asian Conf. Comput. Vis.*, 2016, pp. 136–153.
[27] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: arXiv:1704.04861.

Chunhui Liu received the B.S. degree in computer science from Chongqing University, Chongqing, China, in 2019, where he is currently pursuing the M.S. degree.

His research interests include Internet of Vehicles and mobile-edge computing.

Kai Liu (Senior Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2011.

From December 2010 to May 2011, he was a Visiting Scholar with the Department of Computer Science, University of Virginia, Charlottesville, VA, USA. From 2011 to 2014, he was a Postdoctoral Fellow with Nanyang Technological University, Singapore, the City University of Hong Kong, and Hong Kong Baptist University, Hong Kong. He is currently a Professor with the College of Computer Science, Chongqing University, Chongqing, China. His research interests include Internet of Vehicles, mobile computing, and pervasive computing.

Songtao Guo (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer software and theory from Chongqing University, Chongqing, China, in 1999, 2003, and 2008, respectively.

He was a Professor with Chongqing University from 2011 to 2012 and a Professor from 2013 to 2018, where he is currently a Full Professor. He was a Senior Research Associate with the City University of Hong Kong, Hong Kong, from 2010 to 2011, and a Visiting Scholar with Stony Brook University, Stony Brook, NY, USA, from May 2011 to May 2012. He has published more than 130 scientific papers in leading refereed journals and conferences. His research interests include mobile-edge computing, mobile cloud computing, and wireless *ad hoc* networks.

Prof. Guo has received many research grants as a Principal Investigator from the National Science Foundation of China and Chongqing and the Postdoctoral Science Foundation of China.

Ruitao Xie received the B.Eng. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2008, and the Ph.D. degree in computer science from City University of Hong Kong, Hong Kong, in 2014.

She is currently an Assistant Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. Her research interests include AI networking and mobile computing, distributed systems, and cloud computing.

Victor C. S. Lee (Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 1997.

He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong. His research interests include data dissemination in vehicular networks, real-time databases, and performance evaluation.

Dr. Lee is a member of ACM and IEEE Computer Society. He has been the Chairman of the IEEE, Hong Kong Section, and Computer Chapter from 2006 to 2007.

Sang H. Son (Life Fellow, IEEE) received the B.S. degree in electronics engineering from Seoul National University, Seoul, South Korea, in 1976, the M.S. degree from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1978, and the Ph.D. degree in computer science from the University of Maryland at College Park, College Park, MD, USA, in 1986.

He was a Visiting Professor with KAIST; the City University of Hong Kong, Hong Kong; the École Centrale de Lille, Villeneuve-d'Ascq, France; Linkoping University, Linköping, Sweden; and the University of Skövde, Skövde, Sweden. He was a Professor with the Department of Computer Science, University of Virginia, Charlottesville, VA, USA, and the WCU Chair Professor with Sogang University, Seoul, South Korea. He is the President of Daegu Gyeongbuk Institute of Science and Technology, Daegu, South Korea. His research has been funded by the Korean Government, National Research Foundation, National Science Foundation, DARPA, Office of Naval Research, Department of Energy, National Security Agency, and IBM. His research interests include cyber–physical systems, real-time and embedded systems, database and data services, and wireless sensor networks. He has authored or coauthored more than 340 papers and edited/authored four books in the above areas.

Dr. Son received the Outstanding Contribution Award from the Cyber Physical Systems Week in 2012. He is a member of the Korean Academy of Science and Technology and the National Academy of Engineering of Korea. He has served on the editorial board of the *ACM Transactions on Cyber Physical Systems*, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and *Real-Time Systems*. He is the Founding Member of ACM/IEEE CPS Week and a member of the steering committee for the IEEE RTCSA and Cyber Physical Systems Week.