# When Social Sensing Meets Edge Computing: Vision and Challenges

Daniel (Yue) Zhang, Nathan Vance, Dong Wang
Department of Computer Science and Engineering
University of Notre Dame, Notre Dame, IN, USA
{yzhang40, nvance1, dwang5}@nd.edu

***Abstract*—This paper overviews the state of the art, research challenges, and future opportunities in an emerging research direction: Social Sensing based Edge Computing (SSEC). Social sensing has emerged as a new sensing application paradigm where measurements about the physical world are collected from humans or from devices on their behalf. The advent of edge computing pushes the frontier of computation, service, and data along the cloud-to-things continuum. The merge of the two technical trends generates a set of new research challenges that need to be addressed. In this paper, we first define the new SSEC paradigm that is motivated by a few underlying technical trends. We then present a few representative real-world case studies of SSEC applications and several key research challenges that exist in those applications.Finally, we outline a few exciting research directions in future SSEC. We hope this paper will stimulate the discussions of this emerging research direction in our field.**

***Index Terms*—Social Sensing, Edge Computing, Internet of Things, Smart Cities**

## I. Introduction

Social sensing has become a new sensing paradigm for collecting real-time measurements about the physical world from humans or mobile devices on their behalf [1]–[3]. Examples of social sensing applications include urban traffic monitoring using mobile apps [4], obtaining real-time situation awareness in the aftermath of a disaster using self-reported observations from citizens [5], and smart healthcare monitoring using wearable sensors [6]. A key limitation in the current social sensing solution space is that data processing and analytic tasks are often done on a *"backend"* system (e.g., on dedicated servers or commercial clouds) [3], [7]–[9]. Unfortunately, this scheme ignores the rich processing capability of increasingly powerful edge devices owned by individuals (e.g., mobile phones, tablets, smart wearables, and the Internet of Things). For example, the emerging AI accelerators (commonly called "AI Chip") on smartphones are capable of finishing complex deep learning tasks that are traditionally done on large server racks [10], [11]. These *ubiquitous, powerful, and individually owned* devices are referred to as "edge devices" in this paper.

The advent of edge computing pushes the frontier of computation, service, and data along the cloud-to-things continuum to the edge of the network [12], [13], and brings new opportunities for social sensing applications. By combining social sensing with edge computing, the privately owned edge devices not only serve as pervasive sensors, but also form a federation of computational nodes where the data collected from them can be processed and consumed at the edge [14]–[17]. We refer to the marriage of social sensing and edge computing as Social Sensing based Edge Computing paradigm, or SSEC for short. We illustrate a typical SSEC system architecture in Figure 1. The SSEC system consists of an edge layer, an edge server layer, and a service layer. In the edge layer, privately owned edge devices (e.g., mobile phones, IoT devices, drones) are leveraged to perform the sensing, storage, networking, and computational tasks near the source of the data. The edge server layer[1] (often comprised of local servers, cloudlets, smart routers, or gateways) provides an intermediate layer between the edge devices and the cloud. The edge server layer also provides additional data storage and computing power in locations of close proximity to the edge devices [18]. The service layer (often built into a back-end cloud) provides a global service interface to all users interested in the applications/services.

The advantages of the SSEC paradigm are multi-fold: 1) social sensing applications can process the sensing data right at the edge devices where the data has been collected, which could significantly reduce the communication costs (e.g., bandwidth) and improve the Quality of Service (QoS) (e.g., delay) of the applications; 2) social sensors (e.g., owner of the edge devices) can obtain payoffs/rewards by leveraging the idle resources of their devices to execute the computing tasks for the application; 3) the SSEC architecture does not suffer from a single point of failure and alleviates the performance bottleneck of the "back-end" solutions.

The SSEC paradigm also introduces many research challenges. In particular, SSEC introduces a set of new challenges to real-time resource management by supporting delay-sensitive social sensing applications in edge computing systems. For example, the edge devices (often owned by end users) in SSEC are in general opportunistic and selfish (e.g., they are not committed to or interested in executing the sensing tasks or sharing their private device status unless incentives/payoffs are provided) [19]. This assumption is unique in SSEC and contrasts sharply with the assumption made in the "backend" based solutions in traditional distributed or cloud-based systems where all computational devices are fully committed and information is shared among all devices [9].

[1]The edge server is also commonly referred to as a *fog node* in fog computing literature. We use these two terms interchangeably in this paper.

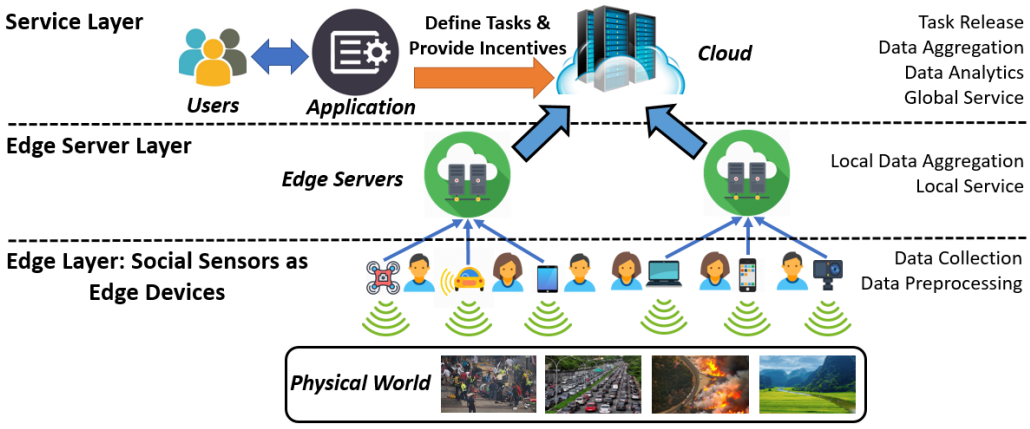


Figure 1: Social Sensing based Edge Computing

Furthermore, the SSEC paradigm calls for close collaboration among end users, infrastructure owners, and application managers. Due to the lack of natural trust among them, none of these parties can be fully trusted as they might be interested in performing privacy and security attacks.

The rest of the paper is organized as follows. In Section II, we present the definition, enabling technologies, and impact of the SSEC paradigm. In Section III, we discuss several important applications and case studies of SSEC. In Section IV, we discuss the unique research challenges and opportunities in SSEC. We outline the future road map of this direction in Section V. Finally, we conclude our vision of SSEC in Section VI.

## II. Social Sensing Edge Computing Paradigm

IoT devices owned by individuals are increasingly equipped with powerful computing and diverse sensing capabilities. The sensing data generated by these devices provides an alternative lens into physical phenomena as compared to traditional sensor networks [1]. Due to the sheer volume of data generated by these devices, it makes sense to explore opportunities for processing the data at the edge of the network. Previous work in edge computing leverage cloudlets [18], [20], micro datacenters [21], [22], and fog computing [23], [24] to address the deficiency of cloud computing when the data is produced at the edge of the network. However, these solutions fail to take advantage of privately owned edge devices as SSEC does, and they instead rely on infrastructure which must be provisioned ahead of time. In this section, we formally define social sensing based edge computing (SSEC) and discuss how SSEC is complementary to existing edge computing frameworks.

### A. What is SSEC?

**DEFINITION 1. Social Sensing based Edge Computing (SSEC)**: an application paradigm that uses humans and devices on their behalf to sense, process, and analyze data collected about the physical world.

In this definition, the devices owned by individuals not only collect data about the physical world, but also actively participate in the application by performing computations and analytic tasks. These privately owned edge devices can be quite heterogeneous, ranging from a GPS sensor, a Raspberry Pi, a robot, to a powerful multi-processor server.

SSEC has two important features: 1) it is human-centric and 2) it has the flexibility to support various applications with different system architectures. We elaborate these features below.

*1) Human-centric Nature of SSEC:* SSEC is human-centric. On one hand, the owners of the edge devices are freelance users and their unique concerns must be carefully considered in the SSEC paradigm. These human concerns includes privacy and security, compliance and churn, and incentives, which will be elaborated in Section IV. On the other hand, we envision that not only can devices engage in the sensing and computational tasks, but people can directly participate as well. In fact, many social sensing applications require input directly from a human, such as reporting traffic congestion [25], or taking videos of an emergency event [16]. Also, SSEC considers the potential of people serving as "social computing nodes" where they directly make inferences using the data. For example, consider an abnormal event detection scenario where edge devices are used to collect video data and infer abnormal events such as an intrusion [26]. Instead of using machine learning algorithms to perform such data analytic tasks, humans can directly identify the abnormal events from the video with high accuracy [27]. We explore the possibility of leveraging humans as computing nodes in a pioneer work [5]. This unique feature of SSEC where human input and intelligence complements the existing edge/cloud computing paradigm promises to enable new applications that wouldn't be possible without it.

*2) Flexibility of SSEC to Support System Variations:* Like traditional edge computing systems that come in many different architectures [28], SSEC has diverse system variations as well (Figure 2). While SSEC focuses more on the privately owned edge devices, it by no means intends to drastically replace the existing cloud or edge computing paradigm by diminishing the existing infrastructure such as cloud servers, large data centers, cloudlets, or near-edge micro

data centers. In fact, SSEC fully takes advantage of existing system infrastructures. The choice of the system architecture is application specific. We summarize a few representative architectures below.

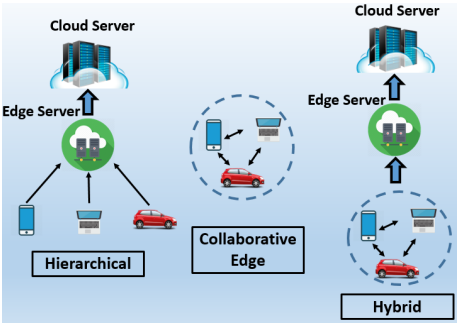


Figure 2: Example SSEC System Variations

**Hierarchical:** A typical cloud-edge hybrid SSEC system architecture is shown in Figure 2. It often follows a hierarchical structure where a remote cloud server manages the application and provides a global interface to the users. The cloud server is often powerful and has a massive storage capability. The application governs a set of spatially distributed edge clusters. An edge cluster consists of a local edge server (e.g., a micro data center or a Road-Side-Unit) and the nearby edge devices that connect to it. In [13], [29], [30], typical edge clusters are illustrated, including a set of devices in a coffee shop connected to a small in-house server owned by the shop; a set of vehicles connected to a Road-side-Unit (RSU) on the same street; and a set of mobile phones connected to the nearest base station. The key characteristic of this hierarchical structure is that the data flow is static: edge devices process the data locally and offload further computational tasks to the edge servers, and edge servers further process the data and send the results to the cloud server for data aggregation tasks and storage.

**Collaborative Edge:** In a collaborative edge architecture, edge devices in close proximity self-organize into a computing cluster and provide peer-to-peer services such as content delivery and computation offloading. This application is particularly suitable for application scenarios where edge or cloud servers are not readily available, or to avoid periodic costs by using these infrastructures. Consider a crowd video sharing application example where a set of spectators at a sporting event (e.g., a soccer game) can take videos of the highlights of the game and can stream them to people in the audience who missed the play or who sit in an undesirable location. To improve performance for devices with poor network connections, the system can encode the video streams to a lower bitrate. In such a scenario, a remote cloud can introduce significant delay for video sharing and local edge servers may not be available (the servers/smart gateways at the stadium may not be accessible by the audience).

**Hybrid:** A hybrid system architecture is a combination of both a hierarchical and collaborative edge, in which self-organized edge devices are connected to the available infrastructure (i.e., edge servers and the cloud). This infrastructure is ideal for scenarios where self-organized edge devices cannot satisfy QoS requirement, so readily available edge servers and the cloud are leveraged to boost performance. Consider a disaster response application where edge devices collaboratively report damages, often by executing image analysis and machine learning algorithms to classify damage severity, during a disaster [5]. A computationally weak edge device such as a video camera can collect image data of the affected area and offload the damage assessment task to a powerful edge device nearby via Bluetooth. The assessment result is further reported to all nearby edge devices in the form of alerts. In the case where edge devices are under-performing due to lack of high-end processors, the collaborative edge can offload tasks to nearby edge servers, such as base stations, or cloud servers for further processing.

### *B. Why We Need SSEC?*

Social Sensing based Edge Computing (SSEC) is motivated by a few key technical trends: i) the IoT devices owned by individuals are becoming increasingly powerful and some of them even have similar computing power as the dedicated servers in traditional edge computing systems [16], [31]. Therefore, it becomes a growing trend to push the computation to the edge devices rather than dedicated remote servers or edge servers [29]; ii) the popularity of mobile payments provides a more convenient way for common individuals to receive incentives by contributing the spare resources on their IoT devices for accomplishing social sensing tasks [32]. We summarize a few advantages of SSEC below.

*1) Coverage and Availability:* One of SSEC's main advantages is its coverage and the availability of edge devices. There are billions of privately owned edge devices worldwide that can collect and process data at a global scale. This natural mobile network is clearly advantageous in terms of coverage as compared to static infrastructure such as data centers or surveillance cameras. Furthermore, SSEC provides mobility as the sensing and computing resources move geographically with their users. This makes SSEC ideal for people-centric sensing and computing tasks as the availability of resources is closely correlated with the prevalence of noteworthy events.

*2) Delay Reduction:* Social sensing applications can process the sensing data on the edge devices where the data has been collected or on devices in close proximity, which could significantly reduce the communication costs (e.g., bandwidth) and improve the Quality of Service (QoS) (e.g., delay) of the applications. This makes SSEC ideal for real time or time-sensitive applications.

*3) Utilization:* SSEC fully leverages the sensing and computing power of the edge devices. Compared to traditional edge computing frameworks that offload computational tasks to edge servers or cloud servers, SSEC envisions that tasks can be executed on smart devices owned by individuals as well. By pushing the tasks to the edge, the SSEC architecture removes the single point of failure and alleviates the performance bottleneck of the "back-end" solution. This enables SSEC to

avoid high deployment costs for sensing tasks, and to save money on the back-end infrastructure.

*4) Reward Earnings:* In SSEC, participants can obtain rewards by contributing the idle resources of their devices to execute computing tasks for the SSEC application. Similar to how unused compute cycles are sold in cloud environments, this creates a new market where the idle resources of edge devices can now be fully utilized.

## III. Real-world Applications

In this section, we discuss a few representative SSEC applications in real world scenarios.

### A. Disaster and Emergency Response

An important application of SSEC is to provide real-time situation awareness during diaster and emergent events (e.g., forest fire, robbery, terrorist attacks) [9]. During such events, human sensors (e.g., citizens, first responders, news reporters) often spontaneously report a massive amount of sensing information that describes the unfolding of the event. SSEC provides a suitable architecture for this category of applications: 1) the edge devices, with close proximity to the human sensors, can collect and extract useful features about the event without sending all data streams back to the cloud; 2) the edge server layer in SSEC can gather processed data and exacted features from edge devices to provide real-time event updates for local citizens; 3) the cloud server aggregates all information collected and provides it to relevant agencies and/or the general public. Figure 3 illustrates a scenario where people use mobile phones and cameras to provide first-hand footage of a terrorist attack at a shopping center. These data can be helpful in tracking a suspect's escape path. The edge server layer provides time-critical alerts of potential threats and offers safety recommendations.

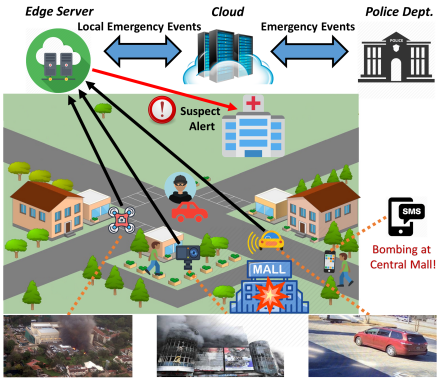


Figure 3: Disaster and Emergency Response Application

### B. Collaborative Traffic Monitoring

Collaborative traffic monitoring in social sensing aims at collecting timely information about traffic conditions (e.g., congestion, accidents, and events) of an area of interest (e.g., city). Such applications are useful for many transportation services such as route planning, traffic management, and fuel efficient navigation [30]. Traditionally, traffic monitoring were performed by analyzing the statically installed traffic cameras such suffers from poor coverage [33]. Moreover, the data generated by these traffic cameras were processed at a remote cloud server, which introduces significant delay and bandwidth cost. SSEC can well address this problem by fully leveraging social sensing and edge devices owned by people. In particular, the personally owned sensing devices on vehicles (e.g., cameras, accelerometers, GPS sensors) offer opportunities to collect a large amount of traffic data in real time. For example, a typical traffic monitoring application can task a set of drivers to use their dashboard cameras to record traffic in front of their vehicles. The processed data (e.g., extracted features) are offloaded to the to nearby edge servers (i.e., RSU) for further analysis of traffic conditions. Additionally, human sensors are also capable of reporting high-level descriptions of the traffic context using their smart phones. An example of such a social sensing application is Waze[2] where drivers collectively report their observations of accidents, road hazards, and traffic jams in real-time. In Figure 4, pedestrians and drivers collaboratively contribute traffic data using their edge devices. The edge server infers the traffic conditions of local streets from the social sensing data and sends accident alerts to the drivers. Transportation agencies can also query the cloud for the road conditions and accidents in their regions of jurisdiction and prioritize accident response, road repair, or traffic control accordingly.

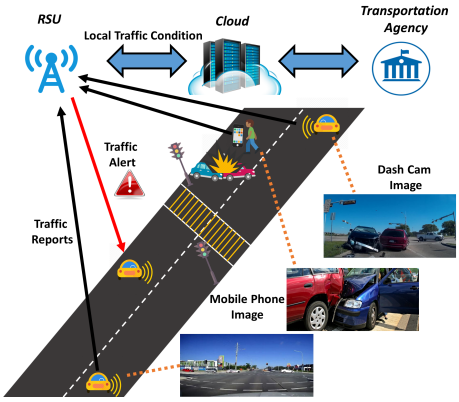


Figure 4: Collaborative Traffic Monitoring Application

### C. Crowd Abnormal Event Detection

The goal of crowd abnormal event detection in social sensing is to generate alerts of abnormal events from data contributed by human sensors and their portable devices (e.g., mobile phones). Traditional abnormal event detection solutions largely depend on video data collected from installed surveillance cameras and utilize image processing techniques to identify these events [34]. Those solutions fail in situations where installed cameras are not available (e.g., due to deployment costs). The prevalence of camera-enabled portable devices has enabled the collection of geo-tagged pictures, videos, and user-reported textual data through social sensing applications. Such multi-modal data can be exploited for

[2]https://www.waze.com/

enhanced situation awareness during abnormal activities (e.g., providing insights for investigating the severity and causes of events). For example, during a soccer game, events such as sudden appearance of unexpected object or malicious behavior of people (e.g., throwing a signal flare into the field) can pose great threats to the safety of players and interrupt the normal course of the game (Figure 5). In our SSEC framework, the audience (as human sensors) can contribute videos, images, and texts to report their observations about the abnormal events. Upon detection of the abnormal events during the game, the cloud-hosted service will send alerts to the fans and the police department for emergency response.

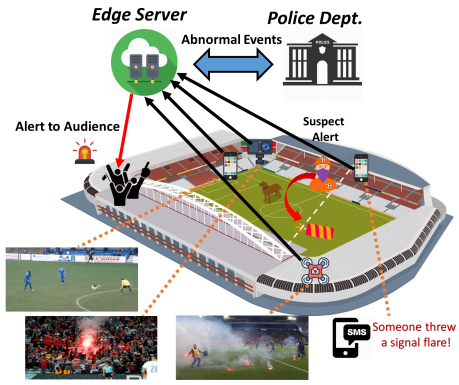


Figure 5: Crowd Abnormal Event Detection Application

### D. Plate Recognition

The plate recognition application (Figure 6) was first introduced in an effort to leverage private vehicles to collaboratively track down suspects of AMBER alerts [35]. In this application, vehicles equipped with dash cameras form a city-wide video surveillance network that tracks moving vehicles using the automatic license plate recognition (ALPR) technique. This system can be used to effectively track down criminal suspects who are on the run in vehicles. It complements existing vehicle searching processes that heavily rely on reports from witnesses who might miss alerts and cannot search enough areas of city [36]. Collecting surveillance video footage can expand coverage. However, analyzing huge amounts of video data in the cloud leads to unreasonable data transmission costs and high response latency. SSEC can significantly reduce the cost of data transmission and response latency by offloading the data to nearby RSUs for real-time processing. SSEC also pushes local processing to be done on these private vehicles to extract features from the raw images and send the processed data to the RSUs instead. This is because the video data collected from the vehicles can also reveal private information of the drivers (e.g., residence location) or the faces of the citizens. Upon detecting the suspect's vehicle, the cloud-hosted service will send alerts to the police department for an immediate response.

### E. Crowd Video Sharing

The crowd video sharing application (Figure 7) uses self-organized edge devices to perform peer-to-peer video content

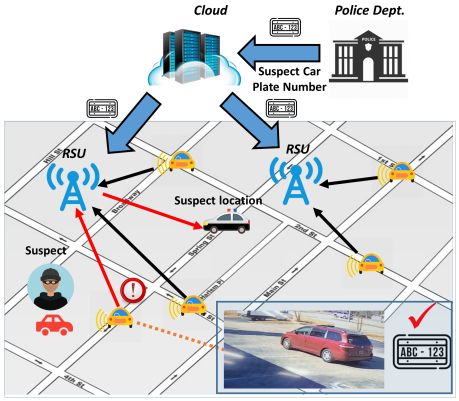


Figure 6: Plate Recognition Application

delivery. This application is most suitable for events where people take interesting videos and want to share it to one another. For example, if a spectator at a soccer match has a good view of some action, then other spectators in less favorable locations may desire to view the footage from the better perspective. In order to facilitate this application the system must 1) employ the participating edge computing resources to avoid bottlenecks as the system scales, and 2) perform video encoding so that devices with poor network connections can be sent smaller video files, thus avoiding network delays. This problem can be solved using SSEC by coordinating edge devices to perform computation and communication tasks, thus providing a source of compute power and bandwidth which scales with the number of participating devices, i.e., demand. A bottom-up game theoretic decision making process optimizes the encoding and transmission of the videos in order to minimize delay in the system.

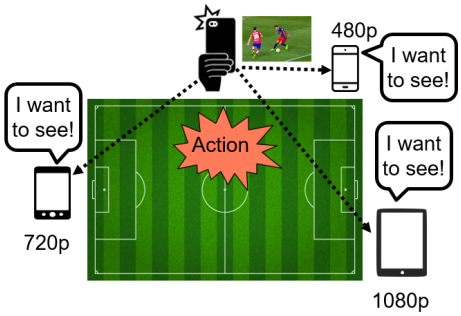


Figure 7: Crowd Video Sharing Application

## IV. Research Challenges and Opportunities

The fusion of social sensing and edge computing pushes the frontier of sensing, computation, and service to the edge of the networks where social sensing occurs. However, utilizing edge devices in the context of social sensing introduces a set of fundamental challenges that are yet to be fully addressed. In this section, we discuss a few critical research challenges and opportunities in SSEC.

### A. Resource Management with Rational Edge

In SSEC, the edge devices are usually owned by end users rather than application providers. Due to the rational nature

of device owners, edge devices and applications often have inconsistent or even conflicting objectives [37]. We refer to this unique feature of SSEC as *"rational edge"*. Due to the rational edge feature, two important issues prevent existing resource managements schemes from being applied to SSEC, namely *competing objectives* and *asymmetric information*.

*1) Competing objectives:* from the application's perspective, it is important to ensure that the edge devices finish the allocated social sensing tasks in a timely fashion to meet the Quality of Service (QoS) requirements (e.g., end-to-end delays). In contrast, device owners are often less concerned about the QoS of the applications but are instead concerned about their costs (e.g., the device's current utilization, energy consumption, memory usage) in running the computational tasks allocated by the applications. They are often unwilling to execute the allocated tasks until sufficient incentives are provided [19]. This is in sharp contrast with traditional distributed computing systems where computational resources are fully cooperative and directly controlled by the application. The mismatch in objectives held by the end users and the application must be carefully addressed by developing a set of new computation allocation models that respect such discrepancies between the two parties.

*2) Asymmetric Information:* another critical challenge in SSEC is that the application server and edge devices usually have different degrees of information, i.e., *"asymmetric information"*. Such asymmetric information makes resource management in social sensing based fog computing systems particularly challenging [38]. The asymmetric information challenge can be viewed from two aspects. On the server side, the application normally has detailed information about the tasks (e.g., the dependencies and criticality of the tasks). This information is important in understanding how tasks are related to the QoS requirements imposed by the social sensing application (e.g., which tasks are more important and should be prioritized; which tasks should have a tighter deadline). In contrast, the edge devices are often less concerned about the details of the tasks and the servers' QoS requirements but more interested in their own device status (e.g., CPU utilization, energy consumption, memory usage). Moreover, an edge device may not share its status information with the server or other edge devices in the system due to various concerns (e.g., privacy, energy, bandwidth). This leads to insufficient information for the server to make optimal computation allocation decisions.

### *B. Constrained Cooperativeness*

In SSEC, edge devices are assumed to be only partially cooperative in finishing their computational tasks due to the rational or selfish nature of end users. This challenge is referred to as *"Constrained Cooperativeness"*. Previous studies showed that collaboration among computation nodes can significantly improve efficiency of resource utilization in distributed systems [39]. Such collaboration between edge devices in social sensing applications is essential to achieve optimized scalability and efficiency in the SSEC system. For example, the execution time of a set of tasks can be significantly reduced if those tasks are allocated to a group of edge devices that run the tasks in parallel and finish them collaboratively. Consider an abnormal event detection application where edge devices (e.g., smartphones, dash cameras) are tasked to take videos/pictures of surroundings to detect abnormal activities. An edge device may not be equipped with a camera and thus is incapable of completing the allocated tasks on its own. However, if it has strong computing power, then it can serve as a "local computation hub" for nearby lower-end edge devices that do have cameras. However, collaboration among edge devices is especially challenging because: i) edge devices are rational actors who are unwilling to collaborate with others unless sufficient incentives are provided; ii) various constraints may prohibit collaboration among edge devices (e.g., latency constraints imposed by the physical distance between devices or trust constraints imposed by the trust between devices); iii) collaboration over task allocation requires explicit consideration of the *task dependencies* of the application.

### *C. Pronounced Heterogeneity*

The heterogeneity in SSEC is often more pronounced than regular edge computing systems. In particular, the edge devices in SSEC often have diversified computing power, runtime environments, network interfaces, and architectures, making it hard to orchestrate these devices to collaboratively accomplish the sensing and computational tasks. The heterogeneity problem in SSEC is particularly challenging because it is not possible for the application to cherry-pick the devices in a fully controlled manner given the fact the devices are owned by individuals [40], [41]; In order to tame the heterogeneity of edge devices in SSEC, several critical research tasks are involved.

*1) Runtime Abstraction:* A critical issue in heterogeneous SSEC is that the devices have diverse runtime environments that may not support the social sensing tasks to be processed. For example, a device may have an incompatible operating system or lack the necessary dependencies to execute a social sensing algorithm (e.g., a deep learning algorithm cannot run on a device without necessary libraries such as Tensorflow or CUDA). Containerization technique such as Docker [42] can be a promising candidate for this runtime abstraction task, which can abstract away the hardware details of the devices and provides a virtual environment that offers a lightweight, portable and high-performance sandbox to host various applications [43]. In particular, the social sensing application developers can "wrap" all necessary dependencies and the OS itself into a Docker container for each social sensing application. Such runtime abstraction can allow the edge devices in SSEC to provide the same interface to the social sensing application developers and offers them the "write once and run anyway" feature despite the heterogeneity of SSEC devices.

*2) Hardware Abstraction:* The hardware abstraction targets at abstracting away the details of heterogeneous hardware specifications of the edge devices for the ease of resource management of SSEC. A possible solution was proposed in

HeteroEdge [15], where the hardware capability of a device can be represented as a set of "workers". HeteroEdge considers three types of workers that are essential in finishing social sensing tasks in SSEC - CPU, GPU, and Sensor workers. Each worker is associated with a capability descriptor in terms of the estimated worst case execution time (WCET) of processing social sensing tasks. The device owners can specify which workers are available to the SSEC application. HeteroEdge follows three important design principles in hardware abstraction in SSEC: i) the set of heterogeneous edge devices should form a unified homogeneous resource pool for the social sensing application ; ii) the device owners should be able to control which resources they would like to provide for an application; iii) the edge device can easily keep track of their own dynamic status and provide necessary context information for the runtime decision and optimization in SSEC.

*3) Networking Abstraction:* The privately owned edge devices in SSEC can have very heterogeneous network interfaces (e.g., Bluetooth, WiFi, Zigbee) and it is essential to abstract away the networking details to allow developers to deploy SSEC applications without worrying about the specific network interface and protocol [44]. A promising technique to this task is Software Defined Networking (SDN) [45]. SDN can orchestrate the network, the services and the devices by hiding the complexities of this heterogeneous network environment from the end users. It provides APIs that can simplify the management of the network, define network flows, and facilitate virtualization within the network.

We found existing resource management work in edge computing cannot sufficiently handle the pronounced heterogeneity in SSEC. A middleware that jointly address the three levels of abstraction above for SSEC has yet to be developed.

### *D. Robustness against Churn and Dynamic Context*

In SSEC, edge devices are most often privately owned and managed, and therefore suffer from churn [46], causing inconsistent availability by devices in edge computing. The inconsistency of edge device availability is aggravated since devices routinely kill tasks for power savings, or are opportunistically contributing compute power and then must stop in order to service their primary purpose [47]. Furthermore, in the case of mobile computing systems, a main criterion in eligibility of a device to perform a task is the location of that device. Should the device move, then it may become unable to serve its function and must be replaced by a device in a more favorable location. To solve this problem in a way that is both scalable and reliable, we introduce buffering into multi-stage streaming applications. In such systems, tasks are broken into multiple stages where different devices perform an operation at each stage of a computational pipeline. If a device along the pipeline unexpectedly quits and must be replaced, then the replacement can be "filled in" by the the devices adjacent to it in the pipeline. Furthermore, this pipeline design lends itself to taking fine-grained advantage of heterogeneous edge computing hardware since each stage can be matched to a specialized computing platform.

Another challenging issue in the SSEC system is that edge devices have volatile status and their willingness to execute social sensing tasks may change dynamically over time. We refer to this challenge as *dynamic compliance*. For example, consider an environment sensing application where edge devices (e.g. mobile phones) are used to collectively monitor the air pollution of a city. Each edge device is tasked to monitor a particular area. An edge device (or its end user) may change the compliance of task execution due to i) changes in the battery status of the device, ii) changes in the physical location of the device with respect to the monitored location, or iii) the system resources taken by the allocated tasks. Failure to capture such dynamics may lead to significantly suboptimal resource allocation where the costs of edge devices to complete a task are prohibitively high. This problem has not been well addressed by existing work due to the difficulty in modeling the compliance of edge devices, which requires deep understanding of devices' dynamic status and willingness to participate.

### *E. Privacy and Security*

SSEC entails potential privacy risks to owners of edge devices in social sensing applications. During the data collection phase, the data collected from edge devices can potentially reveal end users' private information. For example, in the plate recognition application, the image captured by an edge device may contain street information, potentially disclosing user residence or mobility patterns. During the resource management phase, it is of the application's interest to obtain better knowledge on the status of each edge device to maximize the task allocation efficiency. However, the edge devices may not be willing to share such status information due to their privacy configurations. Existing privacy preserving techniques, such as anonymity techniques, can effectively protect the identities of edge devices from curious entities [30]. But such techniques also prevent the application from identifying the contributors of the computational tasks. Therefore, the server will not be able to send the rewards to the contributing edge devices that accomplish the computational tasks if anonymity techniques are directly applied in SSEC. A privacy-aware SSEC system is yet to be developed to meet both the privacy expectations from end users and the QoS requirements from the applications.

Security in SSEC systems is particularly difficult due to the fact that data originates and is processed at the privately owned and managed edge rather than the server. Unfortunately, state-of-the-art solutions often rely on the server being the source of data as is the case in a cluster computing environment, or they rely on the edge devices being owned and managed by the the entity that runs the application [17]. Therefore, it becomes a real issue when an edge device cheats by sending erroneous results to the server. Furthermore, should multiple edge devices conspire against the server, they can form a collusion attack in which they fool the server into believing that falsified results are real, even under sophisticated scrutiny. For example, in an abnormal activity detection application scenario, multiple edge devices can fake the image data being collected or fake

the image processing (running on the edge) results so that the server cannot detect abnormal activities in target areas. On the other hand, edge devices can intentionally delay the processing of allocated computational tasks to compromise the QoS of the application. In addition to these numerous vectors of attack, there are many motives as to why an attacker may wish to target an edge computing system (e.g., obtain the monetary incentive without doing work, steal personal information from other edge devices, or sabotage the social sensing application).

## V. Roadmap for Future Work

### A. SSEC and 5G

5G promises to have an estimated network speed as fast as 10 Gb/s and an reduced network latency as low as 1 ms [48]. We envision 5G will significantly boost the performance of SSEC and enable new SSEC applications. For example, the emergence of 5G networking capabilities will increase the number of connected devices on a network and promote collaboration among private edge devices. The delay requirement of 5G requires base stations to be deployed at higher density, which would also be able to serve as edge servers in SSEC. With 5G networks, SSEC applications such as the crowd video sharing and the plate recognition that involves video content transfer will significantly benefit from boosted Internet speed and ultra low latency. We envision more data intensive and delay sensitive SSEC application will be enabled by 5G and future networking technologies.

### B. SSEC and AI

AI at the edge is a growing trend in both industry and academic research. Many AI-enabled chips have been developed and integrated into video cameras such as DeepLens[3], hand-held devices [11], and hard drives [49]. However, AI capabilities are still far from being pervasive - many edge devices are low-end sensors without processing capabilities or do not have a GPU for supporting AI algorithms. SSEC can further promote AI by developing collaborative intelligent edge where lower end devices can offload AI tasks to edge devices with AI capabilities [15]. Many road blocks must be removed for this vision to become true. For example, performing AI tasks on privately owned edge devices inevitably incurs an energy cost. Considering that the battery is often the most precious resource of an edge device [50], incentive mechanisms must be designed so that a fair market can form to reward those who contribute energy. The collaborative intelligent edge also involves interactions among devices of differing ownership. Therefore, privacy and trust concerns must be carefully addressed.

### C. SSEC and Human-in-the-loop

Human-in-the-loop SSEC enables the integration of human intelligence (e.g., context-awareness, cognitive skills) with the processing and sensing capability of physical devices. We envision that the human component of SSEC can be modeled as a "social edge node" in which the human can perform inference or make decisions in the edge computing framework just as a physical device. This Human-in-the-loop SSEC paradigm can benefit many mission critical tasks by introducing the domain expertise of a human. For example, humans can improve the effectiveness of physical systems in many intelligent tasks (e.g., disaster assessment [5] and traffic abnormality detection [51]). A clear road block of this direction is that people are subjective and can make mistakes. Also, the response delay can be highly unpredictable compared to physical devices. To address these challenges, we envision that a new set of theories for building human-machine hybrid systems must be developed to fully leverage human intelligence in SSEC.

.

[3]https://aws.amazon.com/deeplens/

## VI. Conclusion

In this paper, we present an emerging Social Sensing based Edge Computing (SSEC) framework to exploit the edge-enabled infrastructure and the ever-increasingly powerful IoT devices to improve the scalability and responsiveness of social sensing applications. With the human-centric design, SSEC envisions to integrate human intelligence into the process of data collection, processing, analysis, and decision making. We discuss several emerging applications that are enabled by SSEC, together with a number of open research challenges are to be undertaken by the community. We conclude the paper with a roadmap with future research directions of integrating SSEC with the emerging technological trends such as 5G networks, AI at the edge, and Human-in-the-loop design. We hope this paper would bring the SSEC paradigm to the attention of the community.

## References


[1] D. Wang, B. K. Szymanski, T. Abdelzaher, H. Ji, and L. Kaplan, "The age of social sensing," *Computer*, vol. 52, no. 1, pp. 36–45, 2019.

[2] C. C. Aggarwal and T. Abdelzaher, "Social sensing," in *Managing and mining sensor data*. Springer, 2013, pp. 237–297.

[3] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Proc. ACM/IEEE 11th Int Information Processing in Sensor Networks (IPSN) Conf*, Apr. 2012, pp. 233–244.

[4] Z. Liu, Z. Li, K. Wu, and M. Li, "Urban traffic prediction from mobility data using deep learning," *IEEE Network*, vol. 32, no. 4, pp. 40–46, 2018.

[5] D. Y. Zhang, Y. Zhang, Q. Li, T. Plummer, and D. Wang, "Crowdlearn: A crowd-ai hybrid system for deep learning-based damage assessment applications," in *Distributed Computing Systems (ICDCS), 2019 IEEE 39th International Conference on*. IEEE, 2019.

[6] Y. Zhang, M. Qiu, C.-W. Tsai, M. M. Hassan, and A. Alamri, "Health-cps: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Systems Journal*, vol. 11, no. 1, pp. 88–95, 2017.

[7] D. Wang, M. T. Amin, S. Li, T. Abdelzaher, L. Kaplan, S. Gu, C. Pan, H. Liu, C. C. Aggarwal, and R. Ganti, "Using humans as sensors: an estimation-theoretic perspective," in *Information Processing in Sensor Networks, IPSN-14 Proceedings of the 13th International Symposium on*. IEEE, 2014, pp. 35–46.

[8] R. W. Ouyang, L. M. Kaplan, A. Toniolo, M. Srivastava, and T. Norman, "Parallel and streaming truth discovery in large-scale quantitative crowdsourcing."

[9] D. Y. Zhang, C. Zheng, D. Wang, D. Thain, X. Mu, G. Madey, and C. Huang, "Towards scalable and dynamic social sensing using a distributed computing framework," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 966–976.

[10] Androidheadlines, "AI Chips To Be In Every Third Smartphone By 2019," https://www.androidheadlines.com/2018/09/ai-chips-to-be-in-every-third-smartphone-by-2019.html, 2018, [Online; accessed 6-November-2018].

[11] X. You, C. Zhang, X. Tan, S. Jin, and H. Wu, "Ai for 5g: research directions and paradigms," *Science China Information Sciences*, vol. 62, no. 2, p. 21301, 2019.

[12] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*. IEEE, 2016, pp. 1–8.

[13] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[14] D. Zhang, Y. Ma, Y. Zhang, S. Lin, X. S. Hu, and D. Wang, "A real-time and non-cooperative task allocation framework for social sensing applications in edge computing systems," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2018, pp. 316–326.

[15] D. Y. Zhang, T. Rashid, X. Li, N. Vance, and D. Wang, "Heteroedge: taming the heterogeneity of edge computing system in social sensing," in *Proceedings of the International Conference on Internet of Things Design and Implementation*. ACM, 2019, pp. 37–48.

[16] D. Zhang, Y. Ma, C. Zheng, Y. Zhang, X. S. Hu, and D. Wang, "Cooperative-competitive task allocation in edge computing for delay-sensitive social sensing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 243–259.

[17] N. Vance, D. Y. Zhang, Y. Zhang, and D. Wang, "Privacy-aware edge computing in social sensing applications using ring signatures," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2018, pp. 755–762.

[18] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, no. 4, pp. 14–23, 2009.

[19] T. Giannetsos, S. Gisdakis, and P. Papadimitratos, "Trustworthy people-centric sensing: Privacy, security and user incentives road-map," in *Ad Hoc Networking Workshop (MED-HOC-NET), 2014 13th Annual Mediterranean*. IEEE, 2014, pp. 39–46.

[20] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[21] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for iot," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. IEEE, 2015, pp. 687–694.

[22] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Computer Networks*, vol. 130, pp. 94–120, 2018.

[23] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE, 2015, pp. 73–78.

[24] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[25] A. Artikis, M. Weidlich, F. Schnitzler, I. Boutsis, T. Liebig, N. Piatkowski, C. Bockermann, K. Morik, V. Kalogeraki, J. Marecek *et al.*, "Heterogeneous stream processing and crowdsourcing for urban traffic management." in *EDBT*, 2014, pp. 712–723.

[26] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2720–2727.

[27] Y. Cong, J. Yuan, and J. Liu, "Abnormal event detection in crowded scenes using sparse representation," *Pattern Recognition*, vol. 46, no. 7, pp. 1851–1864, 2013.

[28] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

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

[30] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 146–152, 2017.

[31] D. Zhang and D. Wang, "An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems," to appear in IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019, accepted.

[32] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang, "A blockchain based privacy-preserving incentive mechanism in crowdsensing applications," *IEEE Access*, vol. 6, pp. 17 545–17 556, 2018.

[33] E. D'Andrea, P. Ducange, B. Lazzerini, and F. Marcelloni, "Real-time detection of traffic from twitter stream analysis," *IEEE transactions on intelligent transportation systems*, vol. 16, no. 4, pp. 2269–2283, 2015.

[34] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 851–860.

[35] Q. Zhang, X. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Big data sharing and processing in collaborative edge environment," in *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE, 2016, pp. 20–25.

[36] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Enhancing amber alert using collaborative edges: Poster," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 27.

[37] D. Y. Zhang, Y. Ma, Y. Zhang, S. Lin, X. S. Hu, and D. Wang, "A real-time and non-cooperative task allocation framework for social sensing applications in edge computing systems," to appear in Real-Time and Embedded Technology and Applications Symposium (RTAS), 2018 IEEE. IEEE, 2018, accepted.

[38] S. Narayana, P. Huang, G. Giannopoulou, L. Thiele, and R. V. Prasad, "Exploring energy saving for mixed-criticality systems on multi-cores," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2016 IEEE*. IEEE, 2016, pp. 1–12.

[39] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *arXiv preprint arXiv:1703.06058*, 2017.

[40] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 369–392, 2014.

[41] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, "Taming heterogeneity-the ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003.

[42] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[43] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 11.

[44] T. Laukkarinen, J. Suhonen, and M. Hännikäinen, "A survey of wireless sensor network abstraction for application development," *International Journal of Distributed Sensor Networks*, vol. 8, no. 12, p. 740268, 2012.

[45] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[46] L. M. Vaquero, F. Cuadrado, Y. Elkhatib, J. Bernal-Bernabe, S. N. Srirama, and M. F. Zhani, "Research challenges in nextgen service orchestration," *Future Generation Computer Systems*, vol. 90, pp. 20–38, 2019, https://www.sciencedirect.com/science/article/pii/S0167739X18303157.

[47] X. Zhu, C. He, K. Li, and X. Qin, "Adaptive energy-efficient scheduling for real-time tasks on dvs-enabled heterogeneous clusters," *Journal of parallel and distributed computing*, vol. 72, no. 6, pp. 751–763, 2012.

[48] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[49] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European conference on computer vision*. Springer, 2014, pp. 391–405.

[50] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[51] K. Chen, M. Lu, G. Tan, and J. Wu, "Crsm: Crowdsourcing based road surface monitoring," in *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*. IEEE, 2013, pp. 2151–2158.