HealthEdge: Task Scheduling for Edge Computing with Health Emergency and Human Behavior Consideration in Smart Homes

Haoyu Wang*, Jiaqi Gong†, Yan Zhuang*, Haiying Shen*, John Lach‡
* *Department of Computer Science*
University of Virginia, Charlottesville, VA, Email: hw8c, yz8bk, hs6ms@virginia.edu
† *Department of Information Systems*
University of Maryland in Baltimore County, Baltimore, MD, Email: jgong@umbc.edu
‡ *Department of ECE*
University of Virginia, Charlottesville, VA, Email: jlach@virginia.edu

***Abstract*—Nowadays, a large amount of services are deployed on the edge of the network from the cloud since processing data at the edge can reduce response time and lower bandwidth cost for applications such as healthcare in smart homes. Resource management is very important in the edge computing since it is able to increase the system efficiency and improve the quality of service. A common approach for resource management in edge computing is to assign tasks to the remote cloud or edge devices just according to several factors such as energy, bandwidth consumption, and latency. However, the approach is insufficiently efficient and falls short in meeting the requirements of handling health emergency when being applied in smart homes for healthcare. In this paper, we propose a task scheduling approach called *HealthEdge* that sets different processing priorities for different tasks based on the collected data on human health status and determines whether a task should run in a local device or a remote cloud in order to reduce its total processing time as much as possible. Based on a real trace from five patients, we conduct a trace-driven experiment to evaluate the performance of *HealthEdge* in comparison with other methods. The results show that *HealthEdge* can optimally assign tasks between the network edge and cloud, which can reduce the task processing time, reduce bandwidth consumption and increase local edge workstation utilization.**

I. Introduction

Although the integration of smart Internet of Things (IoT) and cloud computing enables many applications, completely moving all the computing tasks to the cloud is inefficient in several scenarios. For instance, when the bandwidth is limited and response time requirement is strict, uploading the data to the cloud for processing may incur longer response time and occupy large portion of the bandwidth. A study reported that the bandwidth demand nearly double each year [1, 2], which would even increase the response time. Fortunately, since the IoT devices become increasingly powerful nowadays, the data processing can be executed at the edge of networks, namely the edge computing [3]. By allowing the computation near the data source and avoid unnecessary data movement, edge computing gains several benefits, e.g., providing efficient network operation and fast service delivery to ensure the quality of service and improve the user experience [4–7]. Therefore, recently, an increasing amount of services are pushed back to the edge of the network from the cloud to reduce response time and lower bandwidth cost. The wide proliferation of IoT and mobile devices with more powerful computation ability changes the role of edge computing device from data consumer to data producer/consumer in the entire computing paradigm.

Meanwhile, the healthcare domain starts to leverage edge computing to improve healthcare services. The healthcare platform deployed in a smart home enables utilizing sensors and mobile devices, and scaling data storage and processing power, for different kinds of healthcare analysis. It enables the sharing of analytical results and accessing to the processing and storage infrastructure with reduced response time and optimized resource utilization [3, 8–10]. For example, in a standard healthcare scenario, assisted persons are monitored by many different sensors gathering data and processing the data in edge workstations (i.e., servers) or the private cloud data center managed by the hospital. The doctors can diagnose and make decisions with these data processing results. Computing tasks on processing the collected data facilitate inferring complex human behaviors [11] and diagnosing possible diseases like neurologic disorders [12], multiple sclerosis diagnosis [13], fall detection [14], respiratory failure in the elderly [15], alzheimer's disease [16] and coronary heart disease [17]. Also, this healthcare platform provides an infrastructure for large-scale data analysis for health in communities, public and global.

In edge computing, resource management plays an significant role for assigning tasks (that are generated by local devices) to the remote cloud (the central of the network) or local servers/devices (the edge of the network). A common approach for resource management in edge computing is to assign tasks to the remote cloud or local servers according to several factors such as energy, bandwidth consumption and latency. These methods can be generally classified into three categories based on their goals: (1) reducing energy consumption (e.g., [18]), (2) improving system throughput (e.g., [19]), and (3) reducing task completion time (e.g., [18]). For instance, Aazam *et al.* [19] proposed a predication model to dynamically estimate the resource utilization based on users' behaviors and allocate resource accordingly. Huerta-Canepa *et al.* [20] proposed an application offloading decision-making scheme based on the application features. Cuervo *et al.* [21] proposed a resource management strategy to estimate energy consumption of the mobile platform and offload the computation task in order to accommodate the network bandwidth and latency changes. Jia *et al.* [22] proposed an online task management algorithm to minimize the the application completion time by considering the general task dependency in the mobile device.

Though these methods are effective in achieving their goals, they are insufficiently efficient and falls short in meeting the requirements of the aforementioned healthcare platforms at smart homes. First, the healthcare platforms have higher requirement on the task latency in order to ensure patients' safety. Second, compared with other application domains, the healthcare platforms are more sensitive to network dynamics (e.g., changes of available bandwidth and latency). In this paper, we consider the resource management in the healthcare platforms in smart homes between the network edge (e.g., smart home sensors) and the central of the network (e.g., the private cloud data center). Specifically, we propose a task scheduling approach called *HealthEdge* that sets different processing priorities for different tasks based on the collected data on human health status, and determines whether a task should run in a local device or a remote cloud in order to reduce its total processing time as much as possible. In *HealthEdge*, the system architecture is separated into two tiers, where the private cloud data center is in the first tier, the network edge including sensors and edge workstations are in the second tier. All the tasks are generated by the sensors and then the tasks will be transferred to the edge workstation. Our contributions in this paper are as follows:

1. We first formulate the task scheduling resource management problem and then prove that it is an NP-hard problem.
2. We propose a heuristic resource management *HealthEdge* that sets different priorities for different tasks based on the human health status to decide each task's execution location (the edge workstation or the private cloud center) and its location in the task waiting queue, which gives higher-priority tasks to start earlier and also minimize the task processing time.
3. We construct a trace-driven simulation to evaluate the performance of *HealthEdge* on bandwidth utilization, task processing time, edge workstation CPU utilization and the scheduling time, as well as the processing time for emergency tasks.

We believe *HealthEdge* can also improve existing resource management approaches in the edge computing domain for other applications by differentiating different task priority and considering the human status. The security of health tasks is very important to protect the users' privacy but it is not our focus in this paper and we leave it as our future. The rest of this paper is organized as follows. In Section II, we present the background and related work on resource management in healthcare systems and in edge computing. In Section III, we introduce the architecture of *HealthEdge*, formulate the task assignment into mathematical problem and prove that it is NP-head. In Section IV, we present the *HealthEdge* resource management system, which focuses on task scheduling and the queue optimization. In Section V, we present trace-driven experimental results for *HealthEdge* in comparison with other methods. Finally, Section VI presents conclusions and discusses the future work.

II. Related Work

There is an increasing interest in utilizing wireless and mobile technologies for improving the performance of healthcare systems recently, especially with the rise of the cloud computing and Internet of Things (IoT). Meanwhile, there are many increasing interests on the research topics of mobile edge computing and its resource management. Below, we present the related work regarding health IoT systems, interactions between human behavior and health IoT systems, edge computing and its resource management.

A. Health IoT Systems

In the last decade, smart home technologies designed for specific research purpose has been developed for persons with dementia [23], assistive living for seniors [24], individuals with obesity [25], and so on. Recently, researchers started rethinking the technological trend of utilizing wireless and mobile technologies in healthcare applications. For instance, Stankovic [26] discussed the research directions of these systems in wireless and mobile healthcare applications, specifically emphasizing how to develop interventions for patients optimally, and also indicted [27] the potential trend of emerging IoT for healthcare, specifically discussing the knowledge exploration in the vast amount of data generated from the IoT sensors. Bakar [28] reviewed the activity monitoring and anomaly detection in the smart homes.

There is also an emerging technological trend that integrate cloud computing and IoT for healthcare applications. Fortino *et al.* [29] presented a framework for integrating body sensor networks and cloud computing to solve the technical issues such as heterogeneity of sensor data, resources scalability (i.e. storage and processing power) and pricing of services. Gravina *et al.* [30] developed a specific cloud-based activity monitoring system leveraging cloud computing techniques. Abawajy *et al.* [31] developed a pervasive patient health monitoring system and demonstrated how the integration of cloud computing and IoT makes the system flexible, scalable, and energy-efficient. Besides such general

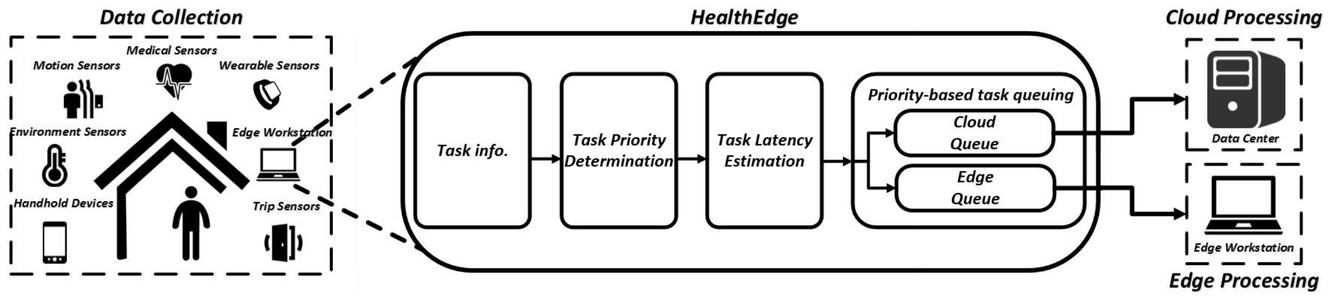

Figure 1: The overview of *HealthEdge*, which consists of three major components: task priority determination, latency estimation and priority-considered task queuing.

purpose systems, Sareen *et al.* [32] developed a specific system to detect epileptic seizures using cloud computing and sensor networks to provide a real-time response to the seizure events.

The significant interconnections between human behavior and health IoT systems have been investigated in various dimensions. This human-centric computing perspective enhanced the performance of the health IoT systems in many ways. For instance, Chen *et al.* [33] integrated human behavior into the energy management system to optimize the operation efficiency. Ji *et al.* [34] utilized the wearable sensor data to analyze human behavior to increase the system awareness of personal demands. Huang *et al.* [35] emphasized that the prediction abilities of smart home technologies could enhance the relationship between the system and human. Also, Yang *et al.* [36] investigated the user acceptance of smart home services from the perspectives about human behavior.

B. Mobile Edge Computing

More recently, several works have been conducted to promote the edge computing research in term of infrastructure development. Sun *et al.* [37] designed a novel architecture, edgeIoT, to process the collected data streams efficiently at the mobile edge. Jang *et al.* [38] developed an edge cloud infrastructure that provides a new abstraction for the application developer to manage the sensors and actuator on the edge cloud side. Nastic *et al.* [39] proposed a middleware infrastructure between the IoT and cloud, which is able to provide a generic and light-weight abstraction to customize and manage the edge applications and resources. Also, the development of mobile edge computing infrastructure facilitates the cloud service and improve the user experience. For instance, Shi *et al.* [3] indicated that utilizing edge computing can reduce the shopping-cart update latency and enhance the user experience for online shopping.

Resource management is an important research topic for mobile edge computing. One key of this problem is to determine which task should be offloaded to the cloud and when it should be offloaded. Many research studies investigated this topic. Rudenko *et al.* [40] claimed that the energy can be saved by offloading task to remote servers. Cuervo [21] proposed a code offload solution for mobile edge computing in order to accommodate the network bandwidth and latency changes. Chun *et al.* [41] presented the CloneCloud, which uses a static code analyzer to determine which task should be offloaded. Aazam *et al.* [19] proposed a predication model to dynamically estimate the resource utilization based on users' behaviors and allocate resource accordingly. However, they did not consider to prioritize the task, therefore, some mission/safety-critical tasks can suffer longer latency.

These computation offloading schemes are unsuitable for healthcare systems. First, the properties of healthcare dataset is quite different with the dataset from other applications, e.g, larger size of the dataset obtained from the long-term monitoring. Beside, these methods failed to consider the impact of human dynamics on the network dynamics. Furthermore, some tasks in the health IoT applications should be prioritized based on clinician or user's preference, e.g., physiological signal monitoring has higher priority over environment monitoring. Therefore, the current computation offloading schemes are not applicable to the health IoT applications and we aim to design a new task scheduling system for the healthcare IoT systems.

III. System Architecture and Problem Formulation

HealthEdge considers task emergency and human behavior in task scheduling for the optimization of the resource management. That is, it determines whether a task should be executed in the cloud or a local server (i.e., edge device). In the rest of the section, first, we describe the architecture of *HealthEdge*. Second, we formulate the problem for the resource management optimization for automatically allocating the tasks and resources to minimize the task latency.

A. System Architecture

Figure 1 demonstrated the architecture of the *HealthEdge* health IoT system for a person's living apartment. Each sensor monitors the variables from the human body directly or the house environment. When the sensors are monitoring their targets, they also send the live data sets to the edge workstation concurrently. The edge workstation periodically (different tasks have different time periods) processes these data sets and then sends the results to the private cloud data center or sends all the unprocessed data sets to the private cloud data center directly. The private cloud data center collects all the results (generated by the private cloud data center or the edge workstation) and then send them to the doctors to make decision for the users. For some emergency situation, reducing the emergency-related task processing time is very important. In the meantime, we also need to fully utilize the edge workstation, save the bandwidth cost to achieve better performance.

The tasks cane be divided into several layers, including monitoring layer, processing layer, a temporary storage layer, security layer, transport layer, and user interface layer. Different layers contain scalable tasks. All of these parameters might be captured in reconfigurable sampling rates which depend on the needs of the processing layer and the inputs from the user interface layer. The tasks in these layers can be deployed in local computation resources such as the microprocessor of the sensor nodes, or in global resources such as the near-the-edge cloud.

HealthEdge sets up a scenario with the consideration of the demands of health monitoring and real-time health services responses at the smart home. The wireless and mobile sensor nodes are equipped for capturing behavioral and environmental parameters. Also, the nodes have abilities for computation, storage, and wireless communication to the Internet. There are edge workstation(s) at the smart home. Near-the-edge private cloud data center is deployed to manage and support the sensor nodes, including computation resources and data storage.

B. Problem Formulation

In this section, we formulate the problem of task assignment among the edge workstations and the private cloud data centers. We focus on edge workstations here and our work can be easily extended to edge devices by including edge devices as task running options. Table I lists the major notations used in this paper. We use $X_{s_i}^{t_k}$, a binary variable, to denote the assignment of task t_k to edge workstation s_i or to the private cloud data center. We use $U_{s_i}^{t_k}$ to denote the assigned CPU capacity for the task t_k in edge workstation s_i. Then, we can have the task assignment f represented as a set of mappings below:

$$\begin{aligned} f =\{&\langle s_1, (X_{s_1}^{t_1} \cdot U_{s_1}^{t_1}, X_{s_1}^{t_2} \cdot U_{s_1}^{t_2}, ..., X_{s_1}^{t_k} \cdot U_{s_1}^{t_k})\rangle, ..., \\ &\langle s_i, (X_{s_i}^{t_1} \cdot U_{s_i}^{t_1}, X_{s_i}^{t_2} \cdot U_{s_i}^{t_2}, ..., X_{s_i}^{t_k} \cdot U_{s_i}^{t_k})\rangle, ...\}\end{aligned}$$

For a set of tasks $\mathcal{J}$ generated in a certain period of time, we aim to minimize the total processing time of the tasks without overloading any edge workstations. Another important consideration is the network traffic load, caused by data transmission from the edge workstations to the private cloud data center. We assume that all the data needed by task t_k is stored in one edge workstation. We use $D_{s_i}^{t_k}$ to denote the size of the data needed by task t_k, and that all

Table I: Notations.

Notation	Description
$\mathcal{M}$	entire edge workstation set
s_i	edge workstation/server i ($s_i \in \mathcal{M}$)
$\mathcal{J}$	entire task set
t_k	task k ($t_k \in \mathcal{J}$)
Γ^{t_k}	emergency level of task t_k
$X_{s_i}^{t_k}$	the execution destination of task t_k
$D_{s_i}^{t_k}$	size of data needed by task t_k, which is stored on s_i
C^{t_k}	computing resources needed by t_k
$U_{s_i}^{t_k}$	assigned CPU capacity for t_k on server s_i
B_{s_i}	available bandwidth between s_i and the cloud
Φ_f	traffic load for task assignment schedule f
Ω_f	total processing time for task assignment schedule f
T	a unit time period

the data is stored in edge workstation s_i. The total network traffic load of all the tasks is calculated by:

$$\Phi_f = \sum_{t_k \in (X_{s_i}^{t_k}=1)} D_{s_i}^{t_k}.$$

We need to minimize network traffic load and fully utilize the computing capacity of edge workstation. Task t_k has an emergency level denoted by Γ^{t_k} ($0 \leq \Gamma \leq 1$) which represents the urgent level of the task. A higher Γ value means the task is more urgent and vice versa. In order to lower the processing time of more urgent tasks, we have the weighted processing time Ω of task t_k. Thus, the total weighted processing time for f is

$$\Omega_f = \sum_{t_k \in (X_{s_i}^{t_k}=0)} \Gamma^{t_k} \cdot \frac{C^{t_k}}{U_{s_i}^{t_k}} + \sum_{t_k \in (X_{s_i}^{t_k}=1)} \Gamma^{t_k} \cdot \frac{D_{s_i}^{t_k}}{B_{s_i}}.$$

We aim to find a task assignment plan f, so that the traffic load and the weighted average processing time need to be minimized. Thus, we formulate the problem of optimized task assignment as a nonlinear programming as:

$$\text{Min} \quad \Phi_f \cdot \Omega_f \tag{1}$$

$$\text{subject to} \quad \sum_{t_k \in \mathcal{J}^T} D_{s_i}^{t_k}/T \leq B_{s_i}, \quad \forall s_i \in \mathcal{M}, \ \forall t_k \in \mathcal{J} \tag{2}$$

$$\sum_{t_k \in \mathcal{J}} U_{s_i}^{t_k} \cdot X_{s_i}^{t_k} \leq U_{s_i}, \forall s_i \in \mathcal{M} \tag{3}$$

$$X_{s_i}^{t_k} \in \{0,1\}, \quad \forall s_i \in \mathcal{M}, \ \forall t_k \in \mathcal{J} \tag{4}$$

$$0 \leq \Gamma^{t_k} \leq 1, \quad \forall t_k \in \mathcal{J} \tag{5}$$

In constraint (2), $\mathcal{J}^T$ is the task set that includes the tasks transferred from edge workstation s_i to the private cloud data center during T. This constraint ensures the tasks can not occupy bandwidth more than the available bandwidth. Constraint (3) ensures that the total computing CPU utilization occupied by the tasks can not exceed the available CPU utilization for each edge workstation. Constraint (4) ensures that the task will be assigned on the edge workstation or the private cloud data center. 0 and 1 represent that the task is assigned on the edge workstation or the private cloud data center, respectively. Constraint (5) limits the emergency level Γ^{t_k} is between 0 to 1.

Lemma: The formulated task assignment problem is NP-Hard.

Proof. Suppose that all the edge workstations are homogeneous with same computing capacity and storage capacity. Assume that the private cloud has sufficient computing capacity to process many tasks, the emergency levels of all the tasks are the same, and we consider the traffic load cost to the cloud. Then, the task assignment problem can be created as a task assignment schedule to minimize the network load traffic under computing capacity constraints and bandwidth limitation between all edge workstations and the private cloud, which is a weighted bin packing problem [42]. Since the weighted bin packing problem is NP-hard, our efficiency task assignment problem is also NP-hard. We then propose our heuristic *HealthEdge* system to solve this problem in section IV.

IV. Design of *HealthEdge*

HealthEdge is a heuristic method to solve the task assignment problem that aims to minimize the task processing time and save the bandwidth consumption while meeting the requirement on handling emergency tasks. It decides whether to assign a task to an edge workstation at home or the private cloud data center owned by the hospital or government, and which edge workstation to choose if the former option is chosen.

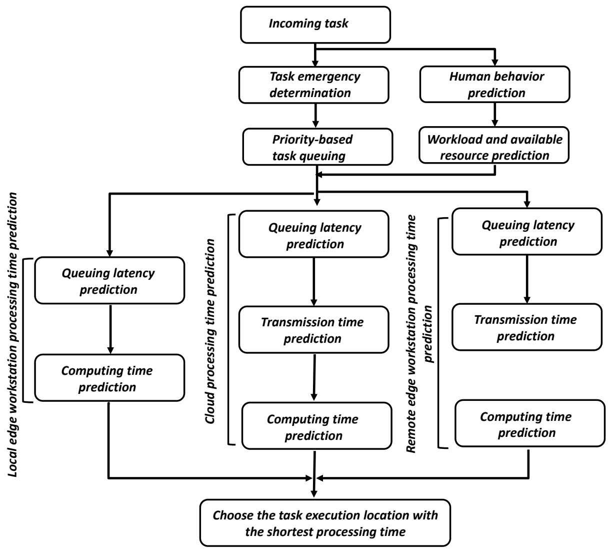
Incoming task
Task emergency determination
Human behavior prediction
Priority-based task queuing
Workload and available resource prediction
Local edge workstation processing time prediction
Queuing latency prediction
Computing time prediction
Cloud processing time prediction
Queuing latency prediction
Transmission time prediction
Computing time prediction
Remote edge workstation processing time prediction
Queuing latency prediction
Transmission time prediction
Computing time prediction
Choose the task execution location with the shortest processing time

Figure 2: Flowchart showing how to assign a task to the edge workstation or the private cloud data center.

Figure 2 shows the flowchart illustrating how to assign a task to the edge workstation or the private cloud data center. Given an incoming new task, *HealthEdge* determines the task emergency level (Section IV-A), and then determines its location in the queue of each possible task execution location (Section IV-B). As we will describe in Section IV-C, *HealthEdge* also predicts human behavior to predict the workload and available resource of the task execution location options. The option can be the local edge workstation that has the task's required datasets, a remote edge workstation that does not have the task's required datasets, and the cloud. Based on the location in the queue and the predicted available resources of each option, it calculates the task processing time of each option. Finally, it chooses the the task execution location with the shortest processing time.

A. Task Emergency Determination

We first introduce how to determine a task's emergency level Γ based on the information extracted from the sensors. This Γ is a metric to describe the emergency level of one certain task, which will affect the schedule about whether the task needs to be processed in the edge workstation or the private cloud data center.

The emergency level is determined by the sensed data or the data processing results. Here, we use simple range checking for the human body sensed data. It can be easily extended to consider other more complex data processing results such as heart attack symptom initial observation. According to the real trace we use, we select five representative features of human body: 1) body temperature from temperature sensor; 2) blood glucose level from glucose monitor; 3) heart beat rate from ECG (Electrocardiogram) sensor; 4) motion information from accelerator and gyroscope sensor; and 5) blood pressure and blood oxygen saturation from pulse oximeter sensor.

According to the historical log of the human behavior features listed above, we generates the *reference range* with upper and lower bounds for each feature. We use t-distribution to estimate the range which is widely used in the healthcare area [43]. Following t-distribution, for a normal sample size of n $(n > 0)$, x is the sample feature like the body temperature. The mean $(\overline{x})$ is computed as

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{6}$$

Then the standard deviation of the sample feature s_x is:

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{7}$$

In this way, the upper bound v_u and lower bound v_l can be represented as:

$$v_u = \overline{x} - \sqrt{\frac{n+1}{n}} \cdot s_x \cdot t_{\alpha,n-1}, \tag{8}$$

$$v_l = \overline{x} + \sqrt{\frac{n+1}{n}} \cdot s_x \cdot t_{\alpha,n-1}, \tag{9}$$

where $t_{\alpha,n-1}$ donates the standard t-distribution coefficient for sample size n. We group samples by age and gender, and calculate the reference range for each group. Given the age and gender information of a person, the reference range of a particular health parameter being monitored should be within v_l and v_u, which represents the normal health conditions. At a given time t, the recorded value of particular health parameter is denoted by v_t. Then Γ at the given time t can be calculated as:

$$\Gamma = \left| \frac{(v_u - v_t)^2 - (v_l - v_t)^2}{(v_u - v_l)^2} \right| \tag{10}$$

A higher Γ value means a higher emergency level and vice versa. Clearly, when $v_t = (v_l + v_u)/2$, $\Gamma = 0$, which represents the most un-emergency situation. Theoretically, in some cases, the value of Γ may exceed 1. In order to unify the value, for all $\Gamma > 1$, we limit the maximum value by changing $\Gamma = 1$ which is the highest emergency level.

For example, for body temperature, the reference rage is $v_l = 35.5\ and\ v_u = 37.5$. Then, if the body temperature is 36.5 degree, the emergency level will be 0, which means it is not an emergency situation. If the body temperature is 38.5 degree, $\Gamma = 0.75$, which represents an emergency situation. Once the body temperature is 39.5 degree, $\Gamma = 1$ ($\Gamma = 3$ and is unified to 1), indicates the most emergency situation. All the emergency tasks need to be processed immediately.

B. Priority-based Task Queuing

Each edge workstation maintains two queues: cloud queue and edge queue. The cloud queue contains all the tasks that the workstation will send to the cloud, and the edge queue contains all the tasks that will run in the workstation. Recall that each task has an emergency level Γ. We set a threshold L; if $\Gamma > L$, the task must be executed immediately. Otherwise, the task is delay tolerable. We call these tasks emergency tasks and non-emergency tasks.

For emergency tasks, rather than using the First-in-first-out (FIFO) rule, we order the tasks in each queue based on the descending order of the task emergency levels. This way, higher emergency tasks will be executed at an earlier time, which will help meet the requirement on emergency handling in the health IoT system. We further propose a method to order the emergency tasks with the same emergency levels, and the non-emergency tasks, as presented below.

Different from the tasks in the edge queue, for the tasks in the cloud queue, their datasets for processing need to be transmit to the cloud, which generate a certain latency. Therefore, if a task with a larger dataset starts earlier than other tasks with smaller datasets, it will delay the other tasks. Also, if a task has waited in the queue for a longer time, it should give a higher priority to start. Otherwise, tasks with lower emergency or larger datasets will be delayed for a long time. In order not to delay the task processing in general, we further determine the priority of emergency tasks with the same emergency levels, and the non-emergency tasks in the cloud queue and edge queue based on the following two equations, respectively.

$$P_{t_k} = \frac{\Gamma^{\alpha} \cdot (T_{t_k})^{\beta}}{D^{t_k}}, \tag{11}$$

$$P_{t_k} = \Gamma^{\alpha} \cdot (T_{t_k})^{\beta}, \tag{12}$$

where α and β are used to give different weights to different factors. A new task is inserted to a queue based on the tasks' priority values. As a result, emergency tasks can run as fast as possible. For delay-tolerant tasks, tasks with longer queuing time and smaller datasets (if the tasks are in the cloud queue) will have a higher priority to be processed earlier. This method not only considers to meet the emergency handling requirement but also helps reduce task processing latency and increase throughput.

C. Task Latency Estimation and Task Scheduling

As we indicated previously, human behaviors influence the workload of edge workstations. Researchers have developed supervised and unsupervised machine learning techniques [44, 45, 11] to extract information from sensed data for human behavior inference[46, 11, 47, 48, 33, 34]. We use the existing methods to predict the human behaviors and then accordingly predict the workloads at each time in a future time period using the machine learning techniques [49]. The workloads include the computation workload for each workstation, the available network bandwidth between each workstation and the cloud.

Therefore, we can use the historical log about a task (such as processing time, dataset size, available bandwidth) to predict resource requirement and processing time of a similar task. Based upon the historical log in the edge workstation, task t_n has the previous computation time $T_n'^{cmp}$ and CPU utilization U_n'. If *HealthEdge* assigns the task t_n on an edge workstation, the edge workstation inserts it into the task queue (denoted by Q_s) which already contains the previous assigned tasks based on the method introduced in Section IV-B. The queuing latency $T_{t_n}^{que}$ of task t_n is the sum of the processing times of the tasks processing task t_n at the head of Q_s:

$$T_{t_n}^{que} = \sum_{i=1}^{n_s-1} T_{t_i}^{cmp},\ \forall t_i \in Q_s, \tag{13}$$

where n_s is the number of tasks proceeding task t_n in the queue. The computing time is inversely proportional to the computing ability. Then, the predicted computation time of a task $T_{t_i}^{cmp}$ is:

$$T_{t_i}^{cmp} = \frac{U_{s_i}^{\prime t_i}}{U_{s_i}^{t_i}} \cdot T_{t_i}^{\prime cmp}, \tag{14}$$

where $U_{s_i}^{t_i}$ means the predicted CPU capacity of the edge workstation that task t_n can use and $U_{s_i}^{\prime t_n}$ means the previous CPU capacity supplied to a similar task as t_i by the edge workstation. Recall that the future CPU capacity is predicted based on the human behavior as explained above. Accordingly, $t_{t_n}^{cmp}$ can be calculated. From Equations (13) and (14), the predicted processing time (denoted by $t_{t_n}^{p}$ of task t_n assigned on the edge workstation equals:

$$T_{t_n}^{p} = T_{t_n}^{cmp} + T_{t_n}^{que} \tag{15}$$

If *HealthEdge* assigns the task t_n to the private cloud data center, the edge workstation puts it in a data transmission queue Q_d which consists the previous assigned tasks to the private cloud data center. The computing ability of data center is much higher than the edge workstation. Furthermore, it has much more computing slots to calculate the tasks. Thus, we neglect the tasks waiting time within data center. The queuing latency equals:

$$T_{t_n}^{que} = \sum_{i=1}^{n_d-1} T_{t_i}^{tran}, \ \forall t_i \in Q_d, \tag{16}$$

where $n_d - 1$ is the number of tasks proceeding task t_n in cloud queue Q_d. The predicted data transmission time of task i can be calculated as:

$$T_{t_i}^{tran} = \frac{D_{s_i}^{t_i}}{B_{s_i}}, \tag{17}$$

where $D_{s_i}^{t_i}$ denotes the size of all data set needed by task t_i and B_{s_i} means the current available bandwidth between the edge workstation s and the private cloud data center. Recall that B_{s_i} is predicted based on the human behavior as explained above. Then, data transmission time for task t_n (T_n^{tran}) can be calculated based on Equation (17).

According to Equation (14), the predicted computing time on the private cloud data center $T_{t_n}^{cmp}$ can be calculated by:

$$T_{t_n}^{cmp} = \frac{T_{t_n}^{cmp}}{\Theta} \tag{18}$$

where Θ is the difference ratio of computing ability between the edge workstation and the data center. This parameter can be set depending on the practical situation. Finally, we calculate the predicted total processing time on the private cloud data center of task t_n, which is the sum of the queuing time, data transmission time and computation time:

$$T_{t_n}^{p} = T_{t_n}^{que} + T_{t_n}^{tran} + T_{t_n}^{cmp}. \tag{19}$$

Using the same method as the above, we can calculate the processing time of task t_n if we assign it to another workstation that does not have its required datasets. Finally, *HealEdge* chooses the edge workstation or the cloud that leads to the shortest task processing time.

V. Performance Evaluation

A. Experimental Setup

In this section, we conduct trace-driven experiment on JAVA-based simulation. In the simulation, there are 50 to 300 workstations and one private cloud data center in our system. Each workstation is located in one apartment and connected to 5 sensors. 1) Temperature sensor; 2) Glucose monitor; 3) ECG sensor; 4) Accelerator and gyroscope sensor; 5) and Pulse oximeter sensor. We assume the storage capacity of sensors and workstations are not the bottleneck in our simulation. We set the bandwidth as 100Mbps between each workstation to the private data center. There are 60 nodes with Intel E5-2650 v4, 32 GB RAM in the simulated private data center. The computing ability of one node in the private data center is 200 times as that of an edge workstation.

B. Workload Description

Our team has been engaging in the development of home monitoring systems and the exploration of these systems in numerous clinical and environmental contexts. The trace for driving the large-scale simulation in this study is from our real-world deployment of a customized monitoring system for persons with dementia in central Virginia area. Our previous work [23] has described the system, sensors, collected data and related tasks in signal processing and modeling. Here, we only highlight the overview of this trace. This one-month (from Dec. 1 to Dec. 31 in 2016) dataset consists of the human behavior dataset (e.g., physiological signal and activity datasets) and environment datasets (e.g., temperature, humidity, light, and noise datasets). Specifically, the physiological signal datasets includes heartbeat and Galvanic Skin Response (GSR) data. The sampling frequency is scalable depending on the resolution of the information required. ECG data in high sampling rate is useful for cardiac disease diagnosis, while ECG data in low sampling rate such as heart rate data could be helpful for energy expenditure estimation or fitness monitoring. The tasks for this data contain data preprocessing (e.g. QRS wave detection, RR interval or heart rate variability extraction, and denoising), anomaly detection, and other medical analysis.

C. Comparison Methods

We compare *HealthEdge* with the following methods: *Distributed*, *Centralized* and *Spanedge* [50]. *Distributed* assigns all the tasks generated within each workstation area on the local workstation as much as possible. If the workstation is overloaded, all the other tasks are added in the waiting queue based on First-in-first-out. *Centralized* assigns all the tasks generated by all the workstations areas on the private data center even though some tasks need huge amount of data (which increases the network load tremendously). *SpanEdge* groups all the tasks into two parts: local-task and global task, where the local-task refers to the task which needs data only from one workstation and a global task refers to the task that requires the results of a group of local-tasks. Based on this classification, local-task is assigned to the edge workstation and the global-task is assigned to the cloud data center. In *HealthEdge*, if the datasets needed by a global task are stored in multiple edge workstations, the task will be assigned to the cloud.

D. Performance on Bandwidth Utilization

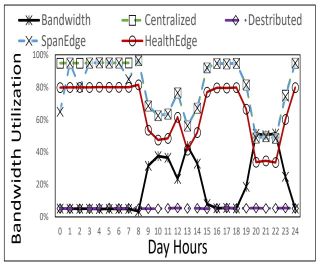

Figure 3: Bandwidth utilization.

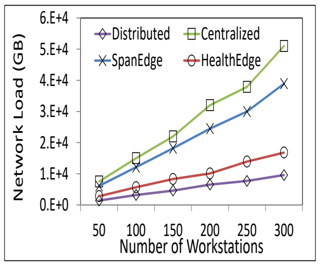

Figure 4: Network load.

In this section, we evaluate the performance on network bandwidth utilization for all methods. We attempt to check if the methods use most of the available bandwidth, leaving little bandwidth to users for daily use and hence adversely affecting user experience. In Figure 3, we show the average bandwidth utilization at each hour per day per smart home from the one-month trace. *Bandwidth* means the average bandwidth utilization by the user's family excluding the bandwidth utilization for the health systems. We can see that the bandwidth utilization varied a lot over time. From 9:00 to 14:00 and 20:00 to 23:00, the bandwidth utilization is almost 50% on average. In order to provide better user experience, a task assignment method should limit the network bandwidth usage and leave enough bandwidth to users for daily use. *Centralized* assigns all the tasks on the private cloud and the total bandwidth utilization is 100%. Because *Centralized* occupies most of the bandwidth, it not only cannot leave sufficient bandwidth to users for daily use, but also generates two problems. First, it increases network cost to the users. Second, it can cause the network congestion, prevents timely data transmission, and finally cannot finish all the tasks on time. *Distributed* generates the lowest bandwidth utilization among all the methods. This is because it assigns all the tasks on the local workstation so that each workstation doesn't need to transfer data to the private cloud, which saves the bandwidth utilization. Although *Distributed* achieves the lowest bandwidth utilization, the computing capacity of local workstation is much lower than the private cloud. It can not complete all the tasks on time especially for those emergency tasks. Though *SpanEdge* only assigns global-tasks to the cloud, it also uses most of the bandwidth, leading to nearly 100% total bandwidth utilization. Thus, *SpanEdge* generates the same problems as *Centralized*. *HealthEdge* assigns the tasks based on their dataset size and predicted available bandwidth at that time. Its total utilization is from 85% to 90%. Therefore, *HealthEdge* can avoid the problems of *SpanEdge* and *Centralized*. That is, *HealthEdge* leaves a certain amount of bandwidth to users for daily use, it constrains network cost and won't cause network congestion which enable to complete tasks timely.

We also measured the total network load in Figure 4. It shows the number of workstations versus the total network load in GB. It shows that the network load performance follows *Centralized*> *SpanEdge*>*HealthEdge*>*Distributed*. *Centralized* assigns all the tasks to the private cloud so that it generates the highest network load. *SpanEdge* assigns the tasks which need the data from other workstations to the cloud. Because it assigns some tasks only use local data, it can achieve lower network load than *Centralized*. *HealthEdge* assigns the tasks based on the data size and predicted available bandwidth. To decrease processing time and network load, the tasks that need less amount of data can be assigned on the private cloud and the tasks that need a large amount of data tend to be assigned on the local workstation. Because it assigns some tasks to the private cloud, it costs higher network load than *Distributed*, which assigns all the tasks on the local workstations. Therefore, Figure 4 shows that *HealthEdge* can achieve lower network load compared with *SpanEdge*.

E. Performance on Task Processing Time

Figure 5 shows the total ask processing time in seconds versus the number of workstations versus. The result follows *Distributed* >*Centralized*>*SpanEdge*>*HealthEdge*. *Distributed* assigns all the tasks to the local workstations which have much less computing capacity than the private cloud. Therefore, it generates the highest processing time. *Centralized* assigns all the tasks to the private cloud. Because the computing capacity of the private cloud is much higher than the local workstations, *Centralized* can achieve much lower task processing time although it costs high network load. *SpanEdge* assigns some tasks which needs data from many workstations to the private cloud so that it can use the private cloud to process large amount of data and achieve the lower processing time than *Centralized*. *HealthEdge* tends to assign computing-intensive task to the private cloud and data-intensive tasks to the local workstation. It fully utilizes the computing capacity of local workstation and saves the data transmission latency. As a result, *HealthEdge* can achieve the best performance on the task processing time.

We consider the tasks with $\Gamma \geq 0.8$ as emergency tasks.

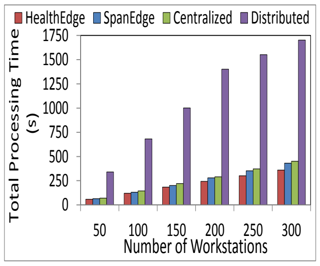

Figure 5: Total processing time.

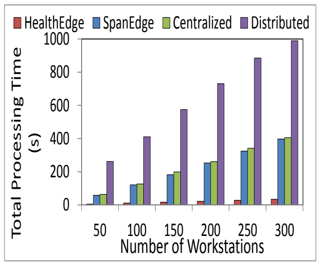

Figure 6: Total processing time on emergency task.

Figure 6 shows the total processing time of the emergency tasks versus the number of workstations. The result follows *Distributed>Centralized >SpanEdge>HealthEdge*. The difference between Figure 6 and Figure 5 is that *HealthEdge* is much lower than *SpanEdge* in Figure 6. In *Centralized*, *Distributed* and *SpanEdge*, all the tasks have the same priority to be processed so that those emergency tasks may not be processed faster than other regular tasks. It leads to higher latency for the emergency tasks than that in *HealthEdge*. *HealthEdge* determines the priority of tasks based on the human health status in the task assignment. For those emergency task (with higher Γ values), they have higher priorities to be processed with shorter queuing latency. Therefore, *HealthEdge* can achieve the shortest processing time for emergency tasks among the different methods.

F. Performance on CPU Utilization

Figure 7 shows the average CPU utilization of local workstation versus the number of workstations. It shows that the CPU utilization percentage follows *Distributed>HealthEdge>SpanEdge>Centralized*. *Centralized* assigns all the tasks to the private cloud so that it costs no CPU utilization of local workstations. *SpanEdge* assigns tasks which need the data from other workstations to the cloud and assigns tasks which need data only from the local workstations on the local workstations. Without considering the computing capacity of the edge workstations, it achieves almost 100% CPU utilization, which is higher than that of *HealthEdge*. *HealthEdge* assigns the tasks based on the predicted processing time on the edge workstation and the private cloud data center. The tasks that need less amount of data but high computing requirement tend to be assigned to the private cloud. The tasks that need a larger amount of data but lower computing requirement tend to be assigned on the local workstation. In the meantime, when *HealthEdge* assigns tasks, it considers the computing capacity on each workstation for latency estimation, so that it achieves higher CPU utilization but lower than 100%, which means it can avoid overloading the workstations. *Distributed* assigns all the tasks on the local workstation so that the CPU utilization is 100%, which actually cannot guarantee the performance of the tasks due to overload. Therefore, the result shows that *HealthEdge* can achieve high CPU utilization of local workstations without overloading them.

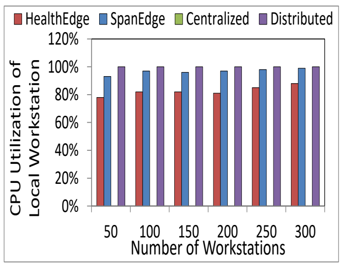

Figure 7: CPU utilization.

VI. Conclusion and Future Work

In this paper, we addressed the task scheduling problem for resource management in the edge computing for health IoT systems at smart homes. The solution determines whether to run a task in an edge workstation or the remote private cloud data center. We first formulate the problem, which has been proved NP-hard. We then proposed *HealthEdge*, a heuristic solution for assigning tasks between the edge workstation and the private cloud data center. Based on human health status from the sensed data, we set different priorities on different tasks based on tasks' emergency levels. Then, we propose a priority-based task queuing method that enables emergency tasks to be processed earlier. Meanwhile, it avoids delaying waiting tasks according to the task waiting time and processing time. Further, *HealthEdge* predicts human behaviors and hence available resources for each server and its bandwidths to the cloud. based on which it estimates the data transmission latency, queuing latency and computing latency to predict the total processing time of a task in each edge workstation and the private cloud data center. Finally, *HealthEdge* assigns the task to the destination with the shortest estimated processing time. We construct a trace-driven simulation to evaluate the performance of *HealthEdge* in comparison with other methods.

Acknowledgment

This research was supported in part by U.S. NSF grants OAC-1724845, ACI-1719397 and CNS-1733596, and Microsoft Research Faculty Fellowship 8300751.

References

[1] D. Gilstrap. Ericsson mobility report. *Ericsson*, 2013.
[2] H. Wang, J. Gong, Y. Zhuang, H. Shen, and J. Lach. Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes. In *Proc. of NAS, poster session*, 2017.
[3] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 2016.
[4] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, and A Neal. Mobile-edge computing introductory technical white paper. *Journal of Mobile-edge Computing (MEC) industry initiative*, 2014.
[5] L. Yan, H. Shen, J. Zhao, C. Xu, F. Luo, and C. Qiu. Catcharger: Deploying wireless charging lanes in a metropolitan road network through categorization and clustering of vehicle traffic. In *Proc. of INFOCOM*, 2017.
[6] L. Kang H. Shen and A. Sarker. Velocity optimization of pure electric vehicles with traffic dynamics and driving safety considerations. In *Proc. of ICDCS*, 2017.

[7] H. Shen, G. Liu, and H. Wang. An economical and slo-guaranteed cloud storage service across multiple cloud service providers. *Trans. on TPDS*, 2017.

[8] H. Wang, H. Shen, and G. Liu. Swarm-based incast congestion control in datacenters serving web applications. In *Proc. of SPAA*, 2017.

[9] G. Liu, H. Shen, and H. Wang. Computing load aware and long-view load balancing for cluster storage systems. In *Proc. of Big Data*, 2015.

[10] G. Liu, H. Shen, and H. Wang. Towards long-view computing load balancing in cluster storage systems. *Trans. of TPDS*, 2017.

[11] W. T. Riley, D. E. Rivera, A. A. Atienza, W. Nilsen, S. M. Allison, and R. Mermelstein. Health behavior models in the age of mobile interventions: are our theories up to the task? *Translational behavioral medicine*, 2011.

[12] Y. K. Axelrod and M. N. Diringer. Temperature management in acute neurologic disorders. *Neurologic clinics*, 2008.

[13] J. Gong, M. D. Goldman, and J. Lach. Deepmotion: a deep convolutional neural network on inertial body sensors for gait assessment in multiple sclerosis. 2016.

[14] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In *Proc. of Wearable and Implantable Body Sensor Networks*, 2009.

[15] P. Ray, S. Birolleau, Y. Lefort, M. Becquemin, C. Beigelman, R. Isnard, A. Teixeira, M. Arthaud, B. Riou, and J. Boddaert. Acute respiratory failure in the elderly: etiology, emergency diagnosis and prognosis. *Critical care*, 2006.

[16] J. Gong, K. M. Rose, I. A. Emi, J. P. Specht, E. Hoque, D. Fan, S. R. Dandu, R. F. Dickerson, Y. Perkhounkova, and J. Lach. Home wireless sensing system for monitoring nighttime agitation and incontinence in patients with alzheimer's disease. In *Proc. of Wireless Health*, 2015.

[17] P. C. van den Hoogen, E. J. Feskens, N. J. Nagelkerke, A. Menotti, A. Nissinen, and D. Kromhout. The relation between blood pressure and mortality due to coronary heart disease among men in different parts of the world. *New England Journal of Medicine*, 2000.

[18] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *Trans. on Networking*, 2016.

[19] M. Aazam and E. Huh. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *Proc. of AINA*, 2015.

[20] G. Huerta-Canepa and D. Lee. An adaptable application offloading scheme based on application behavior. In *Proc. of Advanced Information Networking and Applications-Workshops*, 2008.

[21] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *Proc. of Mobile systems, applications, and services*, 2010.

[22] M. Jia, J. Cao, and L. Yang. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. In *Proc. of INFOCOM workshops*, 2014.

[23] R. Alam, J. Gong, and J. Lach. Besi: Reliable and heterogeneous sensing and intervention for in-home health applications. In *Proc. of CHASE*, 2017.

[24] P. Khosravi and A. H. Ghapanchi. Investigating the effectiveness of technologies applied to assist seniors: A systematic literature review. *International Journal of medical informatics*, 2016.

[25] J. J. Smith, P. J. Morgan, R. C. Plotnikoff, K. A. Dally, J. Salmon, A. D. Okely, T. L. Finn, and D. R. Lubans. Smart-phone obesity prevention trial for adolescent boys in low-income communities: the atlas rct. *Pediatrics*, 2014.

[26] J. A. Stankovic. Research directions for cyber physical systems in wireless and mobile healthcare. *ACM Trans. on Cyber-Physical Systems*, 2016.

[27] J. A. Stankovic. Research directions for the internet of things. *Journal of Internet of Things*, 2014.

[28] U. Bakar, H. Ghayvat, S. Hasanm, and S. Mukhopadhyay. Activity and anomaly detection in smart home: A survey. In *Next Generation Sensors and Systems*. 2016.

[29] G. Fortino, D. Parisi, V. Pirrone, and Di F. G. Bodycloud: A saas approach for community body sensor networks. *Future Generation Computer Systems*, 2014.

[30] R. Gravina, C. Ma, P. Pace, G. Aloi, W. Russo, W. Li, and G. Fortino. Cloud-based activity-aaservice cyber–physical framework for human activity monitoring in mobility. *Future Generation Computer Systems*, 2016.

[31] J. H. Abawajy and M. M. Hassan. Federated internet of things and cloud computing pervasive patient health monitoring system. *IEEE Journal of Communications Magazine*, 2017.

[32] S. Sareen, S. K. Sood, and S. K. Gupta. An automatic prediction of epileptic seizures using cloud computing and wireless sensor networks. *Journal of medical systems*, 2016.

[33] S. Chen, T. Liu, F. Gao, J. Ji, Z. Xu, B. Qian, H. Wu, and X. Guan. Butler, not servant: A human-centric smart home energy management system. *IEEE Journal of Communications Magazine*, 2017.

[34] J. Ji, T. Liu, C. Shen, H. Wu, W. Liu, M. Su, S. Chen, and Z. Jia. A human-centered smart home system with wearable-sensor behavior analysis. In *Proc. of CASE*, 2016.

[35] F. Huang and S. Tseng. Predictable smart home system integrated with heterogeneous network and cloud computing. In *Proc. of ICMLC*, 2016.

[36] H. Yang, H. Lee, and H. Zo. User acceptance of smart home services: an extension of the theory of planned behavior. *Industrial Management & Data Systems*, 2017.

[37] X. Sun and N. Ansari. Edgeiot: Mobile edge computing for the internet of things. *Journal of Communications Magazine*, 2016.

[38] M. Jang, H. Lee, K. Schwan, and K. Bhardwaj. Soul: An edge-cloud system for mobile applications in a sensor-rich world. In *Proc. of Edge Computing (SEC)*, 2016.

[39] S. Nastic, H. Truong, and S. Dustdar. A middleware infrastructure for utility-based provisioning of iot cloud systems. In *Proc. of SEC*, 2016.

[40] A. Rudenko, P. Reiher, G. Popek, and G. Kuenning. Saving portable computer battery power through remote process execution. *Proc. of SIGMOBILE*, 1998.

[41] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proc. of Computer systems*, 2011.

[42] M. R. Gary and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[43] Y. Zheng, X. Ding, C. C. Poon, B. P. Lo, H. Zhang, X. Zhou, G. Yang, N. Zhao, and Y. Zhang. Unobtrusive sensing and wearable devices for health informatics. *Trans. on Biomedical Engineering*, 2014.

[44] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33, 2014.

[45] Subhas Chandra Mukhopadhyay. Wearable sensors for human activity monitoring: A review. *IEEE sensors journal*, 15(3):1321–1330, 2015.

[46] D. Spruijt-Metz and W. Nilsen. Dynamic models of behavior for just-in-time adaptive interventions. *Journal of Pervasive Computing*, 2014.

[47] M. S. H. Aung, F. Alquaddoomi, C. K. Hsieh, M. Rabbi, L. Yang, J. P. Pollak, D. Estrin, and T. Choudhury. Leveraging multi-modal sensing for mobile health: A case review in chronic pain. *IEEE Journal of Selected Topics in Signal Processing*, 10(5):962–974, Aug 2016.

[48] M. Tsai, C. Wu, S. K. Pradhan, Y. Xie, T. Li, L. Fu, and Y. Zeng. Context-aware activity prediction using human behavior pattern in real smart home environments. In *Proc. of CASE*, 2016.

[49] M. Kubat, R. C. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 1998.

[50] H. P. Sajjad, K. Danniswara, A. Al-Shishtawy, and V. Vlassov. Spanedge: Towards unifying stream processing over central and near-the-edge data centers. In *Proc. of SEC*, 2016.