CRII: CNS: With Uncertainty Comes Opportunity: Providing Best-Effort Edge Services with Uncertain and Limited Resources

The goal of this project is to create a runtime system for providing best-effort edge services. Different from clouds with manageable and redundant resources, many edge environments are opportunistic and feature uncertain and limited resources. As a result, edge services cannot guarantee QoS for their clients, a critical obstacle on the way of creating current and emerging edge computing applications. Existing solutions fall short, as they optimize the overall QoS by allocating bottleneck resources, competed for by services. In contrast, we observe that the uncertainty of edge environments also brings opportunity: equivalent functions that do not consume the same resources can satisfy the same service request. Hence, this project will explore how to systematically improve QoS by dynamically orchestrating equivalent functions that consume underutilized resources.

The goal of this project will be realized as two deliverables: (1) a system framework, BesoEdge (Best-Effort Services in Opportunistic Edges); (2) an experimental platform for evaluating the performance of BesoEdge and similar systems. BesoEdge will provide true "best-effort" services by supporting equivalent functions and dynamically optimizing their resource allocations and orchestrations. The experimental platform will provide an ecosystem for developing, deploying, experiencing, and evaluating applications and services in opportunistic edges.

The proposed project, if successful, will benefit both local communities and the edge computing research community. It will provide accessible opportunities for undergraduate students to get hands-on experience in developing innovative smart home/wearable/IoT applications and systems. Integrating low-cost edge devices and sensors into our experimental platform makes it possible to replicate the deployment economically. Located in the Detroit Metropolitan area, the PI's institution provides an excellent opportunity for outreach to local high-schools, thus accelerating end-user adoption of edge computing technologies and attracting underrepresented groups into computing study and research. The technologies developed during the course of this project will accelerate the shift to edge computing and serves as the important first step towards fully realizing the research vision of the PI: providing QoS-guaranteed edge services by opportunistic edges.

1 Introduction and Overview

Compared with cloud computing, edge computing [39] provides services with lower-latency for emerging edge applications, which are increasingly popular in smart buildings, autonomous driving, and other IoT environments. For safety and user experience concerns, many of such applications pose strict requirements on the quality of these services (QoS for short, including latency, reliability, and etc.). It is extremely costly for pervasive edge environments to pre-deploy sufficient resources for fully satisfying the QoS required by fluctuating service requests. To reduce such cost, opportunistic edge computing [30] (a.k.a., mobile edge computing [1]) creates scalable infrastructures at the edge using end-user contributed resources, rendering their resources uncertain and limited. Hence, a fundamental research problem in such opportunistic edges is how to provide QoS guaranteed services by uncertain and limited resources [33]. First, we use an example usage scenario to demonstrate this problem.

1.1 Motivating Example

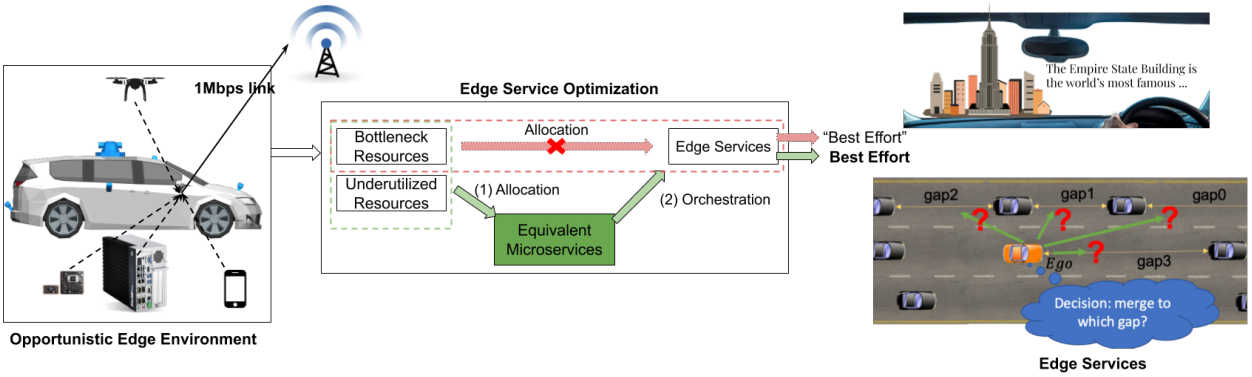

Figure 1: Motivating Example: Service Provisioning in a vehicle-based Opportunistic Edge

Consider an opportunistic edge computing platform in an autonomous vehicle, as shown in Figure 1. The vehicle applies autonomous driving technologies to free its passenger from driving and to provide various entertainment options. In our example scenario, the passenger is interested in sightseeing and requests that information on the buildings along the route be displayed on the front windshield. Meanwhile, the vehicle needs to collect information about its surroundings and make real-time path planning decisions. To avoid transmitting a huge amount of sensor data over the internet, which would reduce the response time, the computers and sensors mounted on the vehicle and the portable devices owned by the passenger form an opportunistic edge computing platform to provide services to these applications. To avoid complicating this problem with security concerns, this example does not consider sharing resources vehicle-to-vehicle.

The edge platform in our example clearly features limited and uncertain resources. With different passengers bringing various devices, the resources in this opportunistic edge environment change in terms of their categories and amounts. Meanwhile, the edge applications' service requests are also highly dynamic, in terms of their functional and non-functional requirements (e.g., reliability, latency, and accuracy). In our example scenario, the passenger may switch to a different application that requires translating the road signs, or she may be in a rush and thus request that the path planning be more proactive and aggressive. Hence, one problem to answer in this example scenario is: **with limited and uncertain resources, how to best satisfy the ever-changing service requests?**

1.2 Problem Analysis

While cloud computing ensures QoS by sharing a seemingly infinite pool of redundant resources among services [47], to be allocated on demand at runtime [49], edge computing is constrained by the computing,

networking, and sensing resources at hand. Duplicating rich resources in pervasive edges, sufficient enough to meet peak requirements, is inefficient and unrealistic [33, 52]. To better handle fluctuating requests, opportunistic edge computing creates scalable infrastructures at the edge using end-user contributed resources. It enables the collocated stationary edge servers deployed by ISPs [48] and mobile/wearable/IoT devices owned by individuals to form a self-organized cluster and cooperatively provide edge services. However, lacking a standardized setup across edge environments and accepting mobile devices as part of the cluster, opportunistic edges usually feature dissimilar resources, while the resources in one edge can also change over time. Therefore, opportunistic edge computing provides additional computing, networking, and sensing resources, while also introducing the problem of resource uncertainty.

We observe that the resource uncertainty in opportunistic edges also brings opportunities: the additional sensing and data processing capabilities brought by user devices make it possible to satisfy a functional requirement by multiple equivalent functions, consuming different resources and providing dissimilar QoS. If the QoS of edge services is constrained by some bottleneck resources, switching to an equivalent function that consumes underutilized resources may further optimize QoS. Moreover, the combined execution of equivalent functions can also optimize QoS (e.g., fail-over for reliability [40], and speculative parallel for latency [17]). Motivated by these observations, the key insight of this proposal is that fully utilizing all available resources in an opportunistic edge can further improve the QoS of edge services.

In the aforementioned example, the most intuitive way to find out what the passenger is looking at is by image recognition, which may consume either networking or computing resources. It can also be implemented by: first read the GPS, orientation, and gaze distance from the passenger's device, then calculate the location of the object the passenger is looking at, and finally obtain the information of the object by querying a cloud server with location as input. Instead of competing with the path planning service on the computing and networking resources, this equivalent function consumes underutilized resources (i.e.,GPS and other sensors) and thereby makes it possible to improve QoS when the computing/networking resources are insufficient. Even in case the sensor-based method is not reliable enough due to the loss of GPS signals, using it first and switching to image recognition if the sensor-based method fails can still reduce the consumption of computing and networking resources while providing the required reliability.

However, the state of the art opportunistic edge system designs lack support for utilizing such equivalent functions. Existing designs for optimizing service QoS can be divided into two interrelated approaches: 1) cache the results or offload the computing procedure to the cloud [11, 12, 50]; 2) when service requests are competing on identical resources, find an optimal resource allocation solution so that the overall utility of the system is maximized [3, 18, 34, 51]. As the first approach relies on storage and network resources, some recent studies of the second approach [45, 58] model the service utility maximization as a multi-constrained optimization problem, with the computing, network, and storage resources being the constraints. None of these approaches systematically support equivalent functions for QoS optimization.

1.3 Overview of the Proposed Activities & Intellectual Merit

To address the aforementioned problem, this project proposes two key research and engineering activities: 1) to design and implement a system framework, called **BesoEdge**, for **B**est-**E**ffort **S**ervices in **O**pportunistic **Edge**s; 2) to build an experimental platform for students to explore edge system and development, for non-expert users to experience edge applications, and for researchers to study realistic opportunistic edges and evaluate their system designs.

BesoEdge — A System Framework for Provisioning Best-Effort Services in Opportunistic Edges: BesoEdge provides best-effort services by dynamically allocating resources for equivalent functions and orchestrating their executions. It applies the virtualization technologies to encapsulate equivalent functions as microservices and deploy microservices on edge devices at runtime. The key innovation of BesoEdge lies in the joint optimization of resource allocation and execution orchestration of equivalent microservices, which

will be achieved in three incremental steps: 1) given the QoS of equivalent microservices, study how to orchestrate their combined executions to best satisfy the multi-dimensional QoS requirements of concurrent edge requests; 2) given the available resources, find all Pareto-optimal resource allocation strategies for each microservice containers, so that the QoS of one microservice cannot be improved without worsening another microservice; 3) explore how to combine the resource allocation for microservices and the orchestration of their combined executions to further improve the QoS satisfaction level of services. Undergraduate students will be hired to implement the key system components of BesoEdge and the joint optimization algorithm.

An Experimental Platform for Evaluating BesoEdge in Realistic Edge Environments: The platform connects edge computing researchers, application developers, and non-expert users who are interested in experiencing novel edge applications. We plan to build an experience center, which provides opportunities for students to explore and develop edge systems, for non-expert users to experience edge applications, and for researchers to study realistic opportunistic edges and evaluate their system designs. The platform integrates BesoEdge for managing the devices and providing services in the experience center. The resource status of devices and the service requests of applications will be collected, and published as a publicly accessible dataset, for the edge computing research community to study the service demands and resource supplies in realistic opportunistic edge environments, as well as to test their system designs. Undergraduate students will be hired to implement the platform's system components and develop edge applications.

Intellectual Merit: The project proposes a brand new solution for improving the QoS satisfaction of edge services, which makes full use of variable resources available in an edge environment. Equivalent functions that consume different resources but provide the same functionality are encapsulated as microservices, with their runtime resources and execution orchestration being managed dynamically, to reach the global optimum QoS for concurrent requests. Our project will also provide practical and theoretical insights about the QoS requirements and resource supplies in realistic opportunistic edge environments. This will inform the future design of opportunistic edge systems and serves as an important first step towards realizing the research vision of the PI: QoS-guaranteed edge services can be provided by opportunistic edges.

PI's Qualifications: The PI's prior research has proven that *the combined execution of equivalent functions can quantitatively improve the QoS performance of edge services.* This finding has been recognized by the edge research community as a promising avenue. One of his papers [41] on providing programming support for the combined execution of equivalent functions received the Best Paper award from IEEE EDGE'19 and another paper [42] about orchestrating the combined execution automatically based on the runtime context at the edge is published by IEEE ICDCS'20. The PI has been working in the distributed computing domain for 10 years, with more than 45 technical publications in international journals and major peer-reviewed conference proceedings. His publication record, together with his experience in supervising undergraduate & graduate students, will contribute to the success of the project and will ease the generation of novel research insights and pragmatic technical solutions.

2 Proposed Research: BesoEdge

Edge computing applications use the computational, sensor, and networking resources of nearby mobile and stationary computing devices. As there is no way to predict what devices will be available at runtime, BesoEdge (Beso means kiss in Spanish) migrates the service-oriented architecture [19] and lightweight virtualization technologies [28] originally designed for cloud computing to edge environments: 1) A monolithic function is broken down into loosely coupled functional units of microservices, whose executions is orchestrated by BesoEdge's central controller to provide services; 2) edge-based function units are encapsulated as container images, to be deployed and initialized on edge devices at runtime. However, different from cloud environments that can allocate seemingly unlimited resources at will, edge environments serve fluctuating requests with limited resources. The goal of BesoEdge is to provide true "best-effort" services for multiple concurrent requests by managing available edge devices.

2.1 Preliminary Work

Orchestrating the Combined Execution of Equivalent Functions: While most existing approaches follow the solutions used in cloud computing and study how to optimally allocate bottleneck resources among competing services, the PI's prior research [4, 41, 42] embraces the wide existence of equivalent functions in opportunistic edges by orchestrating their combined executions.

First, we studied how to improve service QoS qualitatively by executing equivalent functions. Considering that edge devices are unreliable due to their mobility and energy status, our work MOLE [41] provides an intuitive programming interface for developers to specify equivalent functions and their combined execution strategies, including fail-over (for two equivalent functions a and b, execute a first and only execute b if a fails) and speculative parallel executions (execute a and b simultaneously and return the first obtained result without waiting for the late result). Our evaluation shows that MOLE improves the reliability and responsiveness of edge applications with higher execution costs (resource consumption). We also notice that these equivalent functions may be inaccurate, and our work [4] demonstrates that the combined execution of equivalent functions can also improve service accuracy.

Then, we studied how to quantitatively best satisfy QoS requirements by finding the optimal combined execution strategy for equivalent functions. The combined execution of equivalent functions is to trade some QoS aspects for other aspects. For example, fail-over improves reliability by incurring higher latency, speculative parallel reduces latency by incurring higher cost (resource consumption), while the accuracy enhancement also incurs higher cost. To better satisfy the multi-dimensional QoS requirements of services, our work [42] explores how to estimate the QoS of an orchestration strategy for executing equivalent functions, and how to find the optimal strategy in an edge environment with the QoS of equivalent executions detected at runtime. The optimal strategy can be a combination of fail-over and speculative parallelism. For example, for three equivalent functions a, b, and c, one can first execute a, return a's result if it succeeds, or start to execute b and c in parallel upon a's failure and return the first obtained result of b and c.

Adjusting Resource Allocation for Equivalent Functions: However, our previous research hasn't taken into consideration the dynamic resource allocation in edge environments. By applying lightweight virtualization technologies, function executables can be provided as container [8] or unikernel [46] images, to be downloaded, deployed, and executed on edge devices at runtime. With equivalent functions consuming dissimilar resources, our undergoing research explores *how to improve service QoS by adjusting the resource allocation for equivalent functions, and thereby better utilize limited edge resources*.

To demonstrate this problem, consider face recognition in an edge environment. Three implementations can be considered as equivalent: 1) edge-based: perform facial image recognition entirely at the edge, which only consumes *high computing resources*; 2) cloud-based: transfer raw images to be processed by the cloud, which only consumes *high network resources*; 3) hybrid: pre-process the images at the edge and only upload small amount of data to be processed by the cloud, which consumes *low network resources* and *low computer resources*. In an edge environment with *high* computing resource and *medium* network resource, none of these approaches can fully utilize the resources: the edge-based approach is bounded by the computing resource and leave network resource underutilized, while the hybrid approach is bounded by the network and leave computing underutilized. To maximize the system's throughput, the central controller of the edge cluster can deploy both approaches and adjust their resource allocations, so that both network and computing resources can be fully utilized. Our preliminary results show that in an experimental edge system, fully utilizing the resources can improve the system's throughput by 40%.

The findings in the preliminary work can be summarized as, **either optimizing equivalent functions' orchestration or adjusting their resource allocation can improve QoS**. We also observe that these two procedures are correlated: adjusting the resources allocated for equivalent functions will change their individual QoS, and eventually change the optimal orchestration strategy and the resulting QoS. Hence, we plan to explore their joint optimization for further improving QoS.

2.2 Proposed Work: Joint Optimization of Resources

We use an example to show how the joint optimization of resource allocation and orchestration of equivalent functions can further improve the QoS satisfaction level in opportunistic edges. Consider recognizing what building the passenger is looking at in the aforementioned vehicle-based opportunistic edge. Four equivalent functions can satisfy the request: (a) edge-based image recognition; (b) transmitting the image to the cloud for image recognition; (c) querying the edge-based cache system that stores the cloud-based image recognition results; (d) GPS based solution, as introduced in Section 1.2. Solutions (a) and (b) consume heavy computing and network resources, incur higher latency, and don't scale well. Solution (c) and (d) may proceed faster and scale better, but they suffer from low accuracy. Hence, one possible solution is to lean the network and computing resources towards solutions (c) and (d), while only providing basic resources for solutions (a) and (b). The optimal orchestration strategy for such resource allocation can be: 1) first execute (c) and (d) simultaneously and check if their results agree with each other; 2) if so, return the result; 3) if not, execute either (a) or (b) and use its execution result as the final result. This joint optimization allocates more resources for functions (c) and (d) and thereby improves their responsiveness. The orchestration makes sure that the overall latency and accuracy are acceptable by handling most requests with (c) and (d) and passing over only the inaccurate results to (a) and (b).

The key obstacle towards fully realizing this vision is to *find the optimal solution of resource allocation and orchestration for equivalent functions, so that the QoS requirements of services can be best satisfied.* We formulate this problem as an NP-hard multi-dimensional knapsack problem under resource constraints. Let $\mathcal{R} = \{R_1, R_2, ...R_r\}$ denote the amount of resources of r different categories available at a given edge. We use $\mathcal{S} = \{1, 2, ..., s\}$ to denote the service requests, and use $\mathcal{Q} = \{q_1, q_2, ..., q_s\}$ to denote their QoS requirements. For each service s, we use $\mathcal{E}_s$ to denote the equivalent functions, which are provided by the edge service developers.

At runtime, BesoEdge allocates resources for each equivalent functions. We use $\mathcal{C}_e = \{c_e^1, c_e^2, ..., c_e^r\}$ to denote the resource allocation for equivalent function e, and $\tau(C_e)$ to estimate e's resulting QoS under resource allocation C_e. We use $\mathcal{O}_s$ to denote all possible orchestration strategies for service s. For a service s, given its orchestration $o_s \in \mathcal{O}_s$ and the QoS of all its equivalent functions $\mathcal{Q}(C_e) = \{\tau(C_e) \mid \forall e \in \mathcal{E}_s\}$, we can calculate the resulting QoS (denoted as q'_s) by a function $\theta(o_s, \mathcal{Q}(C_e))$ based on our preliminary work [42], with the execution orchestration and resource allocation of its equivalent functions as input. As the QoS is a multi-dimensional vector, we introduce another function $\delta(q_s, q'_s)$ to calculate how the resulting QoS q'_s satisfies the QoS requirement q_s. The joint optimization can be formulated as:

$$
\begin{aligned}
\Big\{\{\mathcal{C}_e, \forall e \in \mathcal{E}\}, o_s \mid \forall s \in \mathcal{S}\Big\} &= \arg\max \sum^{\forall s \in \mathcal{S}} \delta\Big(q_s, q'_s\Big), \\
&= \arg\max \sum^{\forall s \in \mathcal{S}} \delta\Big(q_s, \theta\big(o_s, \{\tau(C_e) \mid \forall e \in \mathcal{E}_s\}\big)\Big), \\
s.t. : \sum^{\forall s \in \mathcal{S}} \sum^{\forall e \in \mathcal{E}} & c_e^r \leq R_r, \forall r \in \mathcal{R};\ o_s \in \mathcal{O}_s, \forall s \in \mathcal{S}
\end{aligned}
\tag{1}
$$

The equation shows that the aim of BesoEdge is to optimally allocate resources for equivalent functions $(\mathcal{C}_e, \forall e \in \mathcal{E})$ and orchestrate their combined executions o_s, so that the overall QoS satisfaction $\delta(q_s, q'_s)$ for all services can be maximized, under the constraints that the total resources allocated to equivalent functions cannot exceed the available resources in the edge. The problem is non-trivial because 1) it is impossible to build a linear projection from the orchestration strategy to the resulting QoS satisfaction level (i.e., formulating θ); 2) due to the different implementations of equivalent functions, the change of resources may have different impacts on different functions. Hence, it is also impossible to build a linear projection from the resource allocation of each equivalent function to its resulting QoS level (i.e., formulating τ).

Deliverables: Our initial study and analysis show the promise of improving QoS satisfaction levels by the joint optimization of resource allocation and orchestration of equivalent functions. The proposed work of BesoEdge includes the following deliverables.

First, we plan to design, implement, and test the optimization algorithm. We will break down the aforementioned research problem into two sub-problems: (1) find the joint optimization solution in an edge environment with identical service requests; (2) extend the findings to non-identical service requests. To solve sub-problem (1), we will first explore an exhaustive search algorithm. The basic idea is to first find all Pareto optimal resource allocation strategies for equivalent functions and all possible orchestration strategies for these resource allocation strategies, and then exhaustively search for the best solution. To make this algorithm practical, we will narrow down the search space by removing QoS-similar orchestrations and resource allocations. Based on the results, we will further explore if there is any greedy algorithm to find sub-optimal solutions by $O(n)$ complexity. We will conduct a small-scale study in our lab to evaluate the performance of the algorithm and adjust it iteratively.

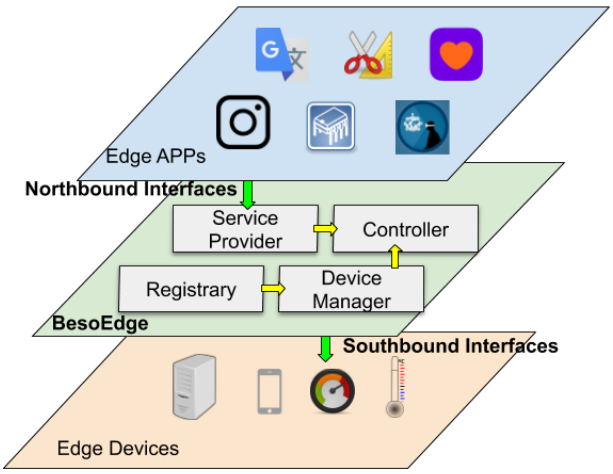

Figure 2: BesoEdge System Design

Second, we will design and implement BesoEdge, a distributed system runtime for opportunistic edges as depicted in Figure 2. BesoEdge will be deployed on a stationary edge device that serves as a gateway. It interacts with edge applications via northbound interfaces for collecting service QoS requirements and for provisioning service, and interacts with edge devices (resource providers) via southbound interfaces for monitoring their status and distributing function executables. BesoEdge contains the following major system components: 1) a local registry for storing executables of microservices and services; 2) an edge device manager for monitoring and managing edge devices; 3) a service provider for handling service requests; 4) a central controller for coordinating edge devices for service provisioning by the joint optimization algorithm.

To make the engineering workload more manageable, BesoEdge will be built upon Docker Swarm [9], a mature resource virtualization system, to leverage its built-in tools. The gateway and the edge devices will form a Docker swarm, where the gateway acts as manager and other edge devices act as workers. BesoEdge encapsulates edge services as application stacks, which contains the equivalent microservices as docker stack services along with an orchestration service. The central controller will be implemented as a standalone application stack, which runs on docker manager, and makes decisions about how to adjust resource allocations and orchestrations for other edge services. Besides, we plan to reuse the PI's previous designs and implementations for opportunistic edge computing systems.

3 Proposed Research: Experimental Platform for Opportunistic Edges

Opportunistic edge computing can allow IoT devices, sensors, edge servers, and personal devices to join the edge cluster in a plug-and-play fashion, and thereby provides more functionality and better performance to edge applications. However, due to the lack of proper runtime support that can provide QoS-guaranteed services to edge applications, existing edge applications and systems are designed for pre-deployed edge environments with fixed resources and service requests. Hence, there is an emerging need for a realistic opportunistic edge computing platform, for researchers to better understand the features of opportunistic edges and test their solutions towards guaranteeing QoS.

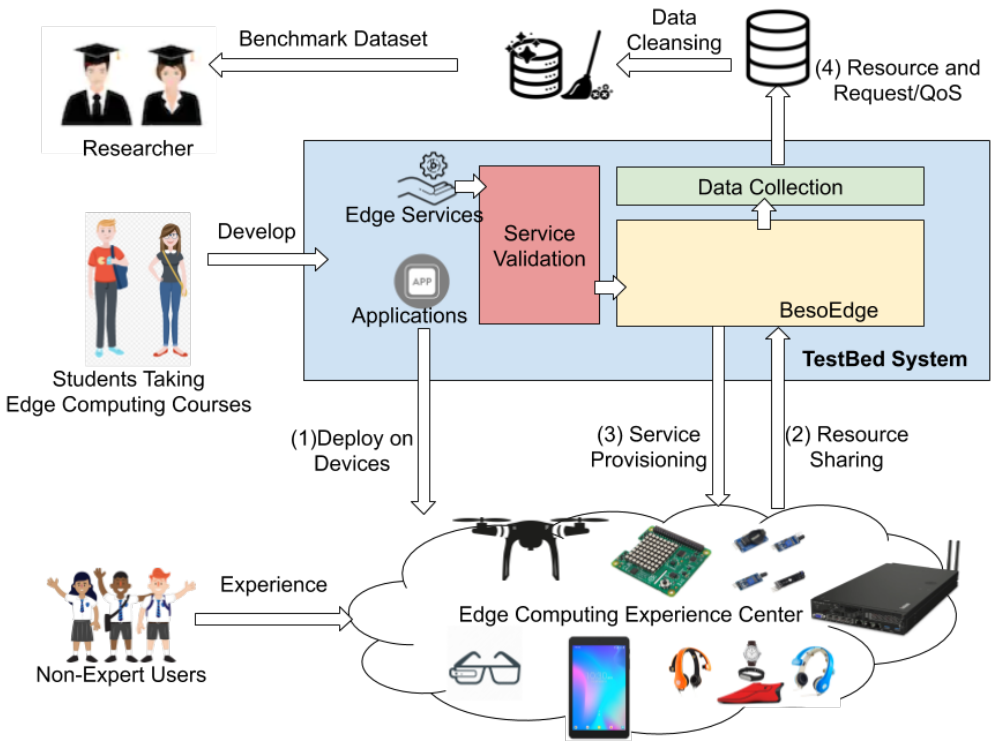

Figure 3: Workflow of the Proposed Experimental Platform

In this project, we propose to set up an experimental platform for opportunistic edge computing, by forming a real opportunistic edge environments and serving real-world edge applications. As shown in Figure 3, the platform will integrate BesoEdge to manage self-organized edge devices and to best satisfy the dynamic QoS requirements of edge applications. The service-provisioning edge devices can be pre-deployed edge devices (i.e., servers, Raspberry Pis, and other IoT devices), or mobile/wearable client devices provided by the platform and carried around by non-expert users (i.e., Drones, smart glasses, and smart phones, see Step 2). These non-expert users run novel edge applications on their client devices, making use of the edge services provided by BesoEdge (see Step 3). The edge applications and edge services will be mainly developed by students taking the edge computing course, and will be deployed on demand on client devices (see Step 1). The experimental platform will collect the time-varying service requests and resource provisioning data to provide publicly accessible datasets for the edge computing research community.

Deliverables: The proposed work of the experimental platform includes the following deliverables:

First, we will design and develop the software system for the experimental testbed. Based on BesoEdge, the platform will further expand two modules: 1) a data collection module which records the time-varying service requests, edge device status, and BesoEdge's decisions for resource allocation and microservice orchestration; 2) a service validation module which integrates existing taint-analysis tools [13, 25] to detect whether sensitive sensor data (i.e., user images and voices) are collected and leaked to suspicious sinking points (i.e., being sent to 3rd party cloud services). The validation module protects the privacy of the non-expert users of our experimental platform.

Second, we will design and deploy an experience center for edge computing applications. We plan to purchase cutting edge hardware and devices, like home robots, drones, VR glasses, wearable devices, smart home sensors, and other IoT devices, to attract non-expert users from local communities like high school students and people working in the automotive industry. We will encourage students taking the Edge Computing / Distributed System / Mobile Computing courses to build novel applications and edge services

as their course projects. The platform will provide programming/debugging supports for the developers to flatten the learning curve.

Third, we will release a dataset about the service requests and device status in our experimental platform. The dataset will contain time-varying data about the devices, the edge applications running on these devices, the service requests and their QoS requirements, as well as the status of in use and idle resources on each edge device. It enables edge computing researchers to study the features of realistic opportunistic edges and to evaluate their service provisioning strategies.

4 Broader Impact of the Proposed Work

The proposed research will have positive broader impacts on education, student involvement, outreach and diversity, as well as the industrial and research communities.

Education Plan: The PI is actively planning to propose a new course on "Edge Computing". The research outcome of this project, including the design of BesoEdge and the experimental platform, will be used to support that course and other courses on "Distributed Computing" and "Mobile Computing". Students taking these courses will gain hands-over experience of developing realistic applications for the cutting-edge devices in the experience center as well as implementing novel edge services for the BesoEdge platform. BesoEdge and the experience center will help the students understand the requirements, system models, and general problems in distributed computing.

Student Involvement and Training: The design and implementation of BesoEdge will actively seek to engage women and minority groups to participate. We aim to recruit women and minorities to participate in engineering BesoEdge and the experimental platform and to appear at events to raise the visibility of minority participation and create role models. The PI actively encourages the participation of undergraduates, women and underrepresented minorities in his research, with a variety of offerings including independent study, summer internship and major qualifying projects. He has co-authored three published papers with undergraduate students. The PI served twice in the Women's Preview Weekend at Virginia Tech and demonstrated his research outcomes in edge computing systems to female high school students. The PI is planning to continue his strong track record of promoting diversity in CS and working with undergraduate students.

Outreach and Diversity: UM-Dearborn's student body reflects the rich cultural, ethnic and religious diversity of the state of Michigan and the city of Dearborn. As a result, the PI anticipates a natural involvement of students belonging to a diverse set of populations, including minorities. The UM-Dearborn CIS department runs many outreach activities for high school and K-12 students. The low-cost sensors of Raspberry Pis and portable edge servers make it possible to deploy our platform during the outreach activities for demonstration and help our audiences gain hands-on experience of engineering modern computer systems and applications. The proposed experience center will also be open to local communities to attract underrepresented groups into computer engineering study and research. Additionally, the Detroit region has many automotive companies and is actively developing autonomous and connected vehicles, which is one of the targeted usage scenarios of our project. We will work closely with the local automotive companies, and prompt perceptual knowledge about edge computing to their employees.

Impact on Industrial and Research Communities: The next decade will see an extensive expansion of IoT systems deployed in pervasive environments like smart homes, autonomous vehicles, and manufacturing plants, which will pose strict requirements on the latency, reliability, and accuracy of the functions they receive from distributed executions. The success carried out of the proposed research will greatly benefit the industry of IoT systems towards fully supporting plug-and-play IoT devices and reducing the cost for pre-deployed edge servers and sensors.

The technologies developed in this proposed research will benefit the research community of edge computing. The theoretical support for optimizing the resource allocation and orchestration of equivalent functions can be applied to solve other problems like robustness and scalability in edge computing systems. The

ability to procure resources from nearby devices, bypassing wide-area networking, will reduce the cellular network traffic, as well as render this technology suitable for disaster recovery scenarios, in which the standard Internet infrastructure is compromised. We will open source the platform so that researchers can adapt it to test their own solutions towards guaranteeing QoS in opportunistic edges. While it is extremely hard to simulate the dynamic resource provisioning and service requests in real edges, the dataset collected by the experimental platform makes it possible to benchmark the performances of different solutions.

5 Current Status of Research and Practice

QoS Optimization: Edge applications in emerging domains like smart buildings [24], autonomous driving [23], and other IoT environments [44,57] pose strict QoS requirements. Although some Internet Service Providers (ISP) are deploying standard edge resources as a part of their 5G network [16], IoT/mobile devices owned by individuals make an important complementary to these servers [12, 57]. The state of the art software-centric solutions mainly focus on better managing edge resources [15], and their approaches can be classified into four major categories as listed below.
1)Deploying redundant computing resources: Some edge applications that only require computing resources assume that edge servers with sufficient resources are pre-deployed [21, 29, 56]. However, it is not practical to assume all sensing resources are available at any edge environments.
2)Using remote resources: remote resources could either be nearby edges [27, 32] or both nearby edges and the cloud [37, 53]. The problem of this category is that the sensing and networking capabilities required by edge applications [23, 56] cannot be replaced by remote resources.
3)context-adaptiveness: Context-adaptive solutions specific the available resources as contexts and automatically adapt the software's execution behaviour in order to better fit dissimilar resources [2, 5, 31, 36]. However, due to the massive amount of different edge resources and their combinations, specifying all possible contexts can be extremely complicated.
4)Dynamic Resource Allocation: As edge platforms adopt lightweight virtualization technologies from cloud [22], some approaches implement their findings towards "best-effort" edge services as system runtime for allocating resources for service containers.

None of these solutions provide QoS guaranteed services on opportunistic edges. Compared with these solutions, the proposed approach makes full use of edge devices in a plug-and-play fashion.

Equivalent functions at Edge and their combined execution: In edge/IoT environments, equivalent functions deliver the same functionality but differ both in their respective QoS characteristics and in their implementations, including hardware/software resource utilization, algorithms, and compositions. For example, [20] demonstrates that the environmental temperature can be captured by a temperature sensor, or be inferred from the CPU temperature; both wireless methods [38] and optical methods [26] have been used to obtain the indoor location of individuals.

Several known patterns incorporate equivalent execution to enhance various performance characteristics. The fail-over strategy first tries executing a function; if it is unavailable or disabled, the execution switches to a back-up function [7]. The speculative parallel execution strategy spawns the execution of all functions simultaneously, proceeding as soon as any of them returns successfully [10, 35]. With both strategies improving reliability, fail-over is cost-efficient and speculative parallel is latency-efficient. The majority voting executes equivalent algorithms, using a consensus result as the final output to improve the accuracy of pattern recognition [54]. Although the combined use of equivalent functions can optimize the QoS of edge applications, how to orchestrate the combined execution of multiple equivalent functions to best satisfy a given QoS requirement remains an untapped problem. Hiratsuka et al. [14] and Cardellini et al. [6] explore the combined use of functional-equivalent microservices to enhance the QoS of web services. However, these works can only roughly estimate the QoS of the generated combined execution.

Testbed for edge computing: Ad hoc testbeds are developed to verify the performance of edge systems

and software [5, 27] in lab environments with pre-deployed edge resources. These testbeds engage huge efforts in setting up the edge systems comprising mobile devices, edge servers, and IoT devices. For other edge systems and applications, rebuilding such testbeds is repetitive and tedious. Most public-available testbeds for edge computing are built upon simulation software built for traditional network infrastructures like the wireless sensor networks (WSNs), such as EdgeCloudSim [43] and IOTSim [55]. However, as pointed out by [37], most testbeds and benchmarks rely on lab-based infrastructures and fail to reveal the resource supply and service request status in realistic edge environments.

6 Plan of Work

The project will involve the PI, one graduate student, and at least two part-time undergraduate students. The proposed project requires both research and engineering efforts. The PI and the graduate student will be responsible for the system design and the algorithm research. The undergraduate students will participate largely in the engineering end, but they will still need to participate in the research to gain enough background for understanding the engineering work. This project will span two years with the following quarter-based schedule and milestones:

Year 1: (1) Complete the design of BesoEdge and the experimental platform. (2) Implement BesoEdge with a very basic central controller. (3) Implement the experimental platform. (4) Integrate the experimental platform into the PI's undergraduate and graduate course.

Year 2: (1) Experimentally run the platform and the experience center. (2) Optimize the BesoEdge controller design based on the evaluation result. (3) Integrate the BesoEdge controller into the system. (4) Release the dataset and setup project website.

7 Results from Prior NSF Support

PI Song has no prior NSF support.

8 Justification for Funding Request

This project requires to build an experience center, serving as an ecosystem that connects different parties. The PI will need a budget for purchasing edge servers, IoT devices, and user devices to build the experience center and hire undergraduate students to engineer such a platform. A graduate student will help the PI design the platform, explore the algorithm, and guide the undergraduate students. We expect to publish our results in major conferences and journals to attract academia and industry collaborators. The program's support will be essential for the PI to hire undergraduate students, purchase necessary devices, work full-time in the summer with a graduate student, and publish the results.

9 Summary

As compared to traditional cloud computing, edge computing provides resource- and data-intensive services at the edge of the network co-located with client requests, thereby reducing network transmission and providing context-awareness. One fundamental difference between edge and cloud is that edge resources are heterogeneous: edge systems rely on the resources provided by devices clustering within one-hop communication range, so different edge systems feature vastly dissimilar but constrained resource setups in terms of their capabilities and capacities. Such resource uncertainty and poverty makes it hard to robustly satisfy the QoS requirements of real-time sensing and control tasks. The proposed research tackles this problem by a software-centric framework that automatically adjusts the resource allocations and executions in the edge to accommodate the available resources and service requests. This project also provides an experimental platform for collecting data in opportunistic edges, providing cutting edge and accessible edge computing experiences for non-expert users, and preparing computer science undergraduate students for engineering edge computing applications and systems.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2017.

[2] A. A. Abdellatif, A. Mohamed, C. F. Chiasserini, M. Tlili, and A. Erbad. Edge computing for smart health: Context-aware approaches, opportunities, and challenges. *IEEE Network*, 33(3):196–203, 2019.

[3] A. Al-Shuwaili and O. Simeone. Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. *IEEE Wireless Communications Letters*, 6(3):398–401, 2017.

[4] A. Bhatia, S. Li, Z. Song, and E. Tilevich. Exploiting equivalence to efficiently enhance the accuracy of cognitive services. In *CloudCom*, pages 143–150, 2019.

[5] M. Breitbach, D. Schäfer, J. Edinger, and C. Becker. Context-aware data and task placement in edge computing environments. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2019.

[6] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola. Moses: A framework for qos driven runtime adaptation of service-oriented systems. *IEEE Transactions on Software Engineering*, 38(5):1138–1159, 2011.

[7] A. Carzaniga, A. Gorla, N. Perino, and M. Pezze. Automatic workarounds: Exploiting the intrinsic redundancy of web applications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):1–42, 2015.

[8] Docker. Docker container. Website. `https://www.docker.com`.

[9] Docker. Docker swarm. Website. `https://docs.docker.com/engine/swarm/`.

[10] H. Guo, J. Huai, H. Li, T. Deng, Y. Li, and Z. Du. Angel: Optimal configuration for high available service composition. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 280–287. IEEE, 2007.

[11] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim. Energy efficient task caching and offloading for mobile edge computing. *IEEE Access*, 6:11365–11373, 2018.

[12] Y. Hao, Y. Miao, L. Hu, M. S. Hossain, G. Muhammad, and S. U. Amin. Smart-edge-cocaco: Ai-enabled smart edge with joint computation, caching, and communication in heterogeneous iot. *IEEE Network*, 33(2):58–64, 2019.

[13] A. Henderson, L. K. Yan, X. Hu, A. Prakash, H. Yin, and S. McCamant. Decaf: A platform-neutral whole-system dynamic binary analysis platform. *IEEE Transactions on Software Engineering*, 43(2):164–184, 2016.

[14] N. Hiratsuka, F. Ishikawa, and S. Honiden. Service selection with combinational use of functionally-equivalent services. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 97–104. IEEE, 2011.

[15] C.-H. Hong and B. Varghese. Resource management in fog/edge computing: A survey. *arXiv preprint arXiv:1810.00305*, 2018.

[16] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[17] M. C. Jeffrey, S. Subramanian, M. Abeydeera, J. Emer, and D. Sanchez. Data-centric execution of speculative parallel programs. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016.

[18] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong. Social welfare maximization auction in edge computing resource allocation for mobile blockchain. In *2018 IEEE international conference on communications (ICC)*, pages 1–6. IEEE, 2018.

[19] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional, 2005.

[20] C. Krintz, R. Wolski, N. Golubovic, and F. Bakir. Estimating outdoor temperature from cpu temperature for iot applications in agriculture. In *Proceedings of the 8th International Conference on the Internet of Things*, pages 1–8, 2018.

[21] K. Li. Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing. *IEEE Transactions on Sustainable Computing*, 2019.

[22] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang. Multiuser computation offloading and downloading for edge computing with virtualization. *IEEE Transactions on Wireless Communications*, 18(9):4298–4311, 2019.

[23] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[24] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang. Intelligent edge computing for iot-based energy management in smart cities. *IEEE Network*, 33(2):111–117, 2019.

[25] V. B. Livshits and M. S. Lam. Finding security vulnerabilities in java applications with static analysis. In *USENIX Security Symposium*, volume 14, pages 18–18, 2005.

[26] R. Mautz and S. Tilch. Survey of optical indoor positioning systems. In *2011 international conference on indoor positioning and indoor navigation*, pages 1–7. IEEE, 2011.

[27] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li. Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 2287–2295. IEEE, 2019.

[28] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott. Consolidate iot edge computing with lightweight virtualization. *IEEE Network*, 32(1):102–111, 2018.

[29] T. Nishio and R. Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[30] R. Olaniyan, O. Fadahunsi, M. Maheswaran, and M. F. Zhani. Opportunistic edge computing: Concepts, opportunities and research challenges. *Future Generation Computer Systems*, 89:633–645, 2018.

[31] G. Orsini, D. Bade, and W. Lamersdorf. Cloudaware: A context-adaptive middleware for mobile edge and cloud computing applications. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, pages 216–221. IEEE, 2016.

[32] S. Pasteris, S. Wang, M. Herbster, and T. He. Service placement with provable guarantees in heterogeneous edge computing systems. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 514–522. IEEE, 2019.

[33] G. Pavlou and I. Psaras. The troubled journey of qos: From atm to content networking, edge-computing and distributed internet governance. *Computer Communications*, 131:8–12, 2018.

[34] J. Plachy, Z. Becvar, and E. C. Strinati. Dynamic resource allocation exploiting mobility prediction in mobile edge computing. In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6. IEEE, 2016.

[35] Z. Qiu. Enhancing response time and reliability via speculative replication and redundancy. 2016.

[36] J. M. N. Ramírez, P. Roose, and M. Dalmau. Distributed interfaces and context-oriented broadcast services in a smart-home environment. In *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8. IEEE, 2016.

[37] F. A. Salaht, F. Desprez, and A. Lebre. An overview of service placement problem in fog and edge computing. *ACM Computing Surveys (CSUR)*, 53(3):1–35, 2020.

[38] F. Seco, A. R. Jiménez, C. Prieto, J. Roa, and K. Koutsou. A survey of mathematical methods for indoor localization. In *2009 IEEE International Symposium on Intelligent Signal Processing*, pages 9–14. IEEE, 2009.

[39] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[40] Z. Song and E. Tilevich. Equivalence-enhanced microservice workflow orchestration to efficiently increase reliability. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 426–433. IEEE, 2019.

[41] Z. Song and E. Tilevich. A programming model for reliable and efficient edge-based execution under resource variability. In *2019 IEEE International Conference on Edge Computing (EDGE)*, pages 64–71. IEEE, 2019.

[42] Z. Song and E. Tilevich. Win with what you have: Qos-consistent edge services with unreliable and dynamic resources. 2020.

[43] C. Sonmez, A. Ozgovde, and C. Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493, 2018.

[44] X. Sun and N. Ansari. Edgeiot: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, 54(12):22–29, 2016.

[45] T. X. Tran and D. Pompili. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Transactions on Vehicular Technology*, 68(1):856–868, 2018.

[46] unikernel. Unikernels: Rethinking cloud infrastructure. Website. `http://unikernel.org/`.

[47] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, 2009.

[48] Verizon. Edge computing and the future of iot & ai. Website. `https://enterprise.verizon.com/business/learn/edge-computing/edge-computing-and-IoT/`.

[49] Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE transactions on parallel and distributed systems*, 24(6):1107–1117, 2012.

[50] J. Xu, L. Chen, and P. Zhou. Joint service caching and task offloading for mobile edge computing in dense networks. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 207–215. IEEE, 2018.

[51] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang. Zenith: Utility-aware resource allocation for edge computing. In *2017 IEEE international conference on edge computing (EDGE)*, pages 47–54. IEEE, 2017.

[52] X. Xu, H. Cao, Q. Geng, X. Liu, F. Dai, and C. Wang. Dynamic resource provisioning for workflow scheduling under uncertainty in edge computing environment. *Concurrency and Computation: Practice and Experience*, page e5674, 2020.

[53] B. Yang, D. Wu, and R. Wang. Cue: An intelligent edge computing framework. *IEEE Network*, 33(3):18–25, 2019.

[54] Q. Zeng, J. Su, C. Fu, G. Kayas, L. Luo, X. Du, C. C. Tan, and J. Wu. A multiversion programming inspired approach to detecting audio adversarial examples. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 39–51. IEEE, 2019.

[55] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan. Iotsim: A simulator for analysing iot applications. *Journal of Systems Architecture*, 72:93–107, 2017.

[56] D. 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 (IoTDI)*, pages 37–48, 2019.

[57] D. Y. Zhang and D. Wang. An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 766–774. IEEE, 2019.

[58] J. Zhao, Q. Li, Y. Gong, and K. Zhang. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(8):7944–7956, 2019.