Task Offloading for Edge-Fog-Cloud Interplay in the Healthcare Internet of Things (IoT)

Farshad Firouzi[1], Bahar Farahani[2], Ehsan Panahi[3], and Mojtaba Barzegari[4]

2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS) | 978-1-6654-3156-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/COINS51742.2021.9524098

***Abstract*—The Smart and Connected Health (SCH) revolution is characterized by the convergence of technologies — from edge computing to cloud computing, Artificial Intelligence (AI), and Internet of Things (IoT) — blurring the lines between the physical and digital worlds. Although these are distinct technologies that evolved independently over time, they are becoming increasingly more intertwined in a way that the capabilities of the technologies are aligned in the best possible way. The public embracement of wearables and the integration of IoT provide greater availability, accessibility, personalization, precision, and lower-cost delivery of healthcare services. Bringing the power of AI offers the ability to wring insights from health data more quickly and accurately. Cloud-IoT has emerged to address some of the major challenges of IoT related to analytics, big data storage, scalability, management, reliability, and heterogeneity. Acting on real-time data compels a move towards edge/fog technology to meet the strict computing time requirement addressing the main drawbacks of Cloud-based IoT solutions. Although the convergence of the edge-fog-cloud in the age of IoT can potentially be a promising paradigm shift, its adoption is still in its infancy phase, suffering from various issues, such as lack of consensus towards any reference models or best practices. Thereby, this paper presents a holistic approach and reference architecture to address the interplay of edge-fog-cloud IoT for healthcare applications. Moreover, a Reinforcement Learning (RL) based offloading technique is presented to distribute the load across edge, fog, and cloud. Finally, a novel case study, ECG-based arrhythmia detection, is presented to better demonstrate and evaluate the efficiency of the proposed model.**

***Keywords*—Internet of Medical Things (IoMT), Internet of Things (IoT), Healthcare, Edge Computing, Fog Computing, Cloud Computing, Offloading**

I. Introduction

The explosive growth of the Intelligent Internet of Things (IoT), Internet of Medical Things (IoMT), Healthcare IoT, Smart and Connected Health (SCH), and widespread adoption of wearables and smart devices is impacting virtually every aspects of the healthcare industry [1], [2]. While Cloud Computing and IoT are distinct technologies that evolved separately, the components of each often complement each other [3]. In recent years, the two technologies have begun to converge resulted in a new paradigm known as the cloud IoT [4], [2], [5], [6], [7], [8]. This new model addresses some primary IoT obstacles like processing power, service composition/management, and data-driven solution development. However, as data variety, velocity, veracity, and volume increase, sending big data from IoT devices to the cloud can be inefficient or impossible because of latency and bandwidth restrictions. Edge Computing (EC) and Fog Computing (FC) can help address the disadvantages of cloud-based IoT solutions because storage and computing resources are distributed between cloud and edge of the network close to endpoint healthcare IoT devices generating data [9], [10], [11], [12], [13]. Hierarchical edge-fog-cloud architecture provides enormous benefits to healthcare IoT as it enables distributing the data processing — including Artificial Intelligence (AI), Machine Learning (ML), and big data analytics — to reach an ideal solution within the specific requirements of every use case [3], [14], [15], [16], [17], [18], [19], [20], [21], [22]. However, achieving an successful fusion of edge, fog, and cloud demands addressing several issues and challenges. Particularly, due to distributed nature of this paradigm, there is a need to effectively load balance the tasks over the computing nodes to meet the promised Service-Level Agreement (SLA), and to optimize the resource utilization, Quality of service (QoS), energy consumption and other use case dependent constraints [23], [3].

Edge-Fog-Cloud IoT model has emerged as a means of bridging the gap between endpoint IoT devices and the cloud. Edge-Fog-Cloud paradigm is highly heterogeneous due to the wide variety of devices that compose the overall computing power. Devices can include vehicles, wearable devices, base stations, sensors, smartphones, and network nodes with extensive functionality. Additionally, the utilization of base stations and presence points needed to locate servers near the edge is becoming more popular. However, edge-fog-cloud paradigm faces a lot of obstacles [21], [3]. Particularly, there is currently no agreement around best practices or a reference model that defines how cloud and edge computing should be used in IoT. Moreover, the heterogeneous nature of the architecture also means that available resources can vary. Resource issues can also be magnified by mobile devices and network traffic. Finally, dynamic task placement and offloading have emerged as a serious issue in Edge-Fog-Cloud IoT model [3], [24], [25], [26]. To achieve lower processing delay and to better utilize available resources, edge-fog-cloud depends on offloading mechanism [15]. Even with improvements in offloading techniques, many barriers remain [3]. For example, delivery, strength, monitoring, fault analysis, service quality, energy consumption, privacy, integration, load balancing, latency, and security are still issues that must be considered. To address some of the above mentioned challenges, this paper

[1]Electrical and Computer Engineering Department, Duke University
[2]Cyberspace Research Institute, Shahid Beheshti University
[3]Department of Electrical and Computer Engineering, University of Tehran
[4]Biomechanics Section, Department of Mechanical Engineering, KU Leuven University

978-1-6654-3156-9/21/$31.00 ©2021 IEEE

will explain a reference model for Edge-Fog-Cloud IoT and also present a latency- and energy-aware offloading technique based on Reinforcement Learning (RL) to address the interplay among computing resources.

The rest of this paper is organized as follows. Section II discusses and compares the state-of-the-art computing paradigms and technologies for IoT. Section III presents a holistic reference architecture for edge-fog-cloud computing for Healthcare IoT to tackle the most challenging barriers of the existing use cases of healthcare IoT. Section IV presents a novel RL-based offloading technique as well as a comprehensive case study to better demonstrate and evaluate the efficiency of the proposed model. Finally, Section V concludes the paper.

II. Computing Paradigms in the Era of IoT

There are a number of computing paradigms that can be utilized in IoT applications. In this section, we briefly discuss the most important ones. Edge computing locates processing resources near edge/endpoint devices. Similarly, fog computing moves processing and intelligence nearer to endpoint devices, but it is a broader concept that makes room for a hierarchical architecture handling storage, computing, control, and networking from IoT devices to the cloud. Mobile Computing is a mainly consumer-facing service related to the evolution of new mobile devices (i.e., tablets and smartphones) with many of the capabilities of conventional personal computers. The Mist Computing paradigm completes computations on IoT devices. Mobile Cloud Computing is a means of using cloud computing to extend mobile computing and overcome the processing/storage obstacles of mobile devices. Mobile Cloud Computing is centered on remote execution of offloaded tasks and uses the synergy between IoT devices, mobile devices, and CC. However, it is not geographically distributed and does not provide ultra-low latency. Multi-access Edge Computing (MEC), previously known as Mobile Edge Computing, covers tasks that are not mobile device-specific and is located at the network's edge inside the Radio Access Network (RAN) in 4G and 5G. It offers cloud computing functionalities and IT services through existing base stations that are able to operate with little or no internet connection. MEC is considered a subset of edge computing because edge computing can also typically use any form of connectivity, including WiFi, Cellular, LAN, etc. It is worth noting that the cloud provider or user is considered the operator in Mobile Cloud Computing (MCC). In contrast, network infrastructure providers are the operators in MEC. In addition, MCC offers greater computing resources than MEC while MEC latency is much lower than that of MCC. Cloudlet, a virtual machine infrastructure, is much like MEC and is well-suited to real-time IoT applications. Cloudlet functions like a smaller localized cloud placed between devices and the cloud to support real-time data interaction. While it is similar to fog computing, fog computing is a more generalized model that allows resources to be placed at any point between the cloud and endpoint devices. Mobile ad hoc Cloud Computing is a subcategory of MCC used with temporary groups of mobile devices gathered in the same location that share resources with one another to create an ad hoc cloud. The same mobile devices also route their own internal traffic [9], [27].

III. Reference Edge-Fog-Cloud Architecture for IoT Healthcare

The majority of current fog models use one fog layer that stretches from the IoT device layer to the cloud in a three-layer IoT model. Such an architecture usually allows fog devices to share data and communicate with one another. However, a few utilize horizontal fog nodes that do not communicate with neighboring nodes. In those examples, fog nodes communicate solely with the IoT and cloud layers. It has also been demonstrated that processing latency can drop if IoT data is uploaded to a local node and the nodes are allowed to transfer data to one another when additional resources are needed [3]. There are some edge-fog-cloud models that are also hierarchical in nature and include multiple tiers. Such models share data across fog layers vertically but do not share data within horizontal layers. Fog nodes are categorized based on proximity to IoT devices, memory, storage, and computing power. Initially, endpoint devices transfer data to the closest fog layer. Then, the fog later transmits data vertically until finding a node with necessary resources. These models have been illustrated with two, three, and four fog layers [28], [27], [29], [3].

By extending our former work [3], incorporating existing architectures and computing models [29], [30], [31], [27], [32], [33], [34], [35], [34], [36], [22], and considering the relevant healthcare use cases, we present a reference architecture for IoT healthcare to minimize unnecessary network traffic, reduce end-to-end latency, and increase energy efficiency. The hierarchical model's ambition is to provide a general architecture, that can meet the requirements of Wearable IoT (WIoT) and IoT healthcare in various healthcare applications, such as social-distance and crowdsensing. Fig. 1 illustrates the edge-fog-cloud computing model and its concepts. The reference model is designed based on a multi-layer hierarchical cluster-based architecture allowing each node communicates vertically and horizontally within fog and also facilitating the connection among clusters. A typical edge-fog-cloud model consists of the following main components [30], [29], [3], [34]:

- Fog Gateway Nodes (FGN): FGNs are gateways or access points serving as the entry points to fogs by establishing a connection between endpoint IoT devices and the fog.
- Fog Orchestration Controller (FOC): FOC is responsible for the control layer of each fog cluster[30]. It provides a ubiquitous view of the whole fog, monitor the available resources and extract a set of resource metrics, distributes the loads and incoming requests among Fog Computing Nodes (FCNs), addresses inter fog communication, communicates with cloud or another fog in the hierarchy, performs data migration, task offloading, and service deployment and also coordinates the corresponding activities while considering latency, energy consumption, privacy and priority requirements. It should also be noted

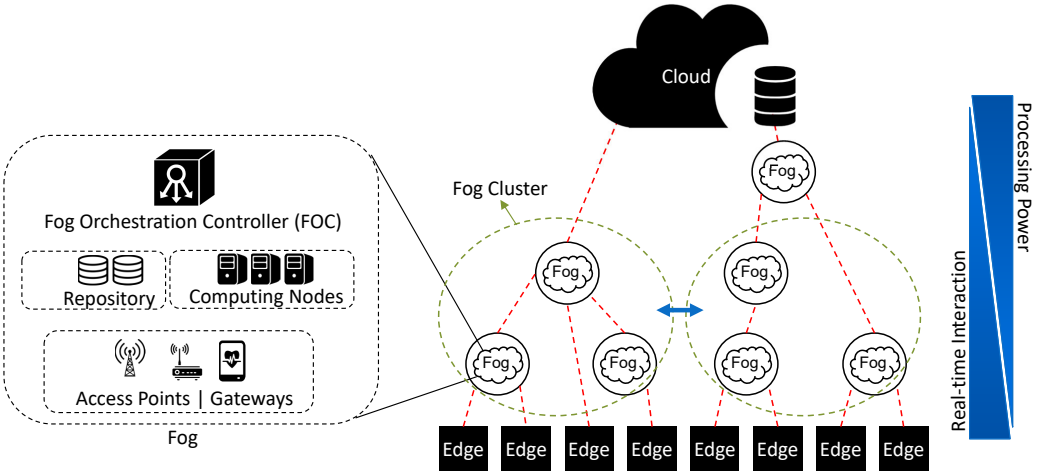


Fig. 1: Reference architecture for edge-fog-cloud computing model in Healthcare IoT.

that the incorporated offloading and Service Placement (SP) techniques can be characterized by the following features [3]:

- Online vs. Offline: Optimizing SP in advance allows precise solution calculations and produces the best possible QoS. Offline optimization is often done with solvers like CPLEX, Choco, and Gurobi. While such solvers often provide good results, they can be very time consuming and are not typically suitable for online use.
- Centralized vs. Decentralized: In the past few years, several offloading and SP techniques have been proposed. The existing solutions can be categorized into Centralized or Decentralized techniques. In the centralized approach, a central orchestrator is located in the cloud with a comprehensive overview of the edge and fog, optimizing SP and meeting the architecture's global goals. In our proposed architecture, we rely on a semi-decentralized approach in a way that FOCs distribute the load between FCNs, but on the other hand, there is no central system to conduct load balancing across fog clusters. In this approach, FCNs communicate with each other as well as the cloud to optimize the system.

- Fog Computing Nodes (FCN): FCNs are general computing nodes that communicate with FOC to execute the assigned tasks. Note that requests are distributed by FOC among FCNs.
- Fog Storage Nodes (FSN): FSN provides and manages distributed database and repositories in the fog.

The proposed reference model provides the following benefits:

- Reduced Network Load: it distributes the network load, and lessens the backhaul system's load.
- Latency-aware Computing: Edge devices also benefit from quicker task processing increasing the Quality of Service (QoS). active Mobility Support: Locating resources nearer to edge devices also allows the network to more quickly and precisely respond to user mobility.
- Providing Context: With resources located nearer to end users, geographic-specific content can also be provided.
- Extended Battery Life: it can lessen power use and increase battery life, which allows higher long-term independence.
- Infinite Load and Storage Support: In conjunction with cloud computing, edge-fog-cloud models can depend on cloud resources to tower over the most powerful of fog nodes. On-demand cloud resources can be utilized o expand limitless resources to meet storage needs.

IV. Case Study

One of the main promising use cases of edge-fog-cloud computing is IoT eHealth that aims at extending the boundaries of the healthcare outside of traditional hospital settings (i.e., transforming hospital-centric to patient-centric) by utilizing ubiquitous and vigilant wearable IoT devices and sensing technologies for continuous and unobtrusive data collection. Leveraging edge-fog-cloud architecture enables us to optimize response time (delay) and energy consumption in IoT eHealth applications [5], [37], [38], [7]. Remote health monitoring and anomaly recognition are among those mission-critical eHealth applications in which transferring a massive amount of raw streaming health data (e.g., vital signs) via wireless networks might result in significant delay and/or energy consumption (translated to lower battery life) depending on the running workload and the working conditions (e.g., the communication network). Distributing the intelligence across edge, fog, and cloud potentially can address these challenges and achieve better energy consumption, and lower latency compared to conventional single-layer systems. In this section, using a case study on Electrocardiography (ECG) based heart anomaly detection, we demonstrate the features and benefits of the hierarchical edge-fog-cloud architecture and task offloading.

Note that the case study is adopted from the previous work of the authors [2], [15].

Analyzing ECG signals is a noninvasive, inexpensive, and widely used clinical method for Cardiac Arrhythmias (CAs) detections. A twelve lead ECG — consisting of three bipolar limb leads (I, II, and III), three unipolar limb leads (AVR, AVL, and AVF), and six unipolar chest leads or V leads (V1, V2, V3, V4, V5, and V6) — is today's standard tool for obtaining ECG signals. Considerable training is required to be able to analyze ECG data. Fortunately, recent studies have shown that ML techniques can automate this process in a very reliable and accurate way with minimal possible human intervention. ML-assisted solutions have the potential to prevent millions of death that heart disease causes world-wide every year. In this case study, we use a Convolutional Neural Network (CNN) to classify ECG signals. To do so, we use MIT-BIH dataset, which contains approximately 110,000 beats of 48 patients consisting of 17 different classes [39]. Among those 17 different classes, the following six categories should be detected for a robust CA detection: NOR (normal beat), LBB (left bundle branch block beat), RBB (right bundle branch block beat), PVC (premature ventricular contraction beat), PAB (paced beat), and APC (atrial premature contraction beat). For creating the model, we properly windowed each beat into a raw signal image with the size of (64, 64). Considering the fact that the dataset is very imbalanced, for the training purpose, some images were augmented using the cropping technique in a way that each image was cropped in 9 different ways, producing ten images for each beat in total. In the end, we created 353690 samples as the input to the CNN training phase [2], [15]. In addition to augmentation, we used random rotation and/or flipping operations to the input images, and also, we shuffled all samples prior to the training phase. In this study, we divided the dataset into train, validation, and test datasets with a ratio of 0.7, 0.15, and 0.15, respectively. Table I and Table II show the architecture of the trained CNN model and its performance, respectively [2].

CNN models are computation-intensive and typically result in a considerable overhead for battery-powered IoT health devices with limited hardware resources. To improve the latency and energy consumption, we can offload a portion of CNN inference computation to fog or cloud to satisfy the SLA or the deadline of the application or extend the lifetime of the battery. Note that SLA is a common parameter that have been used in other offloading and SP techniques as well [40]. To do so, we exploit RL for task offloading in this case study. The exploited RL is an extension of the work proposed by the authors of [40]. RL enables an agent (decision maker) to learn in an interactive unknown finite discrete-time environment by trial and error using feedback/reward from its own decisions and experiences [41]. An RL problem typically consists of five key elements: i) Agent: a living entity that interacts with an environment and takes actions. Naturally, interactions are divided into episodes (e.g., plays of a game, trips through a maze); ii) Environment: Everything agents interact with is called an environment. Agents live in the environment and interact with it; iii) State: it describes the current situation; iv) Reward or motivation: A feedback from the environment to the agent in response to the taken action; v) Policy: a technique to map the state of the agent to actions; vi) Value: future reward that the agent receives when a particular action is taken from a particular state [41].

In RL, the environment is typically defined in the form of a Markov Decision Process (MDP). MDP is an extension of Markov chains. The main difference is that in MDPs we have actions (choices) in each state and by taking an action we expect a reward (giving motivation). MDPs have Markov property meaning that being in each state depends on the previous state and the action that has been taken. An MDP can be represented by a tuple (S, A, P, R, γ). Parameter S is a finite set of states, s_t shows the state at time step t, T is the final time step, and s_T shows the terminal state. $A(s)$ is the finite set of actions that the agent can take in the state s. P is the state transition probability function e.g., the transition probability from the state s to the state s' by taking a particular action. R shows the expected rewards in a way that $R_s = E[R_{t+1}|S_t = S]$ indicates how much reward the agent can expect to get from state S at the moment. Parameter γ $(0 \geq \gamma \leq 1)$ is a discount factor for future rewards and captures the importance of the current reward. For instance, if $\gamma = 0$, it means that the focus of the agent is current rewards (agent is short-sighted), on the other hand, $\gamma = 0$ means that the agent is interested in future rewards (the agent is far-sighted).

The RL agent needs to create a policy $(\pi(S) : S \rightarrow A)$ by living in the environment. The policy controls the balance between exploration and exploitation [41]. Note that in RL, exploitation means that the agent keeps doing what he was doing, on the other hand, exploration means that the agent tries something new. Policy is a mapping from states (S) to actions (A). Policy is a function (e.g., π) that takes a state s and provides a vector with the same size of number of actions that shows the probability of selecting each action in the given state s. At the beginning of the agent's life, each action could be taken with equal probability. However, over the time and with the training of the agent, the probability of each action would change, and finally the algorithm converges to an optimal policy. If the agent extracts the optimal policy, it takes a sequence of actions maximizing the cumulative sum of rewards ($\sum_{i=1}^{T} \gamma^{i-1} r_i$, where T is the length of the episode) gets from the environment. If the following equation is established, then we have found the optimal policy.

$$V^*(s) = \max_{\pi} V^{\pi}(s) = V^{\pi^*}(s) \qquad \forall s \in S, \tag{1}$$

in which * shows the optimal case, π^* corresponds to the optimal policy, $V(s)$ denotes the value-function providing the value of a state, $\max_{\pi} V^{\pi}(s)$ shows the maximum value of state s among all possible policies. Note that the value of a state represents the expected total reward for an agent starting from state s to the terminal state (s_T) using policy π. The

value of a state in RL can be computed based on Bellman Equation:

$$V_\pi(s) = E_\pi[r_{t+1} + \gamma V(S_{t+1})|S_t = s]. \quad (2)$$

Note that the above equation consists of two distinct components: i) immediate reward ($r_{t+1)}$); ii) the value of successor state ($V(S_{t+1})$) considering the discount factor (γ) when we are following a certain policy (π). In this case study, we rely on Q-Learning, a model-free RL algorithm, for task offloading. Q stands for quality indicating that how useful a given action is in gaining some future reward. Algorithm 1 demonstrates the details of the Q-learning algorithm. Simply, Q-learning is based on a function computing the quality of a state-action combination ($Q : S \times A \rightarrow R$). The Q-value of the state–action pair (s_t, a_t) is updated in each step as follows:

$$\hat{Q}(S_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(S_{t+1}, a_{t+1}) - Q(s_t, a_t)), \quad (3)$$

where $\hat{Q}(S_t, a_t)$ shows the new value, $Q(s_t, a_t)$ is the old value, α is the learning rate, r_t is the received reward when moving from state s_t to state s_{t+1}, and $\max_a Q(S_{t+1}, a_{t+1})$ is the estimate of the optimal future value. When the number of states and actions per each state increases, it would not be practical to store all Q values in a table. To address this issue, in deep Q-learning, we exploit a neural network to approximate the Q-value function. A neural network function approximator with weights θ is referred to as a Q-network. To be able to train the Q-network, we can rely on gradient descent algorithm. The objective of this algorithm is to minimize a loss (J_θ), e.g., using RMS optimizer, with respect to actual network parameters (θ):

$$J_\theta = (r_j + \gamma \max_{a'} \hat{Q}(s_j, a', \phi) - Q(s_j, a_j; \theta))^2. \quad (4)$$

TABLE I: The layers of the trained CNN model for ECG classification.

Layer (type)	Output Shape
Input	(64,64,64)
conv2d	(64,64,64)
max pooling	(16,16,64)
conv2d	(16,16,256)
max pooling	(8,8,256)
flatten	(1628)
dense	(256)
dense	(6)

In our case study, the agent of the Q-learning should decide about the location of performing a task (or a block of the task). Based on a simplified version of the reference edge-fog-cloud architecture, there are four places in which the target task can be executed as explained in [40]: i) L_0: the endpoint IoT device (edge node); ii) L_1: the fog node in the close proximity of the IoT device; iii) L_2: the adjacent fog node; iv) L_3: cloud server. We assume the agent is located in L_0. Each task varies in terms of memory usage, CPU usage and bandwidth usage. Therefore, we can define the corresponding state of the system based on the available CPU, memory, and bandwidth of each location as well as the CPU, memory, and bandwidth requirement of each task [40]. The reward is a function of both response time and also energy consumption (note that energy consumption directly impacts the life of the battery). The energy consumption (EC) consists of two distinct components, namely i) energy consumption for the processing of the task; ii) energy consumption for communication (i.e., transferring the data/task from one location to another), which highly depends on the network type and wireless technologies. Note that the response time is also a function of both communication latency and processing time [15], [40]:

TABLE II: The performance of the CNN model.

Classes	Precision	Recall	F1-score
Class 0	0.98	0.98	0.98
Class 1	0.98	0.99	0.99
Class 2	0.96	0.98	0.97
Class 3	0.97	0.95	0.96
Class 4	0.99	0.99	0.99
Class 5	0.96	0.84	0.9

Algorithm 1 Deep Q-Learning for offloading.

```
Initialize replay buffer M to Capacity C
Initialize action-value function Q with random weight θ
Initialize target action-value function Q̂ with random weight φ = θ
for epoch = 1, 2, ... do
    for every time step do
        a_t ← π(s)
        Take action a_t and observe r_t, s_{t+1}
        store (s_t, a_t, r_t, s_{t+1}) in M
        sample (s_j, a_j, r_j, s_{j+1}) from M
        J_θ = (r_j + γ max_{a'} Q̂(s_j, a', φ) − Q(s_j, a_j; θ))^2
        perform gradient descent step on J_θ
        backward θ
        Q̂ = Q
    end for
end for
```

$$EC_{total} = EC_{process} + EC_{transmission}(i, j). \quad (5)$$

$$Reward_{energy}(s, a) = \frac{1}{EC_{total}}. \quad (6)$$

The reward of the agent with respect to the response time can be calculated as follows as proposed in [40]:

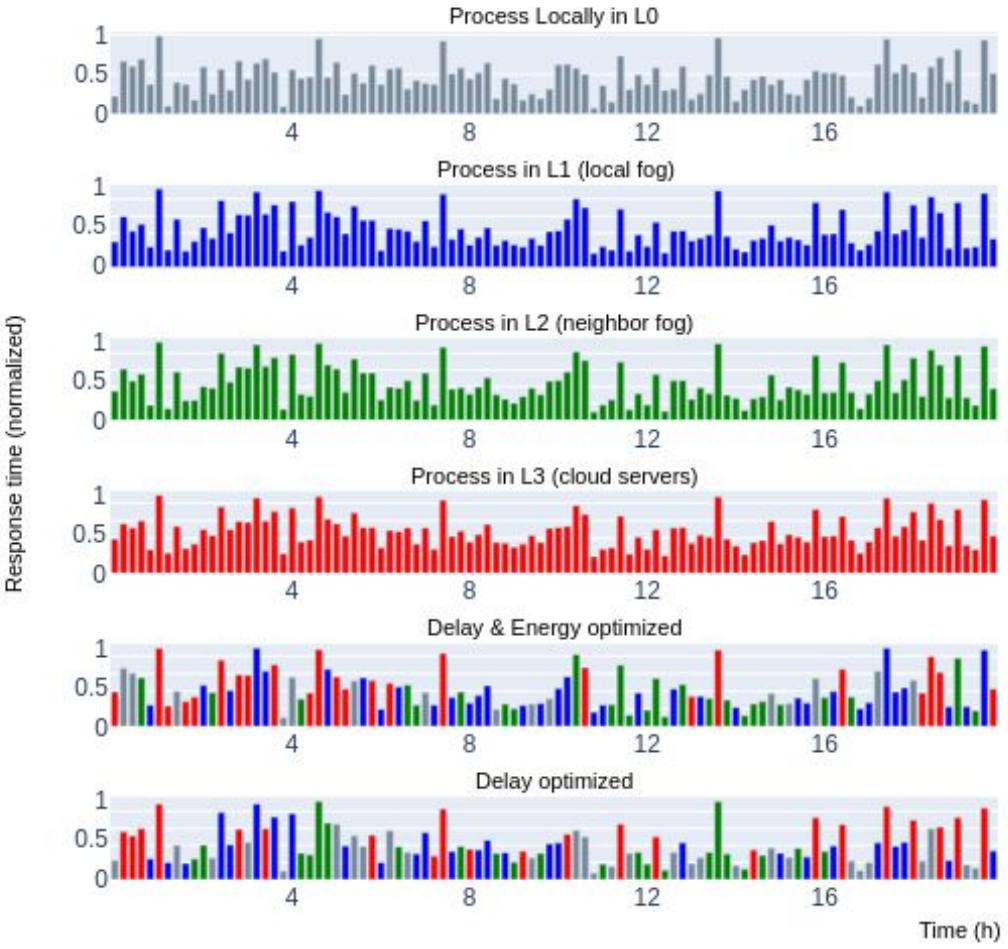


Fig. 2: Comparison of the response time (delay) in different scenarios using a synthetic dataset.

$$Reward_{delay}(s_j, a_i) = \frac{E_{SLA}}{E_{i,j}} - P_{SLA}, \tag{7}$$

$$P_{SLA} = \begin{cases} \frac{E_{i,j}}{E_{SLA}}; & \text{if } E_{SLA} \leq E_{i,j} \\ 0; & Otherwise \end{cases} \tag{8}$$

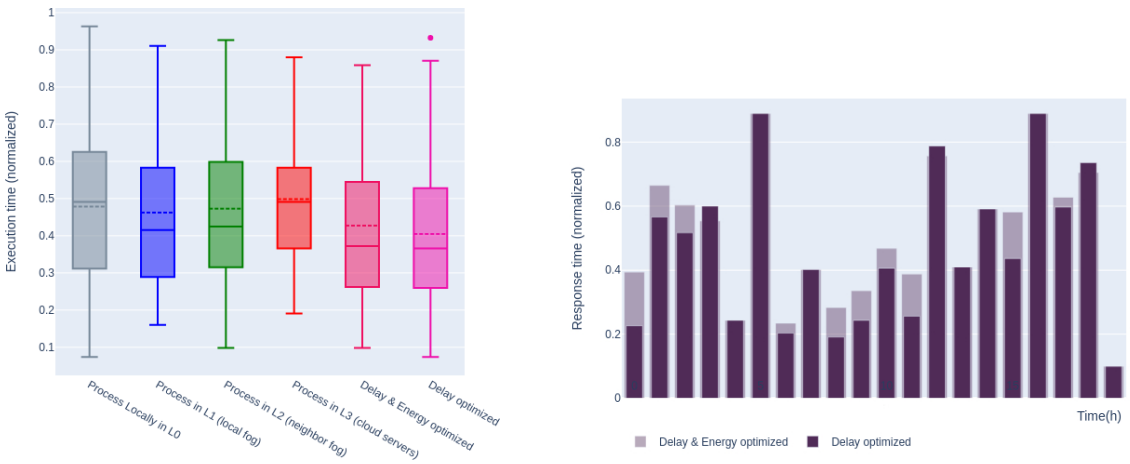
where, E_{SLA} is the expected SLA time or the deadline of the task to be computed. The total reward of the agent is a weighted sum of response time and energy consumption as formulized below:

$$R(s_j, a_i) = \alpha Reward_{delay}(s_j, a_i) + \beta Reward_{energy}(s_j, a_i) \tag{9}$$

To better illustrate the importance of edge-fog-cloud interplay, we considered five different scenarios for a synthetic dataset that we created for this case study: i) process the task locally in $L0$; ii) process the task in $L1$; iii) process the task locally in $L2$; iv) distribute the task across edge-fog-cloud, while our objective is only to minimize the response time ($\alpha = 1.0$ and $\beta = 0.0$); and v) distribute the task across edge-fog-cloud, while our objective is to optimize both response time and energy consumption ($\alpha = 0.85$ and $\beta = 0.15$). Fig. 2 and Fig. 3(a) compare the corresponding response times. As shown in these figures, the response times of scenario v (Delay optimized) and scenario vi (Delay-Energy optimized) are significantly lower (up to 36%) than the other scenarios. Furthermore, Fig.3(b) shows the energy consumption of scenario v and scenario vi over time. According to the results, scenario vi can reduce the energy by 17.6% at the expense of 7.3% increase in response time. In summary, the above case study demonstrates the importance of collaborative intelligence and the interplay between edge-fog-cloud to be able to optimize the response time as well as the energy, which is directly translated to the battery life of the edge devices .

V. Conclusion

As the level of traffic and number of wearables and IoT devices grow, internet service and capacity challenges are emerging, making it difficult to manage context and time-sensitive application needs with only cloud IoT. Therefore, the state-of-the-art healthcare IoT solutions are transitioning away from centralized models and moving toward fog/edge computing resulting in a hierarchical IoT model. This implies that the



(a) (b)

Fig. 3: Delay (Response time) and energy consumption for different scenarios using a synthetic dataset.

edge-fog-cloud interplay becomes the most critical challenge to be able to provide further possibilities in healthcare IoT. Thereby, this paper presented a holistic multi-layer reference model for the edge-fog-cloud IoT enabling the intelligence and tasks to be distributed across layers. Moreover, a novel RL-based technique was presented to address the offloading and service placement challenge. Finally, to evaluate the efficiency of the proposed model, a comprehensive use case, ECG-based arrhythmia detection, was demonstrated.

REFERENCES

[1] F. Firouzi, B. Farahani, M. Weinberger, G. DePace, and F. S. Aliee, "Iot fundamentals: Definitions, architectures, challenges, and promises," in *Intelligent Internet of Things*, pp. 3–50, Springer, 2020.

[2] F. Firouzi, B. Farahani, M. Barzegari, and M. Daneshmand, "Ai-driven data monetization: The other face of data in iot-based smart and connected health," *IEEE Internet of Things Journal*, 2020.

[3] F. Firouzi, B. Farahani, and A. Marinšek, "The convergence and interplay of edge, fog, and cloud in the ai-driven internet of things (iot)," *Information Systems*, p. 101840, 2021.

[4] F. Firouzi and B. Farahani, "Architecting iot cloud," in *Intelligent Internet of Things*, pp. 173–241, Springer, 2020.

[5] F. Firouzi, B. Farahani, M. Daneshmand, K. Grise, J. S. Song, R. Saracco, L. L. Wang, K. Lo, P. Angelov, E. Soares, *et al.*, "Harnessing the power of smart and connected health to tackle covid-19: Iot, ai, robotics, and blockchain for a better world," *IEEE Internet of Things Journal*, 2021.

[6] F. Firouzi, A. M. Rahmani, K. Mankodiya, M. Badaroglu, G. V. Merrett, P. Wong, and B. Farahani, "Internet-of-things and big data for smarter healthcare: From device to architecture, applications and analytics," 2018.

[7] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven iot ehealth: Promises and challenges of iot in medicine and healthcare," *Future Generation Computer Systems*, vol. 78, pp. 659–676, 2018.

[8] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: a survey," *Future generation computer systems*, vol. 56, pp. 684–700, 2016.

[9] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, Sept. 2019.

[10] M. I. Bala and M. A. Chishti, "Survey of applications, challenges and opportunities in fog computing," *International Journal of Pervasive Computing and Communications*, vol. 15, pp. 80–96, June 2019.

[11] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2017.

[12] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[13] A. Aljumah and T. A. Ahanger, "Fog computing and security issues: A review," in *2018 7th international conference on computers communications and control (ICCCC)*, pp. 237–239, IEEE, 2018.

[14] B. Farahani, M. Barzegari, F. S. Aliee, and K. A. Shaik, "Towards collaborative intelligent iot ehealth: From device to fog, and cloud," *Microprocessors and Microsystems*, vol. 72, p. 102938, 2020.

[15] B. Farahani, M. Barzegari, and F. S. Aliee, "Towards collaborative machine learning driven healthcare internet of things," in *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, pp. 134–140, 2019.

[16] N. Mohan and J. Kangasharju, "Edge-fog cloud: A distributed cloud for internet of things computations," in *2016 Cloudification of the Internet of Things (CIoT)*, pp. 1–6, IEEE, 2016.

[17] Z. Nezami, K. Zamanifar, K. Djemame, and E. Pournaras, "Decentralized edge-to-cloud load-balancing: Service placement for the internet of things," *arXiv preprint arXiv:2005.00270*, 2020.

[18] H. Shah-Mansouri and V. W. Wong, "Hierarchical fog-cloud computing for iot systems: A computation offloading game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, 2018.

[19] J. Kang and D.-S. Eom, "Offloading and transmission strategies for iot edge devices and networks," *Sensors*, vol. 19, no. 4, p. 835, 2019.

[20] S. Svorobej, P. Takako Endo, M. Bendechache, C. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, D. Tzovaras, J. Byrne, and T. Lynn, "Simulating fog and edge computing scenarios: An overview and research challenges," *Future Internet*, vol. 11, no. 3, p. 55, 2019.

[21] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–36, 2019.

[22] D. S. Linthicum, "Connecting fog and cloud computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 18–20, 2017.

[23] M. K. Hussein and M. H. Mousa, "Efficient task offloading for iot-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020.

[24] F. A. Salaht, F. Desprez, and A. Lebre, "An overview of service placement problem in Fog and Edge Computing," Tech. Rep. RR-9295, Lyon, France, 2019.

[25] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for IoT applications," *Future Generation Computer Systems*, vol. 90, pp. 149–157, Jan. 2019.
[26] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game," *IEEE Internet of Things Journal*, vol. 5, pp. 3246 – 3257, Aug. 2018. arXiv: 1710.06089.
[27] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the internet of things," *Pervasive and mobile computing*, vol. 52, pp. 71–99, 2019.
[28] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proceedings of the ASE BigData & SocialInformatics 2015*, pp. 1–6, 2015.
[29] O. Consortium *et al.*, "Openfog reference architecture for fog computing," *Architecture Working Group*, pp. 1–162, 2017.
[30] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "Fogbus: A blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, vol. 154, pp. 22–36, 2019.
[31] M. Iorga, L. Feldman, R. Barton, M. Martin, N. Goren, and C. Mahmoudi, "The nist definition of fog computing," tech. rep., National Institute of Standards and Technology, 2017.
[32] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.
[33] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE access*, vol. 6, pp. 47980–48009, 2018.
[34] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management and evaluation of fog computing systems: A survey," *IEEE Internet of Things Journal*, 2020.
[35] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand, "ifogstor: an iot data placement strategy for fog infrastructure," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pp. 97–104, IEEE, 2017.
[36] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for iot," *Wireless Personal Communications*, vol. 92, no. 1, pp. 181–196, 2017.
[37] B. Farahani, F. Firouzi, and K. Chakrabarty, "Healthcare iot," in *Intelligent Internet of Things*, pp. 515–545, Springer, 2020.
[38] F. Firouzi, B. Farahani, M. Ibrahim, and K. Chakrabarty, "Keynote paper: from eda to iot ehealth: promises, challenges, and solutions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 2965–2978, 2018.
[39] MIT-BIH, available at https://www.physionet.org/physiobank/database/mitdb/.
[40] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for iot applications," *Future Generation Computer Systems*, vol. 90, pp. 149–157, 2019.
[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.