QATAR UNIVERSITY

COLLEGE OF ENGINEERING

REINFORCEMENT LEARNING BASED APPROACHES FOR RESOURCE ALLOCATION IN SMART HEALTH SYSTEMS.

BY

AMR ESSAM ABOELENEEN

A Thesis Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computing

June 2022

COMMITTEE PAGE

The members of the Committee approve the Thesis of

Amr Essam Aboeleneen defended on 11/05/2022.

Prof. Amr Mahmoud Salem Mohamed
Thesis Supervisor

Prof. Zhu Han
Committee Member

Dr. Elias Yaacoub
Committee Member

Dr. Mohamed Arselene Ayari
Committee Member

Approved:

Khalid Kamal Naji, Dean, College of Engineering

ABSTRACT

Amr Essam Aboeleneen, Masters : June: 2022, Master of Science in Computing

Title: Reinforcement Learning Based Approaches for Resource Allocation in Smart Health Systems.

Supervisor of Thesis: Prof. Amr Mahmoud Salem Mohamed.

With the emergence of smart health (s-health) applications and services, several requirements for quality have arisen to foresee and react instantaneously to emergency circumstances. Such conditions demand adaptive fast-acting wireless networks and efficient medical IoT devices. Yet, this requires implementing intelligent network selection and resource management schemes that account for heterogeneous networks characteristics and applications' QoS requirements. Although much literature works to solve these two problems, almost none has considered optimizing both sides intelligently at once (Network's and IoT device's side). Thus, In this thesis, we aim to fill this gap by firstly adopting an intelligent Reinforcement Learning (RL)-based network selection scheme on the Internet of Medical Things (IoMT) device. This will enable the IoMT to be more efficient in adjusting the compression ratio and select the most suitable radio access network (RAN) to transfer the acquired data while considering patient state, battery life, and network dynamics. Secondly, we extend the work by optimizing the network side resources using intelligent network slicing. In which we propose a cost-efficient DRL-based network slicing framework that sustains a high level of network's operational performance by supporting diverse and heterogeneous services, while considering key performance indicators (KPIs), e.g., reliability, energy consumption, and data quality.

Specifically, In the second contribution, we aim to find the least-cost route and resources per route for different service flows. Our results from the first contribution show an improvement in the IoT device efficiency demonstrated in longer battery life in addition to a reasonable delay and distortion levels. On the other hand, our approach in the second contribution outperformed the optimal resource allocation algorithm in finding the least cost path and resources per service flow.

DEDICATION

To my family

ACKNOWLEDGMENTS

This work was made possible by NPRP grant # NPRP13S-0205-200265 and NPRP grant # NPRP12S-0119-190006 from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

I want to express all gratitude to my supervisor, Prof. Amr Mohamed, for accepting my thesis and his ongoing encouragement, direction, and support during this research. Also, I would like to express my appreciation to Dr. Alaa Abdelatif for his technical guidance during this project. Finally, I extend my heartfelt gratitude to my family for their unwavering support, without which this project would not have been possible.

Amr Aboeleneen
February 2022

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PUBLICATIONS

[1] A. Abo-eleneen and A. Mohamed, "Mmrl: A multi-modal reinforcement learning technique for energy-efficient medical iot systems," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, IEEE, 2021, pp. 2026–2031.

[2] A. Aboeleneen, A. Awad, and A. Mohamed, "Rlens: Rl-based energy-efficient network selection framework for IoMT *("Accepted in IEEE WTS Conference 2022")*," 2022.

[3] A. Awad, A. Aboeleneen, H. Dawoud, A. Mohamed, A. Erbad, and M. Guizani, "Intelligent-slicing: An ai-assisted network slicing framework for 5g-and-beyond networks*(Under preparation)*," 2022.

CHAPTER 1: INTRODUCTION

This chapter discusses intelligent resource allocation for m-Health systems and its importance. We first introduce the motivation behind the ideas presented in the thesis, discuss the problem statement and conclude with the main contributions and thesis structure.

Motivation and Objectives

Nowadays, healthcare is considered one of the top priorities for all countries. Because of its importance, worldwide expenses in the healthcare sector are on the rise. Moreover, it is predicted that those expenses per GDP will rise to more than 10% compared to 8% in 2015 [1]. One reason for that is the natural population increase, chronic diseases, and worldwide pandemics such as COVID-19 [2]. In addition, the continuous evolution in networks and the medical field allowed for telemedicine services, including the capability of remote surgeries, remote consultation, and other services. These reasons have contributed to increasing the amount of data that needs to be transmitted and processed, thus introducing many challenges to smart healthcare (s-health) systems. Over the years, many solutions have been introduced to solve the mentioned challenges. For example, in the case of remote monitoring, the mobile health (m-health) system flourished, which utilizes edge devices with higher resources than patients' IoT devices to aggregate and process data coming from different sensors before sending them to hospitals. However, this has caused even more challenges to optimize the edge computations to efficiently process data before communication (i.e. edge computing).

Moreover, the advancements in network virtualization and mainly virtualized network function (VNF) opted for flexible virtual network resource allocation. This has

enabled new technologies such as network slicing, which allocates a network partition per medical service (i.e., a slice for remote surgery application). However, this creates a new problem with the adequate resource amount to allocate per service that meets its requirements.

The above challenges have created the primary motivation behind the writing of this thesis. In this thesis, we use artificial intelligence methods, specifically Deep Reinforcement Learning (DRL), to solve the two challenges mentioned above. For each of the challenges, we formally introduce them along with our contribution in a stand-alone chapter form. Indeed, in Chapter 3, we deliver our first contribution where we consider a DRL-based method for optimizing the IoMT devices' efficiency in the case of remote monitoring. Figure 1.1 highlights the scope and primary operations of the first contribution from patient devices to radio access networks (RANs). Additionally, in Chapter 4, we deliver the second contribution as we seek to dynamically optimize the cost of resources allocated for different medical services through DRL-based network slicing. Figure 1.2 illustrates the focus of the second contribution, which represents a second stage after the first contribution, related to the network side's resources.

Therefore, in the following, we summarize the main objectives of the thesis:

- To study the scenarios of remote monitoring and its effect on power-limited IoMT, suggest an improvement to the core operations of the IoMT, and formulate a multi-objective problem accordingly.

- To model the problem of network selection, considering different problem's dynamics, including KPIs such as different routes, and different computing resources per intermediary nodes.

- To model and formulate the scenarios above as sequential decision problems where the health system can be characterized as a Markov Decision Process (MDP) and hence RL can be used to optimize the performance of user and network sides in m-health systems.

- To leverage RL-based techniques to solve the formulated problems and compare them against other baselines through many experiments.

- To test RL's ability to track and adapt efficiently to the complex environment's dynamics that naturally happen in practical health related applications e.g. change in resources of one of the RANs.

In light of the objectives above, we also outline the research questions as follows:

- **RQ-1**: To what extent can the lifetime of IoMT be increased by carefully choosing the RAN and compression level in a user-centric scenario.

- **RQ-2**: How long will an RL agent take to learn the stochastic nature of the complex health scenario with multiple RANs, and varying resources.

- **RQ-3**: Can we increase the Quality of service (QoS) for individual users by carefully allocating the virtual resources in the network.

- **RQ-4**: How well will RL learn the virtual resources allocation at the network level to maximize the users' QoS.

- **RQ-5**: What will be the RL performance against typical optimization methods and state of art baselines.

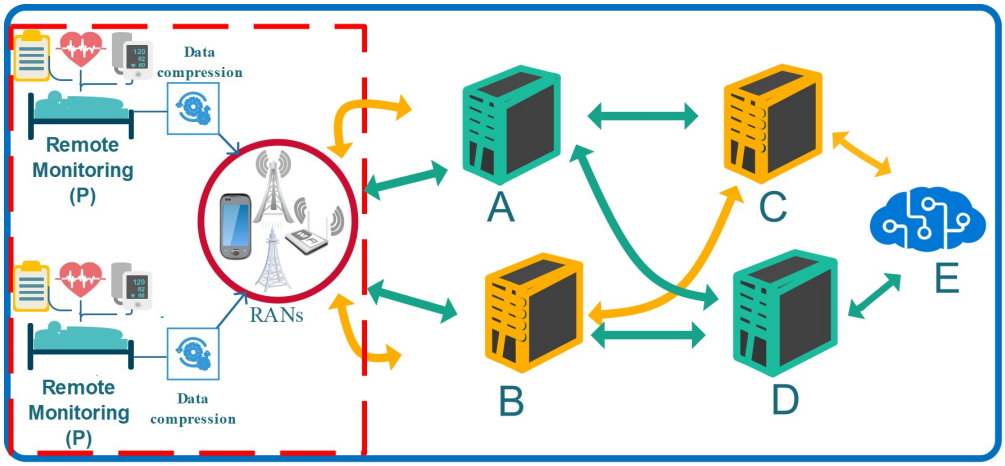


Figure 1.1: The focus of the first contribution, related to IoMT's side

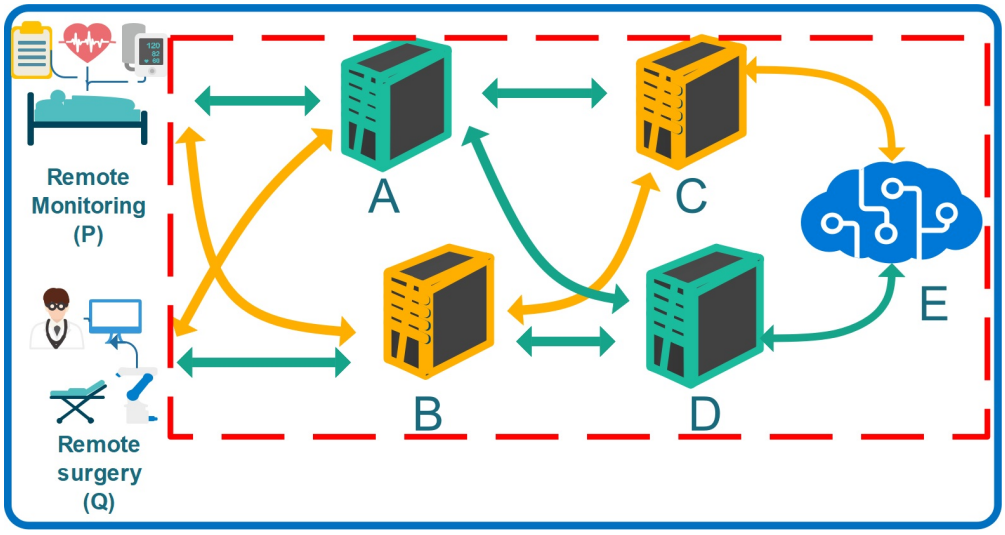


Figure 1.2: The focus of the first contribution, related to network's side

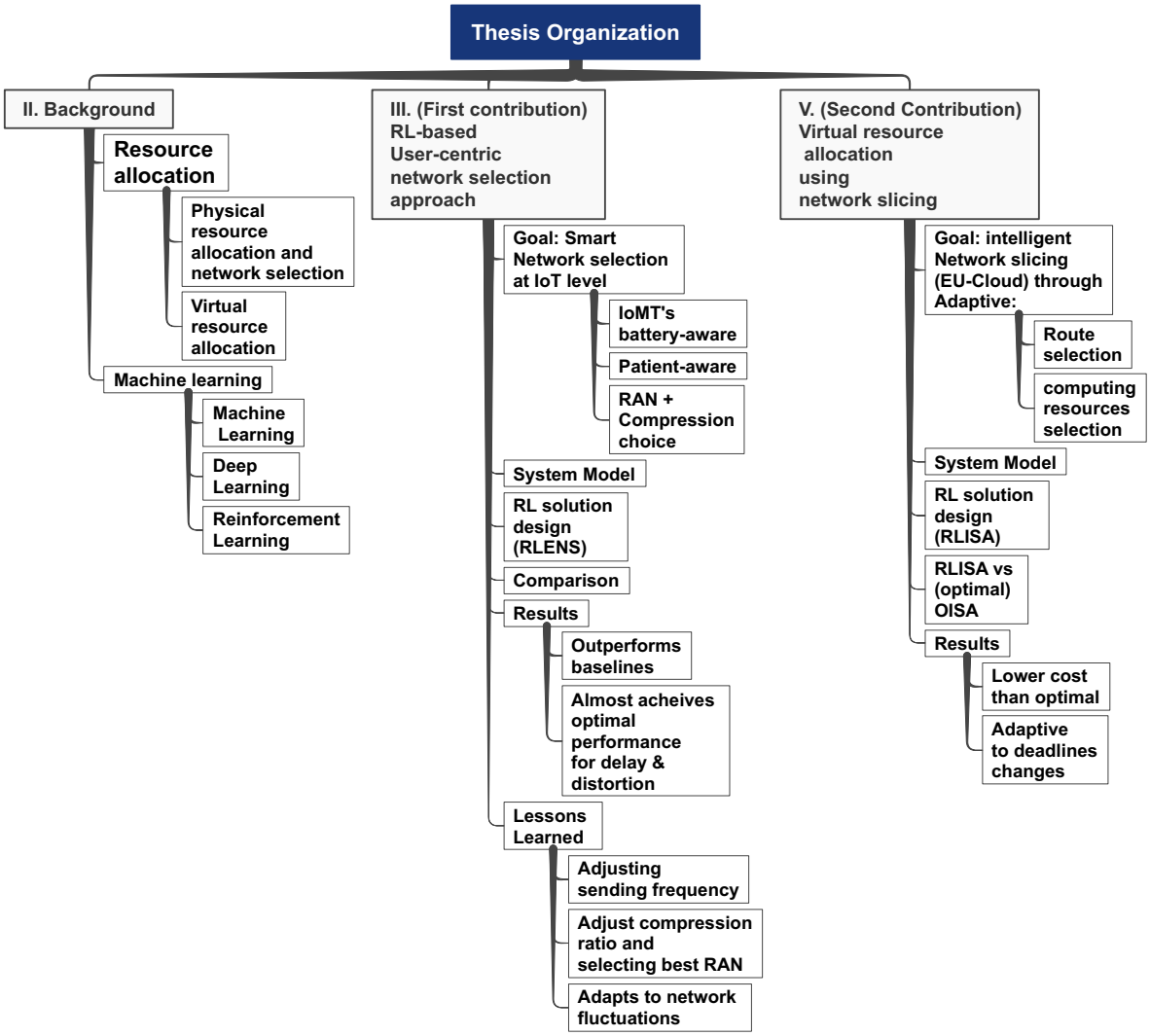


Figure 1.3: Thesis Organization

Thesis Overview

The rest of this thesis is organized as follows: Chapter 2 discusses the underlying context, key concepts, and terminologies. Additionally, we examine and compare our work to that of others in the field. Chapter 3 presents the first contribution related to user-centric physical resource allocation by proposing a method based on DRL. Chapter 4 introduces the notion of network function virtualization and demonstrates how DRL can increase service satisfaction while reducing the total cost. Chapter 5 summarizes the work, discusses main findings, and Chapter 6 recommends some future improvements.

CHAPTER 2: BACKGROUND AND RELATED WORK

This chapter discusses the basic concepts on which this thesis is built. To begin, in Section 2.1, the importance of healthcare is discussed along with the new trends for improving healthcare, such as smart health. Remote monitoring is then discussed as an example of smart health use. After that, smart health's resource allocation problem is discussed in Section 2.2. Finally, in section 2.3, the background is concluded by discussing the methods used to optimize smart health's resource allocation using machine learning techniques.

Smart Health and Remote monitoring

Nowadays, healthcare is considered one of the top priorities for all countries. Because of its importance, the worldwide expenses on the healthcare sector are on the rise. Moreover, it is predicted that those expenses per GDP will rise to more than 10% compared to 8% in 2015 [1]. One reason for that is the natural population increase, chronic diseases, and worldwide pandemics such as COVID-19 [2]. Indeed, the aforementioned reasons have affected healthcare greatly. Taking COVID-19 as an example, the number of related cases worldwide were increasing weekly by more than 200% with a mortality rate of 5% [3], [4]. According to [5], the significant increase of COVID-19's patients has greatly affected hospital activities. Many elective operations have been curtailed, and some hospitals have even declined the admission of critical-condition patients, leading to a rising death rate. Moreover, the public fear of becoming infected from hospital visits prevented people from attending their regular appointments [6]. Furthermore, the continued consumption of hospital resources has left other patients with long queues on imaging, a delay in accessing vital-measurement machines, and fewer follow-ups with

their doctors. These problems even escalate because of readmission of patients or the existence of other disease [7]. All of the above have resulted in a drastic decline in overall public health and domestic mortality with no efficient remote treatment tools.

Thus, a solution that would provide an automated remote monitoring and alerting system would save urgent lives by reducing the time of diagnosis, minimizing patient contact,lowering hospital-based infection and reducing the load on hospital resources. In that sense, smart health was introduced. Smart healthcare (s-health) enables advanced screening tools to provide advanced treatment to patients, and smart healthcare systems to improve healthcare quality by providing patient's biological indicators in real-time. The goal of smart health care is to help patients by providing information about medical issues and their solutions. Smart health care enables patients to take appropriate action in the event of a critical situation [8]. It enables remote check-up services, which reduces treatment costs and assists health care providers in expanding their services beyond location boundaries. With the growth of smart cities, a robust and smart healthcare system is required to ensure that users have access to healthcare services. Aside from well-being, one of the significant contributions is the reduction of healthcare costs through timely diagnostics.

Currently, IoT devices are considered the cornerstone of the smart-health system. These small devices are usually sensors, actuators, and microcontrollers that collect patients' biological data, process it, and send it to remote servers for analysis, alarming the hospital for immediate intervention when needed. Currently, IoT has been used in healthcare to improve a variety of applications, including hospital asset management, behavioral change monitoring, telemonitoring, aged care, medication's effect tracking, and telemedicine[9]. This use of IoT in health care has improved the medical process

and has opened the door for further medical improvements.

However, creating IoT devices to be small and efficient is still a challenge. Since, by design, the specification of these devices is limited, the inefficient use of those devices would lead to an ineffective IoT system that processes and sends out large amounts of data (usually wirelessly), leading to a quickly drained battery. According to [10], this is the biggest problem that affects medical IoT devices sending data to a remote analysis server. Apart from medical IoT, this problem is usually solved if the IoT devices are connected to the power grid. However, many devices, including the Internet of Medical Things (IoMT), are usually made to be mobile and battery-operated. Thus, there is a present need for better ways to increase efficiency and reduce power consumption. To tackle this issue and provide more efficiency to IoT devices, researchers and companies have carried out different efforts. These efforts can be summarized into (1) the development of efficient data-compression algorithms which will reduce the amount of data the IoT device has to transmit, (2) the use of energy-efficient or hybrid transmission technologies for IoT (i.e., NBIOT, LTE-M), (3) pushing the complexity of processing towards an edge device, and (4) introducing energy-harvesting mechanisms which will power these devices from natural sources (i.e., body temperature). Based on the four aforementioned points, a review of the related work regarding IoT efficiency improvement in smart health will be provided.

Since the IoT device usually collects, saves, and publishes data constantly, increasing battery usage, data compression algorithms have been designed to reduce the data's redundancy and the amount of transmitted data. Techniques such as compression sensing, time sampling, and lossy compression have been widely investigated because of their performance effectiveness. It is worth mentioning that the type of data that

the IoT collects drives the type of compression used; critical data will opt for lossless compression while some signals data, such as ECG signals, can tolerate some loss and use lossy compression. Authors in [11] showed a hybrid lossless efficient data compression that achieved about a 40% compression ratio. Bio-signals' (i.e., ECG and RESP) lossy compression was discussed in [12] where the authors used a dictionary-based technique to improve the medical IoT battery cost. The author also showed that the auto-encoder-based method had the lowest energy consumption amongst other compression algorithms. Similar work by [13] achieved a mean compression ratio of 2.1 with 53% less battery usage for lossless compression while achieving a 7.8 compression ratio with 18% less battery usage using a lossy compression scheme. It is essential to mention that a high compression ratio may significantly distort the data on the recipient's side. Thus, papers such as [14], [15] address this issue either by using historical data characteristics or multi-modality. In fact, the latter developed a deep learning multi-modal compression algorithm that reduced distortion by 73.37% from a conventional compression algorithm.

One of IoT's advantages is their ability to communicate with each other or to a remote server. However, the problem in choosing a data transmission technology that provides good range, adequate bandwidth, low power consumption, and high reliability remains. Unfortunately, a solution that combines all of the above does not exist yet and is still challenging. Data transmission technologies in IoT can be partitioned into two main categories: long and short-range communication. Short-range communication protocols include technologies that are usually used for connecting multiple IoT devices together. Examples includes protocols such as Bluetooth low power(BLE or Bluetooth smart)[16], low power WI-FI[17], mesh-based technologies such as Zigbee [18], Z-Wave

[19], 6LoWPAN [20], and finally near field communication such as NFC and RFID. On the other hand, long-range communication includes technologies used to send data to the broad receiver (typically 500M+). Examples include 2G, 3G, 4G, LPWAN, SIGFOX, LoRA, NB-IoT, LTE-M, and LTE CAT-1. A detailed comparison between most of these technologies is presented in [20]. Moreover, application-layer protocols are also crucial in IoT; since IoT devices have limited space, using data-overhead protocols such as HTTP will waste both storage and processing power. Thus, protocols such as CoAP and MQTT are more adequate for data transmission.

Relying on edge devices to process and transmit data is another means of increasing the efficiency of IoT devices. In [21], authors had shown how an intermediary edge device could provide temporary monitoring of a patient's condition using machine learning (ML) without the constant need for transmitting data to a remote place using cellular technology. This work showed a battery reduction of 60%, and it is one of the pillars of our work. However, this assumes the permanent existence of a nearby device, which is not always the case. On the other hand, [22] showed the use of Deep Reinforcement Learning (DRL) to provide the optimal policy for transmitting data securely with the least energy consumption. The researchers showed a significant improvement in terms of battery life against greedy algorithms. However, the work does not consider the criticality of the data sent from the patient's perspective, which results in sending data too frequently and draining the battery. Unlike the previous works, the proposed work in Section 3 considers different aspects of energy consumption, overall delay, and patient's urgency level to optimally allocate resources in the network.

Since the introduction of computer networks, congestion control and resource allocation problems are presented as the most severe problems. Because many different user applications utilize the same network (i.e., sending and receiving data), the data packets can be queued at the router buffer. Many packets can be dropped, consequently reducing the quality of service (QoS) for different users. Although delaying or blocking some users temporarily from sending might prevent network congestion, the resource-allocation mechanism is rather used to guarantee fairness. Resource allocation splits the network resources among the users and controls which users can send data and when.

Physical resource allocation

Allocating Resources to different users may require control from the routers and network equipment (router-centric) or the end-users themselves (host-centric) or both parties. In that regard, many approaches have been explored in the literature that will be discussed next. Authors in [23] divided these efforts into four broad categories, which include cost-based, game-theoretic decision-making techniques, Markov decision processes (MDPs), and optimization. Most of these ideas are applied on a heterogeneous network where the resources are allocated for every node across different Radio access networks (RANs).

Looking at each of the methods mentioned above, resource allocation based on cost was introduced in [24], [25], where [24] perpetrated a Lagrangian distributed technique for reducing the resources cost per node while meeting the different constraints. Although this method showed an overall good performance, it lacks scalability when

different parameters are added to the resource allocation problem[25]. Other works in [26], [27] showed the use of game theory in assessing network bandwidth allocation by different hosts. The introduced methods showed a novel use of game theory, but the convergence of an optimal solution is not always guaranteed, and the complexity is a significant flaw. Moreover, MDP-based approaches have been explored in [28], [29] with the drawback of the scalability of the network. Furthermore, optimization techniques for resource allocation are complicated as the problem might be NP-hard; one strategy for tackling that issue is loosening the constraints, altering variables, or considering online adaptive approaches such as Q-learning [30]. Many works in the literature did not consider energy efficiency in network selection for resource allocation. Thus [31] considers the energy and other parameters such as RAN price and QoS in a multi-objective optimization function in a host-centric approach. Additionally, the author shows that the optimization solution is traffic-aware and can adapt to different dynamics in a reasonable amount of time. However, similar to other optimization functions, scaling up or adapting to high network dynamics is often challenging.

Virtual resource allocation

With the proliferation of cloud computing and scalable computations in the era of beyond fifth-generation (B5G) networks, several issues relating to the rigorous needs of such apps have arisen (such as low latency, excessive data rates, energy consumption, etc.). Thus, the primary function of B5G networks is to provide a diverse range of services for various sectors with varying requirements in a scalable manner. This necessitates meeting distinct performance criteria for each service, posing critical issues for mapping such needs into network design, resource allocation, and packet forwarding

decisions. [32].

Network function virtualization (NFV) has emerged as a possible option for providing flexible management and administration of all network resources while ensuring End-User (EU) Quality of Service (QoS). Indeed, network functions are conducted on a single infrastructure with the use of software-based NFV solutions, simplifying the process of updating, adding, or removing a service from customers. Thus, NFV enables network operators to scale up or down services in response to customer demand while simultaneously cutting overall costs via the use of low-cost network components and topology.

Different sectors provide their service needs through a graph of virtual network functions (VNFs), and network operators translate these requirements into network management choices. This necessitates the development of novel strategies for maximizing the association of radio access networks and the allocation of VNFs across a range of network resources. Notably, the type and quantity of resources given to each service are crucial since their cost and availability substantially impact performance. Network slicing has emerged as a virtual networking design in this context, transforming B5G networks into software-defined networks [33]. Network slicing allows for partitioning a single physical network into many virtual networks that may serve a variety of services.

Because of NFV, the scalability of any network has been possible. However, the physical resources will be depleted without an exemplary resource allocation plan. In that regard, many authors have discussed the idea of optimal VNF placement and intelligent allocation of resources with network slicing[34], [35]. [33], for instance, investigates a management architecture based on NFV components for dynamically deploying instances of virtual tenant networks (VTN). Other authors, such as [36],

reduce the challenge of placing VNFs by exploiting the fluctuation of network traffic demand, ultimately improving resource allocation. Moreover, The researchers of [37] exploited the best network functional split and route from end-users to the central entity to increase the throughput per application.

Additionally, the authors of [38] developed a collaborative decision-making approach that relies on a queuing model called MaxZ that aims to reduce the problem's complexity by isolating the NVF placement and CPU assignment choices while maintaining their dependency. The presented findings demonstrate that MaxZ is near to the optimal. In [39], this approach has been improved to propose a mechanism for allocating resources and networking technologies under some constraints.

Learning techniques

Machine learning

With the help of machine learning, computer systems can perform complicated tasks such as forecasting, analysis, recognition, and planning through learning from historical data. Data and algorithms are critical components for machine learning models. Large datasets and high-quality data can significantly improve the accuracy of machine learning models. Additionally, it is essential to use appropriate algorithms to tackle multiple issues, especially those involving various types of datasets. Machine learning is the general umbrella of various learning methods such as supervised, unsupervised, semi-supervised, and reinforcement learning, which are presented in Figure 2.1.

In Supervised Learning (SL), a machine learning algorithm is provided with some training datasets, which contain input features and a prediction value. The algorithm aims to estimate a specific function (i.e., hypotheses function) that maps the input to the output. The SL categorizes the type of operation according to the output value. If the predicted value is a number, the process is called regression. On the other hand, if the predicted output is a class label, the operation is known as classification. Both regression and classification tasks can be addressed with many algorithms. For example, there are linear regression, polynomial regression, Lasso regression, and others under the regression family. Similarly, in classification, Naive Bayes [40], Support Vector Machines (SVM) [41], and Discriminant Analysis [42] are some examples. The problem with SL is that human intervention is crucial. Not only do people label the output for the training set, but they also select attributes, algorithms, and hyper-parameters. Supervised learning is typically employed in domains where the human

model requires specific knowledge and skill. However, the SL approach necessitates additional data processing for feature selection and anticipates parameter tweaking for optimal algorithm configuration.

In Semi-Supervised Learning (SSL), there are very few labeled data (e.g., 10%-20%) and the rest of the data are unlabeled. The idea of SSL is to use the unlabeled data together with labeled data because building a model with few labeled data is not feasible and often leads to low-performance predictors, as such, in different iterations, unlabeled data points receive some labels or values, and the performance is improved using different iterations.

Unsupervised learning (UL), in contrast to the SL, are mathematical models that employ just training datasets of input vectors and do not include a target variable. Typically, unsupervised learning provides the pattern of input variables and frequently presents various clusters formed from the input data. K-means clustering, Gaussian mixture models, and kernel density estimators are some examples of UL algorithms.

Deep Learning (DL) was recently introduced as a subset of machine learning that uses more advanced techniques than traditional shallow machine learning techniques. The concept of neurons and Multi-Layer Perceptron (MLP) topology existed before the DL was adopted widely. In DL networks, there are many hidden layers, and different types of layers exist, such as a convolutional layer, pooling layer, and dropout layer. These new architectures led to better function approximations, enabling advanced functionality for DL. The paper presented by [43] shows a comprehensive overview of DL advances. Nowadays, the DL field includes many models and structures. These algorithms can be categorized into two main categories.

Firstly, descrimitive models are used for optimal class discrimination for better data

classification. This class includes multilayer perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN). Secondly, Generative Models are the neural networks that focus on learning how the data was created to generate similar data. Below is a summary of different algorithms used in each category above.

- **Convolutional Neural Network (CNN)** : CNN is mostly utilized for handling images and performing operations such as feature extraction and recognition. Numerous CNN versions exist, including visual geometry group (VGG) [44], AlexNet [45], Xception [46], Inception, and ResNet [47]. All of these variants were used in different applications and reusability was achieved by using transfer learning.

- **Recurrent Neural Network (RNN)**: The RNN [48] algorithm is characterized by its capabilities to predict the target based on current and historical data. Since RNN suffered from vanishing gradients problem, which impacted its ability with long sequences of data, few other candidates were introduced, such as Long Short-Term Memory (LSTM) [49], Bi-directional LSTM (BI-LSTM), and Gated Recurrent Units (GRUs) [50]. These methods are usually strengthened by using an attention-based mechanism.

- **Generative models**: These models are usually used to generate data samples. It automatically detects and learns patterns or regularities in input data for the model to create new data samples from the original dataset. Generative Adversarial Network (GAN) [51] is constructed from two neural networks: a generator (G) that generates new data with comparable features to the original data and a discriminator (D) that forecasts the likelihood that the following sample will be pulled

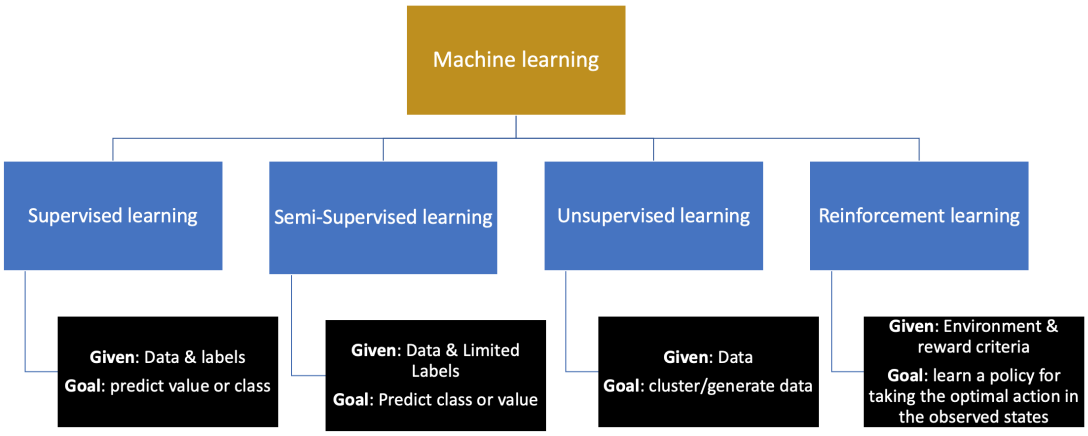


Figure 2.1: Machine learning types

from genuine data rather than the generated data. Thus, both the generator and discriminator are trained to work against one another in GAN modeling. While the generator attempts to deceive and confound the discriminator by providing more data that are genuine, the discriminator attempts to separate original data from G's created data. The family of generative models also includes Auto-encoders (AE) [52], Restricted Boltzmann Machines (RBM) [53], Self-Organizing Maps(SOM) [54], and Deep Belief Networks (DBN) [55].

Although supervised and unsupervised learning techniques have been used in multiple problems, especially networking tasks, both techniques do not possess the intelligence to take actions in a system, not to mention dynamic systems as network systems. Fortunately, reinforcement learning can solve such problems by learning in an interactive environment. In addition, reinforcement learning promises adaptiveness and other features discussed in the next section.

Reinforcement learning

RL [56] is one of the machine learning approaches for an agent to gain knowledge. Unlike other learning techniques, RL is used to find the optimal decision in an interactive environment and does not require a dataset to learn from as it learns directly from the interaction between the agent and the environment. A basic illustration of the RL process is presented in Figure 2.2. Conceptually, RL defines an actor (i.e., agent) and a simulated world/environment, which the agent will interact. The agent runs within a simulated world. The way the world reacts to the agent's particular behavior is specified by a model that is not necessarily known to the agent. The agent may remain in one of the numerous states $s \in \mathcal{S}$ of the environment or may pick one of the multiple actions $a_t \in \mathcal{A}$ to transition between states. The transition probabilities between states determine the agent's final state (P). After applying an action, the environment provides feedback in the form of a reward $r \in \mathcal{R}$.

RL is usually applied in sequential choice tasks, where the current selected action affects the future states such as games, software testing, and others. RL relies on the Markov Decision Process (MDP)[57] and has other concepts such as policy and value function discussed in this section. The fundamental components of an RL system in an MDP problem are ($\mathcal{S}$,$\mathcal{A}$,$\mathcal{T}$,$\mathcal{R}$,γ), where $\mathcal{S}$ represents the list of possible states, $\mathcal{A}$ denotes the list of possible actions at state $\mathcal{S}_t$, and $\mathcal{R}$ denotes discounted reward using a discounted factor $\gamma \in (0, 1]$. In the RL process, the next state $\mathcal{S}_{t+1}$ is determined for each episode based on the following transition probability

$$P(s', r|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]$$

and the reward. In RL, the training process starts with the agent performing random actions $a_t \in \mathcal{A}$ at different state $s_t \in \mathcal{S}$ and receiving rewards r_{t+1}. During this training phase and by using the feedback reward, the agent builds a policy $\pi_t(a|s)$ that associates a specific action $\mathcal{A}_t$ with a specific state $\mathcal{S}_t$ that guarantees the highest accumulated rewards given by the following equation.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

This training phase keeps running until the agent exceeds specific timestamp threshold or number of seconds per episode [58]. The agent is then trained for multiple episodes until converging to the best policy in terms of accumulative rewards.

To decide what action to perform at a given instant, the agent must understand how beneficial it is to be in a particular condition. The *value function* is a technique for determining this operation precisely for each state. It is defined as the total of anticipated rewards following the policy π from the current state onward and is given by the symbol $V_\pi(s)$ and equation:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

Similar equation can also be used to provide the goodness of a specific action at specific state $Q_\pi(s, a)$ given by the following equation.

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

During the training phase, the RL agent collects the interaction trajectories $S_1, A_1, R_2, S_2, A_2, \ldots, S_T$ and learns from them. However, since the RL agent is built

upon MDP, it relies only on the last interaction to get the next reward, which reduces the complexity of the problem.

Different RL algorithms may be used based on the type of the model. If the model of the system is known to the agent, Dynamic Programming (DP) can be adopted for finding the best solution to the environment, and the algorithms used for RL in such an environment are called model-based RL algorithms, examples include [59], and "Model-based value estimation" (MBVE) [60]. On the other hand, if the system model is unknown, the agent must use model-free algorithms that explicitly learn the statistical model of the environment. Model-free RL includes Monte Carlo (MC) and temporal difference (TD) as the most common approaches [58]. These two approaches have been adopted in multiple different learning agents, which can be grouped into three different groups as follows:

- **Value-based**: These methods use TD learning that updates the value function on each step according to a temporal error calculated between old and new value functions. Algorithms that fall into this category are State–action–reward–state–action (SARSA), Q-learning, and Deep Q-network (DQN). These methods are characterized by lower variance, but more bias [61].

- **Policy-based**: Unlike value-based methods, these methods use MC learning that updates a policy at the end of the episode. Changes are done directly to the policy without demanding a particular metric. Examples include Proximal Policy Optimization (PPO) [62].

- **Actor-critic**: These methods harness the power of both value-based and policy-based learning and use MC and TD learning approaches to lower the bias and

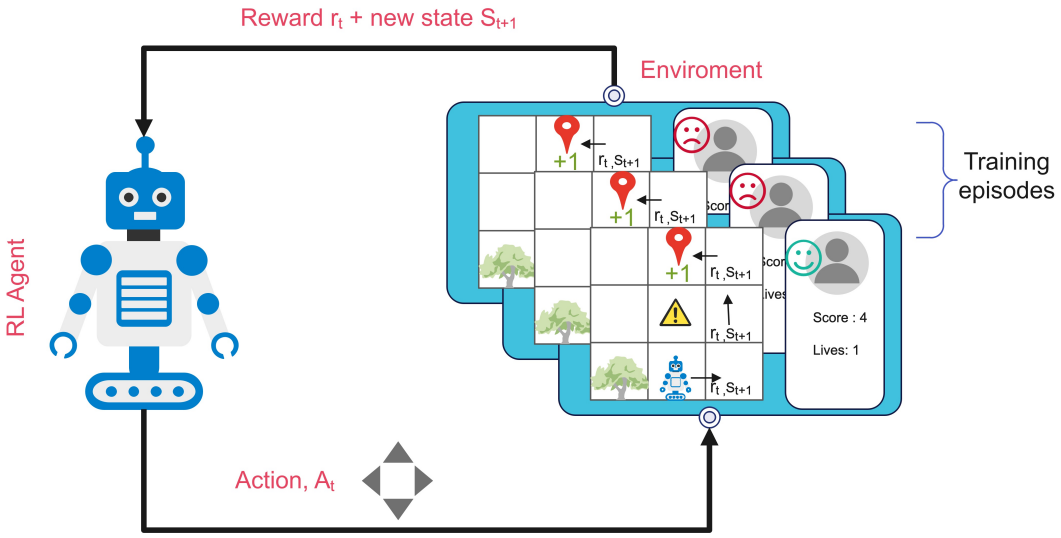


Figure 2.2: Reinforcement Learning cycle

variance from the above-mentioned methods. Examples include Deep Deterministic Policy Gradient (DDPG) [63].

CHAPTER 3: PHYSICAL RESOURCE OPTIMIZATION USING DRL

In this chapter, the first contribution in the thesis is presented where the problem of network selection is tackled by adopting an intelligent Reinforcement Learning (RL)-based network selection scheme. Specifically, we leverage edge computing capabilities to implement a user-centric network selection algorithm at the Internet of Medical Things (IoMT) level, to sense the medical urgency of the patient and act accordingly to adjust the compression ratio and the most suitable radio access network (RAN) to transfer the acquired data while considering different application's requirements, networks dynamics and IoMT state. Our results demonstrate the efficiency of the proposed approach in outperforming the state-of-the-art techniques in terms of battery life by more than 500% while reaching almost 85-90% of the optimal algorithm's performance in delay and distortion.

Several methodologies have been adopted in the literature for solving the network selection problem, including: optimization techniques [64], [65], game theory [66]–[68], Markov decision processes (MDPs) [69], [70], and multi-attribute decision making [71], [72]. However, most of these approaches build on complex mathematical models and instantaneous channels information. Indeed, to guaranty the optimality, while considering diverse networks, applications, and power constraints usually result in an NP-hard problem [64]. Also, leveraging game theory, MDPs, or multi-attribute decision-making approaches is computationally intensive, especially in the case of large networks, and their convergence is not guaranteed. Even if they converge, it is not always guaranteed to converge to an optimal solution.

Accordingly, relying on traditional network selection methodologies, which heavily rely on mathematical models and consider only instantaneous network state, can not cope with the highly-dynamic environments nor the next generation network demand for swift connectivity and quick responsivity. To address these challenges, we opt to leverage the potential of Reinforcement Learning (RL) [73] to develop an intelligent, user-centric network selection scheme for s-health systems. Although few studies have been presented for network selection using the Q-learning method [74], or RL [75], [76], we are still at the beginning level. Thus, this paper aims at advancing the state-of-the-art by:

1. Defining a holistic network selection problem that optimally selects the adequate RAN and compression ratio at the patient level while considering the data distortion, patients' state, and end-to-end delay, i.e., due to processing, transmission,

and queuing.

2. Leveraging efficient RL-based algorithm that considers all system's dynamics to solve the formulated problem. Indeed, we formulate a multi-objective reward function that features the trade-off between energy consumption, delay, and distortion.

3. Conducting comparative experiments to demonstrate the performance of the proposed scheme in comparison to two baselines, namely, energy-greedy and quality-greedy, in addition to a state-of-the-art algorithm.

4. Demonstrating the adaptiveness of our approach to swift network dynamics through introducing some disturbance to the converged RAN. Our results depict how the proposed scheme can adapt to diverse network dynamics while changing the action distributions with a reasonable number of episodes.

The rest of the chapter is organized as follows. Section 3.2 introduces our system model, the environment we are simulating, and the different constraints of the problem. Section 3.3 shows how we were able to transform our problem into MDP and solve it using RL. Finally, Section 3.4 demonstrates our conducted experiments along with our simulation results, while Section 3.5 concludes the chapter.

System Model and Problem Formulation

This section presents the considered system model and the formulated optimization problem for network selection.

System model

This chapter mainly focuses on efficient remote monitoring (Tele-monitoring) for elderly or severely infected patients. The presented model in Figure 3.1 shows the main components of the system. This model considers one home patient who is physically connected to multiple sensors. At each timestep t_i, the sensors will gather two types of data, namely the vital signs and stream-based data. The patient's vital signs (e.g., Blood pressure, Oxygen Saturation (SpO2), lung airflow) will be used to identify the urgency of the patient's condition based on the NEWS system (which was discussed in our earlier paper [77]).

In addition, the sensors will also collect more stream-based data, such as the electrocardiogram (ECG) that will be inserted into the IoT's Buffer (Event Buffer) before getting transmitted to our targets. We consider transmitting stream-based data such as ECG as a generalization of the widely used data type in remote monitoring. Since the IoT device is battery-operated and to reduce the burden on Hospital servers, we opt to maximize the battery state of charge (SOC) by utilizing adaptive edge pre-processing operations (e.g., compression, encoding, and quantization) before transmitting ECG signal data to the target hospital servers or nearby helping device. In this scenario, and for implementation purposes we consider that the IoT device is connected to a tiny battery (i.e., less than 1mAh).

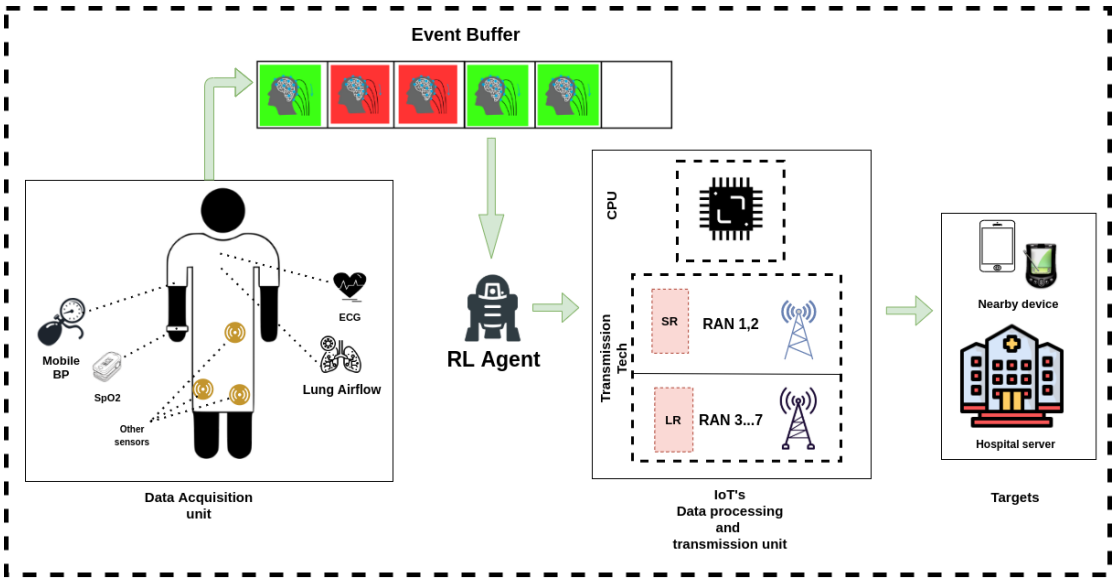


Figure 3.1: System model under study.

Compression model Each ECG signal data consists of B_i bits. Before transmitting the data, our IoMT device has to compress the data with a compression level $\mathbb{C}$ such that the number of bits after compression b_i is calculated as $b_i = B_i \cdot (1 - \mathbb{C})$. Each compression level $\mathbb{C}$ will cause a distortion κ, and will impose a compression energy-cost $\varphi_{compression}$ and compression delay $\mathbb{C}_d$. Since we are using stream-based data, we follow the compression-distortion model used in [78] as follows:

$$\kappa = \frac{c_1(1-\mathbb{C})^{-c_2} + c_3 F^{-4} - c_5}{100} \tag{3.1}$$

where F is the length of wavelet filter used in compressing the signal, and the parameters $c_{1...5}$ are approximated parameters of the ECG's distortion analysis.

The compression energy-cost $\varphi_{compression}$ is calculated as the multiplication of the number of compression cycles $\mathbb{C}_{cycles}$ at the cost of each transformation step E_{comp} in

addition to the DWT per computation energy E_{cs} and is described below

$$\varphi_{compression} = \mathbb{C}_{cycles} \cdot E_{comp} + (1 - \mathbb{C}) \cdot E_{cs} \tag{3.2}$$

The number of compression cycles $\mathbb{C}_{cycles}$ can be calculated from the multiplication of the maximum length of wavelet filter F_{max}, with the number of samples N_s and the number of hierarchical decomposition levels L as follows :

$$\mathbb{C}_{cycles} = F_{max} * N_s \cdot \sum_{l=0}^{l=L} \frac{1}{2^l} \tag{3.3}$$

Moreover, since the IoT device has limited processing capability, the compression task will produce a compression delay $\mathbb{C}_d$ which is dependent on the number of compression cycles and CPU per-second cycles CPU_{cps} which can be calculated as follows:

$$\mathbb{C}_d = \frac{\mathbb{C}_{cycles}}{CPU_{cps}} \tag{3.4}$$

The compressed data will then be transmitted to a target destination such as a hospital server or a nearby device that can analyze and process the data locally before transmitting it to the hospital servers.

Transmission model To transmit the data to any target, we assume the existence of *M* Radio access networks (RANS) which differ in communication range, transfer rates, energy costs $\varphi_{transmission}$, transmission delays T_d and queueing delay Q_d.

Following [65], the energy consumption per RAN $\varphi_{transmission}$ depends on the channel gain G_j, transfer rate r_j, bandwidth w_j, bits to transfer l_i and noise spectral

density N_0. In addition to two parameters δ and c which change according to the selected RAN. The Energy gain can then be expressed by $G_j = K \cdot \alpha \cdot |h_j|^2$, where $K = \frac{-1.5}{log(5BER)}$, α is the path loss attenuation, and $|h_j|$ is the fading channel magnitude. Therefore, $\varphi_{transmission}$ becomes :

$$\varphi_{transmission} = \delta \cdot (\frac{l_i \cdot N_0 \cdot w_j}{r_j \cdot G_j} \cdot (2^{\frac{r_j}{w_j}} - 1)) + c \tag{3.5}$$

Queuing model Since the incoming ECG signal is stored in a buffer (event buffer), a queuing M/M/1 model has been used, which has proven to model the arrival of medical events in the literature [79]. In this scenario, we assume that we have N different ECG signal types where each type has a specific urgency level and are arriving at the same buffer with an arrival rate of λ_i such that $i \in \{1, \cdots, N\}$. In this scenario, the service rate μ is considered to be the network transmission rate and changes according to the selected RAN. Because all network service rates μ are extremely larger than the summation of arrival rates $\sum_{i=0}^{N} \lambda_i < \mu, \forall \mu \in R$, the buffer flow is considered to be low or negligible. Following the queuing delay model presented in [79] for Priority preemptive-resume scheduling, the Average Sojourn Time of different urgency levels S_j can be presented as :

$$S_j = \frac{\frac{1}{\mu} \cdot \sum_{n=1}^{i} \lambda_i}{(1 - (\frac{\lambda_1}{\mu} + \cdots + \frac{\lambda_i}{\mu})) \cdot (1 - (\frac{\lambda_1}{\mu} + \cdots + \frac{\lambda_{i-1}}{\mu}))} + \frac{\frac{1}{\mu}}{(1 - (\frac{\lambda_1}{\mu} + \cdots + \frac{\lambda_{i-1}}{\mu}))} \tag{3.6}$$

Now that we have finished discussing our model, we discuss the problem formulation and the constraints in the next section.

Problem formulation

The ultimate objective of this chapter is to find the optimum compression level and RAN that minimizes the overall delay (including the compression, transmission, and queuing delay components), ECG distortion, while maximizing the battery level in the long run. Hence, the optimization problem was formulated as follows:

$$\textbf{P:} \min_{\mu_j, \mathbb{C}} \left(\sum_{j=1}^{M} \mu_j \upsilon_j \right) \tag{3.7}$$

such that

$$\sum \mu_j = 1, \ \ \forall j \in \{1 \cdots N_{RAN}\}, \tag{3.8}$$

$$\mathbb{C}_{min} \leqslant \mathbb{C} \leqslant \mathbb{C}_{max} \tag{3.9}$$

$$0 \leq (\varphi_{compression} + \varphi_{transmission}) \leq \mathbb{B} \tag{3.10}$$

$$\mathbb{N} \in \{0, 1\} \tag{3.11}$$

$$\mathbb{M} \in \{1, 2, 3\} \, if \, \mathbb{N} = 1 \tag{3.12}$$

$$\mathbb{M} \in \{4, 5, 6, 7\} \, if \, \mathbb{N} = 0 \tag{3.13}$$

where the end to end delay ν is defined as

$$\nu = \mathbb{C}_d + T_d + Q_d$$

and total energy is formed as $\varphi_t = \varphi_{compression} + \varphi_{transmission}$. The utility function υ_j that incorporates distortion, delay, and energy consumption altogether is defined as follows $\upsilon_j = \alpha[\kappa] + \beta[\nu] + \gamma[\varphi_t]$ and α, β and γ are the weight coefficients of each goal such that $\alpha + \beta + \gamma = 1$. While the constraint (3.9) bounds the compression ratio

between 20 and 65% to avoid high distortion in compressing medical data. Equation (3.10) limits the choice of an action that will cost more energy than the left battery charge. The N_{RAN} indicates the maximum number of RANs that exist at a certain point of time. Furthermore, constraints (3.13) and (3.12) ensure that certain RANs (e.g., short-range and long-range) be used according to the existence of a nearby helping device or not. This means that our IoMT cannot use short-range communication RAN to communicate with a nearby connected device if that device does not exist.

Using classical algorithms to solve problem **P** will ignore the dynamics of the problem (e.g., patient's urgency state, the battery level, RAN condition) to achieve the lowest delay distortion, which will increase the average energy consumption. Hence in the next section, we propose using deep reinforcement learning (DRL) to optimize the action selection process of compression and transmission while being aware of the environment's nature and dynamics.

The Proposed RL-based Solution (RLENS)

RL is a type of machine learning which has been widely used in the literature [80] because of its in-need of training datasets and its ability to optimize complex environments under dynamic changes. Rather than learning from datasets, the RL agent learns by interacting with an environment where it will receive a reward or a penalty corresponding to how good the action was. The RL problem is formulated as a Markov Decision Processes (MDPs). An MDP is defined as a tuple of 5 elements, namely ($\mathcal{S}$,$\mathcal{A}$,$\mathcal{T}$,$\mathcal{R}$,γ).Where the agent observes the system state $\mathcal{S}$ and will apply an action $\mathcal{A}$ in which it will receive a reward $\mathcal{R}$ that is discounted by γ and the new state $\mathcal{S}'$. When applying an action, a state is transitioned with probability $\mathcal{T}$. With the help of a learning algorithm, the agent will then be able to successfully map states to actions that will produce the highest cumulative reward, which will refer to as a policy. Thus to solve our problem **P** using RL, we have first to transform the problem into MDP, in which we will explain the environment, actions, and rewards.

State The environment state at timestep t is presented as s_t = ($\mathbb{B}_t$,$\mathbb{F}_t$,$\mathbb{N}_t$,$\mathbb{U}_t$,$\mathbb{S}_t$) where $\mathbb{B}_t$ is the battery level of the IoMT device at time t, $\mathbb{F}_t$ indicates the current data amount stored in the buffer, $\mathbb{N}_t$ is a flag to reflect nearby device existence state, $\mathbb{U}_t$ specifies if the patient is in critical state or not (i.e., $\mathbb{U}_t \in \{0, 1\}$) and $\mathbb{S}_t$ is a number indicating the percentage of data sent out of all data.

Action At each timestep, the RL agent will perform an action, either to send data or to be Idle. If the agent chooses to send the data, a compression level must be specified, and a RAN must be selected. The chosen action a_t is presented as $a_t = \{\mathbb{C}_t, \mathbb{T}_t, \mathbb{I}_t\}$ where $\mathbb{C}_t$

is a real number indicating compression level such that $\mathbb{C}_t \in \mathbb{R}, \{\mathbb{C}_{min} \leq \mathbb{C}_t \leq \mathbb{C}_{max}\}$, $\mathbb{T}_t$ is also a number indicating the RAN index to use and $\mathbb{T}_t \in \mathbb{R}, \{0 \leq \mathbb{T}_t \leq 7\}$ finally, $\mathbb{I}_t$ which is an indicator for being idle (i.e., not compressing nor transmitting data) or not, such that $\mathbb{I}_t \in \mathbb{R}, \{0 \leq \mathbb{I}_t \leq 1\}$ with a deciding-threshold of 0.5. If the agent did not choose to be idle, then the chosen action will affect the battery level, overall delay, and distortion of the data. To simplify and avoid scheduling RAN resources, the chosen RAN is assumed to send the data with its maximum rate of one second for only one second, equivalent to one timestep in our system.

Reward We formulate our reward function, which defines our primary objective of minimizing the overall delay and distortion while maximizing the battery life. To be able to achieve that, the reward was split into immediate and final rewards such that the immediate reward considers delay, distortion, or being idle. And The final reward will assess the battery level left after finishing each episode. Thus the reward function r_t is described as follows:

$$r_t = \begin{cases} (0.5, \cdots, 0) \mapsto & \mathbb{C}_{min} \leq \mathbb{C} \leq \mathbb{C}_{max} \\ (0.5, \cdots, 0) \mapsto & \varrho_{min} \leq \varrho \leq \varrho_{max} \\ 0 & idle \text{ and } \mathbb{U}_t = 0 \\ -1 & idle \text{ and } \mathbb{U}_t = 1 \\ \mathbb{B}_{left} * 100 & if \ \ \mathbb{S}_t = 1 \end{cases} \tag{3.14}$$

In this work, we have decided to use the Soft actor-critic (SAC) [81] algorithm, which provides different features that is paramount in our scenario. Unlike other RL algorithms that maximize the cumulative reward only and use a deterministic policy, SAC learns a stochastic policy that tries to increase both the cumulative reward and

entropy $\mathcal{H}(.)$ of selecting an action by combining them into a single reward function $J(\theta)$ which can be defined as follows:

$$J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_{\pi_\theta}}[r(s_t, a_t) + \alpha\mathcal{H}(\pi_\theta(.|s_t))] \tag{3.15}$$

where α controls the importance of the entropy and $\mathbb{E}$ is the expected value of the reward that is gained from following the policy π_θ in selecting action a_t in the state s_t. Having high entropy is essential in our case. It encourages the agent to explore more action options and can potentially find many optimal solutions that it can switch to with the least number of training iterations. In addition, SAC is an off-policy algorithm, which means that it can use previously collected data and learn from it, which helps in training faster. SAC also supports continuous action space, which is the case in our environment.

Table 3.1: List of used simulation parameters

Parameter	Value
RAN IDs	1,2,3,4,5,6 and 7
RANs service rates λ	2,4,6,8,10,12,14 Mbps
Bandwidth w_j	0.5 Mhz
Discount rate γ	0.90
Training episodes	2000
Replay buffer size $\mathcal{D}$	2*10^6
Batch size	256
Learning rate α	0.0003
Tau τ	0.005
Cumulative Arrival rate μ	300 KB
Queue size	50 MB
Min and Max comp. $\mathbb{C}\_min$, $\mathbb{C}\_max$	20, 65
Min and max delay ϱ_min, ϱ_max	0.05, 0.15 sec
Max Battery capacity	0.9 Joule
Max transmitted data	800 Mb

Performance Evaluation & discussion

In this part, we explain our environment setup along with the performance experiments. We compare our RL-based solution with two different baselines, the first is the energy-greedy and the second is the optimal algorithm. While the first considers the energy without focusing on the patient's urgency state, the second cares about all of the objectives considered in the problem **P**, in experimentation, the latter's performance showed very similar results to the state of the art [76].

Environment setup As previously mentioned and illustrated in Figure 3.1, we simulate the environment of a remote monitoring system, where the patient's sensors collect some data such as ECG. An IoMT device is then responsible for choosing for compressing and transferring that data through different RANs to a specific target (i.e., hospital or nearby device). For simplicity, all environment configuration is presented in Table 3.1. To study and analyze the performance of our RL-based solution, we have conducted different experiments that we describe in what follows.

In the first experiment, we compare three different ways of solving problem **P**, which are energy-greedy, state-of-the-art algorithm[76] and our system RLENS. The Energy-greedy algorithm cares the most for the energy consumed rather than the delay and distortion. In our case, we considered our RL-based solution with more weights on the energy as the greedy algorithm. On the contrary, the state-of-the-art algorithm[76] focuses more on the delay and the distortion without taking into consideration the energy consumption. Finally, our primary solution, RLENS, considers all mentioned parameters.

The first result can be seen in Figure 3.2, and it shows the superiority of the energy-

greedy policy for having the least battery consumption among all with a 40% remaining battery after transferring all data. The energy-greedy agent always tries to compress data with the highest compression level and transfer data frequently with a low-bandwidth RAN. Interestingly, this strategy leads to the lowest delay (as seen in Figure 3.3) and energy consumption while distorting data the most as in Figure 3.4. On the other hand, the state-of-the-art agent will always try to reduce the delay and distortion together. The only way to achieve both objectives is by having the least compression possible and choosing an energy-expensive RAN with a high rate that guarantees low overall delay. This behavior will consume very high energy, depleting the battery as illustrated in Figure 3.2.

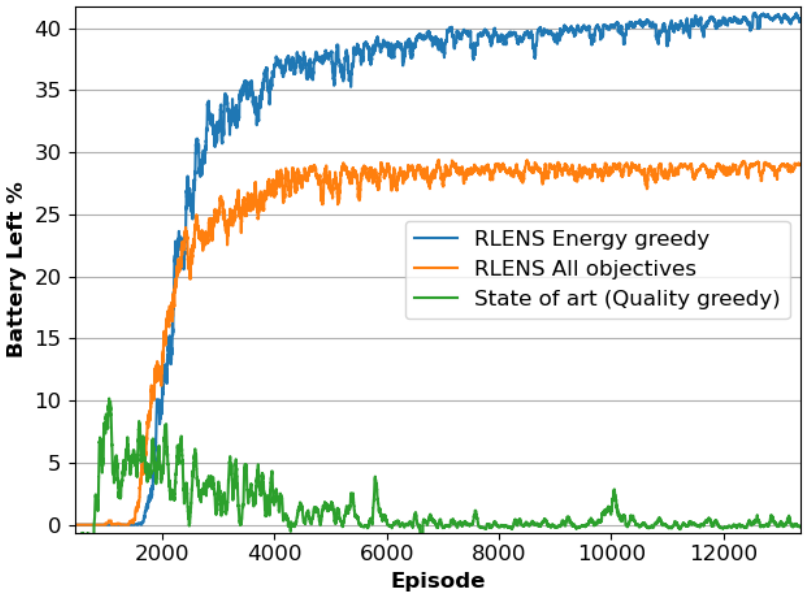


Figure 3.2: Shows different remaining battery levels from different algorithms

On the contrary to the past two methods, the RL-based solution achieves most of the objectives by being idle at specific points of time (e.g., when patient's condition is not urgent) and transferring the data with the adequate compression-ratio through the cheapest RAN that provides the highest bits/joule. This behavior reduces the transmis-

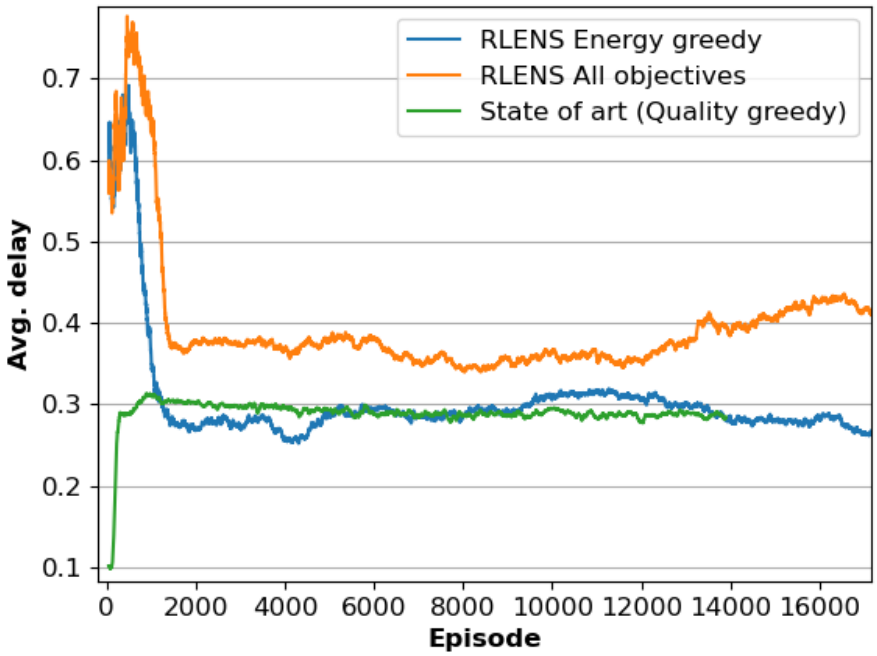


Figure 3.3: Highlights proposed solution (RLENS) has almost same similar average delay (sec) as state of the art algorithm

sion frequency, the unneeded data-quality conserving while increasing the overall delay with a tiny amount. This act that can be seen in Figure 3.2, Figure 3.3 and Figure 3.4 showed to be 85-90% close to the state of art behavior in terms of delay and distortion and 80% close to the energy saving in the energy-greedy algorithm.

In the second experiment illustrated in Figure 3.5, we show how long each agent survives in an environment of a tiny battery against our RL solution. We introduce one new baseline, namely a random agent, that randomly explores all options. Similar to the observation of the aforementioned experiment, the RL agent manages to survive the harsh environment with the help of its learned techniques with more than 20% battery left. In contrast, optimal and random agents fail after 25-60 steps. Giving RL a longer battery life by more than 500%

The third experiment, depicted in Figure 3.6, shows one advantage of using an RL-

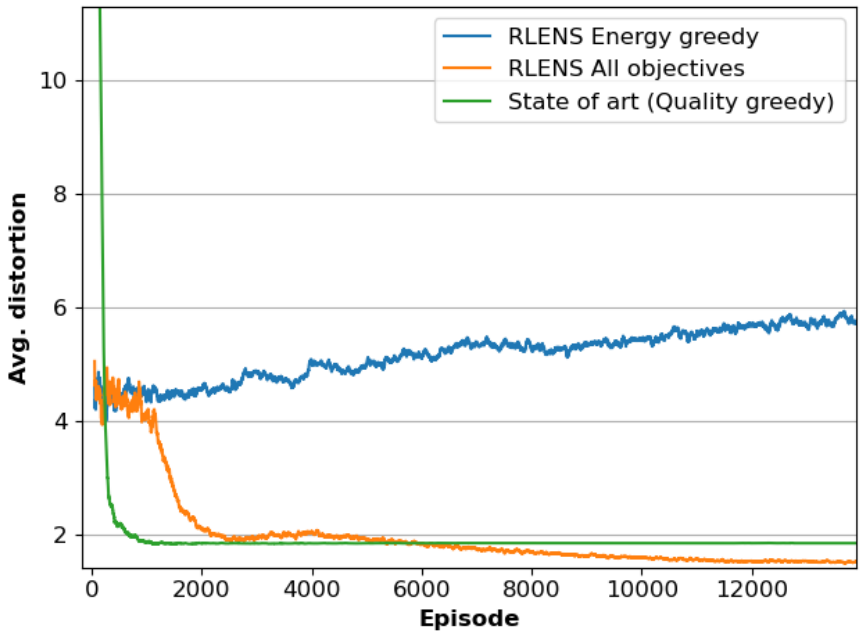


Figure 3.4: Avg.Distortion

based technique over other techniques, which is adaptability. For that experiment, we wait until our agent converges on using a specific RAN and then simulate a significant disturbance that could happen to that RAN. This disturbance will make the energy cost of using that RAN very high. In our case, we introduced a disturbance to RAN 3, which has triggered our agent to change its action distribution instantly to adapt to the current situation and successfully converges after around 1000 episodes. We argue that this convergence would have been much quicker if the agent had been trained for more episodes.

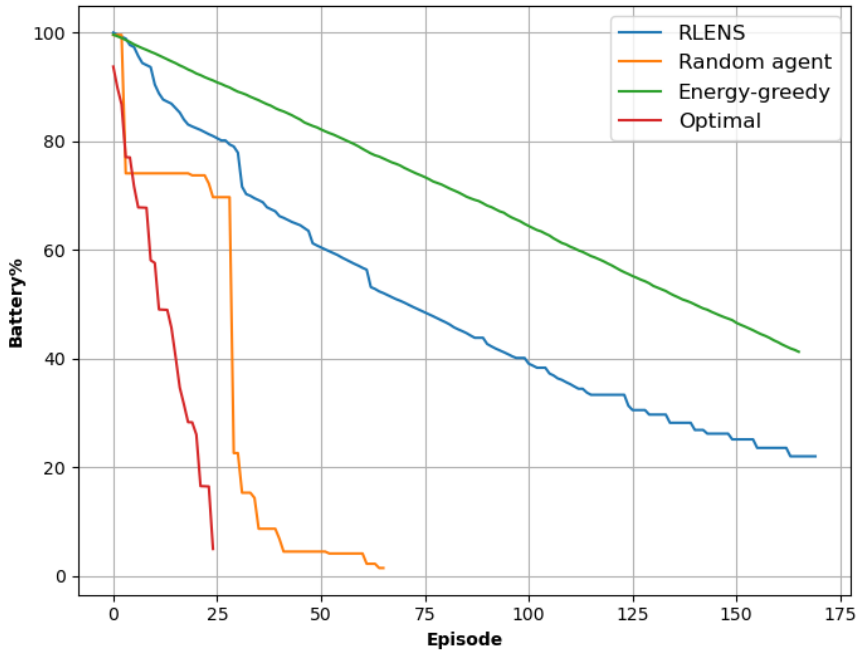


Figure 3.5: Performance comparison

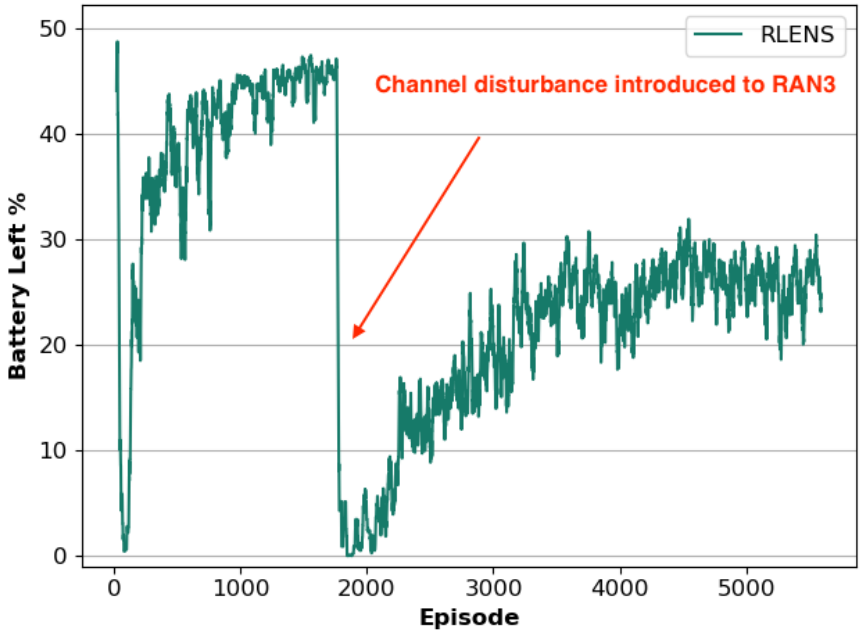


Figure 3.6: RL adaption to changes of channel

Conclusion

In conclusion, we presented RLENS, an RL-based technique for network selection. We then studied the case of remote monitoring and we modeled the system accordingly. Our algorithm's goal was to optimize the selection of compression level and transmission RAN while maximizing the battery life of IoMT, in addition to taking multiple aspects and constraints in the objective. We have demonstrated the performance of our solution using multiple experiments, which showed a 500% improvement over optimal algorithm, an adaptability to the environment and close performance to both energy and quality greedy by 85-90% and 80% respectively.

CHAPTER 4: DRL-BASED VIRTUAL RESOURCE ALLOCATION USING NFV

In the previous chapter, we presented how Reinforcement learning optimized the user-centric network selection scheme considering different dynamics from both the patient and the IoMT.

In this chapter, we consider the network point of view, where we extend the problem of resource allocation to virtual network resource allocation. We perform RL-based network slicing to optimize the cost of different services network slices by dynamically considering the choice of the network, data rate (bandwidth), VNF's computing resources, and paths. Our results show the advantage of using RL over optimal algorithm in getting the least cost slices that abide by all constraints.

In contrast to the VNF placement and network slicing methods described in the literature, our architecture supports various Intelligent-healthcare (I-health) services, with their own key performance indicators (KPIs), while also accounting for the varying resources and their sites, particularly the KPIs from the network's edge to the cloud. Thus, we observe that incorporating various I-health services in conjunction with all relevant KPIs necessitates developing a new problem formulation and creative solutions, which cannot be a simple extension of previous work. As such, the methodology provided in this study attempts to investigate unique methods that integrate artificial intelligence (AI) with distributed optimization techniques to enable intelligent network management while giving practical trade-offs between optimality and complexity. Our contributions particularly enhance the current state of the art in the following areas:

- We model the problem of network slicing considering different problem's dynamics (including KPIs, different routes, and different computing resources per intermediary nodes)
- We present an efficient, decentralized solution that leverages Reinforcement Learning to construct end-to-end network slices, allocating necessary resources while considering multiple KPIs per each service.
- We demonstrate how the network slicing problem is converted to MDP, in which we discuss states, actions, and rewards.
- We then introduce a convex optimization (CVX)-based optimal solution as a baseline. The solution tackles the complicated (NP-hard) network slicing problem

in stages.

- We conduct different experiments to demonstrate how the proposed solution performs against optimal algorithm and show how our solution provides lower complexity, better solution, and better adaptability to different deadlines.

- The suggested solution has been analyzed to demonstrate its efficacy compared to state-of-the-art alternative.

Network slicing architecture

To serve various services that demand different Quality of Service (QoS), the proposed architecture in Figure 4.1 uses ubiquitous network management intelligence at multiple network layers (from the cloud to the network edge devices). Indeed, it integrates SDN controller capabilities at various network tiers to provide a dynamic network slicing architecture that enables the efficient and cost-effective administration of multiple network slices with varying requirements. Overall, the centralized SDN controller, placed in the cloud, is primarily responsible for adapting/managing all network slices. In contrast, the local SDN (LSDN) controllers, located at the access points (i.e., macro base stations), are responsible for scheduling computing and commutation resources for the served users. Without sacrificing generality, this chapter will concentrate on wireless heterogeneous I-health systems (as seen in Figure 4.1). In which each patient (or end-user) may transfer medical data to the Health-cloud (H-cloud) over different Radio Access Networks (RANs). Each RAN will have unique energy consumption, financial cost, and transmission latency features, even though the various RANs will share the same control plane. The available RANs may change over time due to mobility and different traffic load.

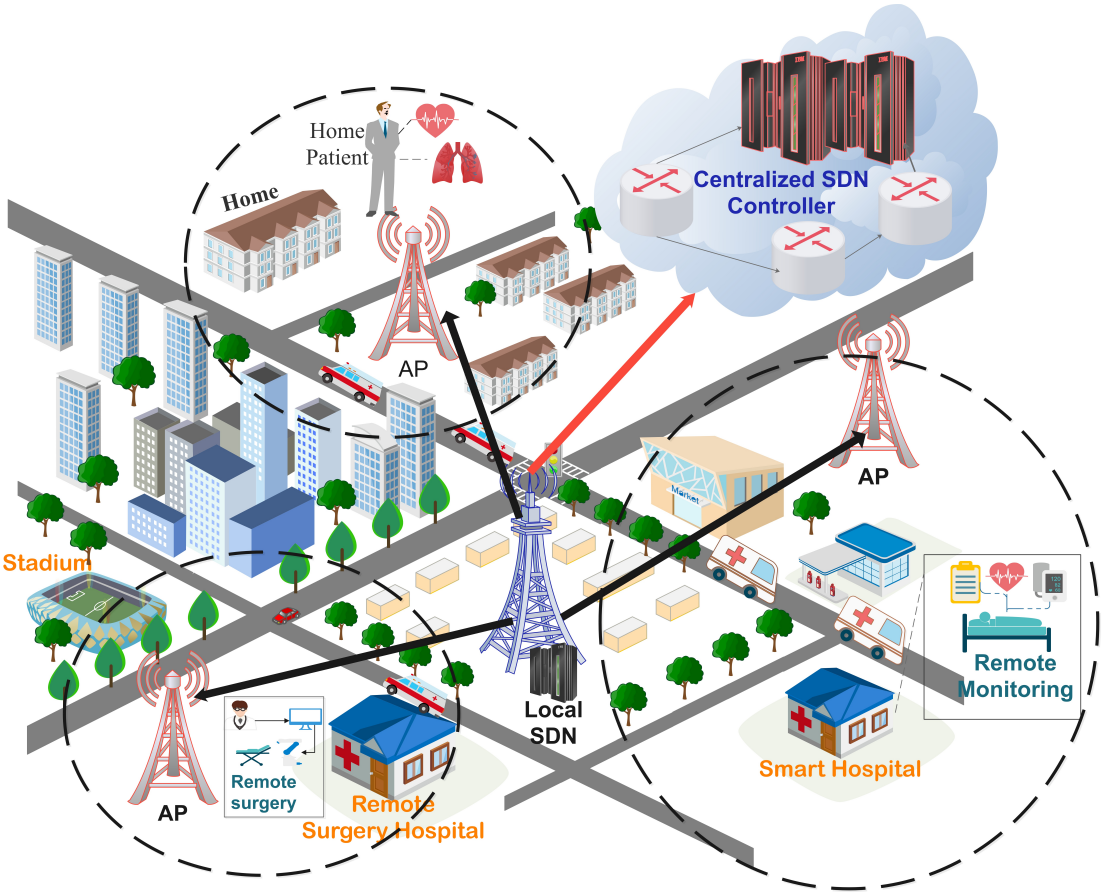


Figure 4.1: Considered system Model

System Model

As illustrated in Figure 4.2, various health care services (like remote monitoring, smart hospitals, and remote procedures) should be allocated a specific network slice to satisfy their strict QoS demands. Each service $s \in S$ consists of different VNFs $v \in V$ connected in a graph with their special order [82]. In this case, a single VNF runs on one or more nodes of different capabilities (i.e., different resource amounts). As seen in Figure 4.3(a), VNFs may represent a variety of functions such as data collecting, event detection, feature extraction, adaptive compression, and database-related functions [83]. Because VNFs may require different resources with big amounts, VNFs are allocated distributed resources from the physical network (i.e., computing nodes can provide computational resources and storage) they are running on. As mentioned earlier, a service is a group of VNFs connected in a specific order. According to the service requirement, different KPIs are considered. For example, in a remote surgery service where a patient is being treated remotely, the latency/delay KPI is more significant than the cost KPI. On the other hand, telemonitoring may care more about cost KPI than latency as the monitoring is done over long periods, thus low-cost communication. We also note that not all KPIs must be fulfilled per service [84]. Table 4.1 summarizes different KPIs used in the cloud. Moreover, after end-users process their data flow $f \in \mathcal{F}$ at the first VNF's node, the size of data flow should follow the flow conservation law as follow,

$$B(f, n, n+1) = B(f, n-1, n) \cdot \kappa(f, n), \quad \forall n \in \mathcal{N}, \forall f \in \mathcal{F} \tag{4.1}$$

While $B(f, n-1, n)$ represents the size of the in-going data flow f to node n (which belongs to VNF v), $\kappa(f, n)$ indicates the compression ratio that n applies to that flow, resulting in the new traffic size $B(f, n, n+1)$.

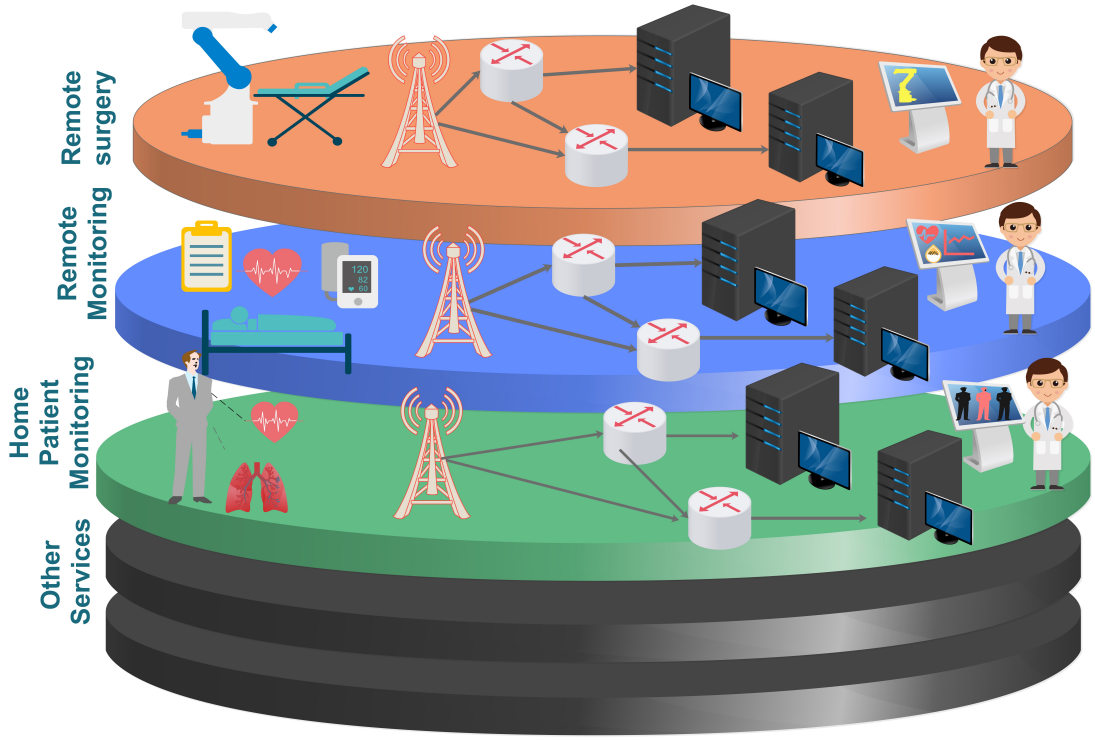
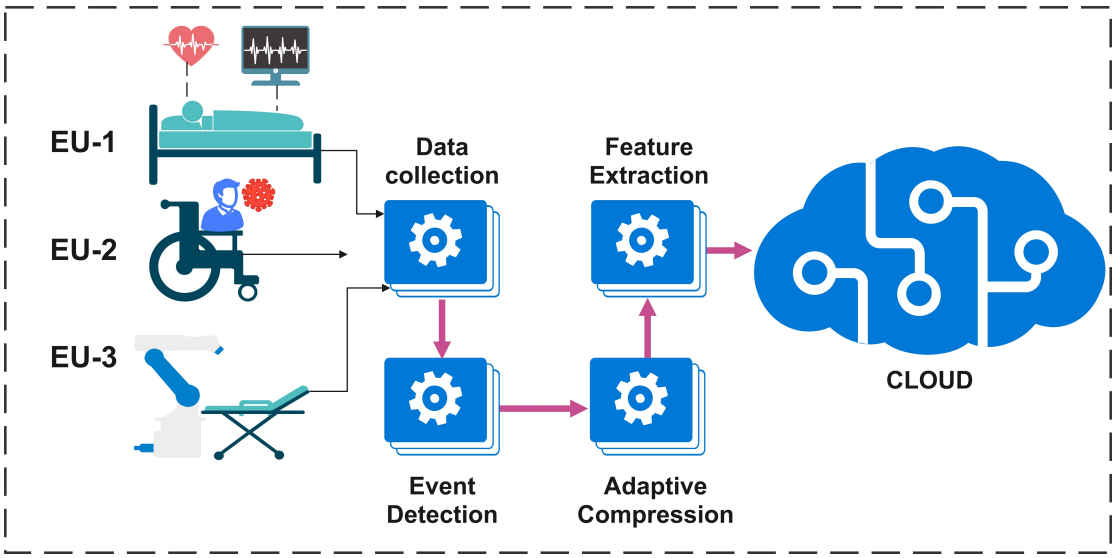
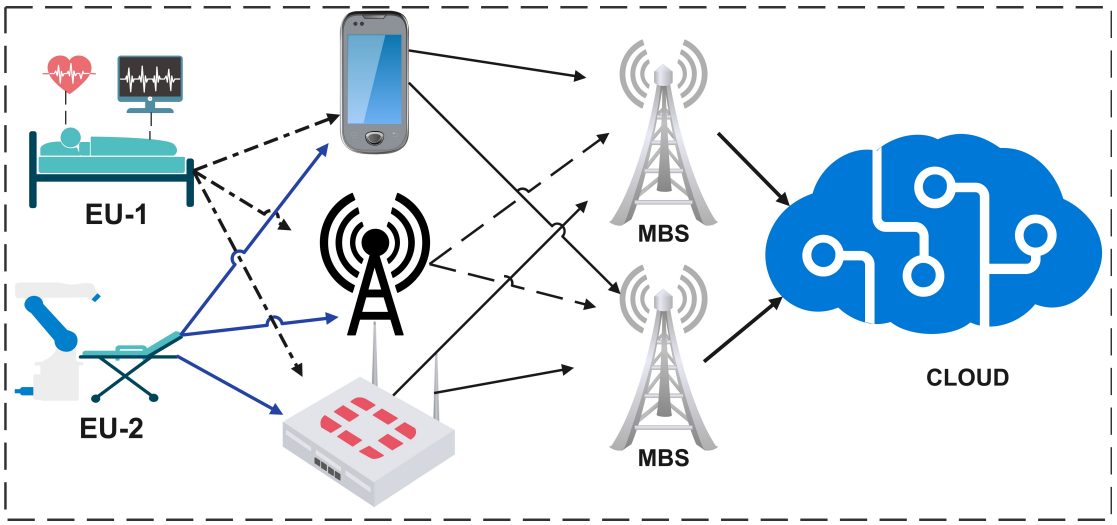


Figure 4.2: I-Health 5G Network slicing



(a) Service graph



(b) Physical graph

Figure 4.3: (a) The service graph for different medical applications, each block represents a VNF. (b) The Physical graph where VNF is running on. In this scenario, data are collected from different patients and are analyzed at the fog-level nodes then forwarded to the cloud with the help of macro base stations (MBS). The arrows represent bidirectional traffic

Table 4.1: Summary of different KPIs in the cloud

KPI	Definition
Traffic Load l	Load for processing and transmission
Delay	The highest permitted end-to-end delay
Reliability	Reliability of all different nodes on service graph
Cost	The anticipated expense of providing the service.
Availability	The availability of a service to customers
Security	The Security of the end-end path of service

Computing and network resources

Figure 4.3(b) depicts the system's physical graph, which illustrates all available processing and communication resources at various levels (i.e., fog, MEC, and cloud). The vertices of the graph represent the various network nodes, while the edges (i, j) reflect the links between them. Because network nodes may have varying levels of computational resources (for example, CPU and memory), the amount of resources of type k usable at node n is defined as $a_n(k)$. In order to maintain uniformity, service and physical flow must match. Each edge (i, j) corresponds to a particular link $l \in \mathcal{L}$, with a communication latency D_l and a capacity W_l. Additionally, in order to simulate a realistic status of links and network nodes, each node $n \in \mathcal{N}$ and link l have reliability parameters $\eta_n(t)$ and $\eta_l(t)$, which represent the node's or link's ability to function correctly at time t. Finally, each physical link l is limited to its capacity; therefore,

$$\sum_{f \in \mathcal{F}} r_{f,l} \leq W_l, \quad \forall l \in \mathcal{L}, \tag{4.2}$$

where $r_{f,l}$ denotes the data amount of flow f via link l.

key performance indicators

As presented earlier in Table 4.1, different KPIs can assist the requirement and performance of different services. Our system considers four different service's KPIs: end-end delay, nodes and links reliability, compression distortion, and energy consumed from transferring the data. Each of the mentioned KPIs will be discussed in detail in the following.

Delay In our system, the delay is composed of two major components, namely network delay and computation delay. While the network delay is caused by data transmission across multiple network links, the computation delay is generated by the computing nodes running different VNFs. Firstly, the network delay calculation, the average network delay is calculated by adding the delays associated with the various links that a flow f passes through on its path p which can be represented as follows

$$d_n(f,p) = \sum_{(i,j)\in p} D_{i,j} = \sum_{(i,j)\in p} \frac{B(f,i,j)}{r_{f,i,j}} + \epsilon_{i,j}, \tag{4.3}$$

Here, $\frac{B(f,i,j)}{r_{f,i,j}}$ represents the transmission time and $\epsilon_{i,j}$ depicts the channel access delay that may exist while on the data flows in link i,j.

Secondly, the processing delay, the VNF instances are modeled as M/M/1-PS queues per the processing paradigm in [82]. The processor sharing (PS) model was chosen to accurately simulate the behavior of a multi-threaded program operating on a virtual machine (without losing generality, additional processing models [85] may also be implemented in our framework). As a result, if the amount of traffic associated with flow f and processed at the instance of VNF v at node n is $\hat{B}(f,n)$, the total processing delay suffered by flow f through path p is stated as:

$$d_p(f,p) = \sum_{n\in p} \lambda(f,n) \frac{1}{a_n(k,cpu) - r_{cpu}(n)\hat{B}(f,n)}. \tag{4.4}$$

where $\lambda(f,n)$ is the percentage of the traffic flow $B(f,n)$ processed at the instance of VNF v hosted on node n, i.e., $\hat{B}(f,n) = \lambda(f,n) \cdot B(f,n)$. When compared to other types of resources, the allocated CPU plays a large role in (4.4). Thus, the allocated CPU

provides an extra degree of flexibility to the trade-off between cost and performance. More CPU results in a shorter processing time but higher expenses. On the other hand, adding more resources from other resource types (e.g., storage space), would not affect the other computing resources. Where D_f is the maximum desired delay for service s flow f, the delay constraint for path p is expressed as follows:

$$d_n(f,p) + d_p(f,p) \leq D_f. \tag{4.5}$$

Reliability Since a single path p consists of a group of nodes and links, therefore the reliability of p is calculated as the multiplication of each node and link reliability as follows:

$$\prod_{n \in p} \prod_{(i,j) \in p} \eta_n(t) \cdot \eta_{i,j}(t) \geq H_f, \tag{4.6}$$

where H_f is the maximum target reliability required for flow f.

Encoding distortion Given the volume of data generated by I-health systems, it is impractical to transmit all raw data from the EU to the cloud. Adaptive compression of gathered data at the network edge appears to be a possible solution to this problem. This reduces the transferred data size and hence the transmission energy usage and total delay. However, using lossy compression may cause encoding distortion. As a result, adaptive compression, energy consumption, delay, and distortion are generally trade-offs. High compression ratios result in lower energy consumption and lower delay to transfer the data but higher distortion. To quantitatively evaluate how compression can cause distortion, we look at the EEG signal encoding model described in [86]. The author used the Root-mean-square difference between recovered EEG data and the

original to measure distortion in this model. Moreover, the Real-time implementation in [86] provides the compression-distortion relations as follows

$$\psi(f,n) = \frac{x_1 e^{(1-\kappa(f,n))} + x_2 \cdot (1-\kappa(f,n))^{-x_3} + x_4}{100} \tag{4.7}$$

where the model parameters $x_1 \rightarrow x_4$ can be estimated using the statistics of the considered EEG encoder [86] and $\kappa(f,n)$ is the compression ratio done to f in node n.

Energy consumption To calculate the energy consumed by transferring data, we use the same energy model presented earlier in Section 3.

To transmit the data to any target, we assume the existence of ***M*** Radio access networks (RANS) which differs in communication range, transfer rates r_{ij}, energy costs E_{ij}.

For each RAN, the energy consumption E_{ij} depends on the channel gain G_j, transfer rate r_j, bandwidth w_j, bits to transfer l_i and noise spectral density N_0. In addition to two parameters δ and c which change according to the selected RAN [87] and can be achieved experimentally. The Energy gain can then be expressed by $G_j = \kappa \cdot \alpha \cdot |h_j|^2$. where K= -1.5/(log(5BER)), α is the path loss attenuation, and $|h_j|$ is the fading channel magnitude. Therefore, E_{ij} becomes :

$$E_{ij} = \delta \cdot (\frac{l_i \cdot N_0 \cdot w_j}{r_j \cdot G_j} \cdot (2^{\frac{r_j}{w_j}} - 1)) + c \tag{4.8}$$

Problem Formulation

In this section, we will describe the problem formulation for our system.

Our network slicing framework's ultimate purpose is to produce end-to-end cost-effective network slices that fulfill all of the KPIs required by various services. This is accomplished by the cloud SDN controller capabilities, which enable efficient and cost-effective administration of many network slices with varying requirements. The primary job of the SDN controller, which is often hosted in the cloud, is to adapt/manage all network slices. To serve the main goal, the SDN controller must perform two different tasks sequentially as follows :

- Collect the different required KPIs from multiple services
- Reserve virtualized network resources and functions (on physical links and nodes).

Thus, our objective function will consider three different aspects per each service slice: the service route, the nodes' capabilities running VNFs along the route, and the amount of reserved resources on each node in the route such that the total cost from reserving these resources is minimized. Estimating the network resource cost per network slice can be modeled as the summation between three different costs as follows:

$$c_{total} = c_n(v) + c_n(k) + c(i,j)$$

Where $c_n(v)$ is the cost for creating VNF instance v on a node n, $c_n(k)$ is the price per unit resource k at node n (cost at node) and $c(i,j)$ is the fee for transferring data unit per time unit on a link (i,j) (cost at network links).

For ease, the different aspects of the system model are presented in Table 4.2.

Table 4.2: Summary of different aspects of our network slicing problem

Aspect	Equation	Description
Delay	$d_n(f,p) + d_p(f,p)$	Summation of delay from: (i) processing at computing nodes. (ii) transferring data on different links per each route.
Reliability	$\prod_{n \in p} \prod_{(i,j) \in p} \eta_n(t) \cdot \eta_{i,j}(t)$	Multiplication of reliability from: (i) Each node. (ii) Each Link in the route
Cost	$c_{total} = c_n(v) + c_n(k) + c(i,j)$	Summation of total cost from: (i) installing VNF on node if not exist. (ii) reserving computational resources on processing nodes along the end-end route (e.g., $/cpu resource). (iii) transferring data traffic across different links in the end-end route (e.g., $/bit).

When a request to deploy a service instance s is received, the centralized-SDN controller begins solving the optimization problem, which is expressed as a cost-minimization problem as follows:

$$\textbf{P1:} \min_{S_{f,p}, a_n(f,k)} (U_1) \tag{4.9}$$

such that

$$S_{f,p}\left[d_n(f,p) + d_p(f,p)\right] \leq D_f,$$

$$\forall f \in \mathcal{F}, \forall p \in \mathcal{P} \tag{4.10}$$

$$\prod_{i,j} \eta_j(t)\eta_{i,j}(t) \geq S_{f,p} \cdot H_f, \forall f \in \mathcal{F}, \forall p \in \mathcal{P} \tag{4.11}$$

$$\sum_{f \in \mathcal{F}} d_{f,p,l} \cdot r_{f,l} \leq W_l, \ \ \forall l \in \mathcal{L}, \tag{4.12}$$

$$B(f,n,n+1) = B(f,n-1,n) \cdot \kappa(f,n),$$

$$\forall n \in \mathcal{N}, \forall f \in \mathcal{F}, \tag{4.13}$$

$$\sum_{l \in p} d_{f,p,l} = N_p \cdot S_{f,p}, \ \ \forall f \in \mathcal{F}, \forall p \in \mathcal{P}, \tag{4.14}$$

$$\sum_{p \in P} S_{f,p} = 1, \ \ \forall f \in \mathcal{F}, \tag{4.15}$$

$$d_{f,p,l} \in \{0,1\}, \ \ \forall f \in \mathcal{F}, \forall p \in \mathcal{P}, \forall l \in \mathcal{L}, \tag{4.16}$$

$$S_{f,p} \in \{0,1\}, \ \ \forall f \in \mathcal{F}, \forall p \in \mathcal{P}, \tag{4.17}$$

In (4.9), the objective function U_1 is defined as,

$$U_1 = \sum_f \sum_p S_{f,p} \cdot \Bigg[\sum_n \sum_v c_n(v) + \sum_n \sum_k c_n(k) a_n(f,k) + \sum_{(i,j)} c(i,j) B(f,i,j) \Bigg].$$

The objective of our optimization problem **p1** is to allocate the necessary resources per service flow and traffic route in such a way that the overall cost is reduced while satisfying different KPIs for all different services.

We establish a path indicator $S_{f,p}$ in (4.9) to indicate the selection decision of route p by flow f, where$S_{f,p} = 1$ when path p is picked and zero otherwise. Therefore, the problem's unknowns are $S_{f,p}$ and $a_n(f, k)$, implying that each service must calculate its transfer route and the quantity of resources to be allocated in all nodes in the network along that path.

Furthermore, as noted in Section 4.3, all flows must meet the latency, reliability, link capacity requirements i.e., (4.10)-(4.13). Additionally, constraint (4.14) assures that when flow f selects path p, all links along this path are reserved for this flow, where $d_{f,p,l}$ is the link selection indicator of link l over path p, and N_p is the number of hops in route p. The constraint (4.15) guarantees that each flow chooses just one route.

Proposed solutions

In this section, we will show two different solutions to our problem **P1** which we have defined earlier. The first solution involves an optimization-based approach, which we name (OISA). We use OISA as a benchmark for the second solution. The second solution is our main contribution, the DRL-based solution, named (RLISA). We then compare both solutions and show the results.

Optimization solution (OISA)

The challenge of simultaneously allocating resources (or VNF placement) and routing traffic is often difficult to solve; an even simplified form of this problem (with just one KPI) has been demonstrated to be an NP-hard problem [88]. The defined inter-slice allocation issue in **P1** may also be viewed as a more sophisticated variant of a multi-constrained path problem, in which the costs associated with distinct links change at each hop [89]. As a result, solving **P1** directly is impractical, urging us to propose an efficient approach known as Optimized Inter-Slice Allocation (OISA). Using OISA, for each flow, we get the optimized decision by solving the problem in three sequential steps as follows:

1. Finding all candidate paths that abide by the delay and network-capacity constraints.

2. For each candidate path, optimize resource selection and cost

3. Select the path with the minimum cost.

This sequential solving of the problem reduces the search space for the second step (i.e., optimizing resources per path). The details of the OISA algorithm are shown in Algorithm 1

Thus OISA reformulates **P1** to be as follows:

$$\textbf{P1-S:} \min_{a_n(f,k)} \left(\tilde{U}_1\right) \tag{4.18}$$

$$\text{subject to (4.10), (4.13)},$$

where $\tilde{U}_1$ is the objective function in (4.9) for a specific flow f and path p.

We should highlight that if two or more flows share the same link, the capacity of the link (bandwidth) can be divided among these flows using any proportional fair method [90]. Furthermore, the OISA's complexity is determined by the number of flows F, the number of potential paths P for each flow, and the complexity of solving **P1-S**, i.e., C_{P1S}. As a result, OISA's worst-case complexity is $P^F \cdot C_{P1S}$.

Algorithm 1 Optimized Inter-Slice Allocation (OISA) Algorithm

Input: $c_n(v)$, $c_n(k)$, $c(i,j)$, $r_{f,l}$, W_l, D_f, and H_f
for f in F **do**
    store all possible paths in set $\mathcal{A}$
end for
for possible paths tuples in $\mathcal{A}$ **do**
    Sample n possible paths taking into consideration the total delay and network-capacity constraint
    Solve the optimization for the selected paths collectively
    Store paths and resources chosen by all F flows and their total cost in set $\mathcal{C}$
end for
From $\mathcal{C}$ find the minimum total cost, the corresponding resources, and paths for the different flows.
For all flows, update selected path $S_{f,p^*} = 1$, and allocate the resources $a_n(f,k)$
Output: $\{S_{f,p^*}, a_n(f,k)\}$

Table 4.3: Simulation parameters.

Parameter	Value
D_1 and D_2	5 ms and 8 ms
H_1 and H_2	0.97
α, β and γ	0.33
M	4
N_u	3
C_{th}	0.3 \$
κ_i^{max}	0.7
DRL parameters	as in [91]

DRL-based solution(RLISA)

In this approach, to develop a scalable, low complexity, and adaptive solution, we opt to leverage the power of DRL. We argue that developing a DRL-based solution can cope with a highly complex and dynamic environment while efficiently solving the inter-slice allocation problem.

DRL has been widely employed in the literature [73], [80] due to its independence of training datasets and its capability to optimize complicated environments under dynamic changes. Rather than learning from datasets, the RL agent learns by interacting with an environment in which it would be rewarded or penalized based on how well it performed. To simplify the learning process, RL adopts Markov Decision Processes (MDPs), where ($\mathcal{S}$,$\mathcal{A}$,$\mathcal{T}$,$\mathcal{R}$,γ) are the five elements that make up an MDP.

The DRL learning cycle starts with an agent that observes the system state $\mathcal{S}$ and takes an action $a \in \mathcal{A}$ to get a reward $r \in \mathcal{R}$, which is discounted by a factor γ, i.e., $\mathcal{R} = \gamma r_t$, plus the new state $s' \in \mathcal{S}$. When the agent acts, the current state of the environment will transition to s' with a probability $\mathcal{T}$. After the RL agent has been trained using a learning algorithm, the agent builds a policy π that makes it able to correctly map different states to the best actions that guarantee the highest cumulative reward. In what follows, we illustrate how to leverage DRL to solve the formulated inter-slice allocation problem in **P1**.

First, we convert our problem into an MDP, defining our environment, state space, actions, and reward.

State To efficiently capture the different states in our environment, we included links' reliability ($l_0, \cdots, l_6$) in addition to the number of data packets to be sent by each service

flow d_{count}. This gives a per timestep state tuple $s_t = (l_0, \cdots, l_6, d_{count})$. Hence, the RL agent can choose a path that guarantees the reliability threshold during the training phase. The above representation will also help the RL agent to adapt to different environmental conditions (when a link or node is down).

Action At each timestep, the RL agent chooses the best paths for each service flow (each path represents the set of links and nodes from the patient to the cloud) while reserving the needed communication and computational resources along each path. For instance, if we consider the network topology in Figure 4.4 with two service flows F_1 and F_2, the chosen paths will be p_{f1}, p_{f2}, respectively, while the reserved resources along each path (for both flows) will be $(res0_{f1}, res1_{f1}, res0_{f2}, res1_{f2})$. Thus, the action space will be $a_t = (p_{f1}, p_{f2}, res0_{f1}, res1_{f1}, res0_{f2}, res1_{f2})$.

$$a_t = (p_{f1}\{0, \cdots, 3\}, p_{f2}\{0, \cdots, 3\}, res0_{f1}\{1, \cdots, 1000\}, \\ res1_{f1}\{1, \cdots, 1000\}, res0_{f2}\{1, \cdots, 1000\}, res1_{f2}\{1, \cdots, 1000\}) \quad (4.19)$$

It is worth mentioning that every node has limited computational capability; hence the RL agent may take illegal actions in the exploration phase (e.g., reserve resources beyond the maximum resources available at a certain node). We have developed two rules to solve this issue:

- If a flow requests more resources than a specific node's maximum capacity, it will be assigned only the maximum available resources at this node.
- Splitting the bandwidth/computational resources equally between the flows sharing a particular link/node if they request more resources than the maximum capacity.

Reward function After defining the desired states representation and actions, we formulate our reward function. This function defines our primary goal of minimizing end-to-end costs while respecting delays and reliability constraints of diverse flows. Firstly, to maintain the required delay constraint for each service flow, the difference between the current and the required delay is used to penalize the agent if the delay constraint (i.e., $c1$) is not satisfied. The same applies to the reliability constraint $c2$.

When the two constraints $c1$ and $c2$ are satisfied, the cost of the resources RL chooses to reserve is calculated. To ensure the RL agent continuously chooses the combination of the resources with the lowest financial cost, we model the rewards based on how close the current cost is from the lowest and highest resources' cost explored by the agent.

To be able to do that, we take two different ranges, $\mathbb{C}[\mathbb{C}_{min}, \mathbb{C}_{max}]$ and $(1, \cdots, 0)$ and map the first range to the second using linear interpolation.

Meaning that the resource combination with the least cost will always have the highest reward as the RL agent keeps exploring.

This will urge the agent to find the actions with the least cost, i.e., near to $\mathbb{C}_{min}$. Therefore, the reward function r_t is defined as:

$$r_t = \begin{cases} (1, \cdots, 0) & \mapsto \mathbb{C}[\mathbb{C}_{min}, \mathbb{C}_{max}],\ if\ \neg(c1, c2) \\ -\Delta(\mathbb{D} - \mathbb{D}_{req}) & if\ \ \mathbb{D} > \mathbb{D}_{req}\ \ (c1) \\ -\Delta(\mathbb{R} - \mathbb{R}_{req}) & if\ \ \mathbb{R} < \mathbb{R}_{req}\ \ (c2) \end{cases} \tag{4.20}$$

To make sure that the agent is always guided towards the least cost, $\mathbb{C}_{min}$ is continually updated by the lowest cost explored by the agent; hence the reward is updated accordingly. This guidance will encourage the agent to reach better results without diverging from its target.

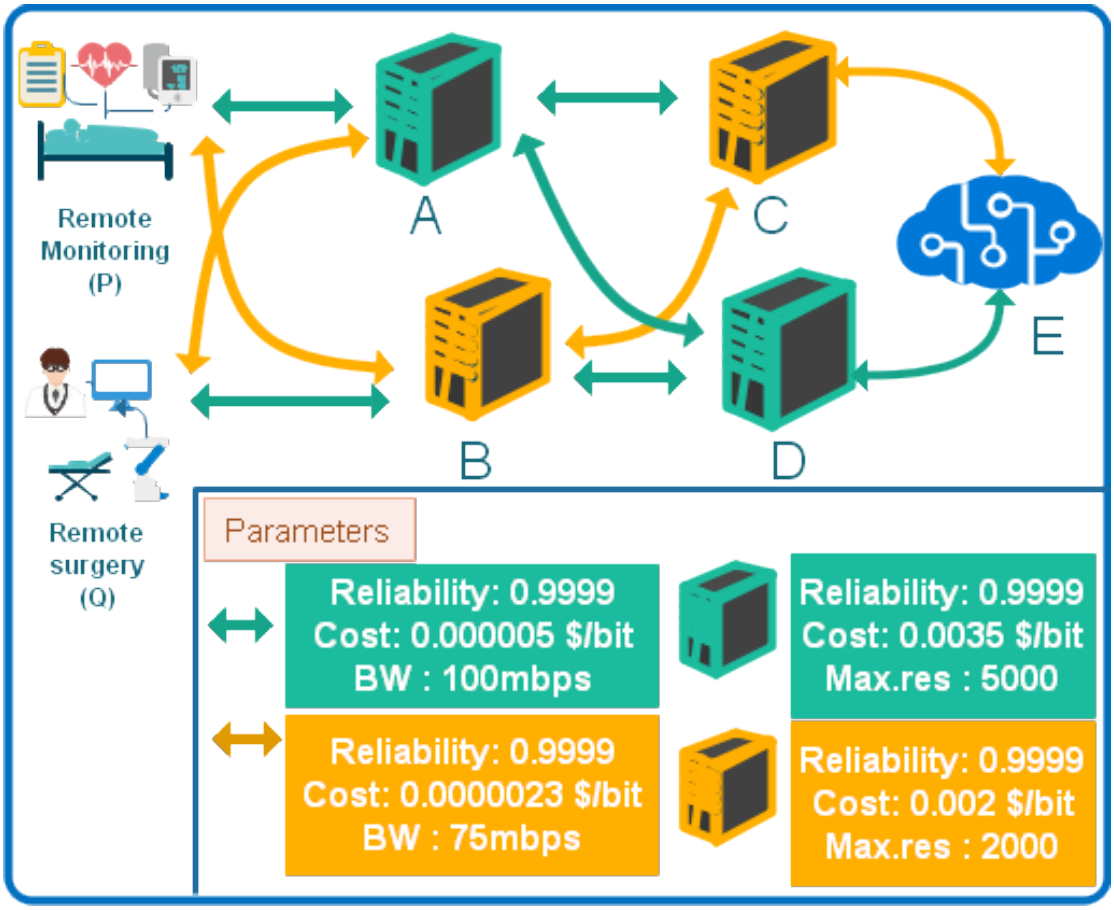


Figure 4.4: An example of remote monitoring and remote surgery services sharing the same physical graph.

Second, we opt to leverage an efficient DRL algorithm, namely Proximal Policy Optimization (PPO) algorithm [92], to solve our inter-slice allocation problem. PPO is a policy gradient algorithm that was found by the OpenAI group in 2017 [92]. The next paragraph describes how PPO works and the reason behind using it.

In simple terms, the PPO algorithm utilizes two local neural networks, namely actor and critic. While the former is used to do different actions, the latter is used to define the quality of the actions taken by the actor. The actor then uses the critic feedback to continuously update its policy using stochastic gradient descent (SGD).

PPO has been widely used in the literature because of its inclusivity of having multiple parallel training agents (brought from the A2C algorithm) and having a Trust

Region for updating the actor policy (which was brought from the TRPO algorithm). The main notion of PPO is that, whenever the actor updates its policy, the new policy is clipped to not deviate too much from the old policy. We highlight that having trust-region and tiny updates are extremely helpful to solve our problem, due to the fine-tuning nature of this algorithm in searching for the optimal paths (e.g., p_{f1}, p_{f2}) and resources (e.g., $(res0_{f1}, res1_{f1}, res0_{f2}, res1_{f2})$) that guarantee the lowest cost (as will be shown in Section 4.5). Moreover, PPO supports high-dimensional observations and multi-discrete actions, which make it the best choice to solve our problem while supporting the scalability needed. In addition to that, PPO uses parallel agents for the training, which allows for reducing the training time while enabling the agent to collect many training trajectories (i.e., $\mathcal{S}_1, \mathcal{A}_1, \mathcal{R}_2, \mathcal{S}_2, \mathcal{A}_2, \ldots, \mathcal{S}_T$) to learn from and reduce the variance. Finally, the proposed solution supports dynamical and real-time exploration and testing for the environment; thus, it updates the reserved paths and resources in real-time, which is crucial for highly dynamic networks and critical applications. In what follows, we present the details of the proposed RL-based Inter-Slice Allocation algorithm (RLISA).

As presented in Algorithm 2, RLISA firstly starts by sampling an action using the PPO algorithm from the action space (4.19). Secondly, forming an allocation map AM, where we store all links and nodes used by all services flows to make it easier to resolve the conflicts (resulting from sharing the same node, same link, or illegal actions) in the next step. After that, we resolve any conflicts and update the allocation map to be ready to apply the action, reach the new state, and get the reward. The resulted trajectory is then stored into the replay buffer, where the PPO's actor and critic networks will learn in a mini-batch fashion.

Algorithm 2 RL-based Inter-Slice Allocation (RLISA) Algorithm

Input: # of episodes K and maximum data to be sent d_{max}
Initialize PPO Neural network parameters θ_{π_0}, PPO value function network parameters and replay buffer R
for Each episode E **do**
    Reset all links and nodes reliabilities and data count d_{count}
    Release all nodes' allocated resources
    while $d_{count} < d_{max}$ **do**
        Using PPO policy, select a_t tuple $(p_{fn}, res0_{f1}, res1_{f1})$ per each flow
        Create an allocation map AM for all links and nodes
        if Multiple flows share same link ($\sum_{f \in \mathcal{F}} r_{f,l} > W_l$) **then**
            Divide link bandwidth equally among all flows & update AM
        end if
        if Multiple flows share same node and their total usage $> a_n(k)$ **then**
            Divide node resources proportionally & update AM
        end if
        Based on AM, apply action a_t from (4.19) and transfer data
        Calculate reward r_{t+1} as in (4.20)
        Decay $\{l_0, \cdots, l_6\}$ and increment d_{count}
        Save rollout $(s_t, a_t, r_{t+1}, s_{t+1})$ in R
    end while
    for m in M **do**
        Sample random mini-batches from R
        Calculate rewards-to-Go $\hat{R}_t$
        Estimate advantage estimate $\hat{A}_t^{\pi_k}$
        Update PPO networks
    end for
end for

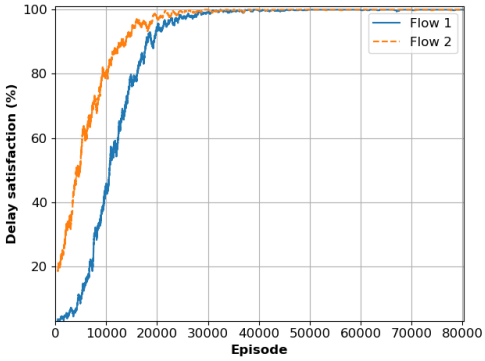
Performance evaluation

This section assesses our suggested solution in relation to OISA. First, we start by examining the convergence behavior of the RLISA solution proposed. Then, comparing it to the OISA technique while increasing the maximum delay deadlines for the various flows.

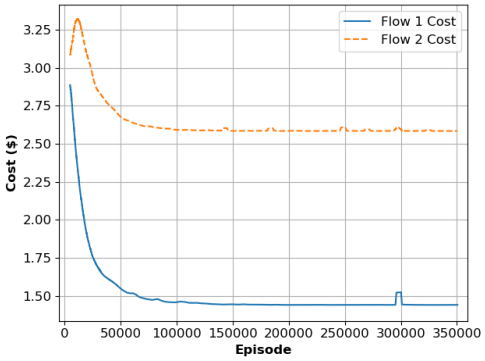
To test and validate the DRL solution, we created the simulation according to the scenario presented in Figure 4.4 with the RL parameters shown in Table 4.3. In that case, we set a delay deadline of 8 and 5 milliseconds for flow 1 and flow 2 respectively, and trained our RL agent. The results are illustrated in Figure 4.5. Figure 4.5(a) shows the reward convergence by the agent. On the other hand, Figure 4.5(b) and Figure 4.5(c) respectively show the least cost selected path and resource along with the delay constraint being fulfilled.

To further investigate the performance of DRL against the optimal solution and to show the effect of increasing the deadline on the cost presented by the convex optimization (CVX) solver, we have ran the same scenario in Figure 4.4 with an increased maximum delay deadline (i.e., (8,5), (9,6), (10,7) for flows 1 and 2 respectively). The results, illustrated in Figure 4.6, show how DRL outperforms the optimal solution in almost all cases. Although both optimal and DRL solutions chose the same paths for flows 1 and 2, the DRL solution chose more fine-tuned values for the resource reservation. The reason behind that was the fine-tuning nature of the PPO DRL algorithm, as it creates a trust-region and slowly moves in the direction of the highest reward. On the other hand, the optimal algorithm was more conservative as it cared mostly about meeting the constraint by reserving more resources than DRL's. This experiment showed us how

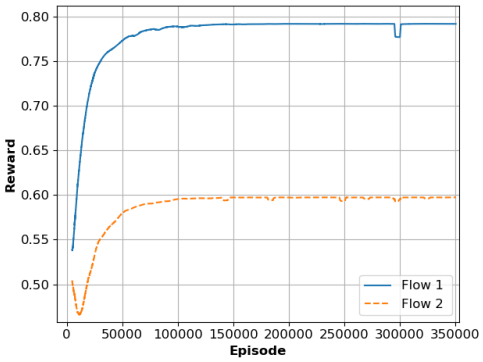
DRL could adapt to different delay deadlines and effectively obtain the lowest cost with less complexity and without rerunning as in OISA.



(a) Delay satisfaction



(b) Cost Convergence



(c) Reward convergence

Figure 4.5: Convergence behavior of the proposed RLISA solution with increasing the number of episodes, (a) cumulative reward, (b) cost, and (c) delay satisfaction percentage.

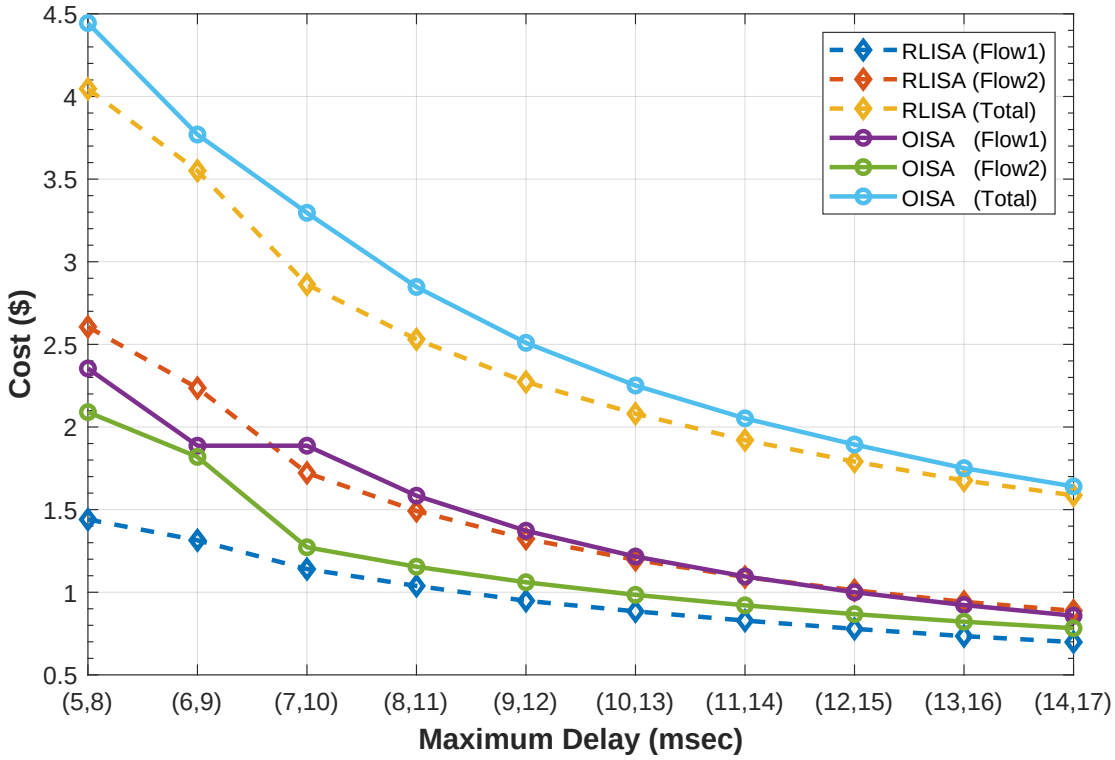


Figure 4.6: Cost variation of inter-slice allocation problem for OISA and RLISA, while increasing the maximum delay deadline of flows 1 and 2

Conclusion

In summary, we presented RLISA, an RL-based algorithm that optimizes virtual resource allocation through the dynamic choice of optimal service path and intermediate node resources for different service flows. We have compared our solution against an optimization-based algorithm. We examined the convergence of the RL agent and showed how our proposed solution outperforms the optimization solution with lower complexity. Moreover, our solution presented an adaptive behavior to different deadlines changes than optimization-based one.

CHAPTER 5: CONCLUSION

In this thesis work, we investigated the use of deep reinforcement learning (DRL) in improving physical and virtual network resource allocation for m-health systems. After introducing the main concepts related to health care and machine learning in Chapter 2. We presented our first contribution (RLENS) in Chapter 3, an RL-based user-centric technique for improving network selection for battery-operated IoMTs that considers the conditions of the patient, data aging, and the IoMT. We demonstrated the advantage of RLENS by studying the case of remote monitoring and modeling the problem accordingly. Particularly, our goal was to optimize the selection of compression level and transmission radio access network while maximizing the battery life of IoMT and considering multiple constraints. To verify our solution, we have created a set of experiments that demonstrated the advantage of RLENS against two baselines, in which RLENS showed a reduced distortion level, lower delay, and long battery life.

As Chapter 3 focused on optimizing the network selection from the user's IoT perspective (user side), Chapter 4 shifts the focus on optimizing the network side. Specifically, we optimize the allocation of different services' end-end network virtual resources through RL and network slicing. In this case, we optimize the end-end route and the resource reserved per intermediary nodes for different service flows. The main goal behind this usage is to lower the cost of different slices while abiding by different KPIs coming from different services. We then verify our RL-based solution (RLISA) against an optimization-powered algorithm (OISA) and perform different experiments. Our results show how RLISA outperforms OISA in complexity, least cost, and adaptability to changes.

CHAPTER 6: FUTURE WORK

For the first contribution, the following can be considered:

- Considering multi-modal vital signs, where IoT device can compress and transmit different signals e.g., ECG, EEG, each with its own custom compression ratio.
- Adding a mechanism for incentive renting of resource on the nearby devices and including it in the objective function.
- Formulating the problem as a cooperative multi-agent optimization, with different agents with a local view on the IoT side and others with a global view on the network side who reserve the bandwidth per local agents. This can provide more scalable solution while considering all different patient states
- The mechanism for detecting patient's urgency can be updated to use GANs or Restricted Boltzmann Machine (RBM) in which it can dynamically adapt to usual levels of sensors and only alert when the overall condition is quite different from the normal values.

For the second contribution, these points can be considered:

- In this problem, we considered that data flows in only one path to the destination per service, this can be upgraded to include multiple path split where data flow per service can be choose multiple routes to destination (i.e., load balancing). Moreover, investigating the optimal number of paths per service flow.
- In this work, we reserved network slices per service, however many users can join a single service, in which a second partitioning algorithm can be used to split data according to each user's state. In other words, a granular RAN slicing.

- More KPIs can also be a good addition to the problem formulation, for example the security of intermediary nodes, in which the SDN controller will need to perform an additional encryption step before handing over the data to them.
- Adaptive network slicing might be added, where a network slice changes according to the changing total amount of data produced by different patients.

REFERENCES

[1] L. Lorenzoni, A. Marino, D. Morgan, and C. James, "Health spending projections to 2030," no. 110, 2019. DOI: https://doi.org/https://doi.org/10.1787/5667f23d-en. [Online]. Available: https://www.oecd-ilibrary.org/content/paper/5667f23d-en.

[2] H. Yeganeh, "An analysis of emerging trends and transformations in global healthcare," *International Journal of Health Governance*, 2019.

[3] H. M. Zawbaa, H. Osama, A. El-Gendy, *et al.*, "Effect of mutation and vaccination on spread, severity, and mortality of covid-19 disease," *Journal of medical virology*, vol. 94, no. 1, pp. 197–204, 2022.

[4] T. Fisayo and S. Tsukagoshi, "Three waves of the COVID-19 pandemic.," *Postgraduate medical journal*, Aug. 2020, ISSN: 1469-0756. DOI: 10.1136/postgradmedj-2020-138564.

[5] J. D. Birkmeyer, A. Barnato, N. Birkmeyer, R. Bessler, and J. Skinner, "The Impact Of The COVID-19 Pandemic On Hospital Admissions In The United States," *Health Affairs*, vol. 39, no. 11, pp. 2010–2017, Nov. 2020, ISSN: 0278-2715. DOI: 10.1377/hlthaff.2020.00980.

[6] R. Barranco, L. Vallega Bernucci Du Tremoul, and F. Ventura, "Hospital-acquired sars-cov-2 infections in patients: Inevitable conditions or medical malpractice?" *International Journal of Environmental Research and Public Health*, vol. 18, no. 2, p. 489, 2021.

[7] B. M. Kuehn, "Hospital Readmission Is Common Among COVID-19 Survivors," *JAMA*, vol. 324, no. 24, p. 2477, Dec. 2020, ISSN: 15383598. DOI: 10.1001/jama.2020.23910.

[8] P. Sundaravadivel, E. Kougianos, S. P. Mohanty, and M. K. Ganapathiraju, "Everything you wanted to know about smart health care: Evaluating the different technologies and components of the internet of things for better health," *IEEE Consumer Electronics Magazine*, vol. 7, no. 1, pp. 18–28, 2017.

[9] J. M. C. Brito, "Trends in wireless communications towards 5g networks—the influence of e-health and iot applications," in *2016 International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, IEEE, 2016, pp. 1–7.

[10] Y. Lian, "Challenges in the design of self-powered wearable wireless sensors for healthcare Internet-of-Things," in *Proceedings - 2015 IEEE 11th International Conference on ASIC, ASICON 2015*, Institute of Electrical and Electronics Engineers Inc., Jul. 2016, ISBN: 9781479984831. DOI: 10.1109/ASICON.2015.7517022.

[11] S. Nosratian, M. Moradkhani, and M. B. Tavakoli, "Hybrid data compression using fuzzy logic and huffman coding in secure iot," *Iranian Journal of Fuzzy Systems*, vol. 18, no. 1, pp. 101–116, 2021, ISSN: 17350654. DOI: 10.22111/ijfs.2021.5875.

[12] M. Hooshmand, D. Zordan, D. Del Testa, E. Grisan, and M. Rossi, "Boosting the Battery Life of Wearables for Health Monitoring Through the Compression of Biosignals," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1647–1662, Oct. 2017, ISSN: 23274662. DOI: 10.1109/JIOT.2017.2689164.

[13] C. J. Deepu, C. H. Heng, and Y. Lian, "A Hybrid Data Compression Scheme for Power Reduction in Wireless Sensors for IoT," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 245–254, 2017, ISSN: 19324545. DOI: 10.1109/TBCAS.2016.2591923.

[14] M. Danieletto, N. Bui, and M. Zorzi, "RAZOR: A compression and classification solution for the internet of things," *Sensors (Switzerland)*, vol. 14, no. 1, pp. 68–94, Dec. 2013, ISSN: 14248220. DOI: 10.3390/s140100068.

[15] A. B. Said, M. F. Al-Sa'D, M. Tlili, *et al.*, "A Deep Learning Approach for Vital Signs Compression and Energy Efficient Delivery in mhealth Systems," *IEEE Access*, vol. 6, pp. 33 727–33 739, Jun. 2018, ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2844308.

[16] P. Bulić, G. Kojek, and A. Biasizzo, "Data Transmission Efficiency in Bluetooth Low Energy Versions," *Sensors*, vol. 19, no. 17, p. 3746, Aug. 2019, ISSN: 1424-8220. DOI: 10.3390/s19173746. [Online]. Available: https://www.mdpi.com/1424-8220/19/17/3746.

[17] A. Abedi, O. Abari, and T. Brecht, "Wi-LE: Can WiFi replace bluetooth?" In *HotNets 2019 - Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, New York, NY, USA: Association for Computing Machinery, Inc, Nov. 2019, pp. 117–124, ISBN: 9781450370202. DOI: 10.1145/3365609.3365853. [Online]. Available: https://dl.acm.org/doi/10.1145/3365609.3365853.

[18] H. A. H. Alobaidy, J. S. Mandeep, R. Nordin, and N. F. Abdullah, "A Review on ZigBee Based WSNs: Concepts, Infrastructure, Applications, and Challenges," *International Journal of Electrical and Electronic Engineering & Telecommuni-*

cations, pp. 189–198, 2020, ISSN: 23192518. DOI: 10.18178/ijeetc.9.3.189-198.

[19] G. A. Naidu and J. Kumar, "Wireless Protocols: Wi-Fi SON, Bluetooth, ZigBee, Z-Wave, and Wi-Fi," in *Lecture Notes in Networks and Systems*, vol. 65, Springer, 2019, pp. 229–239. DOI: 10.1007/978-981-13-3765-9{_}24. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-3765-9_24.

[20] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of Things (IoT) communication protocols: Review," in *ICIT 2017 - 8th International Conference on Information Technology, Proceedings*, Institute of Electrical and Electronics Engineers Inc., Oct. 2017, pp. 685–690, ISBN: 9781509063321. DOI: 10.1109/ICITECH.2017.8079928.

[21] A. Awad Abdellatif, A. Emam, C. F. Chiasserini, A. Mohamed, A. Jaoua, and R. Ward, "Edge-based compression and classification for smart healthcare systems: concept, implementation and evaluation," *Expert Systems with Applications*, vol. 117, pp. 1–14, Mar. 2019, ISSN: 09574174. DOI: 10.1016/j.eswa.2018.09.019.

[22] M. S. Allahham, A. A. Abdellatif, A. Mohamed, A. Erbad, E. Yaacoub, and M. Guizani, "I-SEE: Intelligent, Secure and Energy-Efficient Techniques for Medical Data Transmission Using Deep Reinforcement Learning," *IEEE Internet of Things Journal*, pp. 1–1, Sep. 2020, ISSN: 2327-4662. DOI: 10.1109/jiot.2020.3027048.

[23] A. Awad, A. Mohamed, and C.-F. Chiasserini, "Dynamic network selection in heterogeneous wireless networks: A user-centric scheme for improved delivery," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 53–60, 2016.

[24] T. T. Doan and C. L. Beck, "Distributed lagrangian methods for network resource allocation," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2017, pp. 650–655.

[25] R. Trestian, O. Ormond, and G.-M. Muntean, "Enhanced power-friendly access network selection strategy for multimedia delivery over heterogeneous wireless networks," *IEEE Transactions on Broadcasting*, vol. 60, no. 1, pp. 85–101, 2014.

[26] P. Naghavi, S. H. Rastegar, V. Shah-Mansouri, and H. Kebriaei, "Learning rat selection game in 5g heterogeneous networks," *IEEE Wireless Communications Letters*, vol. 5, no. 1, pp. 52–55, 2015.

[27] D. Niyato and E. Hossain, "Dynamics of network selection in heterogeneous wireless networks: An evolutionary game approach," *IEEE transactions on vehicular technology*, vol. 58, no. 4, 2008.

[28] M. El Helou, M. Ibrahim, S. Lahoud, K. Khawam, D. Mezher, and B. Cousin, "A network-assisted approach for rat selection in heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 6, pp. 1055–1067, 2015.

[29] C. Sun, E. Stevens-Navarro, and V. W. Wong, "A constrained mdp-based vertical handoff decision algorithm for 4g wireless networks," in *2008 IEEE International Conference on Communications*, IEEE, 2008, pp. 2169–2174.

[30] Q. Wu, Z. Du, P. Yang, Y.-D. Yao, and J. Wang, "Traffic-aware online network selection in heterogeneous wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 1, pp. 381–397, 2015.

[31] A. Awad, A. Mohamed, and C.-F. Chiasserini, "User-centric network selection in multi-rat systems," in *2016 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, 2016, pp. 97–102.

[32] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[33] M. Condoluci and T. Mahmoodi, "Softwarization and virtualization in 5g mobile networks: Benefits, trends and challenges," *Computer Networks*, vol. 146, pp. 65–84, 2018.

[34] F. Z. Yousaf, M. Gramaglia, V. Friderikos, *et al.*, "Network slicing with flexible mobility and qos/qoe support for 5g networks," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2017, pp. 1195–1201.

[35] G. Einziger, M. Goldstein, and Y. Sa'ar, "Faster placement of virtual machines through adaptive caching," in *IEEE INFOCOM*, 2019, pp. 2458–2466.

[36] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A geo-clustering approach," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 787–796, 2018.

[37] B. Ojaghi, F. Adelantado, A. Antonopoulos, and C. Verikoukis, "Slicedran: Service-aware network slicing framework for 5G radio access networks," *IEEE Systems Journal*, pp. 1–12, 2021. DOI: 10.1109/JSYST.2021.3064398.

[38] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "Vnf placement and resource allocation for the support of vertical services in 5g networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 433–446, 2019.

[39] J. Martin-Perez, F. Malandrino, C. F. Chiasserini, and C. J. Bernardos, "Okpi: All-kpi network slicing through efficient resource allocation," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul. 2020. DOI: 10.1109/infocom41043.2020.9155263. [Online]. Available: http://dx.doi.org/10.1109/INFOCOM41043.2020.9155263.

[40] K. P. Murphy *et al.*, "Naive bayes classifiers," *University of British Columbia*, vol. 18, no. 60, pp. 1–8, 2006.

[41] N. Cristianini and E. Ricci, "Support vector machines," in *Encyclopedia of Algorithms*, M.-Y. Kao, Ed. Boston, MA: Springer US, 2008, pp. 928–932, ISBN: 978-0-387-30162-4. DOI: 10.1007/978-0-387-30162-4_415. [Online]. Available: https://doi.org/10.1007/978-0-387-30162-4_415.

[42] T. Li, S. Zhu, and M. Ogihara, "Using discriminant analysis for multi-class classification: An experimental investigation," *Knowledge and information systems*, vol. 10, no. 4, pp. 453–472, 2006.

[43] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, no. 6, pp. 1–20, 2021.

[44] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV].

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[46] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[48] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: https://aclanthology.org/D14-1179.

[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[50] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*

(EMNLP), Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: https://aclanthology.org/D14-1179.

[51] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[52] D. Bank, N. Koenigstein, and R. Giryes, *Autoencoders*, 2021. arXiv: 2003.05991 [cs.LG].

[53] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07, Corvalis, Oregon, USA: Association for Computing Machinery, 2007, pp. 791–798, ISBN: 9781595937933. DOI: 10.1145/1273496.1273596. [Online]. Available: https://doi.org/10.1145/1273496.1273596.

[54] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990. DOI: 10.1109/5.58325.

[55] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[56] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *Robotica*, vol. 17, no. 2, pp. 229–235, 1999.

[57] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[58] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM Sigart Bulletin*, vol. 2, no. 4, pp. 160–163, 1991.

[59] S. Racanière, T. Weber, D. Reichert, *et al.*, "Imagination-augmented agents for deep reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[60] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," *arXiv preprint arXiv:1803.00101*, 2018.

[61] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[62] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[63] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[64] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2706–2716, 2013.

[65] A. A. Abdellatif, A. Mohamed, and C.-F. Chiasserini, "User-centric networks selection with adaptive data compression for smart health," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3618–3628, 2018.

[66] Y. Zhu, J. Li, Q. Huang, and D. Wu, "Game theoretic approach for network access control in heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9856–9866, 2018.

[67] J. Cui, D. Wu, and Z. Qin, "Caching ap selection and channel allocation in wireless caching networks: A binary concurrent interference minimizing game solution," *IEEE Access*, vol. 6, pp. 54 516–54 526, 2018.

[68] A. Awad, A. Mohamed, C.-F. Chiasserini, and T. Elfouly, "Network association with dynamic pricing over d2d-enabled heterogeneous networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2017, pp. 1–6.

[69] M. El Helou, M. Ibrahim, S. Lahoud, K. Khawam, D. Mezher, and B. Cousin, "A network-assisted approach for rat selection in heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 6, pp. 1055–1067, 2015.

[70] J. G. Andrews, S. Singh, Q. Ye, X. Lin, and H. S. Dhillon, "An overview of load balancing in hetnets: Old myths and open problems," *IEEE Wireless Communications*, vol. 21, no. 2, pp. 18–25, 2014.

[71] C. Desogus, M. Anedda, and M. Murroni, "Real-time load optimization for multimedia delivery content over heterogeneuos wireless network using a mew approach," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2017, pp. 1–4.

[72] I. Bisio, C. Braccini, S. Delucchi, F. Lavagetto, and M. Marchese, "Dynamic multi-attribute network selection algorithm for vertical handover procedures over mobile ad hoc networks," in *IEEE International Conference on Communications (ICC)*, 2014, pp. 342–347. DOI: 10.1109/ICC.2014.6883342.

[73] A. A. Abdellatif, N. Mhaisen, Z. Chkirbene, A. Mohamed, A. Erbad, and M. Guizani, "Reinforcement learning for intelligent healthcare systems: A comprehensive survey," *arXiv preprint arXiv:2108.04087*, 2021.

[74] F. Slimeni, Z. Chtourou, B. Scheers, V. Le Nir, and R. Attia, "Cooperative q-learning based channel selection for cognitive radio networks," *Wireless Networks*, vol. 25, no. 7, pp. 4161–4171, 2019.

[75] D. D. Nguyen, H. X. Nguyen, and L. B. White, "Reinforcement learning with network-assisted feedback for heterogeneous rat selection," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 6062–6076, 2017.

[76] Z. Chkirbene, A. Awad, A. Mohamed, A. Erbad, and M. Guizani, "Deep reinforcement learning for network selection over heterogeneous health systems," *IEEE Transactions on Network Science and Engineering*, 2021.

[77] A. Abo-eleneen and A. Mohamed, "Mmrl: A multi-modal reinforcement learning technique for energy-efficient medical iot systems," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, IEEE, 2021, pp. 2026–2031.

[78] A. Awad, M. Hamdy, A. Mohamed, and H. Alnuweiri, "Real-time implementation and evaluation of an adaptive energy-aware data compression for wireless eeg monitoring systems," in *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, IEEE, 2014, pp. 108–114.

[79] A. A. Abdellatif, L. Samara, A. Mohamed, *et al.*, "Medge-chain: Leveraging edge computing and blockchain for efficient medical data exchange," *IEEE Internet of Things Journal*, 2021.

[80] K. Gai and M. Qiu, "Optimal resource allocation using reinforcement learning for iot content-centric services," *Applied Soft Computing*, vol. 70, pp. 12–21, 2018, ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2018.03.056. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494618302540.

[81] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[82] J. Martín-Pérez, F. Malandrino, C.-F. Chiasserini, and C. J. Bernardos, "Okpi: All-kpi network slicing through efficient resource allocation," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, IEEE, 2020, pp. 804–813.

[83] K. Kamran, E. Yeh, and Q. Ma, "Deco: Joint computation, caching and forwarding in data-centric computing networks," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 111–120.

[84] T. Norp, "5G requirements and key performance indicators," *Journal of ICT Standardization*, pp. 15–30, 2018.

[85] J. Prados-Garzon, P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, P. Andres-Maldonado, and J. M. Lopez-Soler, "Performance modeling of softwarized network services based on queuing theory with experimental validation," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1558–1573, 2021. DOI: 10.1109/TMC.2019.2962488.

[86] A. Awad, M. Hamdy, A. Mohamed, and H. Alnuweiri, "Real-time implementation and evaluation of an adaptive energy-aware data compression for wireless EEG monitoring systems," *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE)*, pp. 108–114, Aug. 2014.

[87] K. Mahmud, M. Inoue, H. Murakami, M. Hasegawa, and H. Morikawa, "Measurement and usage of power consumption parameters of wireless interfaces in energy-aware multi-service mobile terminals," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 2, pp. 1090–1094, Nov. 2004.

[88] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molisch, "Approximation algorithms for the NFV service distribution problem," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9. DOI: 10.1109/INFOCOM.2017.8057039.

[89] G. Xue, A. Sen, W. Zhang, J. Tang, and K. Thulasiraman, "Finding a path subject to many additive qos constraints," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 201–211, 2007. DOI: 10.1109/TNET.2006.890089.

[90] Y. Bello, A. A. Abdellatif, M. S. Allahham, *et al.*, "B5g: Predictive container auto-scaling for cellular evolved packet core," *IEEE Access*, pp. 1–1, 2021. DOI: 10.1109/ACCESS.2021.3126048.

[91] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of*

Machine Learning Research, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html.

[92] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347 [cs.LG].