DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
University of Mohamed Khider - BISKRA
Faculty of Exact Sciences, Natural Sciences and Life
Computer Science Department

Order Number: RTIC /M2/2020

THESIS

PRESENTED TO OBTAIN THE ACADEMIC MASTER DIPLOMA IN

COMPUTER SCIENCE

OPTION: Network and Information and Communication Technology (RTIC)

QoS based Architectural Adaptation of IoT Systems

By:
Bounhas Mohamed Aymen

Supervisor:
Dr. Tarek Zernadji

UNIVERSITY OF MOHAMED KHIDER - BISKRA

Abstract

Faculty of Exact Sciences, Natural Sciences and Life

Computer Science Department

QoS based Architectural Adaptation of IoT Systems

by Bounhas Mohamed Aymen

Currently, Self-adaptation is considered to be the best solution to dynamically manage a system in the occurrence of deviations from the expected quality of service (QoS) parameters. However, systems such as IoT applications face different issues from the interference to the load traffic in network; that make QoS stability even harder. Usually adaptation techniques make use of reactive approaches that start when the system deviates from the expected QoS parameters. What we propose is a proactive approach to anticipate the changes before the event of a QoS deviation. More specifically, we propose proactive architectural adaptation of IoT system that uses machine learning for decision making and in aiding for the predictive process. The approach continuously monitors the QoS parameters and predicts based on historical data the possible deviations for the QoS parameters and choose the best adaptation action.

Keywords: Architecture-based adaptation , Self-adaptation Feedback loop , Internet of Things, Quality of Service.

Acknowledgements

First, I would like to thank **God** for granting me life and good health as well as supreme protection and guidance throughout my studies.

I am thankful to my supervisor" **Dr.Tarek Zernadji** " for his informative and useful guidance and suggestions throughout the journey for project design.

I would like to express my gratitude to the friends and colleagues who gave me their moral and intellectual support throughout my journey.

A big thanks to my parents for encouraging me and for supporting me morally every day in the construction of this memoir. Without them, I would not be here.

Contents

List of Figures

List of Tables

Introduction

Nowadays, the field of Internet of Things (IoT) is an emerging paradigm that aims at giving objects around us the ability to interact and cooperate with each other and users to accomplish various tasks. IoT is applied in various domains, e.g., home automation, health monitoring, smart manufacturing, precision agriculture, and area surveillance, etc.

In general, IoT systems have various Quality-of-Service (QoS) requirements, such as low energy consumption, low packet loss and latency, etc. The priority of these requirements depends on specifics of the domain. However, ensuring the Quality of Service (QoS) is hard as IoT applications are subject to a variety of uncertainties, such as sudden changes in traffic load, and communication interference. Traditionally, the network settings are required of human intervention to deal with uncertainties, resulting in continuous network maintenance or inefficiencies.

Human operators monitor a system and make adjustments when they detect problems, or, more generally, observe opportunities to improve the performance of the system. While humans are better at understanding the overall problem context than computers, human operators are disposed to long reaction time, fatigue, errors, and varying and potentially inconsistent expertise. Self-adaptation provides the means to automate tasks that humans would otherwise perform. It is designed independently of, and external to, the target system in order to realise the adaptation goals. To that end, a feedback loop deployed on top of the network to monitor and estimate the motes, and the environment to autonomously adapt the IoT system. However, it is mostly reactive in nature, where the adaptation is performed to react to an arising issue. Performing the different possible adaptation action at run time will be difficult on the highly dynamic IoT system.

Machine learning techniques can be considered towards aiding adaptation as it ensures that the system can learn from multiple data and improve over a period. In the case of proactive method, it needs a prediction method which leads that the different adaptation actions are needed to be defined before the deviation of the required goals.

Hence, our goal in this work is to propose a proactive approach where the architecture has the ability to learn, predict the QoS deviation (before they happen), generate the best adaptive plan and proactively adapt based on the data generated by the IoT system.

The memory is organized as follows. In the first chapter, we give basis notions in the field of Internet of Things and Self-Adaptation. In the second chapter, we present Machine learning and neural networks. In the third chapter, we give in details the used conception and the implementation of our approach for self-adaptive IoT system.

Part I

Internet of Things and Self Adaptation

Chapter 1

Internet of Things

1.1 Introduction

Every single day as times passes we are all experiencing new things. Most of all this new things come with the invention of the computer and the arrival of internet the more we rely on them the easier things get.

On the Internet of Things (IoT) model, many of the objects that surround us will be on the global computing network where everyone and everything will be linked to the Internet. IoT is continuously evolving and is a hot research topic where opportunities are infinite. Imaginations are unlimited which have put it on the verge of reshaping the current form of the internet into a modified and integrated version. The number of devices availing internet services is rising every day and having all of them connected by wire or wireless will put a powerful source of information at our fingertips. In a way of starting a new era, the term Internet of Things has spring into our lives and started to dominate the field pretty well.

In this chapter, we will present the main concepts of Internet of Things (IoT), including its application domain, challenges and the definition of Internet of things itself.

1.2 Definition

The term Internet of Things does not yet has an official, unified definition, which can be explained by the fact that the expression is still fresh and that the concept is still being constructed.

International telecommunication union (itu) for instance now defines the internet of things as a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies (itu 2012).

The Internet of Things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (uids) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction[1].

So the internet of things is a new technical term, symbolizing the new form in which technical devices have started to communicate via the internet. Thus, the term internet is a description of the state of communication between technical devices via the internet without human intervention; you now find washing machines capable of sending a notice, for example.

1.3 Application fields

Internet of Things has a wide range of applications in many fields.

1.3.1 Wearables

Wearable devices developed with the internet of things technology make people's lives easier. They are installed with sensors and software which collect data and information about the users. This data is preprocessed to extract essential insights about user.

1.3.2 Smart home

Commonly the term smart home is used to define a residence that has appliances, lighting, heating, air conditioning, TVs, computers, entertainment audio and video systems, security, and camera systems qualified for communicating with one another and can be controlled remotely by a time schedule, from any room in the residence, as well as remotely from any place in the world by phone or internet.

1.3.3 Health care

The usage of Io healthcare field is not just simply collecting data or viewing it locally, but the purpose and the effect is much more than originally thought of. This important role (IoT combined with big data) is represented in data gathering and analysis all at the real time.

1.3.4 Smart cities

Currently, some countries implement IoT in parking systems and monitoring, efficient lighting, and the garbage collection and disposal system. They have even implemented IoT in the elder care facilities.

The smart city concept is based on the automation of all the processes providing a city's livelihood with quick and effective solutions. Integrated real-time sensors collect data from citizens and devices, and all the information is analyzed to identify the problems, solve them, and predict alternative models for solving emergency situations.

1.3.5 Agriculture

The advancement of IoT technology in agriculture operations obtained by building a system for monitoring the crop field with the help of sensors (light, humidity, temperature, soil moisture, etc.) and use them in every step of the farming process like making the irrigation system automatic and how much time and resources a seed takes to become a fully grown vegetable.

1.3.6 Industrial automation

The Industrial Internet of Things (IoT) described the IoT as it is used across several industries such as manufacturing, transportation, energy/utilities, mining and metals, aviation and other industrial sectors. IoT allows for remote access and monitoring, but importantly, it allows for data acquisition and collection, exchange and analysis of a different data sources. This has enormous potential for improving productivity and efficiency.

1.3.7 Smart grid

The Smart Grid is an energy production, transmission and distribution network enhanced by digital control, monitoring and telecommunication capacities that not only allow the flow of energy, but also a huge amount of useful information. Internet of Things technology receives the quality and efficient energy it needs from smart grids and enables security in energy controllers and communication systems. It also plays an important role for the development of technology and economic innovations.

1.4 The IoT architecture

There is no single IoT architecture that is agreed universally. Recently, there are two IoT architectures are suggested 3–layer architecture and 5-layer architecture.

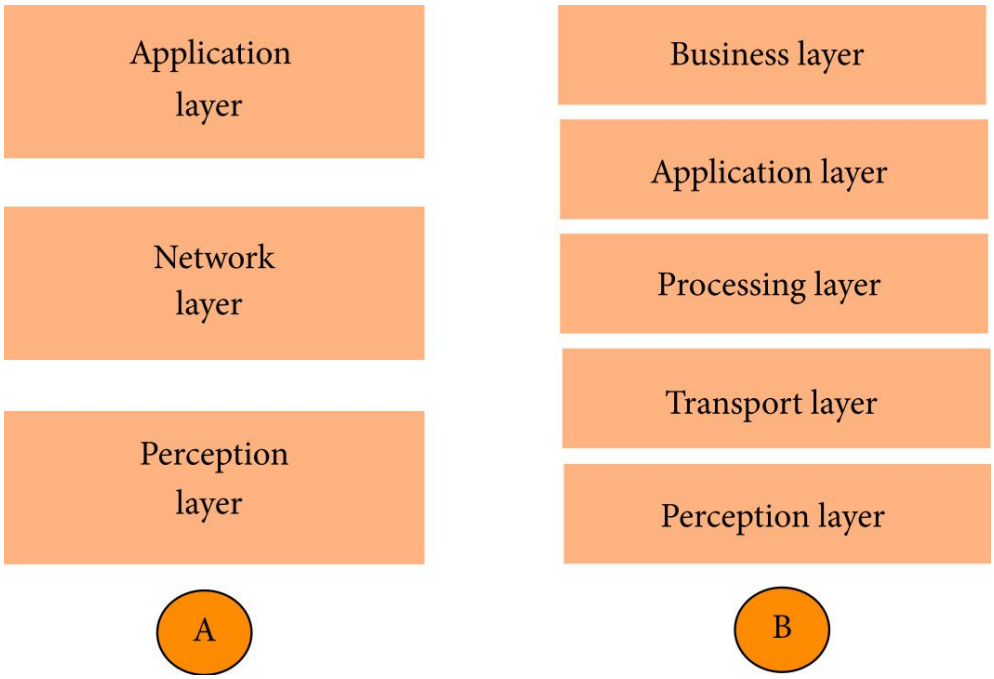


FIGURE 1.1: Architecture of IoT (A: three layers) (B: five layers). [2].

1.4.1 Three layers:

It is a very basic architecture and fulfills the basic idea of IoT. It was proposed in the early stages of development of IoT. It consists of three layers which are:

- **Perception Layer** , This is the physical layer or sensing layer, which includes sensors that are responsible for measuring the physical environment, identifying and locating things or devices, collecting data and sending it to the other layers for processing and storage.

- **Network Layer** , This layer acts like a bridge between perception layer and application layer. It carries and transmits the information collected from the physical objects through sensor also responsible for processing this information and connecting the smart things, network devices and networks to each other.

- **Application Layer** , It represents all applications that use the IoT technology or in which IoT has deployed also it has the responsibility to provide the services to the users based on the data collected from the sensors. The applications of IoT can be smart homes, smart cities, smart health, etc.

1.4.2 Five layers

The three-layer architecture defines the main idea of the Internet of Things. However, it is not sufficient for research on IoT as a result research often focuses on finer aspects of the Internet of Things. That is why there are more proposed architecture one of them is the five-layer architecture.

The role of the perception and application layers is the same as the architecture with three layers.

- **Transport Layer** , This is similar to the network layer of the three-layer architecture. It sends the data gathered in the perception layer to the processing layer and vice versa.

- **Processing Layer** , This is also known as the middleware layer. It collects, stores, analyzes, and processes the information that comes from the transport layer. It employs many technologies, such as databases and cloud computing.

- **Business Layer** , it acts like a manager of the whole IoT system. It has responsibilities to manage and control applications, business and user privacy.

1.5 IoT distribution patterns

IoT distribution patterns classify the architectures according to elements collaboration.

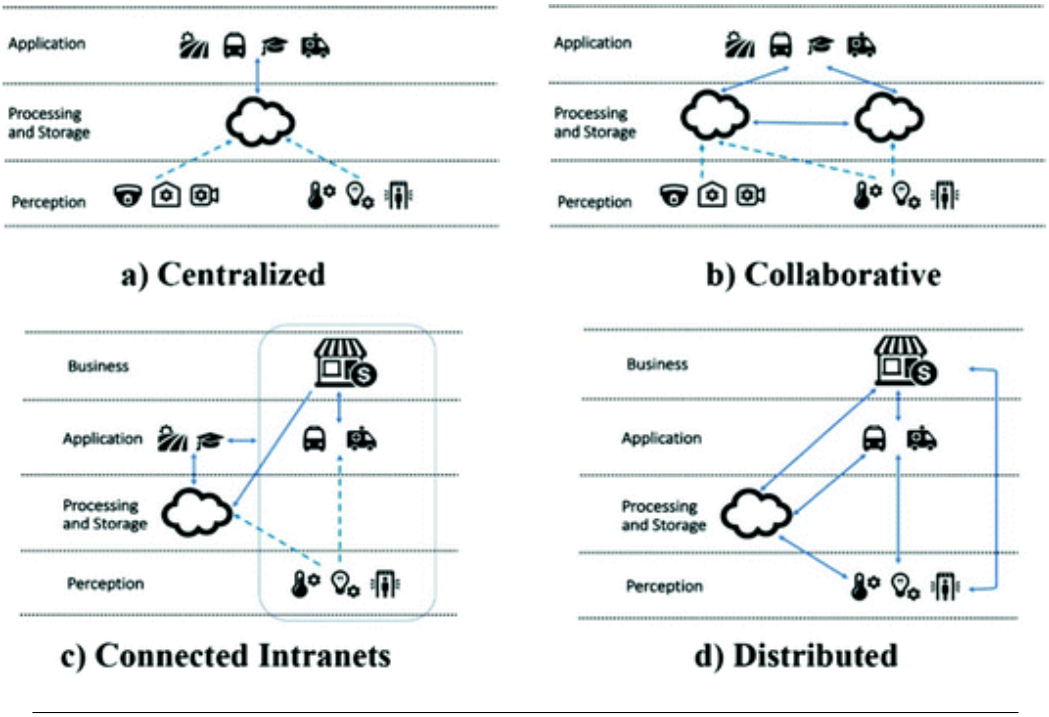


FIGURE 1.2: IoT architectural patterns.

- In **Centralized** distribution pattern, the perception layer provides data for the central processing and storage component to prepare for a service in the next layer. In order to use the IoT service, one must connect to this central component. The central component can be a server, cloud, or a fog network connected to cloud.

- In **Collaborative** pattern, a network of central intelligent components can communicate in order to form and empower their services.

- In **Connected Intranets** pattern, sensors provide data within a local intranet to be used locally, remotely, and centrally. Here the advantage is that if the central component fails, local service is still in access. The disadvantage is that there is no fully distributed framework to facilitate the communication among components.

- In a **Distributed** pattern, all components are fully interconnected and capable to retrieving, process, combine, and provide information and services to other components towards common goals [3] .

1.6 Challenges of Internet of Things

Realizing dependable Internet-of-Things (IoT) comes with numerous challenges. Most of these challenges are:

1.6.1 Interoperability

Each type of intelligent object on the Internet of Things has different information storage, processing and communication capabilities. Different intelligent objects would also be subject to different conditions such as energy availability and communications bandwidth requirements. To this end, in order to facilitate the communication and cooperation of these objects, common standards are required.

1.6.2 Security and Confidentiality

The security and protection aspects of the Internet, such as the confidentiality of communications, the integrity of the messages as well as the authenticity and reliability of communication partners must be ensured in an IoT environment as an example in some cases there is a need to access certain services to accomplish a task or to prevent them from communicating with other objects in the IoT for preventive security reasons.

1.6.3 Connectivity

With the raise of IoT application, obviously will lead to huge number of devices that get connected to the internet. The connectivity plays important role on it by transporting the data from the sensors and transmission of the instruction to the actuators. So, to have a reliable IoT network the connectivity must be ensured.

1.6.4 Autonomic control

Traditional computers demand users to configure and adapt them to diverse application domains and diverse communication environments. However, objects in IoT network

should create connections spontaneously, and organize/configure themselves to suit the platform they are operating in.

1.6.5 Quality of Service

Internet of Things (IoT) aims to allow the interconnection of many of smart devices (things) using a combination of networks and computing technologies. But the flow that comes from the interconnected things makes a greater demand on the underlying communication networks and affects the quality of service (QoS).

In general, IoT systems have various Quality-of-Service (QoS) requirements, such as low energy consumption, low packet loss and latency, etc.

Quality of Service "QoS" is typically defined as an ability of a network to provide the required services for selected network traffic. Also, QoS represents the set of techniques necessary to manage network like bandwidth, delay, jitter, and packet loss, etc. The most representative QoS properties in IoT are presented as follows.[4]

Latency (delay)

The delay indicate to the time required to transmit a packet or a group of packets from transmit end to receive end. It involves the transmission delay and processing delay.

Throughput

It is a measure of a number of data packets sent or received over the network. It can be defined as the actual bandwidth that is available to a network, measured in bits per second (bps). With the increase in network latency, the throughput of the network decreases.

Availability

The percentage of time the network is active and operational when required for use, is defined as availability. In communication, availability is affected by Internet connection.

Security and Privacy

Satisfying customers' need of security and privacy is not easy. Some technologies have been invented to achieve the required security and privacy level.

While network security is the action taken to protect the accessibility and integrity of the communication channel and the information flowing through it. A variety of encryption techniques are used to provide security to the network.

Reliability

The reliability in communication can be formulated in terms of the connection between the origin and destination node pairs. The communication medium is said to be reliable only if it provides assurance of delivering the information to the intended user without any loss or security breach.

Energy Efficiency

As we know that most things in IoT network are using sensors. They have limit resources like small batteries, so to maintain the lifetime of the network we need to minimize the energy consummation.

IoT Layer	Cross Layer QoS Parameters	QoS Parameters and QoS Metrics
		• Service time, services availability,
		• Service delay, Service accuracy, Service load, Service priority
Application		• Information accuracy
		• Cost of network deployment, Cost of service usage
Layer		
		• Maximum # of resources available per unit price & penalties for service degradation
	• IoT coverage	• Fault tolerance
		• Services perform cost, perform time, load, Reliability
	• Response time	
		• Bandwidth, Delay, Packet loss rate, Jitter
	• Energy	
Network	• Consumption/Efficiency	• Utilization of network resources
Layer		• Life time of sensing networks
		• Reliability, Throughput, Real-time
		• Sampling parameters
Perception		• Time synchronization
Layer		• Location/mobility
		• Sensing and actuation coverage

FIGURE 1.3: QoS parameters of three IoT Layers and cross layer parameters [5].

The Quality attributes that we will accomplish are reducing packet loss to maintain reliability and minimize the energy consumption of individual motes, and the whole network. Also, we will maintain connectivity while minimizing packet loss by adding and removing links.

1.6.5.1 Signal-to-Noise Ratio (S/N or SNR)

The SNR (Signal-To-Noise Ratio) is a measure of signal strength relative to background noise. In other words, SNR is the ratio of signal power to the noise power; it is usually measured in a decibel (dB). Also, a ratio greater than 0 dB signifies more signal than noise. The lower the SNR, the higher the interference, resulting in higher packet loss. The noise is simply interference on the same frequency created by electronic devices, but can also include external events that affect the measured phenomenon wind, vibrations, variations of temperature, variations of humidity, etc.

1.6.5.2 Distribution Factor

Represents the percentages of the messages sent by a source mote over a link to one of its parents. The total sum of the distribution factors for one mote is normally 100.

1.6.6 Self-Adaptation

In IoI one particular challenge is handling different operating conditions, such as interference in the wireless communication between devices and gateways or changing availability of services. Without proper adaptation to these conditions, dependability goals of IoT systems, such as reliability, energy efficiency and QoS may be affected.

1.7 Conclusion

In this chapter, we have presented a detailed study on the Internet of Things, its definition and fields of application then the IoT architecture, and its challenges. One of the major challenges of the IoT is Self-adaptation, one of the techniques to ensure these concepts is Architecture-based Adaptation which will be presented in the next chapter.

Chapter 2

Self-adaptation

2.1 Introduction

The aim of autonomic computing is to automate human tasks in system management to achieve high level objectives. At first, human operators monitor a system and make adjustments when they detect problems, or, more generally, observe opportunities to improve the performance of the system. While humans are better at understanding the overall problem context than computers, human operators are disposed to long reaction time, fatigue, errors, varying and potentially inconsistent expertise.

Modern software systems are exposed to various types of uncertainties, such as Internet-of-Things applications which operate under highly dynamic conditions where both the entities and their interconnections are subject to continuous change. Such dynamics are difficult to predict and goals may change during operations.

In recent years, the emergence of autonomic computing offers an alternative approach, where self-adaptation is designed independently of, and external to, the target system in order to automate tasks that humans would otherwise perform. Self-adaptation equips a software system with a feedback loop that collects additional knowledge at runtime, monitors the system and adapts it when necessary to maintain its quality goals, regardless of uncertainties.

2.2 Self Adaptation

The term self-adaptation is not precisely defined in the literature.

Cheng et al. Refer to a self-adaptive system as a system that "is able to adjust its behavior in response to their perception of the environment and the system itself"[6].

Brun et al. Add to that, "the self-prefix indicates that the system decides autonomously (i.e., without or with minimal interference) how to adapt or organise to accommodate changes in its context and environment" [7].

Andersson et al. Refer in this context to "disciplined split" as a basic principle of a self-adaptive system, referring to an explicit separation between a part of the system that deals with the domain concerns and a part that deals the adaptation concerns [8].

From these references we introduce two basic principles that complement one another and determine what a self-adaptive system is as follows:

- **External principle** , A self-adaptive system is a system that can handle changes and uncertainties in its environment, the system itself and its goals autonomously (i.e., without or with minimal human interference).
- **Internal principle** , A self-adaptive system comprises two distinct parts: the first part interacts with the environment and is responsible for the domain concerns (i.e., concerns for which the system is built); the second part interacts with the first part (and monitors its environment) and is responsible for the adaptation concerns (i.e., concerns about the domain concerns).

2.3 Overview Self Adaptive Systems

To allow managed systems to self-adapt with minimal human intervention this requires closing the loop of control. Traditionally Software systems have been designed as open-loop systems: once a system is designed for a certain function and deployed, its extra-functional quality attributes typically remain relatively unchanged. If something goes wrong, in most cases humans must intervene, usually by restarting the failed subsystem or taking the entire system offline for repair. These results in high costs in system downtime, personnel costs, and decreased revenue through system unavailability.

Some researchers have proposed a different approach to solve this problem that uses external software mechanisms to preserve a form of closed-loop control over the target system. Such mechanisms allow a system to self-adapt dynamically, with reduced human intervention.

Closed-loop control consists of mechanisms that monitor the system, reflect on observations for problems, and control the system to preserve it within acceptable bounds of behavior. This kind of system is known as a feedback control system in control theory.

The external controller requires an explicit model of the target system in order to reflect on observations and to configure and repair the system [9]. This External control separates the concerns of system functionality from those of adaptation. With the adaptation mechanism as a separate entity, engineers can more easily modify and control its adaptation logic.

Concept model

The concepts that correspond to the basic elements of a self-adaptive system are kept abstract and general, but they respond with the two basic principles of self-adaptation. The conceptual model involves four basic elements: environment, managed system, adaptation goals, and managing system.

- **Environment** , The environment refers to the part of the external world which the self-adaptive system interact with; it effects will be observed and evaluated.
- **Managed System** , The managed system comprises the application code that realizes the system's domain functionality. To realize its functionality, the managed system senses and effects the environment.
- **Managing System** , The managing system manages the managed system. To that, the managing system composes the adaptation logic that deals with one or more adaption goals. To realize the adaptation goals, the managing system monitors the environment and the managed system and adapts when there is a need.
- **Adaptation Goals** , The adaptation goals are concerns of the managing system over the managed system; they usually related to the software qualities of the managed system.

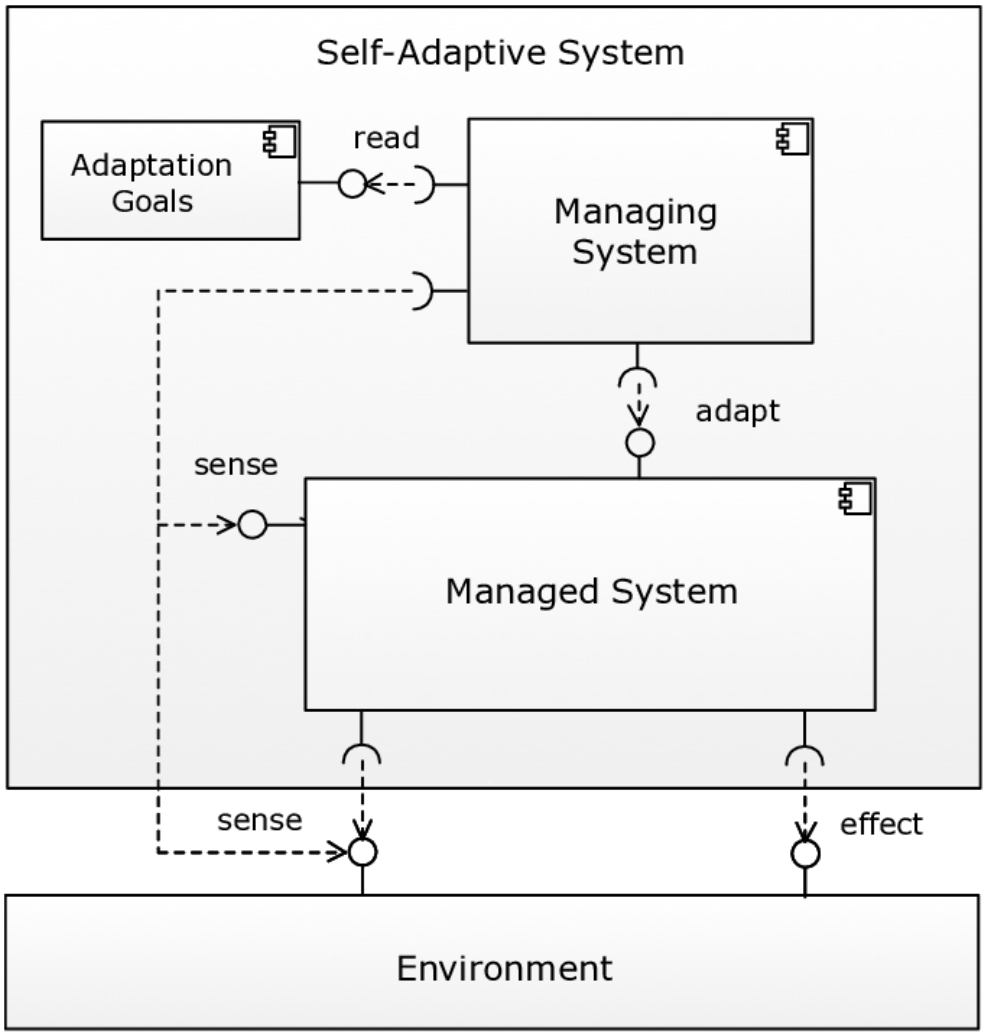


FIGURE 2.1: Conceptual model of a self-adaptive system. [10]

2.4 The IBM Autonomic Framework

As external, feedback control approach IBM introduced its model which is Monitor-Analyze-Plan-Execute- (MAPE) model. This four elements realise the basic functions of any self-adaptive system. These elements share common Knowledge, thus the model of an autonomic manager is usually referred to as the MAPE-K model.

The architecture describes two types of system components an autonomic manager and one or more managed element. An autonomic manager is a component that implements a particular control loop. A managed element is what the autonomic manager is controlling [11].

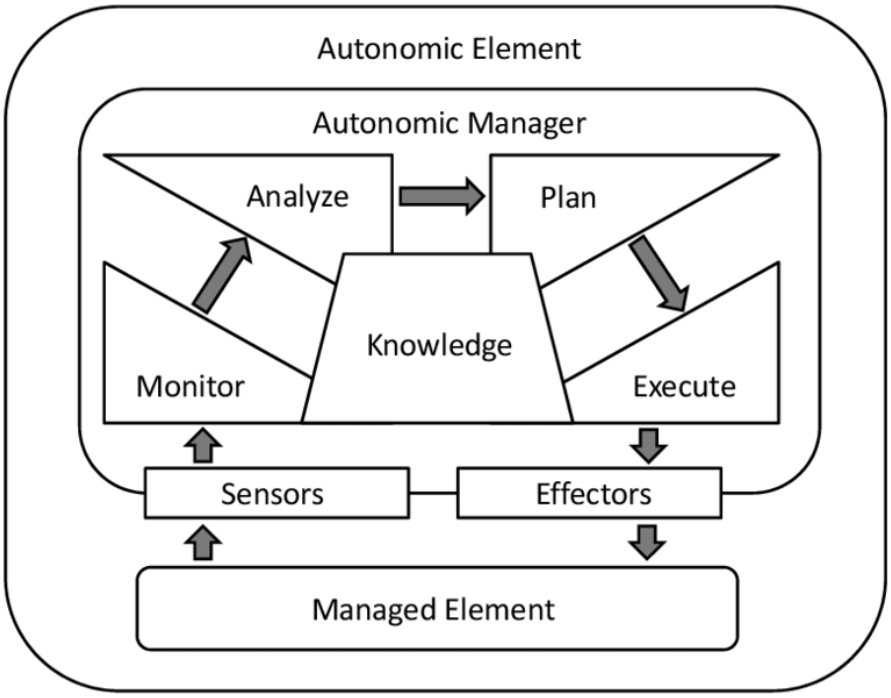


FIGURE 2.2: The IBM reference model for autonomic computing MAPE-K.

- **Monitor** , Collects the details such as metrics and typologies from the managed elements, extracting information properties or its states and filters these details until it determines a symptom that needs to be analyzed.
- **Analyze** , Perform complex data analysis and reasoning on the symptoms provided by the monitor function. If adaptations are required, an adapt request is logically passed to the plan function.
- **Plan** , is concerned with selecting a course of action to achieve goals and objectives of the managed element once a problem is detected.
- **Execute** , Changes the behavior of the managed element using effectors, based on the actions recommended by the plan function.
- Shared between these four phases is the **Knowledge** component, which contains models, data, and plans or scripts to enable separation of adaptation responsibilities and coordination of adaptations.

The Knowledge that is shared among the MAPE components contains various types of runtime models, including models of representative parts of the managed system and the environment, models of the qualities that are subject to adaptation, and other working models that are shared among the MAPE components. The Monitor collects runtime data from the managed system and the environment and uses this to update the content of the Knowledge, resolving uncertainties (e.g., the interference of the links in an iot network is tracked to update the relevant runtime models).

Based on the current knowledge, the Analyzer determines whether there is a need for adaptation of the managed system using the adaptation goals. If adaptation is required, the Planner puts together a plan that consists of a set of adaptation actions that are then enacted by the Executor that adapts the managed system as needed.

2.5 Architecture-Based Self-Adaptation

In the external model the main problem is determine the appropriate kind of models to use for software-based systems. Each type of model has certain advantages in terms of the analyses and kinds of adaptation it supports. In principle, a model should be abstract enough to allow straight forward detection of problems in the target system, but should provide enough fidelity to determine remedial actions to take to fix the problem. State machines, queuing theory, graph theory, differential equations, and other mathematical models have all been used for model based, external adaptation of software systems.

An architectural model provides a high-level view of a system as a collection of components and connectors, annotated with properties that indicate component and system attributes such as reliability, performance, and security [12].

Most of researches use a system's software architecture as the external model for dynamic adaptation. The architecture of a software system is the structure of its components, their interrelationships, and principles and guidelines controlling their design and evolution over time; which provides a global perspective on the system and exposes the important system-level behaviors and properties [13]. The use of software architecture as the basis for self-adaptation, called architecture-based self-adaptation.

The key part in architecture-based adaptation is the separation between the managed system that deals with domain goals, and the managing system that deals with the adaptation goals (Domain goals concern the environment in which the system operates,

while adaptation goals concern the managed system itself).

2.6 Self-adaptation patterns

Their six control patterns based on MAPE-K loop (Monitoring, Analysis, Planning, Execution) that model different types of interacting loops with different level of decentralization. **Figure** 2.3 shows the self-adaptation control patterns. In this figure, (MS) refers to manage subsystems.

- A **Centralized** self-adaptation pattern performs the adaptation through a central control loop.
- In a **Regional Planning** self-adaptation pattern, a physical space can be divided into different regions and the regions local planners coordinate to find the best adaptation solution for a local or global problem. It provides one P for each region to supervise the other elements of loop, in a way to interact different regions P one to another.

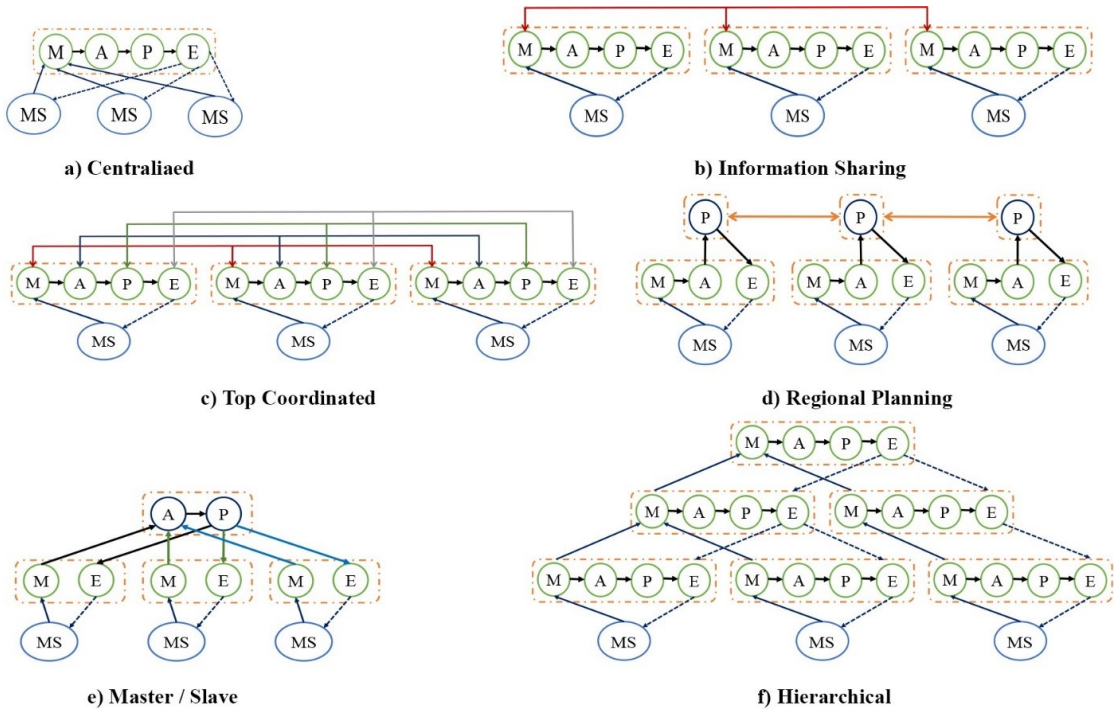


FIGURE 2.3: Self-adaptation patterns [3].

- **Coordinated control and Information Sharing** both are based on a fully decentralized approach, however, with a different level of components coordination. More precisely, in the coordinated pattern, all MAPE components coordinate with their corresponding peers, whilst in information sharing, only M components communicate with one another.

  The other three patterns, are based on a hierarchical distribution model.

- In **Master/Slave** pattern, a hierarchical relationship between one centralized master component (A and P) and multiple slave components (M and E) is created.

- The **Hierarchical** control pattern provides a layered separation of concerns to manage the complexity of self-adaptation as a hierarchy of MAPE-K loops.

2.7 Proactive and reactive self-adaptation

In reactive self-adaptation the system detects a change, and it adapts to continue to satisfy requirements, or to maximize some form of instantaneous utility. These reactive approaches have the advantage of incurring the overhead of the defense approach only when it is needed or affordable to do so. Their disadvantage is that, because they react to changes in the environment, they lag behind the state of the environment.

A promising approach that balances these two extremes is proactive adaptation, which uses predicted information about the near future state of the environment to avoid the overhead of defenses when they are not needed, and to adapt the system in time for predicted upcoming situations.

Proactive adaptation leverages predictions of the near future state of the system/environment to make better, proactive adaptation decisions. A system that has a prediction about the near future load on the system can not only avoid unnecessary adaptations, but also adapt to be in a configuration that suits better the environment.

2.8 Conclusion

As we see in Reactive approach the system detects the changes in environment and adapts to continue to satisfy its requirements but that made the approach lag behind the state of the environment. Otherwise, proactive approach uses predicted information about the near future state of the system/environment to avoid unnecessary adaptations and to make better adaptation decisions.

To maintain predicted information about the system we need to use machine learning algorithm which will be presented in the next chapter.

Part II

Machine Learning

Chapter 3

Machine Learning

3.1 Introduction

With the rapid increase in devices and applications connected to the Internet of Things, the sheer volume of data being created will continue to grow at an incredible rate. It is simply impossible for individuals to analyze and understand such quantities of data manually. Machine learning is helping to collective all of this data from countless sources and touch points to deliver powerful insights, spot actionable trends, and uncover user behavior patterns.

Machine learning has widely used in the field of Internet of Things, and it allowed achieving a spectacular results. In our work, we will use two of powerful machine leaning techniques which are called Reinforcement Learning and Artificial Neural Networks, so that, we get a very high total reward which is in our case Quality of Service (QoS) on Internet of Things (IoT) system. In this chapter, we will describe the basis of those concepts.

3.2 Definition

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves [14].

The process of learning begins with observations of data, like examples, direct experience,

or instruction, to find patterns within this data and make higher choices in the future based on the knowledge that we provide.

The primary goal is to permit the computers to learn automatically without human intervention and adjust actions based on it.

3.3 Application field

With the rise in big data, machine learning has become a key technique for solving problems in areas such as:

- Computational finance, for credit scoring and algorithmic trading.
- Image processing and computer vision, for face recognition, motion detection, and object detection.
- Computational biology, for tumor detection, drug discovery, and DNA sequencing.
- Energy production, for price and load forecasting.
- Automotive, aerospace, and manufacturing, for predictive maintenance.
- Natural language processing, for voice recognition applications.

3.4 Classification of Machine Learning

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Machine learning implementations are classified into three major categories which are as follows:

3.4.1 Supervised Learning

Supervised learning, in the context of artificial intelligence (AI) and machine learning, is a type of system in which both input and desired output data are provided. Input

and output data are labeled for classification to provide a learning basis for future data processing[15].

The idea of supervised learning is like having a teacher supervise the learning process.

In general, supervised learning occurs when a system is given input and output variables with the intentions of learning, although they are mapped together, or related. The goal is to produce an accurate enough mapping functions that when new input is given, the algorithm can predict the output. This is often an iterative process, and every time the algorithm makes a prediction, it is corrected or given feedback until it achieves a suitable level of performance.

Applications of supervised learning are typically broken down into two categories:

- **Classification** is a problem that is used to predict which class a data point is part of which is usually a discrete value.
- **Regression** analysis is a subfield of supervised machine learning. It aims to model the relationship between a certain number of features and a continuous target variable.

3.4.2 Unsupervised Learning

Unsupervised learning is a machine learning technique, where you do not need to supervise the model. Instead, you need to allow the model to work on its own to discover information. It mainly deals with the unlabeled data[16].

Unsupervised learning is wherever you simply input data (X), plus no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution within the data to acquire more about it.

These are known as unsupervised learning because unlike supervised learning above there is no precise answer and there is no teacher. Algorithms are left to their own devises to find out and present the interesting structure in the data.

3.4.2.1 Clustering

Clustering is an important concept when it involves to unsupervised learning. It mostly deals with finding a structure or pattern in a collection of uncategorized data. Clustering

algorithms will process your data and find natural clusters (groups) if they exist within the data.

It does this without having been told how groups should look ahead of time. As we may not even know what we're trying to find, clustering is used for knowledge discovery instead of. It provides an insight into the natural groupings found inside data.

3.4.2.2 Association

Association learning is a rule based machine learning and data mining technique that finds important relations between variables or features in a data set.

Association rule finds interesting associations and relationships among large sets of data items. This rule shows how frequently an item set occurs in a transaction.

3.4.3 Reinforcement learning

Reinforcement learning technology develops from some subjects such as statistics, control theory, psychology and so on, and includes a very long history, but it is not till the late 80s and early 90s that reinforcement learning technology finds the wide research and application in some fields like artificial intelligence, machine learning, and automatic control and so on.

Definition: Reinforcement Learning (RL) is frequently mentioned as a branch of artificial intelligence, and has been one of the central topics in a broad range of scientific fields for the past two decades.

Reinforcement learning comes from the animal learning theory. RL does not need prior knowledge, it can autonomously get optional policy with the knowledge obtained by trial-and-error and continuously interacting with dynamic environment [17].

In RL, the agent takes actions in an environment and receives reward (or penalty) for its actions. After a set of trial-and error runs, it should learn the best policy (optimal policy), which is the sequence of actions that maximize the total reward.

3.4.3.1 Reinforcement learning elements

Beyond the agent and therefore the environment, one can categorize four main sub elements of a reinforcement learning system: a policy, a reward signal, a value function, optionally, a model of the environment [18].

The **policy** is the strategy that the agent employs to determine the next action based on the current state. The policy is responsible for controlling behavior of reinforcement learning agent.

A **reward signal** explains the goal in a reinforcement learning problem. On each time step, the environment leads to the reinforcement learning agent a single number called the reward. The agent's sole objective is to maximize the whole reward it receives over the long run. The reward signal therefore defines what the good and bad events are for the agent.

A **model** is something that reproduce the behavior of the environment and allows inferences to be made about how the environment will react. Models are used for predicting the dynamics of the environment and planning on the best action to take considering the future reward returned by the environment.

Methods for solving reinforcement learning problems that use models are referred to as model-based methods, however methods that do not use models are model-free methods.

3.4.3.2 Reinforcement learning model

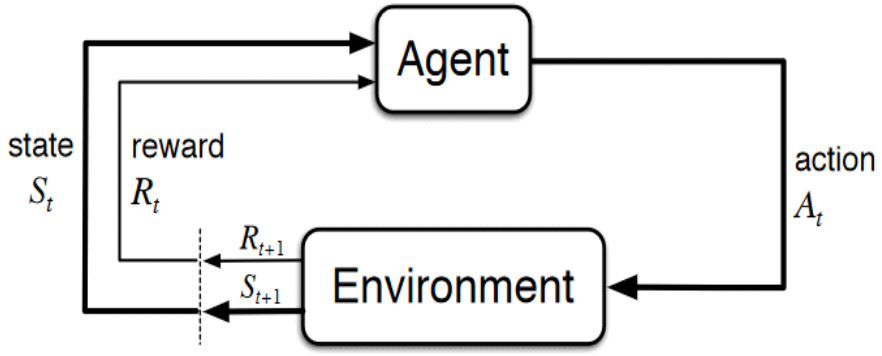


FIGURE 3.1: The agent-environment interaction in a Markov decision process

Basic reinforcement is modeled as a Markov decision process. The learner and decision maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment [18].

Markov decision process can be defined by four factors(S, A, R, P):

- **S;** a set of environment and agent states.
- **A;** a set of agent actions.
- **P**(S,S_{t+1})is the probability of transition (on time **t**) from state **S** to state. S_{t+1} under action **A**.
- **R**(S,S_{t+1}) is the immediate reward after transition from **s** to S_{t+1} with action **A**.

More specifically, the agent and environment interact at each of time steps. In every time step **t**, the agent receives the environment's state S_t ; and based on that selects an action A_t .One time step later, as a consequence of its action, the agent receives a reward, R_{t+1}, and finds itself in a new state,S_{t+1}.

Reinforcement learning system's goal is to learn an action strategy š. the strategy enables the action of the system choice to obtain the largest cumulative reward value of environment The basic theory of reinforcement learning technology is: If a specific system's action causes the positive reward of the environment, the system generating this action lately will strengthen the trend, this can be a positive feedback process; otherwise, the system generating this action will diminish this trend.

3.4.3.3 Reinforcement learning algorithm

Many proposed reinforcement learning algorithms require large amounts of training data before achieving acceptable performance. Typical reinforcement learning method based on the MDP model includes two kinds, direct (model-free) and indirect (model-based), when continuous actions are available.

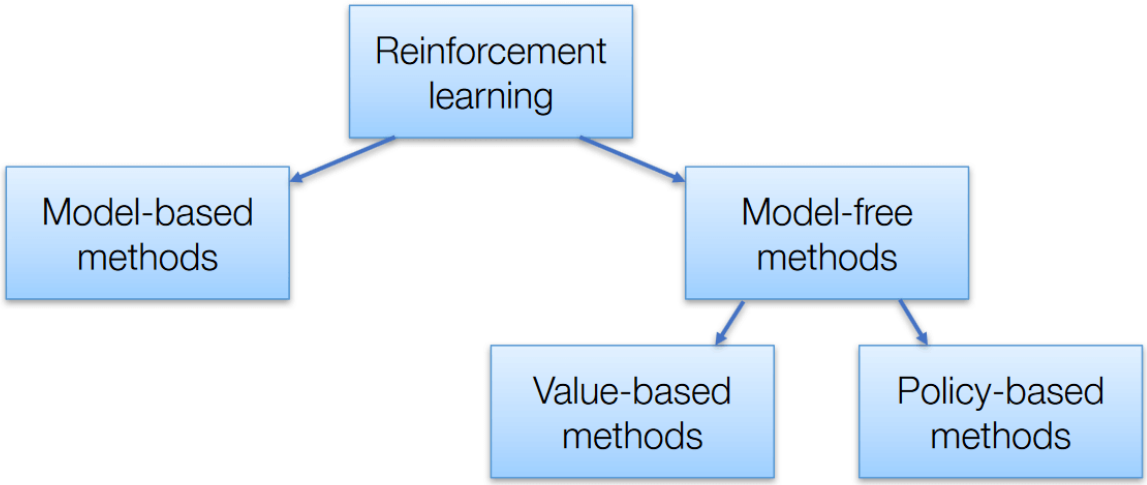


FIGURE 3.2: Reinforcement Learning algorithm

Model based reinforcement learning Model-based algorithm is an algorithm that uses the transition function (T) and the reward function(R) in order to estimate the optimal policy. In general, in a model-based algorithm, the agent can potentially predict the dynamics of the environment (during or after the learning phase), because it has an estimate of the transition function and reward function.

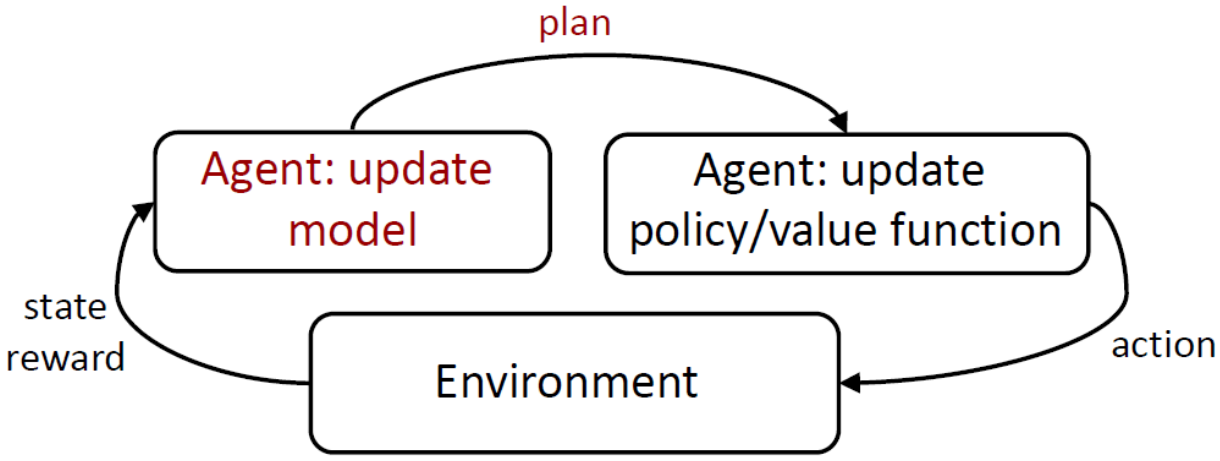


FIGURE 3.3: Model based reinforcement learning [19].

In other word, the agent learn a model of how the environment works from its observations and then plan a solution using that model. That is, if the agent is currently

in state S_1, takes action A_1, and then observes the environment transition to state S_2 with reward R_2, that information can be used to improve its estimate of $\mathbf{T}(S_2 \mid S_1, A_1)$ and $\mathbf{R}(S_1, A_1)$.

Once the agent has adequately modeled the environment, it can use a planning algorithm with its learned model to find a policy. RL solutions which follow this framework are model-based RL algorithms.

Model free reinforcement Learning Model-free means that the agent tries to maximize the expected reward only from real experience, without a model/prior experience. It does not know which state it will be in after taking an action, it only cares about the reward associate with the state/state-action.

Generally Model-free methods are less computational-heavy compare to model-based methods (they are trying to get the optimal policy, not learn the entire dynamic of the environment).

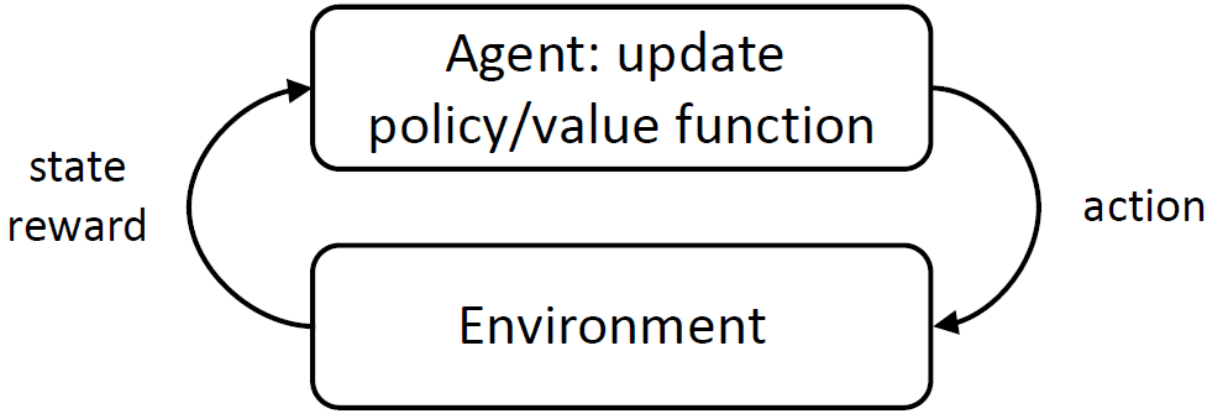


FIGURE 3.4: Model free reinforcement learning
[19].

Value based and Policy based methods Nearly all reinforcement learning algorithms include estimating value functions (functions of states or of state–action pairs).

Value function is that the estimate of how good it is to perform a given action during a given state. The concept of "how good" here is defined in terms of future rewards that can be expected. Of course, the rewards the agent will expect to receive within the future depend on what actions it will take.

Value based methods, the agent uses its experience with the environment to maintain an estimate of the optimal action-value function .The optimal policy is then obtained from the optimal action-value function estimated.

Policy based methods directly learn the optimal policy while not having to maintain a separate value function estimate.

On-policy and Off-policy In RL algorithm there are two phases: the learning (or training) phase and behavior phase (after the training phase). The distinction between on-policy and off-policy algorithms only concerns the training phase.

During the learning phase, the RL agent needs to learn an optimal value (or policy) function. Given that the agent still does not know the optimal policy.

During training, the agent faces a dilemma: the exploration or exploitation dilemma. In the context of RL, exploration and exploitation are different concepts: exploration is the selection and execution (in the environment) of an action that is likely not optimal (according to the knowledge of the agent) and exploitation is the selection and execution of an action that is optimal according to the agent's knowledge (that is, according to the agent's current best estimate of the optimal policy).

An ***off-policy algorithm*** is an algorithm where a target policy of the agent is learnt during training that is different from the optimal policy it tries to estimate (the optimal policy), however a different behavior policy is used to generate the behavior of the agent. For example Q-learning which often uses a greedy policy.

An ***on-policy algorithm*** is an algorithm that, during training, chooses actions using a policy that is derived from the current optimal policy, while the behaviors are also based on the current optimal policy. For example SARSA.

3.4.3.4 Q-Learning:

Definition: (Watkins, 1989) O-learning is a form of model-free reinforcement learning. It provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build maps of the domains.

Q-Learning is an off-policy algorithm, i.e. the agent learns the value of the state-action pair independently of the performed action, because their updates are done regardless

from the current action, but with respect to the action that maximizes the value of the next state-action pair [20].

3.4.3.5 How it works

Q-Learning is a basic form of Reinforcement Learning which uses Q-values (also called action values) to iterative improve the behavior of the learning agent.

Q-values are defined for states and actions. Q(s,a)is an estimation of how good is it to take the action a at the state s. This estimation of Q(s,a) will be iterative computed using the Temporal Difference.

Temporal-Difference (TD) methods learn online directly from experience, do not require a model of the environment, offer guarantees of convergence to optimal performance, and are straightforward to implement [21].

Q-Learning defined as (Bellman Equation) :

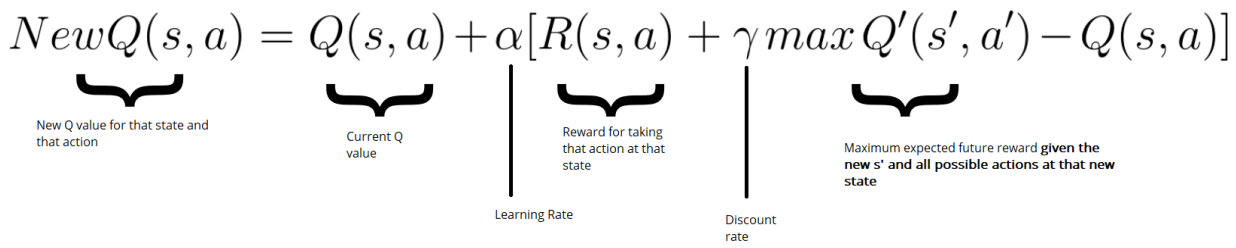


FIGURE 3.5: Bellman Equation.

- **s** Current State of the agent.
- **a** Current Action Picked according to some policy.
- s , Next State where the agent ends up.
- a , Next best action to be picked using current Q-value estimation, i.e. pick the action with the maximum Q-value in the next state.
- **R** Current Reward observed from the environment in Response of current action.
- γ(>0 and <=1) Discounting Factor for Future Rewards. Future rewards are less valuables than current rewards, so they must be discounted. Since Q-value is an estimation of expected rewards from a state, discounting rule applies here as well.

- α Step length taken to update the estimation of Q(S, A) [22].

Before, the agent ends up in one of the terminating states that means there are no further transition possible which means the completion of an episode.

In Q-learning the agent repeats the following 4 steps until the task are done:

1. Agent senses its environment, using this information to determine its current state.

2. Agent takes an action and obtains a penalty or reward.

3. Agent senses its environment again - to see what effect its chosen action had.

4. Agent learns from its experience (and so makes 'better' decisions next time).

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

FIGURE 3.6: Q-Learning algorithm
[23].

Q-Learning is a fast algorithm, it can explore and it can exploit.

Its speed arises from employing a greedy algorithm that may be a problem-solving heuristic which makes the locally optimal choice at each stage in the hope of finding a global optimum. This shallow deliberation is fast, which may be vital in the real-time decision-making once we can't afford to wait for a response.

3.4.4 Artificial Neural Network

Artificial neural networks are inspired from human brain. Artificial neural networks are an attempt of modeling the information processing capabilities of nervous systems [24].

Neural networks employ a massive interconnection of simple computing cells referred to as "neurons" inspired by the natural neurons to achieve the best performance possible.

Artificial Neural Network acts like the brain in two-way:

- Knowledge is acquired by the network over learning process.
- Inter neuron connection strengths referred to as synaptic weights are used to store the knowledge.

3.4.4.1 Neural networks components

An artificial neural network consists of a collection of multiple neurons. Every neuron is a node which is connected to other nodes with links that correspond to biological axon-synapse-dendrite connections. Each link has a weight, which determines the strength of one node.

3.4.4.2 Neurons

ANNs are composed of several artificial neurons. Each artificial neuron has inputs and produce one output which may be sent to multiple different neurons. The inputs often are the feature values of a sample of external data, like images or documents, or they can be the outputs of other neurons.

3.4.4.3 Layers

Contain neurons and help pass information around. There are at minimum two layers during a neural network: Input and Output layer.

The layers, other than the input and output layers, are called hidden layers.

3.4.4.4 Connections and weights

The network consists of connections, each connection providing the output of one neuron as an input to a different neuron. Each connection is assigned a weight that represents its relative importance .The weights are the coefficients that used to amplify or minimize an input signal, which represent the strength of the connection between neurons, and decides how much influence the input will have on the output.

3.4.4.5 Biases

A bias has its own connection weight, what makes sure that even when all the inputs are none there's going to be an activation in the neuron.

Biases are numerical values which are added once weights are applied to inputs.

3.4.4.6 Activation Function

Essentially activation functions is a mathematical formula (algorithm) that normalizes the output before it is passed on to the next or previous neurons in the chain. These functions help neural networks learn and improve themselves.

A function that is used to transport values through the neurons of a neural net's layers. Usually, the input values are added up and passed to an activation function, which generates an output [25].

3.4.4.7 Hyperparameters

Hyperparameters are the variables which determines the network structure and the variables which determine how the network is trained [26].

Hyperparameters are set before the learning process begins. Those are different hyperparameter:

- Layer size.
- Momentum, Learning rate.
- Activation function.
- Dropout.

- Weight initialization strategy.

3.4.4.8 Deep Learning

The adjective "deep" in deep learning comes from the utilization of multiple layers within the network. That means that Deep learning is a neural network with more than two layers.

Deep learning models can achieve accuracy, sometimes exceeding human-level performance. These models are trained by using a large set of labeled data and neural network architectures that contain several layers.

3.4.4.9 Where are Neural Networks being used

Neural networks are applied in solving a wide variety of problems.

Basically, most applications of neural networks fall under the following categories:

- **Prediction** Uses input values to predict some output. e.g. pick the most effective stocks within the market, predict weather, identify people with cancer risk.
- **Classification** Use input values to determine the classification. e.g. the input the letter A, is that the blob of the video data a plane and what kind of plane is it.
- **Data association** Like classification, however it also recognizes data that contain errors. e.g. not only identify the characters that were scanned but identify once the scanner is not working properly.
- **Data Conceptualization** Analyze the inputs in order that grouping relationships can be inferred. e.g. extract from a database the names of those most likely to by a specific product.
- **Data Filtering** Smooth an input signal. e.g. take the noise out of a telephone signal.

3.5 Conclusion

In this chapter, we discussed machine learning techniques and its different types also an overview about neural networks and deep learning. So far We have demonstrated the theoretical part of our research, which has to be mentioned to understand the scientific fundamentals and rules, which allows an understanding of our work proposed in the following sections .The rest of this research is organized as follows: Part 03, discusses the objective and some of the related work, and finally we will present our proposed approach and the results obtained.

Part III

Design and Implementation

Chapter 4

Design

4.1 Introduction

To solve the problem of optimizing and reconfiguring the IoT network. Many researches have been applied in the field of self-adaptive systems. The purpose of our approach is to realise an adaptive IoT application with implementation of an architecture-based adaptation solution to the case. We start with an overview of the general architecture and a description of system components; then we illustrate the quality models that we used to estimate the quality attributes and adaptation goals. Finally, we describe the overall functioning of the system.

4.2 Objective

This work aims to design and implement a system that allows an IoT (Internet Of Things) system to adapt itself at runtime to changes to maintain its required quality goals (exp; The average packet loss over 24 h should not exceed 10% , or energy consumption should be minimized). The developed system should dynamically reconfigure the IoT system from the deviations of expected quality of service (QoS) parameters. More specifically, adaptation actions should be taken proactively, which means that the system should anticipate the change before the occurrence of QoS deviation. This could be done by predicting possible QoS deviations by using machine learning techniques.

4.3 Related Work

A number of recent efforts have explored the application of self-adaptation in IoT System. D. Weyns et al. [27] present MARTAS approach by Applying Architecture-Based Adaptation to Automate the Management of Internet-of-Things. H. Muccini et al. [3] provide a literature based knowledge to learn and evaluate IoT distribution patterns, and self-adaptation control patterns. Robbe Berrevoets and Danny Weyns [28] investigated A QoS-aware Adaptive Mobility Handling Approach for LoRa-based IoT Systems.

Some papers on the use of Machine Learning for self-adaptive systems in other domains have been presented as well. Kunal Shah and Mohan Kumar [29] present Distributed Independent Reinforcement Learning (DIRL) Approach to Resource Management in Wireless Sensor Networks.

Karthik Vaidhyanathan and Henry Muccini [30] apply A Machine Learning-driven Approach for Proactive Decision Making in Adaptive Architectures.

From the related works reported above, our approach share some similar point with DIRL Approach [29] that identifies the need for implementation of Q-Learning algorithm at sensor nodes. To allow each individual sensor node to self-schedule its tasks and generate appropriate action in any given state to maximizing total amount of reward over time. Also, we added and applied a Deep learning model at the MAPE-K loop on top of the IoT system to determine proactively the optimal adaptation action.

4.4 Architecture of The Approach

We start with a general overview of the approach to realise the adaptive IoT application.

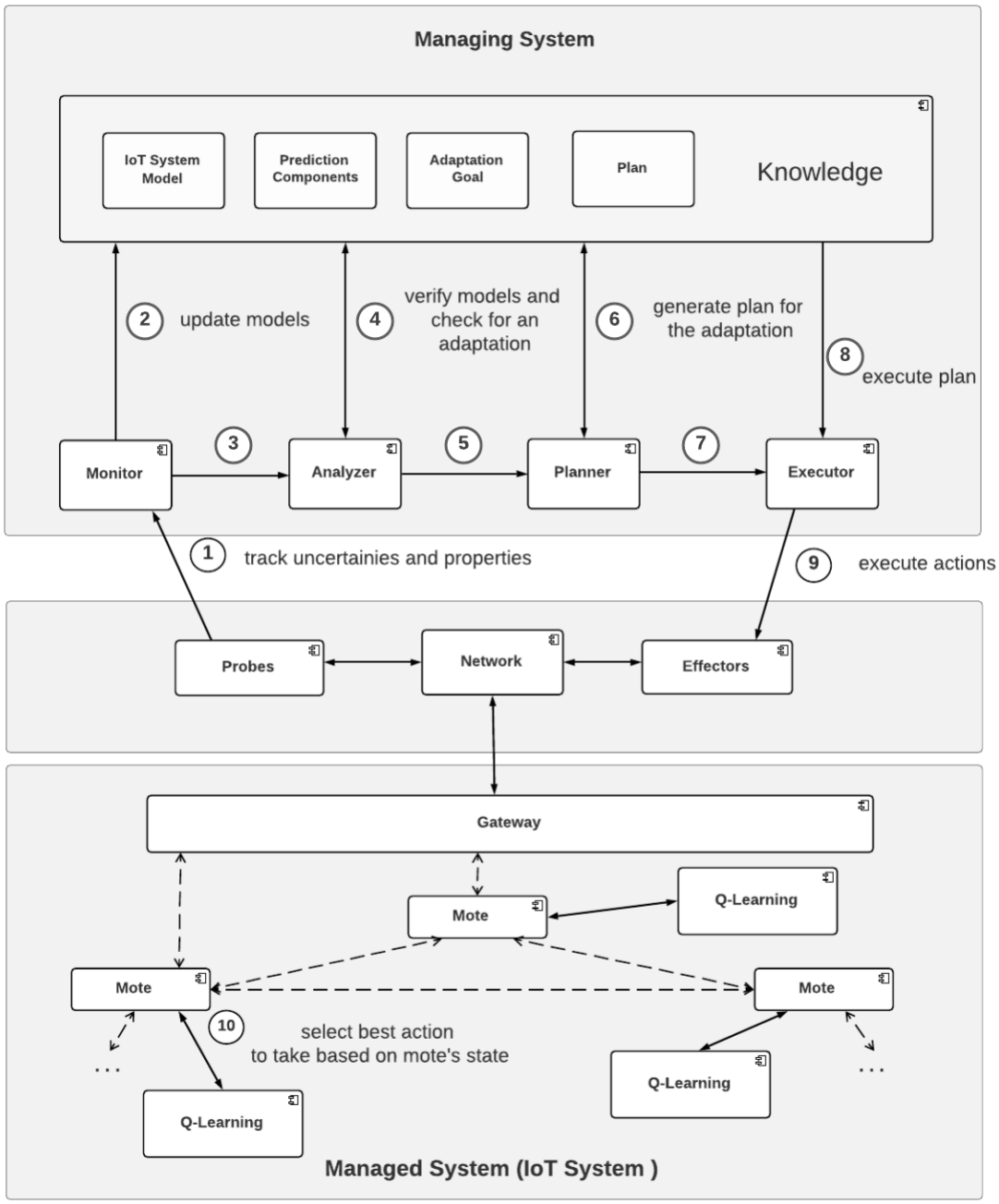


FIGURE 4.1: General architecture of the approach.

The bottom layer consists of the managed system with the network of motes and the gateway. The middle layer comprises a network engine, or we can present as client that uses the IoT network and offers an interface to it via a probe and an effector. The probe can be used to monitor the IoT network (status of motes and links, data about the packet loss, the energy consumption, etc.). And the effector adapts the mote settings (power settings of the motes, distribution of packets sent to parents, etc.). As for the top layer is added to the system that automatically adapts the adaptation goals of the IoT network.

The approach works as follows:

1. The monitor tracks uncertainties (such as Network interference and noise and the changes of packets produced by the motes) and relevant properties of the managed system and the environment in which the system is deployment.
2. The collected data is used to update the models.
3. The monitor triggers the analyzer.
4. The analyzer reads the adaptation goals and also the updated models from the knowledge, as well it estimates the expected deviation of the adaptation goal.
5. If an adaptation required the analyzer triggers the planner.
6. The planner generates a plan for the best adaptation.A plan consists of a series of adaptation actions.
7. The planner triggers the executor.
8. The executor executes the plan.
9. That is, the executor executes the adaptation actions to adapt the managed system.
10. One from its adaptation action is to trigger the Q-Learning algorithm, which selects the best action to take based on mote's state.

4.5 Detailed description of the architecture

Detailed architecture describes the components that make up the approach; our work is composed of three components. We start with MAPE-K loop that optimize and reconfigure the IoT network. Then, we illustrate the prediction components that we used to estimate the deviation of quality of service. The last component is the Q-Learning algorithm that selects the adaptation action. In this part, we will talk and discuss each approach component in detail.

4.5.1 MAPE-K Loop

As we saw in **Figure 2.3** (that shows Self-adaptation patterns.). In our approach we used a centralized adaptation pattern which has a focal MAPE-K loop that is responsible for managed system adaptation. Also, we used it since our study is to reduce Packet loss delivery and minimize energy consumption.

This pattern can only be combined with centralized IoT distribution pattern since the adaptation should take place in one central processing component [3].

The MAPE-K loop is the system that automatically adapts the adaptation goals of the IoT network. The process starts with the monitor which uses the probe to track the traffic load and network interference's as well as everything related with the managed system and quality attributes. This data is used to update a set of models in the knowledge repository, including a model of the IoT system with the relevant aspects of the environment, and a set of quality models (we will discuss below what are the models that have been used).

The **IoT system model** contains (status of motes and links, data about the packet loss, the energy consumption, latency of the network, etc.)

Sub-Component

- The **Monitor** Collects the data and settings from IoT network and updates the knowledge repository.
- The **Analyzer** predicts the expected deviation in adaptation goals by using the prediction components.

- The **Planer** Selects the adaptation action to achieve quality goals and objectives of the network.
- The **Executor** executes the adaptation actions to adapt the IoT system.

4.5.2 Prediction components

Quality Models are used to estimate the expected quality attributes (energy consumption and packet loss).

4.5.2.1 Packet Loss Model

This phase is divided into two steps: the first step is a way of collecting mote's information (link data). The second step is to label every data with a score (0 or 1) relying on adaptation if it required or not. As well, we have a tendency to train deep learning model on our pre-labeled data-set. Deep learning is easy to build and particularly functional for large data sets, and it is known to outperform even highly sophisticated classification methods. Since it is a one-of-a-kind algorithm whose performance continues to improve as more the data fed, the more the classifier is trained on resulting in outperforming more than the traditional models/ algorithm.

This model will be present as Neural Network started by an input and a two hidden layers ended.

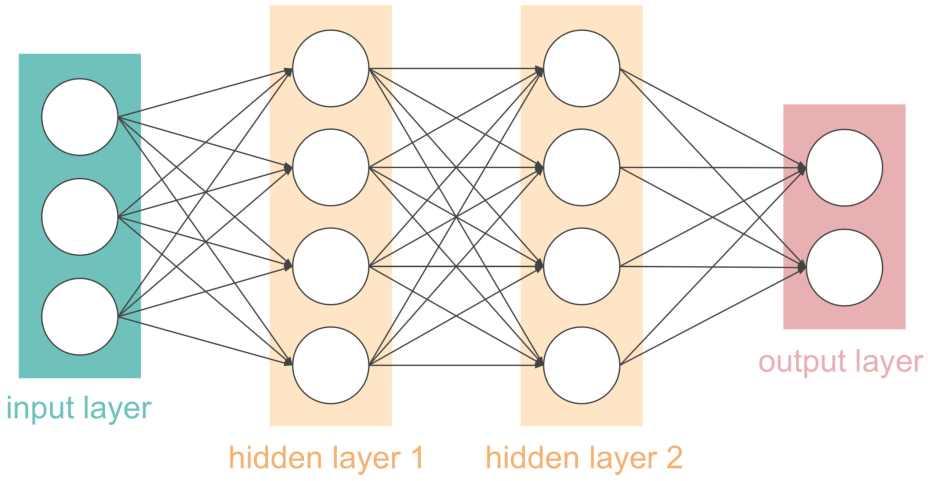
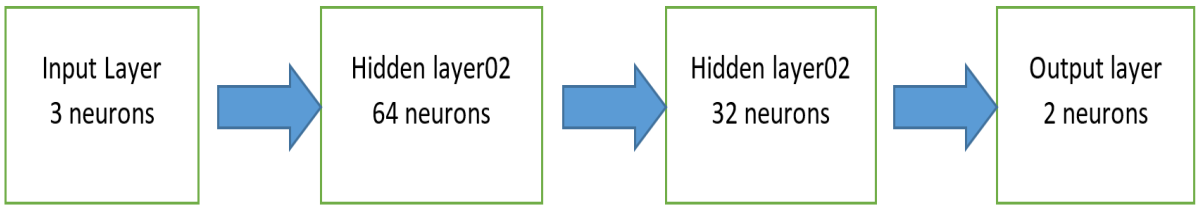


FIGURE 4.2: A neural network with 2 hidden layers.

Our Neural Network will be presented as below:



The Data-set that used for training the model was collected from motes data generated by the Simulation process. One run in simulation means one day of process (period of 24 h). So we collected the data from 1000 simulations (1000 days) to be trained and tested by our model.

Distribution	SNR	Link Power	Adaptation
100.0	0.505263	15.0	1
100.0	3.596491	10.0	1
90.0	-2.210526	9.0	0
50.0	1.001754	11.0	1
100.0	3.584211	15.0	1
50.0	3.592982	2.0	1
40.0	-4.015789	15.0	0
90.0	-3.084211	13.0	0
70.0	3.056140	14.0	1
40.0	1.710526	9.0	1

FIGURE 4.3: A sample of packet loss data-set.

As we see our data-set is simple and organized so there is no need of preprocessing it.

Below shows an example information, with the model predicting zero, which indicates an adaptation required.

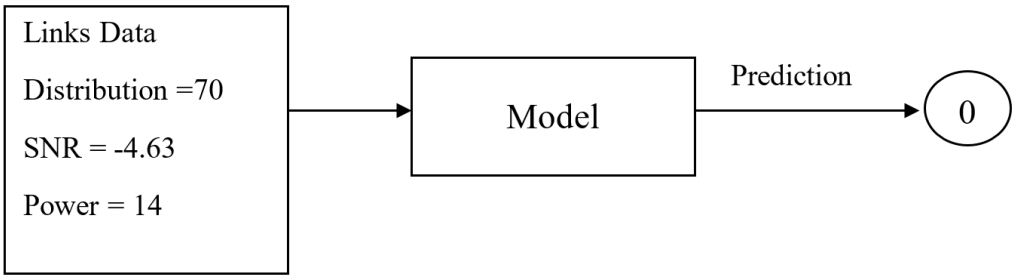


FIGURE 4.4: Example of the prediction process of Packet Loss.

4.5.2.2 Energy Consumption Model

we used the same Neural Network structure as Packet Loss Model that has an input and a two hidden layers, but with different Data-set.

we collected the data from 1000 simulations (1000 days) to be trained and tested by our model. **Figure 4.5** shows the data-set of energy consumption model.

Distribution	Link Power	TrafficLoad	QueueSize	Adaptation
40.0	4.0	0.07	10.0	1
20.0	3.0	0.07	10.0	1
10.0	8.0	1.00	19.0	1
40.0	12.0	1.00	30.0	0
40.0	5.0	1.00	30.0	0
...	...	...	...	...
30.0	14.0	1.00	30.0	0
10.0	11.0	1.00	30.0	0
50.0	10.0	1.00	10.0	1
30.0	10.0	0.85	10.0	1
60.0	10.0	1.00	10.0	1

FIGURE 4.5: A sample of energy consumption data-set.

Below shows an example information, with the model predicting zero, which indicates an adaptation required.

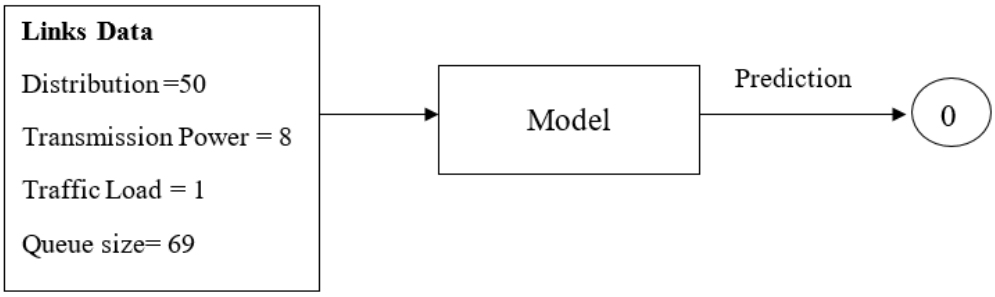


FIGURE 4.6: Example of the prediction process of energy consumption.

4.5.3 Q-Learning

Q-Learning is quite simple, demands minimal computational resources and doesn't require a model of the environment in order to operate. Hence, it is ideal for implementation on resource-constrained sensor nodes. This algorithm embedded in every mote on the network so that the mote will take the best action based on its state in the environment. After the mote state changes by taking the adaptation action, the data will flow to the MAPE-K loop for managing the whole system.

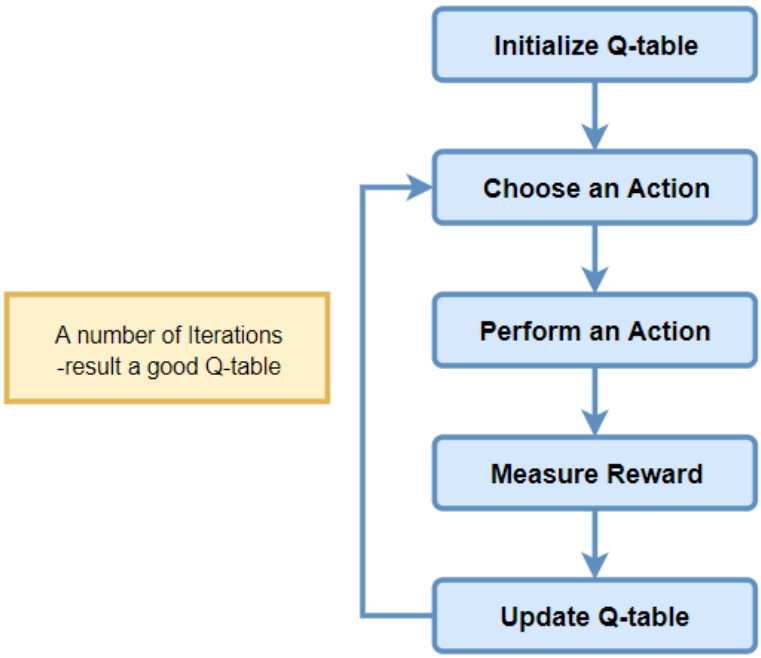


FIGURE 4.7: Training Q-Learning Algorithm [31].

The states

The states are the information about the motes on network .The number of its links, their distribution factor, power settings of links and the Signal Ratio Noise (SNR).

Machines understand numbers rather than letters. So, we will code this information to numbers.

Mote's information	**States**
Other states	0
Link SNR >0 and Link Transmission Power >0	1
Link SNR <0 and Link Transmission Power <15	2
Links Size >1 and (links SNR and Transmission Power are different)	3
Links Size >1 and (links Distribution and Transmission Power are the same)	4
Links Size >1 and (links Distribution and Transmission Power are different)	5

TABLE 4.1: Mote's information coded as states.

The actions

The actions that the mote take to adapt to the network uncurtains are mapped like:

Possible actions	**Actions**
Reduce link's Transmission Power (-1)	0
increase link's Transmission Power (+1)	1
Reduce link's Distribution (-10)	2
increase link's Distribution (+10)	3
change (link's Distribution and link's Transmission Power)	4

TABLE 4.2: Possible actions coded as Actions.

Step 1: Initialize the Q-Table

We will use Q-table to guide us to the best action at each state.

First, the Q-table has to be built. There are n columns, whereas n= number of actions. There are m line, where m= number of states.

In our approach n=4 and m= 6. First, let's start with the values 0.

We also have Reward-table that have the same size as Q-table.at first,we start with the values 0 and in the learning process if the mote at a given state take action ,and that lead to send packet with success will give him a reward(r=1) else will give him punishment(r = -1).

Step 2 and 3: Choose and Perform an Action

The combination between the step 2 and the step 3 is achieved for an undefined amount of time. These steps run until the time training is stopped.

First, as mentioned earlier, when the episode initially starts, every Q-value should be 0. We will use *Epsilon greedy strategy* , this concept identify the mote's exploration ratio in the network. In the beginning, the epsilon rates will be higher. The mote will explore in the network and randomly choose actions.

Logically this concept occurs like this, since the mote does not know anything about the network. As the mote explores, the epsilon rate decreases and the mote starts to exploit the environment, which mean an action (a) in the state (s) is chosen based on the Q-Table.

During the process of exploration, the agent progressively becomes more confident in estimating the Q-values.

Steps 4: Measure Reward

Now we have taken an action and observe an outcome and reward.

The rewarding scheme as follows:

Reward when send packet with success = +100

Reward when packet failed to send =-10

Reward when take bad action =-1

Reward when do nothing=0

Steps 5: Evaluate

In this step we need to update the function Q(s, a).

This process is repeated until the learning is stopped. In this method the Q-Table is been updated and the value function Q is maximized.

4.6 Functional Overview of the system

The global overview of our system will be presented on this diagram (Figure).

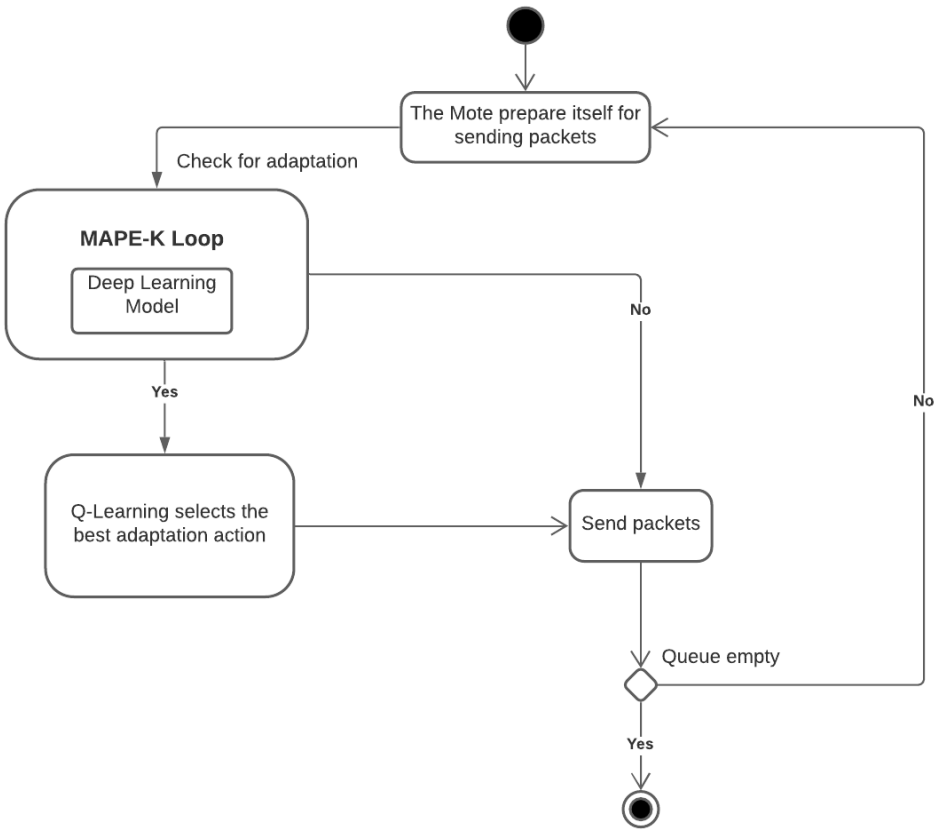


FIGURE 4.8: Global overview of the system.

The process begins with the mote preparation for sending the packets that received from children or the locally generated packets. The MAPE-K Loop will monitor the collected motes data and intervene by generating an adaptation action if an adaptation required. In addition, The Q-Learning algorithm will be triggered to estimate the optimal action to take based on the mote state by effecting the link distribution and power settings. The adaptation is estimated for each mote on the network in order to provide the adaptation goals. The mote then sends the packets in its queue to its parents one by one. As soon as the queue is empty the mote returns to the idle state.

4.7 Conclusion

In this chapter we have presented an approach of self-adaptive IoT system by applying an architecture-based adaptation to it. Also we discuss the quality models (Q-Learning and Deep Learning) that estimate the deviation in the quality attributes (Packet loss and Energy consumption) to guaranty the adaptation goals (QoS).

In the next chapter we will present the implementation of our system and the results of the experiment.

Chapter 5

Implementation

5.1 Introduction

After presenting in detail our Architecture-based adaptation approach on IoT network. This chapter will be devoted to the implementation phase, we will show how we made and implemented our system. We start by presenting the software environment used, through the presentation of the tools and the programming language. Finally, the evaluation process and comparison of our approach versus MARTAS approach.

5.2 Used Softwares and Materials

5.2.1 Materials

Laptop HP ProBook 450 G3.

Processor: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz (4 CPUs),$\sim$ 2.4GHz.

Memory: 4096MB RAM.

Graphics: Intel(R) HD Graphics 520.

Operating System: Windows 10 Professional 64-bit.

5.2.2 Programming language

5.2.2.1 Java

Java was originally developed by James Gosling at SunMicrosystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems platform. Java is defined as an object-oriented language similar to C++, but simplified toeliminate language features that cause common programming errors [32].

Java produces software for multiple platforms. When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS. Java derives much of its syntax from the C and C++ programming languages.

In simple words, this language is free to access and can run on all platforms.

5.2.2.2 Python

Python is probably the easiest-to-learn and nicest-to-use programming language in widespread use. Python code is clear to read and write, and it is concise without being cryptic. Python is a very expressive language, which means we can usually write far fewer lines of Python code then would be required for an equivalent application written in, say, C++ or Java.

Python is a cross-platform language: In general, the same Python program can be run on Windows and Unix-like systems such as Linux, BSD, and Mac OS X, Rasbian, simply by copying the file or files that make up the program to the target machine, with no "building" or compiling necessary. It is possible to create Python programs that use platform-specific functionality, but this is rarely necessary since almost all of Python's standard library and most third-party libraries are fully and transparently cross-platform [33].

5.2.3 Development tools and frameworks

5.2.3.1 Eclipce

Eclipse is an integrated development environment (IDE) for developing applications using the Java programming language and other programming languages such as C/C++, Python, PERL, Ruby etc. The Eclipse platform which provides the foundation for the Eclipse IDE is composed of plug-ins and is designed to be extensible using additional plug-ins. Developed using Java, the Eclipse platform can be used to develop rich client applications, integrated development environments and other tools. Eclipse can be used as an IDE for any programming language for which a plug-in is available [34].

The Eclipse software development kit (SDK) is free and open-source software, which includes the Java development tools, is meant for Java developers.

5.2.3.2 PyCharm

PyCharm is an extremely popular Python IDE. An Integrated Development Environment or IDE features a code editor and a compiler for writing and compiling programs in one or many programming languages.

A company called Jetbrains has developed PyCharm as a cross-platform IDE for python. In addition to supporting versions to 2.x and 3.x of python, PyCharm is also compatible with Windows, Linux, and even Mac OS and at the same time, the tools and the features provided by PyCharm helps programmers to write a variety of software applications in Python very quickly and efficiently [35].

However, you can also create HTML, CSS, and JavaScript files using it. PyCharm gives support for these programming languages because Python is also used for building web-applications.

5.2.3.3 Javafx

JavaFX is a Java library that is used to develop Desktop applications as well as Rich Internet Applications (RIA). The applications built in JavaFX, can run on multiple platforms including Web, Mobile and Desktops.To develop GUI Applications using Java programming language, the programmers rely on libraries such as Advanced Windowing Toolkit and Swing. After the advent of JavaFX, these Java programmers can now develop GUI applications effectively with rich content [36].

JavaFX provides more functionalities than swing. Also, it provides its own components and doesn't depend upon the operating system. It is lightweight and hardware accelerated. It supports various operating systems including Windows, Linux and Mac OS.

5.2.3.4 SceneBuilder

JavaFX Scene Builder is a visual layout tool that lets users quicklydesign JavaFX application user interfaces, without coding. Users can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in the background. The result is an FXML file that can then be combined with a Java project by binding the UI to the application's logic [37].

5.2.3.5 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to do good research. And its primary author and maintainer is François Chollet, a Google engineer. Keras is an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential

abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity [38].

5.3 Simulator

We used the DeltaIoT exemplar which offers a simulator for self-adaptation experimentation. This simulator enables to test and compare new adaptation solutions fast. In the simulator, the activities of the network during a specified period of wall clock time can be simulated in one run; the default period is 15 minutes. **Figure5.1**, shows the architecture of the DeltaIoT simulator.

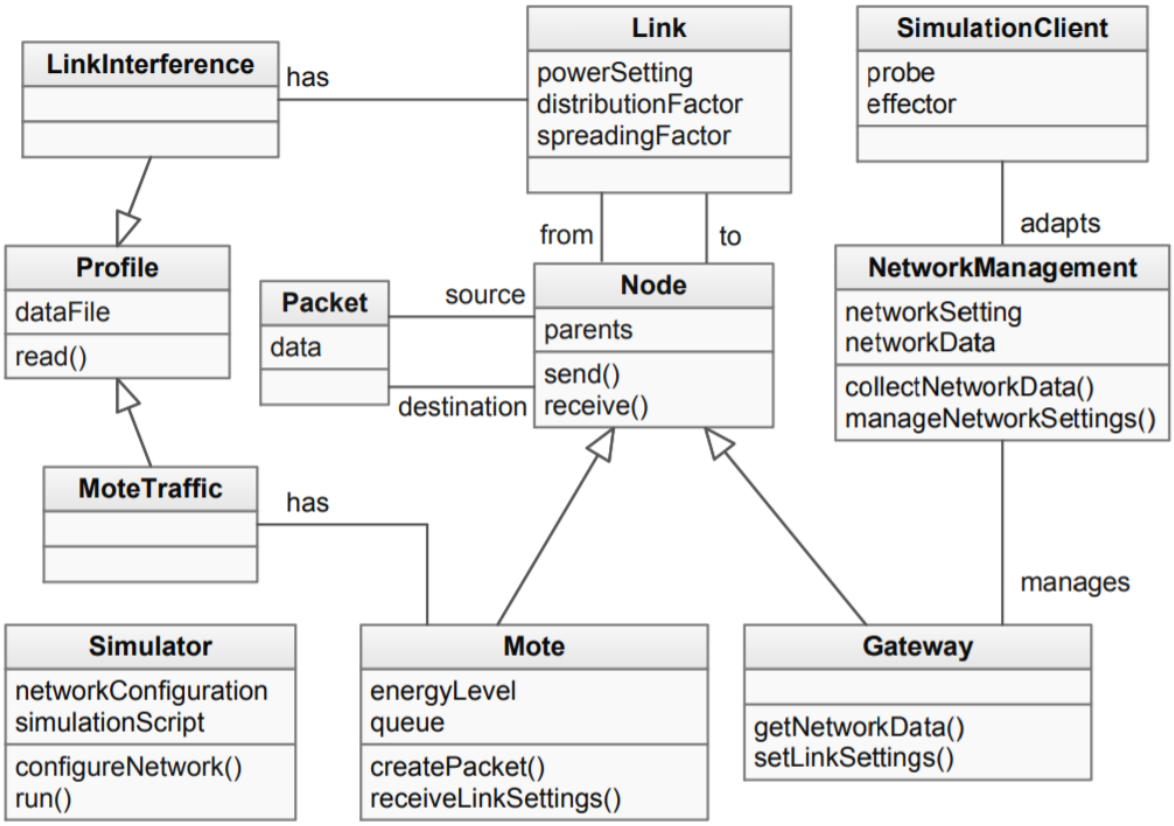


FIGURE 5.1: Architecture of DeltaIoT Simulator [39].

Node, Gateway, Mote, Link, and Packet are the basic elements of the IoT network. These elements correspond to the IoT Network Tier and the Gateway Tier.

The *Simulation Client* offers a probe and effector to apply self-adaptation to the IoT network. The concept of a *Profile* to specify uncertainties in the simulated system.

A *profile* is defined by a file that contains a series of values that represent a property of the system or its environment over time.

Link Interference defines the levels of interference on a link over time, while *Mote Traffic* defines the traffic generated by a mote over time.

Finally, *Simulator* enables a user to perform a simulation of a network configuration.

Figure 5.2 shows the map topology of the network component which contain 14 motes and one Gateway

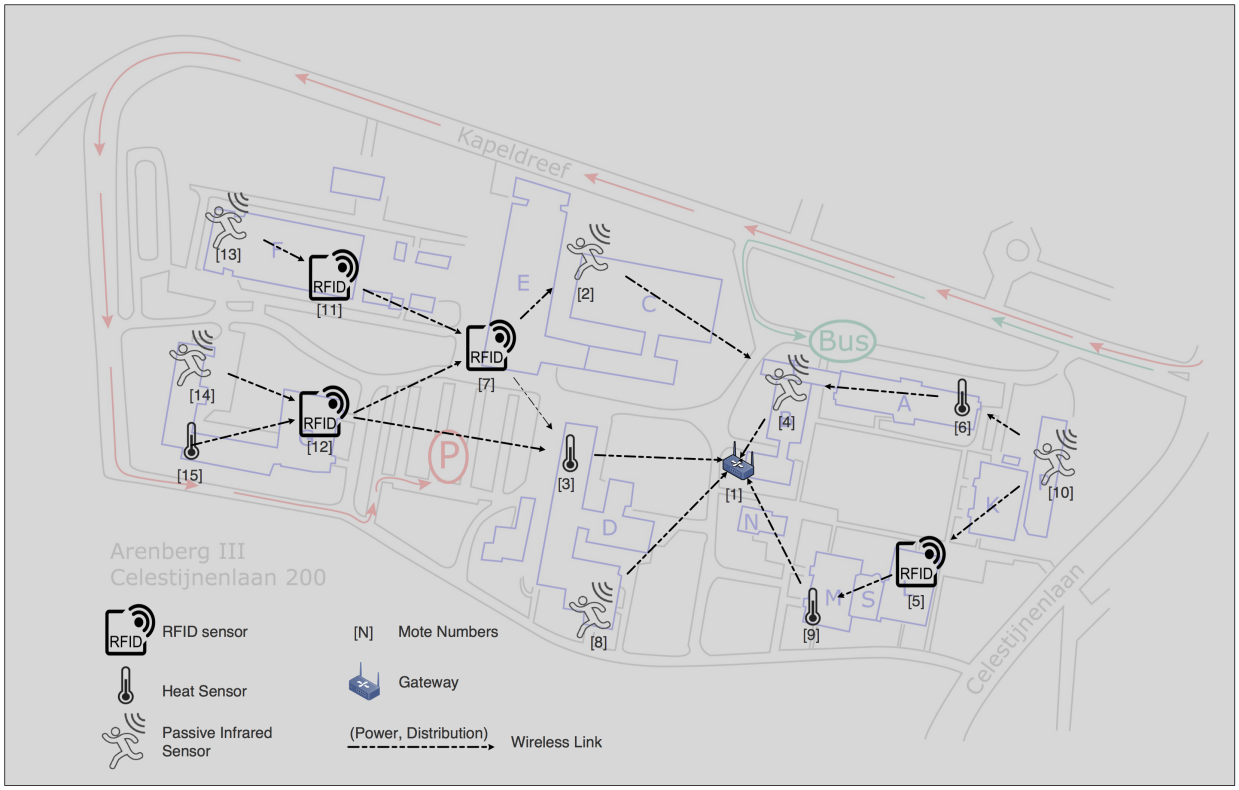


FIGURE 5.2: DeltaIoT network topology.

The simulation process begins with defining turn order for each *mote* in the network in the order they communicate *packets*. When a mote gets its turn ,the *probability* that it will send *packets* is determined based on its recently observed traffic load .The *mote* then sends the *packets* in its *queue* to its parents one by one (the *packets* it received from

children and the locally generated packets). As soon as the *queue* is empty the mote returns to the idle state. All the packets goes to the *gateway* and when he gets its turn, it computes the average energy consumption that was required to communicate packets and the percentage of *packet loss* in the cycle.

5.4 Project structure

Figure 5.3 shows the structure of the project after implementing our approach in the simulator.

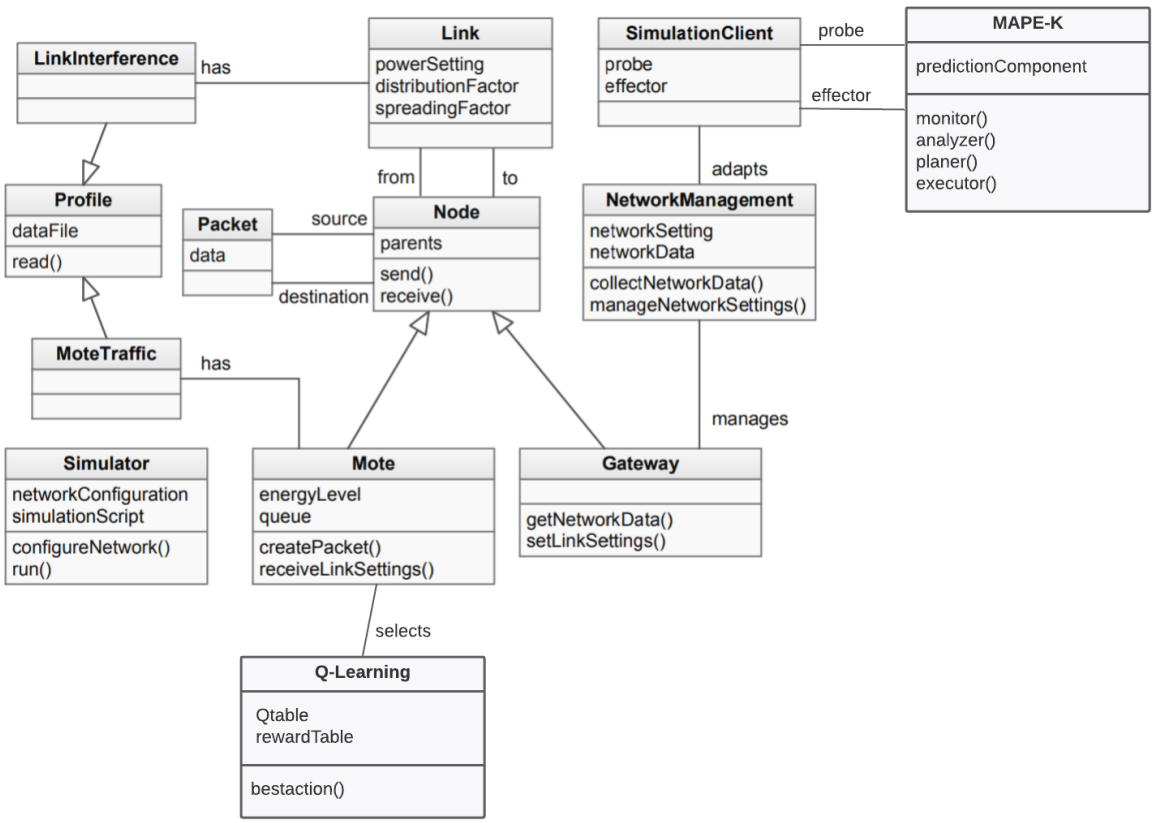


FIGURE 5.3: Structure of the project.

5.5 MAPE-K Pseudo code

Monitor:

```
K.motes = probe.getAllmotes()
invoke Analyzer()
```

Analyzer:

```
foreach(mote in K.motes) {
    foreach(link in mote.links) {
// call the model for the prediction
// Put the link information
//(link Distribution, SNR, Powertransmission)
  // check the result if 1 or 0

    Result=predictPacketloss (Distribution, SNR, Power)
   Result2=predict Energy (Distribution,
       Power, Traffic Load, Queue size)

    if (result==0) {
          adaptationRequired = true;
          addAdaptation(PACKET_LOSS, link, mote)}
    else if(result1==0){
           adaptation Required = true;
          adddaptation(ENERGY, link, mote)

    } else{
    adaptationRequired = false;
   }
} }
if (adaptation Required) invoke Planner
```

Planner:

```
foreach (adaptation in K.Adaptationsteps) {
   if (adaptation == PACKET_LOSS || adaptation == ENERGY ) {
steps.add(new PlanningStep(Step.TRRIGER_QL,MoteId)
  }
```

```
}
invoke Executor
```

Executor:

```
foreach(step in K.planningSteps) {
 if (step.type == TrigerQlearning) {
    effector.trigerQL(MoteId)
}
}
```

5.6 Q-Learning code

Figure 5.4 and **Figure 5.5** presents Q-Leaning code that is divided into two phases: First phase select an action, and the second phase update the Q-table.

```
 // Select one among all possible actions for the current state
 int maxaction;
 int statetest=id;

if(maxQ1(state_id)==0) { // if the values in Qtable different from 0 than select random action

 maxaction= rand.nextInt(possible_action.length);

} else {

if(Math.random() <epsilon) {// choose random value for the exploration process
        maxaction= rand.nextInt(possible_action.length);
    }
    else
    {  // select the best action
        double maxQ1 = maxQ1(state_id);
        maxaction=maxQid1(state_id,maxQ1);

    }
}
```

FIGURE 5.4: Selecting an action.

```
// Using this possible action, consider to go to the next state
// using the Bellman equation to update Q table

double q = Q(state_id, maxaction);
double maxQ = maxQ1(nextState); // bring the q value drom next state
int r = R(state_id, maxaction);

double value = q + alpha * (r + gamma * maxQ - q);

setQ(state_id, maxaction, value); // update Qtable
```

FIGURE 5.5: Updating the Q-table.

5.7 Building Deep Learning model

For building a deep learning model, we need to define the layers (Input, Hidden, and Output). Here, we used Python library named Keras to define layers automatically.

5.7.1 Packet loss model

We will focus on defining the **input layer**. This can be specified while creating the first layer with the input dim argument and setting it to **3** as our input settings.

Next, define the number of **hidden layer(s)** along with the number of neurons and activation functions. The right number can be achieved by going through multiple iterations. Higher the number, more complex is your model. To start with, we are using two hidden layers. One has **64** neurons and the other has **32** neurons with the same activation function - **"softmax"**.

The softmax function squashes the outputs of each unit to be between 0 and 1, just like a sigmoid function. The output of the softmax function is equivalent to a categorical probability. It expresses the probability that any of the classes are true.

Finally, we need to define the **output layer** with **2** neurons to predict the if an adaptation required or not.

By using keras we can resume a lot of work in some of python codes as bellow :

```
# define a function to build the keras model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(64, input_dim=3, kernel_initializer='normal', activation='softmax'))
    model.add(Dropout(0.5))
    model.add(Dense(32, kernel_initializer='normal', activation='softmax'))
    model.add(Dropout(0.2))
    model.add(Dense(2, activation='softmax'))

    # compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

FIGURE 5.6: Creating model with keras.

As we see in **Figure 5.6** We configured the model for training. We set the optimizer to change the weights and biases, and the loss function and metric to evaluate the model's performance. Here, we used "Adam" as the optimizer, "accuracy" as the loss metric. Depending on the type of problem we are solving, we can change our loss and metrics. For binary classification, we used "categorical crossentropy" as a loss function.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 66
=================================================================
Total params: 2,402
Trainable params: 2,402
Non-trainable params: 0
```

FIGURE 5.7: Model description.

The final step of model building is fitting the model on the training data-set. After we split the data between trainable and testable data(trainable=0.8 and testable=0.2), we need to provide the number of training iterations, i.e. epochs. Here, we have taken 30 epochs.

After building our deep learning model using Keras, we got a learned model after 30 Epochs. The **Figure 5.8** below shows the Packet loss model results.

	precision	recall	f1-score	support
0	1.00	0.86	0.93	27739
1	0.93	1.00	0.97	53674
accuracy			0.95	81413
macro avg	0.97	0.93	0.95	81413
weighted avg	0.96	0.95	0.95	81413

FIGURE 5.8: Packet loss Model results.

5.7.2 Energy consumption model

As for Energy consumption model , we used the same structure as Packet loss model. We remain the same Deep Learning Model with slightly changing of input Layer (same hidden layers and output layer) from 3 settings to 4 settings.
Figure 5.9 shows the Energy consumption model results.

	precision	recall	f1-score	support
0	0.99	0.99	0.99	22265
1	1.00	0.99	1.00	42362
accuracy			0.99	64627
macro avg	0.99	0.99	0.99	64627
weighted avg	0.99	0.99	0.99	64627

FIGURE 5.9: Energy consumption Model results.

The results of prediction from our learned model are:

Accuracy of adaptation required (0) = 99 % .

Accuracy of no adaptation required (1) = 100 % .

As for the model accuracy is 99,383570857048035 % .

Note:

the accuracy of the two models was gained by using data-set from simulations process (**Figure4.5, Figure 4.3**). This is why we have high accuracy in both models.

5.8 Obtained Results

The interface of the Simulator is shown in **Figure 5.10**; also the simulation results for the packet loss and energy consumption are presented as well. The Start Adaptation Button executes the simulation process with implementation of our adaptation approach as against of Start Simulation button which run without it.

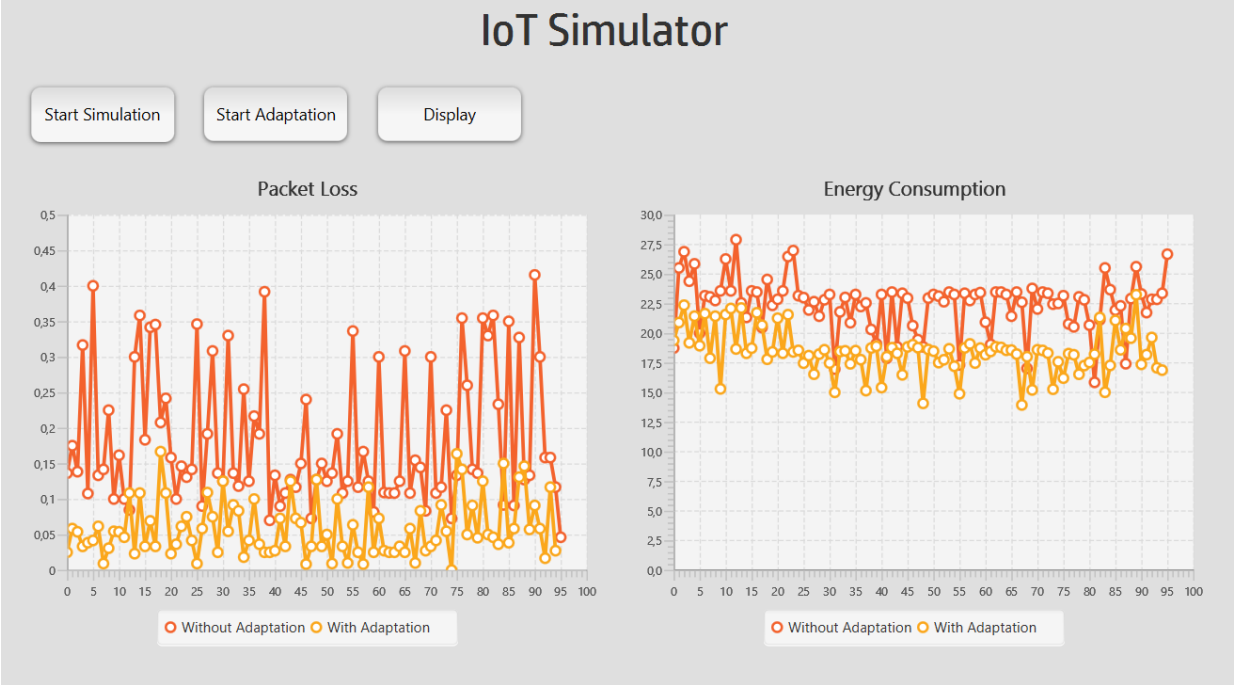


FIGURE 5.10: Simulation results.

Figures 5.11 and **Figure 5.12** show the state of the IoT network before and after the adaptation process.

5.8.1 Scenario 1

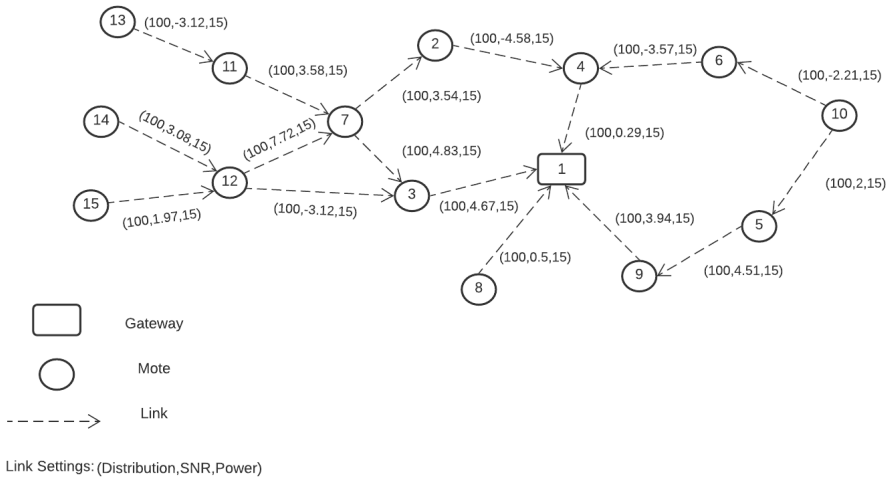


FIGURE 5.11: State of IoT network before the adaptation.

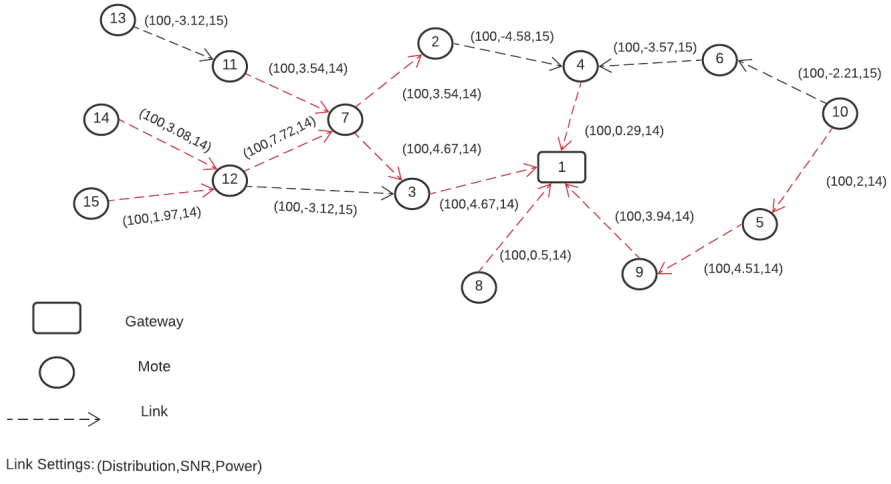


FIGURE 5.12: State of IoT network after the adaptation.

Now let us discuss the adaptation actions occurred that was generated by our approach: The red links represent the adaptation actions that were occurred. In this scenario as the transmission power at its limit, the aim here is to minimize energy consumption by adapting the transmission power of motes (decrease power for links with lower levels of messages lost).

5.8.2 Scenario 2

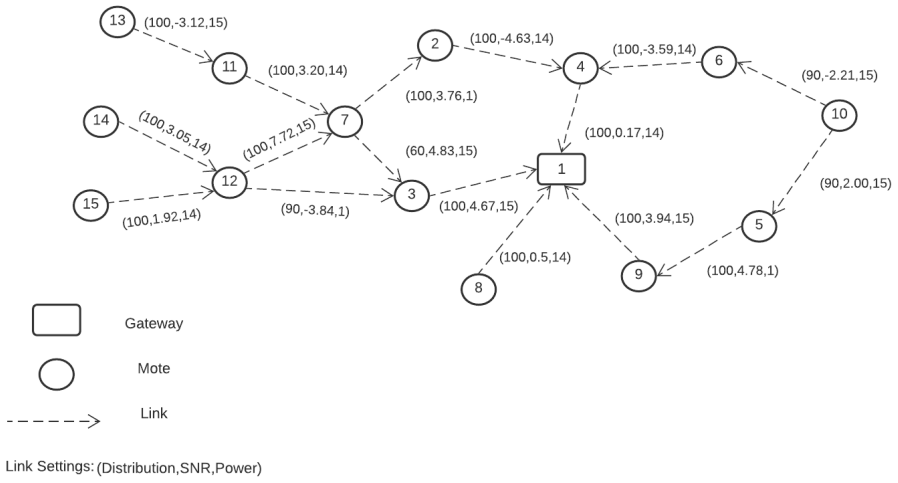


FIGURE 5.13: State of IoT network before the adaptation.

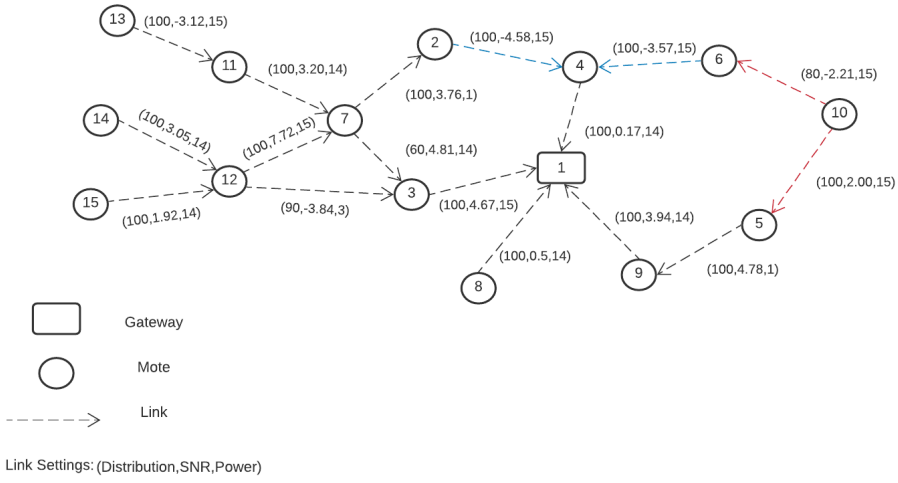


FIGURE 5.14: State of IoT network after the adaptation.

For this scenario, we have two categories red links and blue links. The blue links represent the increase in transmission power of a mote for a link (sets the power setting of mote 6 and mote 2 from 14 to 15). As for the red links, the approach adapts the distribution of packets sent over a link to a parent of a mote (sets the distribution of packets send by mote 10 over the link to mote 6 from 90 to 80 and from mote 10 to mote 5 the distribution changed from 90 to 100). The aim for these adaptation actions is to reduce packet loss over the network.

5.9 Evaluation

The evaluation process is done by comparing our approach and MARTAS approach which is an Architecture-based adaptation approach that combines formal models with statistical techniques at runtime to make adaptation decisions.
We evaluated the packet loss and energy consumption of the IoT network for both approaches over a period of 24 h.

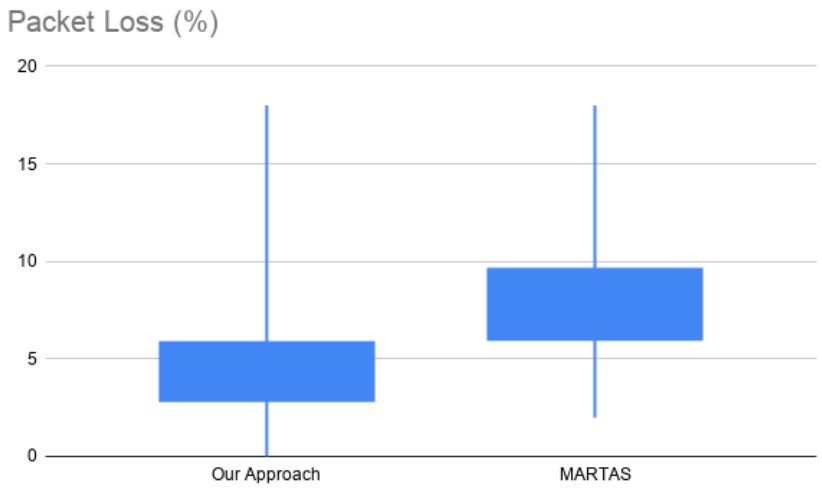


FIGURE 5.15: Test results for Our Approach versus MARTAS in Packet loss.

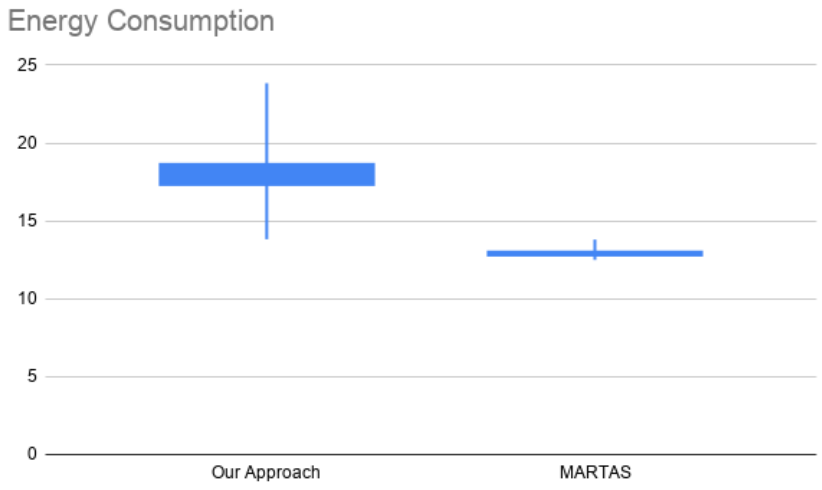


FIGURE 5.16: Test results for Our Approach versus MARTAS in Energy Consumption.

The boxplots show that the Packet loss of our system is significantly better compared to the MARTAS approach (mean of 5.36% for Our Approach versus 9.11% for MARTAS).as For the average energy consumption MARTAS is substantially better optimizing compared for our system (mean of 18 versus 12.70). This refers to our interest in trying to reduce packet loss in the network while minimizing the energy consumption in this process.

5.10 Conclusion

To validate the design of a system, it must be implemented using the appropriate tools. In this chapter we have presented some details concerning the realization of our QoS architectural based adaptation in IoT system with an integration of Machine Learning algorithms (Q-Learning and Deep learning).

From the results obtained from the implementation process, we can confirm that the results are favorable and that Architecture based adaptation is very effective in self-configuring and self-adapting internet of things system.

Conclusion

Internet of Things (IoT) are capable of connecting various smart things together with the internet making life more safe and comfortable by reducing the costs and risk involved. Considering the importance of IoT in the day to day life, QoS metrics in IoT system needs to be defined and placed at priority.

In order to achieve the QoS requirements in IoT, end nodes need to adapt dynamically their settings. In our work, we put packet loss to ensure reliability and energy consumption in the spotlight, two key qualities for IoT systems with battery-powered mobile end nodes.

To automate the management of Internet-of-Things (IoT), this approach used architecture based adaptation on IoT system with a feedback loop on top of it. This MAPE-K loop employs runtime models and used its data to adapt the system to ensure the required goals.

To achieve the adaptation goals, first we implemented Q-Learning algorithm to each mote in the system to take an optimal action which leads to provide the required goals. Also, we used Deep learning to enrich the Knowledge and runtime models with motes link data to estimate the deviation of adaptation goals.

All this data will finally reach the MAPE-K loop to handle uncertainties of the system (The interference of the network links in IoT system or the traffic generated by the motes) And generate the best adaptation action to provide the quality goals.

The main benefits of our approach that automate the management of IoT systems are handle changes whenever they occur faster and longer system lifetimes. As for the risks are the need for predictive components that have perspective view of the whole IoT network by exploiting different machine learning algorithms. Also, using several adaptation options with different configurations to maintain flexible adaptation goals at runtime changes.

In Our future Work, we plan to study more adaptation problems with more different types of uncertainties, such as uncertainties of security attacks, mobility of motes, etc... Finally, we plan to study how self-adaptation can be applied in systems that require multiple feedback loops that need to work together to solve an adaptation problem.

Bibliography

[1] M. Rouse. (Feb. 2020). What is iot (internet of things) and how does it work?, [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT.

[2] P. Sethi and S. R. Sarangi, "Internet of things: Architectures, protocols, and applications", *Journal of Electrical and Computer Engineering*, vol. 2017, no. 25, Jan. 2017. [Online]. Available: https://www.hindawi.com/journals/jece/2017/9324035/.

[3] H. Muccini, R. Spalazzese, M. T. Moghaddam, and M. Sharaf, "Self-adaptive iot architectures: An emergency handling case study", in *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, 2018, pp. 1–6.

[4] M. Singh and G. Baranwal, "Quality of service (qos) in internet of things", in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, 2018, pp. 1–6.

[5] R. C. Bhaddurgatte* and V. K. BP, "A review: Qos architecture and implementations in iot environment", *Research Reviews: Journal of Engineering and Technology*, pp. 6–12, Nov. 2016.

[6] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Di Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle. (2009). Software engineering for self-adaptive systems: A research roadmap. B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds., [Online]. Available: https://doi.org/10.1007/978-3-642-02161-9_1.

[7] Y. Brun, G. D. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, "Engineering self-adaptive systems through feedback loops", in *Software engineering for self-adaptive systems*, Springer, 2009, pp. 48–70.

[8] J. Andersson, R. De Lemos, S. Malek, and D. Weyns, "Reflecting on self-adaptive software systems", in *2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, IEEE, 2009, pp. 38–47.

[9] D. Garlan, B. Schmerl, and S.-W. Cheng, "Software architecture-based self-adaptation", *Autonomic Computing and Networking*, pp. 31–55, Apr. 2009.

[10] W. Danny, *Engineering Self-Adaptive Software Systems – An Organized Tour*. Sep. 2018.

[11] "Autonomic computing toolkit user's guide", *IBM Systems Journal*, Feb. 2004. [Online]. Available: `https://www.ibm.com/developerworks/autonomic/books/fpu0mst.htm#Header_3`.

[12] C. Shang-Wen, G. David, and S. Bradley, "Stitch: A language for architecture-based self-adaptation", vol. 85, no. 12, pp. 2860–2875, Dec. 2012.

[13] B. S. David Garlan and S.-W. Chengn, "Software architecture-based self-adaptation", *Autonomic Computing and Networking*, 2009.

[14] (May 2020). What is machine learning? a definition, [Online]. Available: `https://expertsystem.com/machine-learning-definition/`.

[15] $\text{eccentric}_{d}ata_{s}cientist$. (Feb. 2019). Supervised machine learning, [Online]. Available: `https://medium.com/@or_1_eq_1/supervised-machine-learning-7039d90ac8c1`.

[16] (). Unsupervised machine learning: What is, algorithms, example, [Online]. Available: `https://www.guru99.com/unsupervised-machine-learning.html`.

[17] Q. Wang and Z. Zhongli, "Reinforcement learning model,algorithms and its application", *International Conference on Mechatronic Science, Electric Engineering and Computer*, pp. 11 437–1146, Aug. 2011.

[18] R. S. Sutton and A. G. Barto, "Reinforcement learning:an introduction second edition", 2015.

[19] P. Poupart. (May 2018). Cs885 reinforcement learninglecture 9, [Online]. Available: `https://www.coursehero.com/file/43960726/cs885-lecture9pdf/`.

[20] M. Corazza and A. Sangalli, "Q-learning and sarsa: A comparison between two intelligent stochastic control approaches for financial trading", *Microeconomics: General Equilibrium Disequilibrium Models of Financial Markets eJournal*, 2015.

[21] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction", 1998.

[22] K. kumar Chanda. (Apr. 2020). Q-learning in python, [Online]. Available: `https://www.geeksforgeeks.org/q-learning-in-python/`.

[23] L. Mao. (Mar. 2019). On-policy vs off-policy in reinforcement learning, [Online]. Available: `https://leimao.github.io/blog/RL-On-Policy-VS-Off-Policy/`.

[24] K. Nygren, "Stock prediction – a neural network approach", *Master's Thesis, Royal Institute of Technology, KTH, Stockholm*, Mar. 2004.

[25] J. Fröhlich. (2004). Propagation function, [Online]. Available: `https://www.nnwj.de/propagation-function-term.html`.

[26] P. Radhakrishnan. (Aug. 2017). What are hyperparameters ? and how to tune the hyperparameters in a deep neural network?, [Online]. Available: `https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a`.

[27] D. Weyns, M. U. Iftikhar, D. Hughes, and N. Matthys, "Applying architecture-based adaptation to automate the management of internet-of-things", in *European Conference on Software Architecture*, Springer, 2018, pp. 49–67.

[28] B. Robbe and W. Danny, "A qos-aware adaptive mobility handling approach for lora-based iot systems", pp. 130–139, Sep. 2018.

[29] M. K. Kunal Shah, "Distributed independent reinforcement learning (dirl) approach to resource management in wireless sensor networks", *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, pp. 1–9, Nov. 2007.

[30] K. V. Henry Muccini, "A machine learning-driven approach for proactive decision making in adaptive architectures", *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 242–245, Mar. 2019.

[31] C. Shyalika. (Nov. 2019). A beginners guide to q-learning, [Online]. Available: `https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c`.

[32] V. Beal. (2020). What is java programming language?, [Online]. Available: https://www.webopedia.com/TERM/J/Java.html.

[33] M. Summerfield, "Programming in python 3: A complete introduction to the python language", 2010.

[34] (2020). Eclipse - overview, [Online]. Available: https://www.tutorialspoint.com/eclipse/eclipse_overview.htm.

[35] D. K. Taft. (Oct. 2010). Jetbrains strikes python developers with pycharm 1.0 ide, [Online]. Available: https://www.eweek.com/development/jetbrains-strikes-python-developers-with-pycharm-1.0-ide.

[36] (2020). Javafx - overview, [Online]. Available: https://www.tutorialspoint.com/javafx/javafx_overview.htm.

[37] (2020). Javafx scene builder, [Online]. Available: https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html.

[38] (). About keras, [Online]. Available: https://keras.io/about/.

[39] M. U. Iftikhar, G. S. Ramachandran, P. Bollansée, D. Weyns, and D. Hughes, "Deltaiot: A self-adaptive internet of things exemplar", in *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, IEEE, 2017, pp. 76–82.