

Internet of Things

Rajkumar Buyya
Lalit Garg
Giancarlo Fortino
Sanjay Misra *Editors*

New Frontiers in Cloud Computing and Internet of Things

 Springer

Internet of Things

Technology, Communications and Computing

Series Editors

Giancarlo Fortino, University of Calabria, Rende (CS), Italy

Antonio Liotta, Free University of Bozen-Bolzano, Bolzano, Italy

The series Internet of Things - Technologies, Communications and Computing publishes new developments and advances in the various areas of the different facets of the Internet of Things. The intent is to cover technology (smart devices, wireless sensors, systems), communications (networks and protocols) and computing (theory, middleware and applications) of the Internet of Things, as embedded in the fields of engineering, computer science, life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in the Internet of Things research and development area, spanning the areas of wireless sensor networks, autonomic networking, network protocol, agent-based computing, artificial intelligence, self organizing systems, multi-sensor data fusion, smart objects, and hybrid intelligent systems.

Indexing: *Internet of Things* is covered by Scopus and Ei-Compendex **

Rajkumar Buyya • Lalit Garg • Giancarlo Fortino •
Sanjay Misra

Editors

New Frontiers in Cloud Computing and Internet of Things



Springer

Editors

Rajkumar Buyya
School of Computing and Information
Systems
University of Melbourne
Parkville, VIC, Australia

Giancarlo Fortino
University of Calabria
Rende, Cosenza, Italy

Lalit Garg
Information and Communication
Technology
University of Malta
Msida, Malta

Sanjay Misra 
Department of Computer Science and
Communication
Østfold University College
Halden, Norway

ISSN 2199-1073

Internet of Things

ISBN 978-3-031-05527-0

<https://doi.org/10.1007/978-3-031-05528-7>

ISSN 2199-1081 (electronic)

ISBN 978-3-031-05528-7 (eBook)

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2022, corrected publication 2022

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

In the post-COVID-19 world, organisations are implementing innovative ways to continue business by facilitating work from home and avoiding contact using the Internet of Things (IoT) for remote monitoring. Schools and other educational institutions are implementing e-learning, and hospitals and health organisations are implementing e-medicine and remote patient monitoring via the Internet of Medical Things (IoMT). Online shopping, contactless delivery and virtual social events are becoming popular. Authorities are using sensors for contact tracing, traffic control and safety. Secure and reliable cloud infrastructure availability made these applications possible.

Cloud computing and IoT are becoming ever more essential in our lives. Cloud computing provides relevant, on-demand and flexible network connectivity, allowing users to contribute computing resources that integrate dynamic data from many sources. On the other hand, IoT enables thousands of devices to connect and communicate with one another, exchanging information, knowledge and data to improve the quality of our lives. Computing tasks are carried out on fog or cloud machines (physical or virtual) capable of robust data analytics, deep learning and blockchain mining.

However, IoT and cloud-based applications face security, privacy, reliability, scalability, quality of service, sustainability, performance, user experience, energy efficiency, power requirements, management, integration, compatibility, interoperability, vendor lock-in, maintenance, compliance and ethical issues.

This book aims to present the latest developments in cloud computing and IoT paradigms, including technology, infrastructure, architecture, protocols, and practical applications and solutions to the above challenges. The book also identifies potential future research directions and new upcoming issues. We expect the book to serve as a reference for practitioners, developers, policymakers, academicians, researchers and students.

The Book Organisation

There is a total of 14 chapters authored by experts in cloud computing and IoT from various countries. The book contents are organised in two parts:

- I. Cloud Computing
- II. Internet of Things

Cloud Computing

Part I contains eight chapters describing novel cloud computing applications, fundamental concepts, architectures, algorithms, issues and solutions. The first chapter is the introductory chapter highlighting and critically analysing the recent trends and directions in cloud computing and IoT. The second chapter critically reviews the literature on energy-efficient cloud resource management solutions in cloud environments and proposes a complete taxonomy for energy-efficient solutions. The third chapter offers a multi-objective stochastic release time aware dynamic virtual machine consolidation (SRTDVMC) algorithm to minimise energy consumption and virtual machine migration in the presence of numerous physical machines and heterogeneous workload and evaluates its performance through CloudSim. The fourth chapter proposes and implements an optimisation algorithm called ‘Genetically Enhanced Shuffling Frog Leaping Algorithm’ (GESFLA) for the VM allocation and tasks execution to minimise power consumption, run application costs and optimise the resource usage.

The fifth chapter proposes a heuristic-based algorithm to solve the scheduling problem with minimised inter-node communication and presents a prototype scheduler named D-Storm, extending the original Apache Storm framework into a self-adaptive architecture. The sixth chapter proposes a novel concept of serverless edge computing and analyses the performance of various serverless platforms (OpenFaaS, AWS Greengrass, Apache OpenWhisk) for the single-board computers (SBCs) set up. The seventh chapter of this part proposes an iterative iTree Learning (ITL)-based algorithm to handle high-dimensional data and detect anomalies. Finally, the eighth chapter offers a digital twin of a cloud data centre to help administrators better monitor and manage their data centres.

Internet of Things (IoT)

Part II comprises seven chapters on advanced IoT concepts, novel IoT architectures, protocols, algorithms, applications, issues, and solutions. The ninth chapter critically reviews the state-of-the-art IoT protocols, architectures, and their elements and models available in the literature for various industrial applications. The tenth

chapter analyses the latest developments in IoT and their applications from the sustainability perspective and recommends strategies to ensure the IoT role in global sustainable development. The eleventh chapter looks over IoT and cloud computing applications and prospects in the post-COVID-19 world in various sectors such as education, entertainment, transportation, manufacturing, healthcare and agriculture.

The twelfth chapter proposes an IoT and cloud-based architecture using RFID to monitor students' academic progress by observing their on-campus activities based on the location visited. The thirteenth chapter describes a novel IoT application in tracking network faults and their impact on the overcurrent protection scheme of the distribution network to ensure the safety of equipment and personnel. The fourteenth chapter investigates the application of IoT and cloud computing in heart disease prediction by implementing a Naïve Bayes classifier model. Finally, the fifteenth chapter critically reviews the state-of-the-art progress in intelligent medical IoT devices for monitoring patients' health in real-time.

Parkville, VIC, Australia
Msida, Malta
Rende, Cosenza, Italy
Halden, Norway

Rajkumar Buyya
Lalit Garg
Giancarlo Fortino
Sanjay Misra

Acknowledgements

First and foremost, we take this opportunity to appreciate all the contributing authors for their time, effort and understanding during the preparation of the book. They are the real champions who made this book possible.

Raj would like to thank his family members, especially his wife, Smrithi, and daughters, Soumya and Radha Buyya, for their love, understanding and support during the preparation of the book.

Lalit thanks his mother, Dr Urmila Garg; father, Dr LilaDhar Garg; wife, Ramandeep; and daughter, Rhythm.

Giancarlo thanks his family.

Sanjay dedicates this book in memory of his beloved late father Shri Om Prakash Misra who expired on 24 December 2019.

Finally, we would like to thank the staff at Springer Nature, particularly Mary James and Arun Pandian KJ, for brilliantly managing the book project. They were terrific to work with.

Parkville, VIC, Australia
Msida, Malta
Rende, Cosenza, Italy
Halden, Norway

Rajkumar Buyya
Lalit Garg
Giancarlo Fortino
Sanjay Misra

Contents

Part I Cloud Computing

1	Cloud Computing and Internet of Things: Recent Trends and Directions	3
	Mohammad Goudarzi, Shashikant Ilager, and Rajkumar Buyya	
2	Recent Advances in Energy-Efficient Resource Management Techniques in Cloud Computing Environments	31
	Niloofar Gholipour, Ehsan Arianyan, and Rajkumar Buyya	
3	Multi-objective Dynamic Virtual Machine Consolidation Algorithm for Cloud Data Centers with Highly Energy Proportional Servers and Heterogeneous Workload	69
	Md Anit Khan, Andrew P. Paplinski, Abdul Malik Khan, Manzur Murshed, and Rajkumar Buyya	
4	Energy-Efficient Resource Management of Virtual Machine in Cloud Infrastructure	107
	H. Priyanka and Mary Cherian	
5	Dynamic Resource-Efficient Scheduling in Data Stream Management Systems Deployed on Computing Clouds	133
	Xunyun Liu, Yufei Lin, and Rajkumar Buyya	
6	Serverless Platforms on the Edge: A Performance Analysis	165
	Hamza Javed, Adel N. Toosi, and Mohammad S. Aslanpour	
7	ITL: An Isolation-Tree-Based Learning of Features for Anomaly Detection in Networked Systems	185
	Sara Kardani Moghaddam, Rajkumar Buyya, and Kotagiri Ramamohanarao	
8	Digital Twin of a Cloud Data Centre: An OpenStack Cluster Visualisation	209
	Sheridan Gomes, Adel N. Toosi, Barrett Ens	

Part II Internet of Things

9 Industrial IoT Technologies and Protocols	229
Rahul Devkar, Princy Randhawa, and Mahipal Bukya	
10 IoT for Sustainability	253
Brian Davison	
11 Applications of IoT and Cloud Computing: A COVID-19 Disaster Perspective	287
Kshitij Dhyani, Thejineaswar Guhan, Prajwal Gupta, Saransh Bhachawat, Ganapathy Pattukandan Ganapathy, and Kathiravan Srinivasan	
12 Analytics of IoT-Based System for Monitoring Students' Progress in Educational Environment.....	323
Moses Kazeem Abiodun, Joseph Bamidele Awotunde, Emmanuel Abidemi Adeniyi, Roseline Oluwaseun Ogundokun, and Sanjay Misra	
13 Power System Protection on Smart Grid Monitoring Faults in the Distribution Network via IoT	343
Owolabi Peter Bango, Sanjay Misra, Oluranti Jonathan, and Ravin Ahuja	
14 Medical Data Analysis for IoT-Based Datasets in the Cloud Using Naïve Bayes Classifier for Prediction of Heart Disease	365
Babatunde Gbadamosi, Roseline Oluwaseun Ogundokun, Emmanuel Abidemi Adeniyi, Sanjay Misra, and Nkiruka Francisca Stephens	
15 The Internet of Things in Healthcare: Benefits, Use Cases, and Major Evolutions	387
Raj Shree, Ashwani Kant Shukla, Ravi Prakash Pandey, Vivek Shukla, and K. V. Arya	
Correction to: Recent Advances in Energy-Efficient Resource Management Techniques in Cloud Computing Environments	C1
Index	407

Part I

Cloud Computing

Chapter 1

Cloud Computing and Internet of Things: Recent Trends and Directions



Mohammad Goudarzi, Shashikant Ilager, and Rajkumar Buyya

Contents

1.1	Introduction	3
1.1.1	Cloud Computing	4
1.1.2	Internet of Things (IoT)	6
1.2	Key Cloud Technologies and Services	8
1.2.1	Infrastructure as a Service (IaaS)	9
1.2.2	Container as a Service (CaaS)	10
1.2.3	Platform as a Service (PaaS)	11
1.2.4	Function as a Service (FaaS)	12
1.2.5	Software as a Service (SaaS)	13
1.3	Key IoT Technologies and Applications	14
1.3.1	Things	15
1.3.2	Communication Protocols	15
1.3.3	IoT Frameworks	19
1.3.4	Deployment Models	20
1.3.5	Applications	22
1.4	Modeling and Simulation Toolkits	22
1.5	Open Challenges	24
1.6	Summary	25
	References	26

1.1 Introduction

The twenty-first-century digital infrastructure and applications are driven by Cloud and Internet of Things (IoT) technologies. Applications built using IoT and Cloud computing technologies have become ubiquitous and influence our modern digital

M. Goudarzi · R. Buyya (✉)

CLOUDS Lab, School of Computing and Information Systems, The University of Melbourne,

Parkville, VIC, Australia

e-mail: rbuyya@unimelb.edu.au

S. Ilager

Technische Universität (TU) Wien, Vienna, Austria

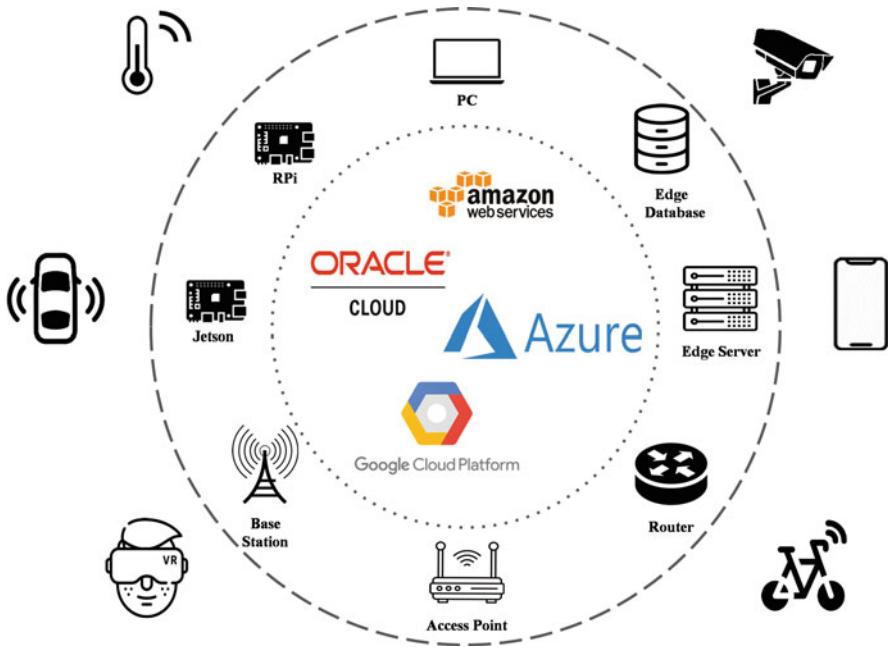


Fig. 1.1 A high-level view of contemporary distributed system involving IoT, Cloud and Edge/Fog components enabling various applications

society in every aspect. The advancement in high-speed networking technologies, pervasive computing devices, and IoT-based sensor networks has resulted in unprecedented growth in data generation. Cloud computing and emerging paradigms such as Edge/Fog computing provide infrastructures and tools required to store, process, and analyze this massive amount of data.

The development in computing and networking technologies over the last many decades are attributed to underlying technologies of Cloud computing and IoT. Cloud and Edge/Fog computing enable IoT-based applications such as smart cities, emergency healthcare applications, and autonomous vehicles, as illustrated in Fig. 1.1. Therefore, a fundamental understanding of technologies associated with Cloud and IoT and the capabilities and limitations are crucial. Therefore, in this chapter, we discuss Cloud and IoT technologies and open challenges associated with them.

1.1.1 *Cloud Computing*

Cloud computing has seen tremendous growth in recent years. The transition from ownership-based on-premises IT Infrastructure to subscription-based Cloud has changed the way computing services are delivered to end-users. Cloud computing's

fundamental principle is to provide computing resources as utility services (similar to water and electricity) [1]. Based on actual resource usage, it offers on-demand access to elastic resources with a pay-as-you-go model. This unique and flexible service delivery model ensures that individuals and businesses can easily access required computing services.

Cloud computing services are traditionally categorized into three types. First, the Infrastructure as a Service (IaaS) model offers computing, storage, and networking resources either in the virtual or physical form. Second, the Platform as a Service (PaaS) model offers tools for rapid application development and deployment such as middleware platforms, Application Programming Interfaces (APIs), and Software Development Kits (SDKs). Finally, the Software as a Service (SaaS) model offers direct access to application software to the users, and the Software is developed and managed by service providers completely. In addition to these, recently new service categories have been added in the Cloud computing paradigm such as Container as a Service (CaaS) and Function as a Service (FaaS) to match the technology advancement and modern application requirements.

The rapid growth in digital services, Internet of Things (IoT), Industry 4.0, and 5G-based application scenarios are creating a massive demand for Cloud services. According to Gartner, by 2022, 60% of organizations will use external Cloud service providers [2], and by 2024, Cloud computing alone accounts for 14.2% of total global IT spending [3]. Clouds have become application's back-end infrastructures for essential resources (compute, storage, and network) for these modern IT services. Along with remote Clouds, recently, Cloud services have been delivered from the edge of the network to satisfy Quality of Service (QoS) requirements for latency-sensitive applications such as autonomous vehicles, emergency healthcare services [4]. Cloud computing uses massive network-based infrastructures, mainly Data Centers (DCs), the core and backbone infrastructure in this networked system. By default, Cloud workloads are always-on, required to provide 24×7 access to deployed services. For instance, the Google search engine is expected to achieve an almost 100% availability rate [5]. Similarly, Amazon Web Services (AWS) delivers thousands of Elastic Compute (EC2) instances created in a day through its automated APIs [6].

To cater to Cloud services' demand, major Cloud service providers such as AWS, Microsoft Azure, Google Cloud, Oracle Cloud, and IBM Clouds are deploying many hyper-scale DCs in multiple regions worldwide. Many Cloud service providers are increasingly adopting multi-cloud solutions to increase the reliability and availability of Cloud services they offer. DCs have seen huge growth both in number and size. There are over eight million DCs globally, from private small-scale to hyper-scale DCs, and they are estimated to grow rapidly at 12% annually [7]. As their numbers and size grow, it creates new challenges in managing security and providing uninterrupted power to these Cloud DCs [8].

The emergence of the Internet of Things (IoT), diverse mobile applications, smart grids, innovative industries, and smart cities has resulted in massive data generation. Thus, increasing the demand for computing resources to process this data and derive valuable insights for users and businesses [9, 10]. Besides, new application and

execution models like microservices and serverless or FaaS computing [11] are becoming mainstream, significantly reducing the complexities in the design and deployment of software components. On the other hand, this increased connectivity and heterogeneous workloads demand distinct QoS levels to satisfy their application requirements [12, 13]. These developments have led to the building of hyper-scale DCs and complex multi-tier computing infrastructures.

While the fundamental infrastructure capabilities and the number of Cloud services offered by providers are rapidly increasing, new challenges are also emerging that should be addressed diligently for the continued success of Cloud computing technology. Recent advancements have also witnessed Cloud-like services extending to the edge of the network and embedding computing services within network infrastructures, namely, Edge/Fog Computing [14]. The IoT-based latency-sensitive applications mainly drive this need for computing services at the edge of the network since remote Clouds have significant latency due to their geographical distance to the remote Cloud DCs. These new paradigms significantly benefit a few special classes of applications, especially supporting latency-sensitive IoT-based applications. However, we should simultaneously address issues with both Edge/Fog and Cloud infrastructures, including increasing infrastructure efficiency, avoiding resource wastage, reducing power consumption, new pricing and economic models, and managing security.

1.1.2 Internet of Things (IoT)

The Internet of Things (IoT) has become an integral basis of the digital world, thanks to the continuous development of super-cheap and tiny-sized computer chips and ubiquitous access to the Internet. In IoT, “Things” refers to any entities (e.g., smart devices, sensors, human beings) that are context-aware and able to communicate with other entities without any temporal and spatial constraints [15]. Small devices and sensors act as distributed data aggregators with Internet access that forward collected data to the larger computer platforms for processing and permanent storage [16]. This low-cost and distributed paradigm draws a promising future and provides a great opportunity for developers and businesses to form/transition their functionalities to smarter ones.

IoT applications span almost all vital aspects of today’s life, such as healthcare, security, entertainment, transportation, and industrial systems [15, 17, 18]. According to the report from Cisco [19], Norton [20], and Business Insider [21], 15 billion IoT devices will be connected to the Internet by 2023, 21 billion by 2025, and 41 billion by 2027, which will create substantial economic opportunities. Bain & Company [22] estimates the size of the IoT market in 2021 (including hardware, software, systems integrations, and data services) to be around 520 billion U.S. dollars, while Statista [23] and Business Insider [21] expect the size of IoT market will reach to 1 trillion U.S. dollars by 2022 and over 2 trillion U.S. dollars by 2027, accordingly. Among different categories of IoT applications, the fastest growing one

is expected to be the connected vehicles (e.g., navigation, diagnostics) by roughly one-third of market size, according to Cisco [19] and Statista [23]. Also, the second major category is expected to be smart cities (e.g., smart homes) [19].

Considering the ever-increasing number of IoT devices and IoT applications, a tremendous amount of data has been being generated by IoT devices. The statistics depict that 18.3 ZB of data was produced by IoT devices in 2019 while International Data Corporation (IDC) [24] predicts about a 400% increase in upcoming years, which hits 73 ZB of data by 2025. The real power of IoT resides in collecting and analyzing the data circulating in the environment [15]. The processing, storage, and transmission of this gigantic amount of data require special considerations. Inevitably, the Big Data mechanisms and techniques (e.g., data acquisition, filtering, transmission, and analysis) should be adapted and deployed to meet the requirements of the IoT.

Cloud computing is one of the main enablers of IoT that offers on-demand services to process, analyze, and store the data generated from IoT devices in a simplified, scalable, and affordable manner [15, 25]. Recent advances in Cloud computing services such as serverless platforms, FaaS, transactional databases, Machine Learning (ML), and autonomous data warehouses open new horizons in the field of data acquisition and analysis of IoT data [26]. Besides, IoT businesses and companies that deploy their applications and systems on the Cloud can reduce their expenses (e.g., infrastructural and operational costs), which leads to more affordable services and products for the end-users. To illustrate, let's consider the IoT in the healthcare industry. According to Help Net Security [27], the healthcare industry experienced a 14.5% growth in IoT spending in 2020. Also, the healthcare industry is expected to remain as one of the major categories of IoT applications in the future. The e-Health and remote patient monitoring are among the most promising applications of healthcare [26]. These types of applications are directly in touch with the wellness, safety, and convenience of users, especially chronic and elderly patients, that improve their confidence, autonomy, and self-management. Thus, these applications require resources with high availability, scalability, security, and affordability for smooth and efficient execution. Cloud computing satisfies such requirements by providing elastic resources with 24/7 availability and monitoring. Consequently, considering the new advances in Cloud computing and the IoT-enabled healthcare industry, Appinventiv [28] reports that the estimated waiting time of patients has been reduced by 50%, and workforce productivity in the healthcare industry has increased by 57%. In addition, new business models in the healthcare industry have increased by 36%.

Latency-critical and real-time applications (e.g., smart traffic flow control) have necessitated bringing Cloud-like services (e.g., processing, storage) to the edge of the network, called Edge/Fog computing paradigm. In the Edge/Fog computing paradigm, the constituent parts of latency-critical IoT applications can be completely or partially deployed on the Edge/Fog servers, distributed in the vicinity of end-users [14, 29]. Edge/Fog servers are accessible with lower latency and higher data rate compared to Cloud services. Gartner [30] predicts that 75% of enterprise-generated data will be processed at the edge by 2025, which is a big

jump from 10% in 2018. However, the resources of Edge/Fog servers are usually limited compared to Cloud resources. Consequently, Edge/Fog computing does not compete with Cloud computing, but they complement each other to satisfy the diverse requirements of heterogeneous IoT applications and systems. According to the Forrester analyst [31], the Edge-Cloud market had a 50% growth in 2020, proving the dominance of applications requiring different levels of QoS.

While new IoT-enabled business models and applications have been exponentially increasing in recent years, new challenges are also emerging that require constant and precise research and analysis to guarantee the continued success of IoT. With the highly heterogeneous nature of IoT devices, applications, and their required QoS level, hybrid computing platforms are required for satisfying such requirements. Hence, different Cloud service providers have started planning for efficient multi-cloud services alongside extending their services to the edge of the network [32]. Besides, big-tech companies are bringing more computational power to the low-level Edge devices (e.g., Nvidia Jetson Platform, and Google Coral Edge TPU) to empower these devices for running more resource-hungry applications and analysis tools. Simultaneously, softwares such as machine-learning-based analysis tools and container orchestration frameworks are being adapted to run on Edge devices. Although these novel advances open new horizons for IoT, such a highly heterogeneous environment requires fast adaptation of resource management techniques and protocols, considering new hardwares and softwares (e.g., scheduling mechanisms, scalability procedures, efficient power management techniques (especially for the Edge devices)), software frameworks and programming models suiting the IoT and Edge/Fog computing environment's characteristics. In addition, open standards, regulatory policies, and pricing mechanisms for collaboration among Edge service providers and privacy-preserving mechanisms, among others.

1.2 Key Cloud Technologies and Services

The advancement in multiple technologies has enabled the realization of Cloud computing, especially virtualization. Virtualization enables service providers to offer customized virtual resources (compute, network, and storage) on a shared physical machine, simultaneously providing an isolation environment for the user applications [33]. This has enabled service providers to reduce resource fragmentation and increase infrastructure utilization to achieve economic sustainability in offering Cloud services. Virtualization, improved network connectivity and speed, and standardized web services and REST APIs have helped Cloud computing become successful. The AWS is the first public Cloud service provider started with offering Virtual Machines (VMs), now the AWS itself has more than thousands of different services in their offering [6]. Moreover, the public Cloud service providers have rapidly increased, including Microsoft Azure, Google Cloud, Oracle Cloud, Alibaba Cloud, and many others, capturing significant Cloud computing market share.

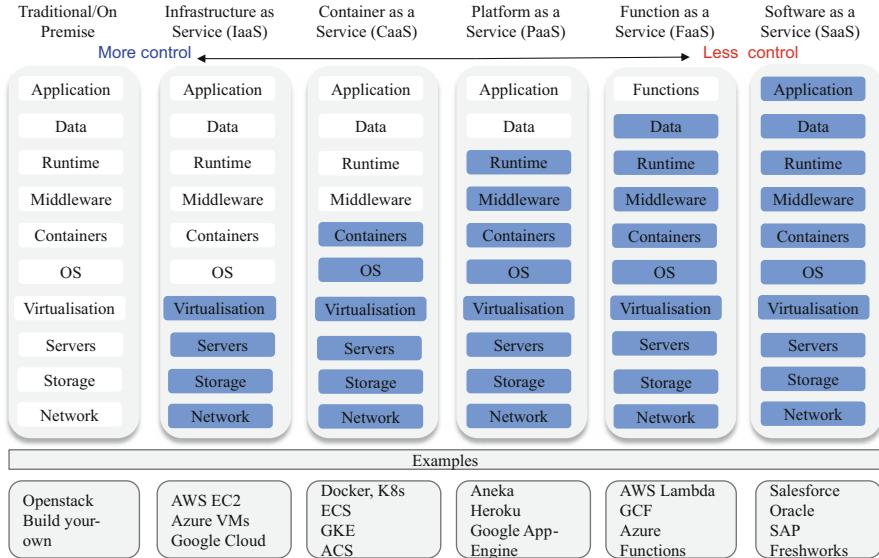


Fig. 1.2 Cloud computing services and their level of manageability

While the main aim of Cloud computing is to provide managed services to users, different services provide different levels of managed services suiting the user and application requirements. Figure 1.2 illustrates the critical services with their respective level of managed services offered by Cloud service providers. In this section, we explore key Cloud technologies categorized into different service models and discuss their characteristics and example technologies associated with them.

1.2.1 *Infrastructure as a Service (IaaS)*

Infrastructure as a Service (IaaS) provides computing resources (compute, network, and storage) for users. Service providers manage the physical infrastructure in this service while Operating Systems (OSs) to application management are left to users. In addition to this, Cloud service providers offer many tools for monitoring and managing subscribed resources, such as autoscaling, failover management, and cost management. Resource virtualization is a primary underlying technology that enables IaaS by providing customized resources to end-users. Hypervisors allow spin up new VMs based on user requirements. Along with computing virtualization, storage and network resources are also virtualized using storage area network and Network Function Virtualization (NFV) technologies [34].

AWS was the first public Cloud service provider to offer IaaS with their AWS Elastic Compute (EC2) service. As Cloud adoption has gained rapid growth in recent years, Cloud service providers have also increased. Microsoft Azure, Google Cloud, Oracle, IBM, Alibaba, and others provide different IaaS services for end-users.

Due to privacy and policy requirements, many organizations prefer to build their own private Cloud infrastructures. The open-source IaaS frameworks such as OpenStack provide middleware platforms to build private Cloud [35]. It provides virtualization, networking, storage, DNS, and other essential services to create and manage private Cloud infrastructure easily. Moreover, to accommodate growing business needs and elasticity in Cloud infrastructures, it is common to establish a Hybrid Cloud where confidential data and partial application services are hosted on private Cloud and other components of applications are hosted into public Clouds [36, 37]. The public Cloud service providers offer new IaaS tools to quickly create hybrid Cloud environments that seamlessly deploy and manage applications across private and public Clouds. For instance, *Azure arc* can run and manage Azure data services, and *Azure Stack* helps to develop, deploy, and run Cloud-compatible applications across hybrid Cloud infrastructures.

1.2.2 *Container as a Service (CaaS)*

Containers as a Service (CaaS) aims to provide applications with a virtualized, lightweight, isolated execution environment. Although the VM-based resource allocation provides complete isolation, each VM consumes many resources and increases the boot-up time of new VMs [38, 39]. While VMs enable virtualization of physical machines having their own OSs, the container provides OS-level virtualization and is isolated from one another. A container bundles its software, libraries, dependency packages, and configuration files; they can communicate with each other through well-defined APIs. The container technology has been present for a long time in Linux environments; the recent advancements have made it easier to use and allowed software applications to be broken down into smaller services, known as microservices.

Containers are lightweight compared to VMs since they do not need their own entire OSs. As a result, they have a faster boot-up time compared to VMs. More importantly, the cost of deploying an application with container frameworks significantly reduces since they consume fewer resources compared to VM, and Cloud service providers can serve more users using containers with the same amount of physical resources as opposed to VMs. However, it is essential to note that not all applications would benefit from containers [40]. An enterprise application requiring a significant amount of resources and complete OS functionalities might benefit from VMs. Nevertheless, a large class of applications can be easily containerized and deployed using CaaS offerings from Cloud service providers.

Docker [41] is a popular container management open-source framework, and it implements a Docker engine that enables the composition and creation of containers on Linux environments. Docker also supports Docker swarm for composing and orchestrating multiple containers for application deployment. Kubernetes [42] is another popular open-source framework developed by Google to enable container orchestration in large clusters and provide auto-scaling and application deployment mechanisms. Kubernetes has become a standard container management framework for containerized applications on private and public Cloud clusters.

Many public Cloud service providers offer managed CaaS. Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that provides a secure, reliable, and scalable way to run containerized application services. Microsoft Azure also provides a fully managed Kubernetes container orchestration service that integrates with Azure Active Directory. It allows building microservice applications and managing containers at scale. Google Kubernetes engine also has managed Kubernetes services. Similarly, many other Cloud providers provide a variety of managed container services for users. This lightweight virtualization technology has enabled rapid application development and deployment at a scale using managed services.

1.2.3 Platform as a Service (PaaS)

The Platform as a Service (PaaS) service model provides frameworks, tools, and SDKs to rapidly build and deploy the applications. In this model, physical infrastructures, Oss, and users leverage the Cloud Application Platforms (CAPs) APIs to deploy their application on the Cloud [1]. Users are relieved from procuring and managing the computing infrastructure. For instance, Heroku enables developers to build, run, and operate applications entirely in the Cloud. Google App engine provides fully managed APIs to deploy applications without scaling and managing the underlying resources.

Aneka [43] is another example of a Platform-as-a-Service framework supporting multiple programming models for the rapid development of applications and their deployment on distributed heterogeneous computing resources. Aneka provides a rich set of .NET-based APIs to developers for exploiting resources transparently available in Cloud infrastructure and expressing the business logic of applications by using the preferred programming abstractions. Moreover, system administrators can leverage a collection of tools to monitor and control the deployed infrastructure. The infrastructure can be built upon a public Cloud available to anyone through the Internet or a private Cloud constituted by a set of nodes with restricted access. More importantly, the Aneka can also be used to set up a hybrid Cloud that includes computing nodes from both private and public Clouds. It supports multiple programming models, and developers can choose a suitable model to build the Cloud-native applications according to the application needs; the programming models include task programming, thread programming, and map-reduce programming models.

The PaaS tools are not only limited to application management; they can also extend their services into data management by providing managed data stores and databases where users can leverage APIs and directly create and manage their data without worrying about storage infrastructures. Similarly, it applies to networking resources where PaaS tools help to create secure and dynamic networks for user applications.

1.2.4 Function as a Service (FaaS)

The emergence of the microservice application paradigm has introduced a new type of FaaS Cloud service model. The FaaS models enable applications and developers to design software as a set of functions that communicate with well-defined APIs [44]. This has led to a new computing paradigm popularly known as serverless computing, an architectural pattern of designing Cloud applications without needing to provision explicit resources by users. For instance, users need not lease VMs to deploy their applications. Instead, each function is ephemeral in this model, and once the request is generated to a particular function, the FaaS platform creates a new containerized function that terminates itself after it executes. Serverless computing and FaaS are often interchangeably used. However, we can say that FaaS implements serverless computing, a paradigm in the Cloud where developers can decompose application business logic into a set of functions that are executed in Linux containers fully managed by a FaaS platform.

In serverless computing, the resources (servers) required to execute an application are fully managed by service providers [45, 46]. It is important to note that the function execution state can be returned to the caller or managed externally by saving the results to a persistent database. The resources are temporarily provisioned for the execution of functions whenever they are invoked, and often these functions are containerized and executed to increase boot-up time and reduce resource usages. Due to their unique architectural pattern, the service provider charges the application execution based on the number of invocations to a function and also the duration of that Function. Since each Function in this paradigm is stateless and ephemeral, service providers have a number of restrictions on function characteristics, including its size and maximum execution duration. Hence, decomposing application into meaningful functions suiting the FaaS platforms is necessary. However, not all applications can be converted or designed based on the FaaS model due to application complexities and limitations of FaaS models. Many issues need to be addressed, such as efficient FaaS middlewares and cold-start bottlenecks of functions [44].

The FaaS model enables Continuous Integration (CI) and Continuous Delivery (CD) in application development where developers can independently update or change the individual Function and deploy in runtime without affecting the entire application stack, reducing engineering cost, and accelerating innovation rate in organizations. This Cloud service model is supported by many Cloud service

providers in the market and supports major programming language environments. AWS offers a Lambda service that is tightly integrated with its other Cloud services. Oracle Cloud Functions, Microsoft Azure Functions, and Google Cloud functions also deliver similar services implementing the FaaS model. The open-source platforms such as OpenFaaS, Kubeless, Nucleo, Fission, and IBM's OpenWhisk platforms enable application implementation using the FaaS model and deploy them on private or on-premises infrastructures.

1.2.5 Software as a Service (SaaS)

Software as a Service (SaaS) offers on-demand subscription-based software solutions based on a pay-as-you-go basis from a Cloud service provider. In this service, users remotely access the required services hosted in centralized Cloud DCs, and service providers fully manage the computing stack from hardware to application management [47]. These services are accessed directly from the Internet, either through web applications or mobile applications. The underlying physical infrastructure, middleware platforms, applications software, and data repository are hosted in the Cloud service provider's DC. The Cloud service provider ensures QoS through Service Level Agreements (SLAs) [Citation error]. The SLAs will guarantee the availability and the security of the applications and user data. This service delivery model enables users and organizations to reduce any upfront cost of getting access to required software solutions since all the required elements in the application and computing stack are fully managed by service providers.

All major Cloud service providers, including AWS, Google, Oracle, and Microsoft Azure, offer different SaaS applications. However, the software-centric nature of this service and the ability to leverage infrastructure from existing large Cloud service providers have allowed many small to medium-scale companies to develop and offer specialized SaaS services for users. Generally, SaaS applications can be broadly categorized into two types: Business to Business (B2B) SaaS and Business to Customers (B2C) SaaS.

B2B SaaS: These software systems are broadly developed to serve the business needs of organizations and enterprises. Software systems such as Customer Relation Management (CRMs) and enterprise data management fall into this category. Examples of SaaS service providers include Salesforce, Zoho, Freshworks, and SAP, among many others. For instance, Salesforce is one of the largest SaaS providers that offers CRM and enterprise software focusing on customer service, marketing, and analytics. Similarly, Zoho provides various project management and business automation tools, and Freshworks provides SaaS tools for businesses to support customers through email, phone, website, and social networks. The B2B SaaS applications enable business organizations to adopt digital services rapidly without having expensive in-house technical capabilities to build sophisticated software systems.

B2C SaaS: These software systems are developed to serve the end users' requirements directly. Most major Cloud service providers offer many SaaS applications for users; example services include emails, calendars, offices, social media applications, and personal Cloud storage systems such as Dropbox, Google Drive, and Microsoft OneDrive. Due to the rapid adoption of Cloud technologies from most Internet-based service providers, directly or indirectly, these digital services are part of the SaaS Cloud service delivery model.

1.3 Key IoT Technologies and Applications

In the IoT paradigm, many hardware and software components work closely together to satisfy the requirements of various types of IoT applications. These components are responsible for the contextual perception of the environment as raw data, transferring raw data, management and scheduling of incoming data, processing and storage of raw data, and the interaction with the target environment based on the information (i.e., processed data). According to the data flow, the continuum of the IoT paradigm can be defined as things, communication protocols, frameworks, deployment models, and applications. Figure 1.3 represents an overview of the IoT continuum.

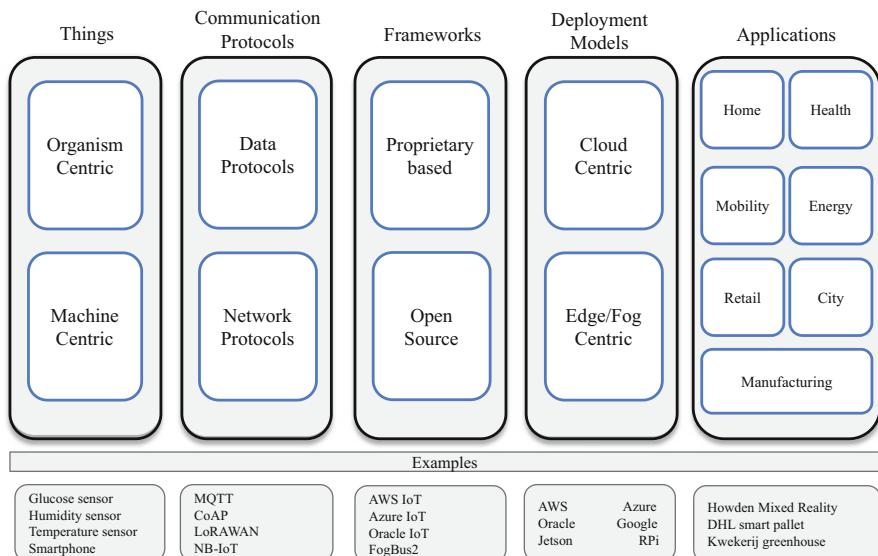


Fig. 1.3 Internet of Things (IoT) technology continuum

1.3.1 *Things*

There are several definitions for Things in the literature. Here, we consider Things as the most generic form and refer to it as any physical objects/entities that can perceive the change in the environment (i.e., embedded with sensor), has a unique identifier, and embedded with a communication module to communicate with other objects/entities without any constraints [15]. Based on the source of data, Things can be categorized into two different categories: (a) organism-centric and (b) machine-centric.

- *Organism-centric*: This category contains any living organisms (i.e., living plants, animals, and humans) that are either embedded with biochips or interact with some types of sensors while they are the main source of data. Some examples of this category include, but are not limited to, human beings embedded with any implantable sensors (e.g., blood pressure, heart rate), animals with tracking sensors, and the sap flow sensors for plants (used as an indicator of transpiration).
- *Machine-centric*: This category contains any artifacts that are not physically integrated with living organisms, such as smart cars, smart buildings, robots, industrial machines, etc.

Once the Things (and their sensors) are set up and connected, they perceive their target environment and forward the obtained data for processing and storage using the integrated communication modules (wired or wireless). Also, Things may receive feedback to act on their target environment based on the processed data.

1.3.2 *Communication Protocols*

In the IoT paradigm, communication protocols are a fundamental part of the IoT technology stack, enabling a large number of IoT devices to communicate and interact together. One of the main differentiating factors between an ordinary and smart device is the capability of communication and interaction among smart devices. Hence, a medium (i.e., a common language) is required for each interaction, provided by the IoT communication protocols. Such protocols play a principal role in the evaluation of the cost and specification of IoT applications based on supporting communication features such as quality, credibility, power consumption, connectivity, and communication range. IoT's communication protocols and standards can be broadly classified into (a) IoT data protocols and (b) Network protocols, described in what follows.

1.3.2.1 IoT Data Protocols

Data protocols can provide end-to-end communication among IoT devices, while the connectivity can be obtained using wired or wireless networks. These protocols sit in the presentation and application layers of the standard OSI network architecture. Some examples of the popular IoT data protocols include:

- Message Queuing Telemetry Transport (MQTT): It is a lightweight protocol that works based on the publish-subscribe messaging model and provides a reliable and low power consumption for devices. The MQTT is widely adapted for IoT devices, and it is proved as one of the best choices for wireless networks with occasional bandwidth constraints and unreliable connections. Facebook has used the MQTT for the online chat of Facebook Messenger.
- Constrained Application Protocol (CoAP): It is a specialized web transfer protocol that works based on the request-response messaging model for constrained nodes and networks in the IoT. The CoAP is designed for machine-to-machine applications such as smart energy and building automation.
- Advanced Message Queuing Protocol (AMQP): It is an open standard protocol for business message passing between applications or organizations that work based on the publish-subscribe messaging model. Despite the security and reliability, the AMQP is not widely used in the IoT, especially for devices with limited memory, due to its heaviness. For financial institutions, the AMQP is used for Business to Business (B2B) transactions.
- Data Distribution Service (DDS): It is considered as the first open international middleware standard directly addressing publish-subscribe communications for real-time and embedded systems. The DDS is a scalable protocol that can be used for a diverse range of devices, from Cloud servers to tiny IoT devices.
- Extensible Messaging and Presence Protocol (XMPP): It is a set of open technologies for instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data, that works based on publish-subscribe and request-response messaging models. It supports near real-time and low-latency communication while it does not provide any QoS guarantee.
- Zero Message Queuing (ZMQ): It is a lightweight messaging library that works based on the request-reply and publish-subscribe messaging model. It extends the standard socket interfaces and provides abstractions of asynchronous message queues, message filtering, etc.

While these IoT data protocols are available to use for any IoT system, selecting the suitable protocol in the development phase is a complex design choice that requires a comprehensive understanding of the requirements of the target IoT system and the features provided by each protocol. Table 1.1 summarizes several important features of these data protocols.

Table 1.1 IoT data protocols comparisons

Protocol name	Transport protocol	Messaging model	Security	QoS support	Sample use case
MQTT	TCP	Publish/subscribe	✓ Medium optional	✓ Medium 3 levels	IoT messaging
CoAP	UDP	Request/reply	✓ Medium optional	✓ Medium 2 levels	Utility field
AMQP	TCP	Publish/subscribe	✓ High optional	✓ Medium 3 levels	Enterprise integration
DDS	TCP, UDP	Publish/subscribe	✓ High optional	✓ High 23 levels	Military, big organizations
XMPP	TCP	Publish/subscribe, request/reply	✓ High mandatory	x	Remote management
ZMQ	UDP	Publish/subscribe, request/reply	✓ High optional	✓ Medium 3 levels	Big organization, IoT messaging

1.3.2.2 IoT Network Protocols

The network protocols belong to lower layers of standard OSI network architecture and are typically used over the Internet to connect devices over the network. Some examples of the popular IoT network protocols include:

- ZigBee: It is one of the most effective low-power communication protocols for IoT which provides high scalability and security. It is widely used for home automation products.
- Z-wave: It is an increasingly popular IoT protocol based on low-power radio frequency communication technology. It provides high scalability, durability, and security, while its price is higher than ZigBee. Similar to ZigBee, it is widely used for home automation products.
- Bluetooth: It is one of the most important communication protocols for short-range communications, which is highly suitable for mobile devices. Bluetooth Low Energy (BLE), with reduced power consumption, is a version of Bluetooth which is used by many IoT companies because of its low latency, responsiveness, scalability, and cross-vendor interoperability.
- Long Range Wide Area Networking (LoRAWAN): This protocol is a low-power and noncellular protocol for long-range wireless communication of IoT devices. LoRAWAN is strongly adopted and deployed, specifically for IoT products to be deployed in areas without cellular service. Besides, it is widely used for smart city and industrial use cases.
- Narrow Band IoT (NB-IoT): It is categorized as a 5G technology, specifically designed for low energy consumption, supporting massive connection density, security, and low latency.
- WiFi: It is the most popular wireless communication protocol for Local Area Networks (LAN), which provides easy deployments, scalability, and high speed.

Table 1.2 provides a summary of important features of these IoT network protocols.

Table 1.2 IoT network protocols comparisons

Protocol name	Frequency	Range	Data rate	Power draw	Topology
ZigBee	2.4 GHz (US)	~100 ft	250 Kbps	Low	Mesh
Z-wave	915 MHz (US)	~100 ft	40 Kbps	Low	Mesh
BLE	2.4 GHz	~300 ft	1–2 Mbps	Low	Mesh, P2P
LoRAWAN	150 MHz–1 GHz	~20 miles	50 Kbps	Low	Star
NB-IoT	<1 GHz	~10 miles	100 Kbps	Low	Star
WiFi	2.4, 5 GHz	~230 ft	7 Gbps	High	Star

1.3.3 IoT Frameworks

Considering the ever-increasing popularity of the IoT paradigm with new applications, design, and integration technologies, many IoT frameworks have been developed to satisfy the requirements of different IoT use cases. We consider frameworks as a suite of software that provides a high-level abstraction, including management, security, interoperability, and flexibility of services, systems, and devices [48]. Some of these frameworks are proprietary-based such as AWS IoT and Microsoft Azure IoT, while other frameworks are fully open-source such as Kaa [49], ThingSpeak [50], and FogBus2 [51]. In this section, we study some of these frameworks and briefly discuss some of their key features.

1.3.3.1 Proprietary-Based

The proprietary frameworks work on specific platforms or require a subscription fee to use the partial or complete set of services. Some examples of proprietary-based IoT frameworks include AWS IoT, Microsoft Azure IoT, Oracle IoT, IBM Watson IoT, Siemens MindSphere, SAP Leonardo IoT, Alibaba Cloud IoT, Braincube, and Hitachi Lumada, just to mention a few. These frameworks are mostly Cloud service providers that optimize some/full suite of their services while considering different requirements of IoT devices. To illustrate, AWS IoT provides several Cloud-based services for IoT scenarios. AWS IoT provides software that helps integrate IoT devices into AWS IoT-based solutions. When IoT devices connect to AWS IoT, it connects them to the Cloud services that AWS provides (e.g., analytics, databases, networking).

1.3.3.2 Open Source

The open-source frameworks usually do not depend on specific platforms, and they can be deployed on different hardware configurations. Besides, due to their open-source nature, developers can extend these frameworks based on their requirements. Some examples of open-source IoT frameworks include:

- Kaa IoT [49]: It is one of the highly flexible and multi-purpose frameworks for implementing end-to-end IoT solutions. It enables information exchange among devices, real-time monitoring of devices, Cloud integration, and visualization of the system.
- ThingSpeak [50]: It is an IoT analysis platform that allows aggregation, visualization, and analysis of live IoT data streams in the Cloud and sending monitoring alarms and feedback. It uses the power of MATLAB to make sense of the IoT data.

- Zetta IoT [52]: It is built on Node.js for creating IoT servers that run across distributed computers and the Cloud. It combines REST APIs, WebSockets, and reactive programming, which suits assembling many devices into data-intensive and real-time applications.
- DeviceHive IoT [53]: It is an IoT platform distributed under Apache 2.0 license and provides deployment options for Docker and Kubernetes. It supports Android and iOS and can run batch analysis and ML techniques.
- FogBus2 [51]: It is a containerized framework for the integration of heterogeneous IoT devices with the multi-cloud platform and Edge servers. It supports dynamic resource scheduling, scalability, profiling, resource discovery, multi-database, and blockchain.

1.3.4 Deployment Models

Recent advancements in hardware and software technologies provide new opportunities for the deployment of IoT systems and applications. The majority of IoT systems and applications were previously deployed in a centralized manner using the resources and services provided by Cloud service providers. However, the emergence of the Edge/Fog computing paradigm alongside advancements in communication technologies make the efficient distributed deployment of IoT systems and applications practically possible. Selecting the optimal deployment type for IoT systems and applications is an important design choice, requiring precise analysis of important contributing factors such as the requirements of IoT systems and applications, the availability of resources, scalability, and pricing, just to mention a few.

1.3.4.1 Cloud-Centric IoT

Cloud service providers offer numerous types of on-demand and pay-as-you-go Cloud services with different levels of QoS for their customers. Generally, the resources at the Cloud are richer and more powerful than distributed resources at the edge of the network. Besides, the Cloud service providers offer high availability services with a diverse range of flexible pricing schemes for their customers. Hence, many large organizations and companies integrate their IoT systems with these Cloud service providers and send their resource-hungry computations or streams of data to the Cloud for processing and storage. To illustrate, ML services of AWS and Microsoft Azure are being used by a wide range of startups and large organizations for market prediction, finding target outliers, and processing of large and high-quality video files and streams. Moreover, Oracle Cloud offers a range of database solutions such as the Autonomous Database to enable optimized Big Data analysis on very large datasets. It is important to note that IoT systems can also be integrated

with multiple Cloud platforms to become beneficiaries of the optimized suite of services that are provided by individual Cloud service providers.

Alongside the suite of centralized services that the majority of Cloud service providers are currently offering, some Cloud service providers such as AWS and Microsoft Azure started expanding their resources at the edge of the network to offer edge-optimized services. While the complete suite of services is yet to be optimized for use at the edge of the network, it shows the future plans of large Cloud service providers such as AWS and Microsoft Azure to be predominant on the Edge/Fog computing platforms [36].

1.3.4.2 Edge/Fog-Centric IoT

The majority of Cloud service providers offer centralized services on Cloud servers; however, many emerging IoT applications such as online gaming, healthcare, and online transportation systems require significantly low latency and real-time services. Proposed solutions that only rely on Cloud computing for processing and storage are not scalable and cannot satisfy the latency requirements of real-time applications. Besides, forwarding the huge amount of data generated by IoT devices may result in congestion at the Cloud servers and interrupt their services [14, 16].

In Edge/Fog computing paradigm, geographically distributed servers (from backbone gateways and routers to racks of Jetson, RPis, and small servers) are being used for pre-processing, processing, and storage of generated traffic from distributed IoT devices with low latency and high access bandwidth. Not only does this distributed computing paradigm bring about the possibility of new IoT systems and applications, but it also helps re-considering and re-designing previous centralized deployments for the more efficient distributed or hybrid ones. Besides, several edge accelerators such as Google Coral Edge TPU Coprocessor have been introduced that further strengthen the computing power of on-premises Edge resources. Alongside the advancement of hardwares, software companies have been optimizing their libraries and products to be efficiently deployed on the Edge resources. Some examples of these software frameworks are Tensorflow Lite, PyTorch Mobile, and the Nvidia Jetson framework.

Although each Edge server may solely not be sufficient for processing and storage of large amounts of incoming requests, the Edge/Fog deployment model enables federated processing and storage of incoming data on several physically or virtually clustered Edge servers. Moreover, the Edge/Fog deployment model enables the possibility of integration of Edge servers with one or multiple Cloud service providers to further solidify the services it offers and extend the range of services. While the Edge/Fog deployment model provides services with less latency and higher scalability, performing optimized resource management and security mechanisms for its heterogeneous distributed resources is more complex compared to the Cloud-centric deployment model.

1.3.5 Applications

The number of IoT systems and applications has been rapidly increasing due to recent advancements in hardware and software. In this section, we describe the main scopes of IoT applications [9, 12, 15].

- Smart Home: It is one of the most practical applications of IoT to bring higher levels of convenience and security for users. Some examples of smart home applications include metering daily electricity and water usage using sensors, automatic illumination systems, and automatic surveillance systems for homes.
- Healthcare: It is one of the most critical applications of the IoT to improve human wellness. To illustrate, IoT enables real-time data acquisition to monitor the overall condition of patients with critical diseases through implantable or attachable body areal sensors.
- Mobility: It is usually among the largest IoT applications area in recent years. Some IoT-based mobility and transportation applications include fleet management solutions and connected cars.
- Retail: It offers many digital solutions, including, but not limited to, in-store digital signage, cashier-less payment systems, customer tracking, real-time goods monitoring, and warehouse management.
- Energy: IoT is significantly transforming the energy industry from generation to interaction process with end-users. To illustrate, the companies in this area are providing some real-time analysis on the usage patterns of customers, predicting their periodic usage, and providing their customers with periodic feedback and energy management advice.
- Smart City: There are many projects to combine connected technologies with infrastructural requirements such as smart parking, traffic management, smart waste, public safety, and environmental monitoring.
- Manufacturing: Big technology companies such as Microsoft, AWS, and Siemens are among the pioneers of digital transformation in manufacturing and industrial industries. These applications support a wide range of heterogeneous connected devices inside and outside the companies for remotely controlling the connected machinery and equipment monitoring, just to mention a few.

1.4 Modeling and Simulation Toolkits

The most effective approach to study, analyze, and evaluate the performance of any services, IoT applications, and resource management techniques in real-world deployment is the ability to perform inexpensive and repetitive experiments. The real implementation and building of a prototype system in heterogeneous computing environments (consisting of many servers in tiered or flatted order, different resource management techniques, different services, and IoT applications) is challenging and tedious work. Besides, the scalable experiments in such environments are

infeasible since heterogeneous IoT devices and Edge/Fog servers are required for conducting scalability tests. Also, the real deployment with tuning and adjusting the configuration of each device leads to significantly high implementation time. To address these constraints, modeling and simulation environments can be used. Simulation helps design and evaluate customized and scalable experiments while allowing the opportunity to repeat experiments under different settings. In the following, we briefly discuss several modeling and simulation toolkits relevant to Cloud and IoT technologies.

- CloudSim [54]: CloudSim is a widely adopted modeling and simulation toolkit for the evaluation of Cloud computing resource management and application scheduling algorithms. It is based on Java programming and provides tools for modeling data center elements such as virtual machines and physical machines. The ability to extend different application scenarios has made CloudSim a popular toolkit to evaluate resource management algorithms in Cloud computing.
- iCanCloud [55]: It is a C++-based simulator based on OMNET++ [56], which can simulate different physical layer entities. It allows the modeling of hypervisor, virtual machine, and physical machine infrastructure to evaluate the management of Cloud applications.
- ContainerCloudSim [57]: This is an extension of core CloudSim designed to evaluate the performance of scheduling and allocation policies in containerized Cloud DCs. It provides several use cases demonstrating methods using and comparing container scheduling and provisioning policies in terms of energy efficiency and SLA compliance.
- CloudSimSDN [58]: It is another extension of CloudSim designed for modeling and evaluation of SDN-enabled cloud environments. It is a lightweight and scalable simulation environment that allows the evaluation of the network allocation capacity policies.

Along with the above-mentioned important simulation frameworks, there have been many extensions of CloudSim such as CloudAnalyst [59], NetworkCloudSim [60], and CloudSimDisk [61] for different use cases and adding extended functionalities. Similar to Cloud Computing, the following are the important modeling and simulation toolkits for IoT and Edge/Fog computing domains.

- iFogSim [62, 63]: It is a simulation environment to imitate the different scenarios in IoT and Cloud/Fog/Edge computing environments. It is built on top of the CloudSim simulator [54] and extends those features to imitate characteristics of IoT devices and Edge/Fog computing environments. The new release of iFogSim [62] further extends its capabilities by supporting several real and random mobility models, service migration, microservice orchestration techniques, and the dynamic formation of clusters.
- EdgeCloudSim [64]: It is a simulation environment, working based on the CloudSim [54], to conduct experiments that consider both computational and networking resources at the edge of the network. It considers the static deployment and coverage area for the gateway nodes and assumes the link quality between IoT and gateway nodes remains always the same despite their distance.

- FogNetSim++ [65]: A toolkit to simulate a distributed Fog computing environment, working based on OMNeT++ [56]. It can imitate different mobility models for the IoT devices, such as random waypoint and linear mobility while supporting customization of different mobility models.
- MobFogSim [66]: It extends iFogSim to enable modeling of device mobility and service migration in Fog computing. However, the mobility support system of MobFogSim only deals with the IoT gateways and Cloud DCs instead of tiered Edge/Fog infrastructure and limits the scope for creating clusters in Edge/Fog computing environments.
- PureEdgeSim [67]: It is a simulator for Edge/Fog and Mist computing environments. It allows evaluating the performance of resources management strategies in terms of network usage, latency, resources utilization, and energy consumption. Besides, it supports mobility-aware application management while its default policy is complex and difficult to customize. It also has limitations in forming node clusters and augmenting microservice management techniques.
- STEP-ONE [68]: This simulator imitates the operations of fog-based opportunistic network environments, which is working based on the ONE simulator [69] with advanced mobility and messaging interfaces. Although it supports the real-world dataset, it lacks default policies for mobility management, node clustering, and microservice orchestration.
- IoTNetSim [70]: It simulates end-to-end IoT services, their granular details, and networking characteristics. While it supports the mobility of IoT devices, it lacks benchmark policies for mobility-driven service management and dynamic clustering.

1.5 Open Challenges

Distributed Resource Management: Due to the highly distributed nature of the IoT paradigm and Edge/Fog computing, contemporary centralized resource management techniques cannot be efficiently adapted. As a result, lightweight and distributed resource management techniques such as distributed scheduling and orchestration techniques are required to be deployed at the Edge devices to enable low-latency access and reduce the service ready time. As the IoT applications generate different workloads (e.g., streaming and batch), such resource management techniques should be able to manage hybrid workloads.

Privacy-Aware Computing: The emerging concerns about providing confidentiality and privacy for users' data directly affect the design and implementation of digital applications. The application deployment should comply with the new privacy policies such as General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA). Hence, it is requisite to move the computational units where the data resides to comply with such policies.

Security: As the number of connected devices with remote access increases, security challenges become more prevalent. The distributed hardwares are openly exposed to everyone without proper security policies and they can be a potential target for attackers to break the system. Also, the IoT data may transmit over unreliable and insecure networks which are open to attackers. Moreover, there are many vulnerabilities in software frameworks and applications that manage the IoT data while these frameworks and applications are not continuously being updated and patched to address security concerns. Furthermore, while there are many tutorials and solutions to connect IoT devices together, there is a lack of IoT and security skills for those who design and develop IoT systems and applications.

Interoperability: There are obviously many heterogeneous connected devices (from things to servers) in the concept of IoT; however, there are few standards, protocols, or regulatory policies to manage interoperability among a wide variety of connected devices. Defining regulatory policies for networking protocols, authentication, and QoS levels are among the important open challenges for interoperability.

Sustainability in Cloud and Edge/Fog Computing: The increasing demand for computing resources has created a new energy challenge. Cloud DCs consume massive amounts of energy, on the other hand, Edge/Fog infrastructures are often powered through batteries or have limited power supply. Hence, energy-efficient resource management is a challenging task that should be addressed.

Systems for ML: Due to advancements in ML techniques and their rapid adoptions across many business and consumer applications, new demand for specialized hardware resources and software frameworks was created. New systems and software frameworks should be built to support the massive computational requirement of these AI workloads.

ML for Systems: While ML systems themselves are becoming mature and adopted into many critical application domains, it is equally important to use these ML techniques to design and operate large-scale systems. Adopting the ML techniques to solve different resource management problems in Edge/Fog and Cloud is crucial to manage these complex infrastructures and workloads.

1.6 Summary

Cloud and Edge/Fog computing have become pervasive technologies transforming the modern economy and digital society. The utility-based service delivery models have truly realized the potential of delivering computing services with ease of access to users by building planet-scale networked computing infrastructure. Cloud computing has matured and continued to evolve and offer different types of new

services, influencing the emergence of new architectural patterns such as Serverless computing. Similarly, IoT-based application scenarios have created an emerging Edge/Fog computing paradigm to serve the requirements of latency-sensitive IoT-based applications. In this chapter, we have explored state-of-the-art Cloud and IoT technologies and provided a detailed discussion on different services and key technologies associated with them. In addition, we have discussed open challenges that should be addressed to make Cloud and IoT technologies robust and secure.

References

1. R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **25**, 599–616 (2009)
2. Gartner: Gartner forecasts worldwide public cloud revenue to grow 17 percent in 2020 (2019), <https://www.gartner.com/en/newsroom/press-releases/2019-11-13-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2020>
3. Gartner: Gartner forecasts worldwide public cloud end-user spending to grow 18 percent in 2021 (2020), <https://www.gartner.com/en/newsroom/press-releases/2020-11-17-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-18-percent-in-2021>
4. M. Satyanarayanan, The emergence of edge computing. *Computer* **50**, 30–39 (2017)
5. S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* **30**, 107–117 (1998)
6. Amazon: Amazon Web Services, <https://aws.amazon.com/>
7. S. Maybury: How much energy does your data centre use?, https://www.metronode.com.au/energy_usage/
8. S. Ilager, K. Ramamohanarao, R. Buyya, ETAS: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation. *Concurr. Comput.* **31**, e5221 (2019)
9. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **29**, 1645–1660 (2013)
10. L. Atzori, A. Iera, G. Morabito, The internet of things: A survey. *Comput. Netw.* **54**, 2787–2805 (2010)
11. I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, et al., Serverless computing: Current trends and open problems, in *Proceedings of the Research Advances in Cloud Computing*, (Springer, 2017), pp. 1–20
12. A.V. Dastjerdi, R. Buyya, Fog computing: Helping the Internet of Things realize its potential. *Computer* **49**, 112–116 (2016)
13. Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson, et al., An open source benchmark suite for microservices and their hardware software implications for cloud and edge systems, in *Proceedings of the Twenty Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, (ACM, 2019), pp. 3–18
14. R. Mahmud, R. Kotagiri, R. Buyya, Fog computing: A taxonomy, survey and future directions, in *Internet of Everything*, (Springer, 2018), pp. 103–130
15. F. Khodadadi, A.V. Dastjerdi, R. Buyya, Internet of things: An overview, in *Internet of Things*, (Morgan Kaufmann, 2016)
16. M. Goudarzi, H. Wu, M. Palaniswami, R. Buyya, An application placement technique for concurrent IoT applications in edge and fog computing environments. *IEEE Trans. Mob. Comput.* **20**, 1298–1311 (2020)

17. R. Ranjan, O. Rana, S. Nepal, M. Yousif, P. James, Z. Wen, S. Barr, P. Watson, P.P. Jayaraman, D. Georgakopoulos, Others: The next grand challenges: Integrating the Internet of Things and data science. *IEEE Cloud Comput.* **5**, 12–26 (2018)
18. M. Goudarzi, M. Palaniswami, R. Buyya, A distributed application placement and migration management techniques for edge and fog computing environments, in *Proceedings of the 16th Conference on Computer Science and Information Systems (FedCSIS 2021), Sofia, Bulgaria*, (IEEE Press, 2021)
19. Cisco: Cisco Annual Internet Report (2018–2023) White Paper (2020), <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. Last accessed 2021/10/07
20. Norton: The future of IoT: 10 predictions about the Internet of Things (2019), <https://us.norton.com/internetsecurity-iot-5-predictions-for-the-future-of-iot.html>
21. Business Insider: The Internet of Things 2020, <https://www.businessinsider.com/internet-of-things-report>. Last accessed 2021/10/07
22. Bain & Company: <https://www.bain.com/about/media-center/press-releases/2018/bain-predicts-the-iot-market-will-more-than-double-by-2021/>. Last accessed 2021/10/07
23. Statista: Internet of Things spending worldwide 2023, <https://www.statista.com/statistics/668996/worldwide-expenditures-for-the-internet-of-things/>. Last accessed 2021/10/07
24. IDC: IoT growth demands rethink of long-term storage strategies, <https://www.idc.com/getdoc.jsp?containerId=prAP46737220>. Last accessed 2021/10/07
25. M. Goudarzi, Z. Movahedi, M. Nazari, Mobile cloud computing: A multisite computation offloading, in *Proceedings of the 8th International Symposium on Telecommunications (IST)*, (2016), pp. 660–665. ieeexplore.ieee.org
26. E.G.M. Petrakis, S. Sotiriadis, T. Soulantopoulos, P.T. Renta, R. Buyya, N. Bessis, Internet of Things as a Service (iTaaS): Challenges and solutions for management of sensor data on the cloud and the fog. *Internet Things* **3–4**, 156–174 (2018)
27. Help Net Security: Global IoT spending dropping significantly in 2020, but expected to rebound in 2021, <https://www.helpnetsecurity.com/2020/06/22/2020-iot-spending/>. Last accessed 2021/10/07
28. Appinventiv: Understanding the impact of IoT in healthcare, <https://appinventiv.com/blog/iot-in-healthcare/>. Last accessed 2021/10/07
29. M. Goudarzi, M. Palaniswami, R. Buyya, A fog-driven dynamic resource allocation technique in ultra dense femtocell networks. *J. Netw. Comput. Appl.* **145**, 102407 (2019)
30. Gartner: What edge computing means for infrastructure and operations leaders, <https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders>. Last accessed 2021/10/07
31. Forrester: Predictions 2020: Edge computing makes the leap, <https://www.forrester.com/blogs/predictions-2020-edge-computing/>. Last accessed 2021/10/07
32. M. Goudarzi, Q. Deng, R. Buyya, Resource management in edge and fog computing using FogBus2 framework. arXiv preprint arXiv:2108.00591 (2021)
33. A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., Above the clouds: A Berkeley view of cloud computing. University of California, Berkeley, Rep. UCB/EECS-2009-28 (2009)
34. S. Ilager, R. Muralidhar, R. Buyya, Artificial Intelligence (AI)-centric management of resources in modern distributed computing systems, in *2020 IEEE Cloud Summit*, (IEEE, 2020), pp. 1–10
35. O. Sefraoui, M. Aissaoui, M. Eleuldj, OpenStack: Toward an open-source solution for cloud computing. *Int. J. Comput.* **55**, 38–43 (2012)
36. Y. Perry, Azure hybrid cloud: Azure in your local data center, <https://cloud.netapp.com/blog/azure-cvo-blg-azure-hybrid-cloud-in-your-data-center>. Last accessed 2021/10/07
37. A.N. Toosi, R.N. Calheiros, R.K. Thulasiram, R. Buyya, Resource provisioning policies to increase IaaS provider's profit in a federated cloud environment, in *Proceedings of the IEEE International Conference on High Performance Computing and Communications*, (IEEE, 2011), pp. 279–287

38. M.G. Xavier, M.V. Neves, F.D. Rossi, T.C. Ferreto, T. Lange, C.A.F. De Rose, Performance evaluation of container-based virtualization for high performance computing environments, in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, (IEEE, 2013), pp. 233–240
39. C. Pahl, A. Brogi, J. Soldani, P. Jamshidi, Cloud container technologies: A state-of-the-art review. *IEEE Trans. Cloud Comput.* **7**, 677–692 (2019)
40. Z. Li, M. Kihl, Q. Lu, J.A. Andersson, Performance overhead comparison between hypervisor and container based virtualization, in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, (IEEE, 2017), pp. 955–962
41. D. Merkel et al., Docker: Lightweight Linux containers for consistent development and deployment. *Linux J.* **2014**, 2 (2014)
42. D. Bernstein, Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Comput.* **1**, 81–84 (2014)
43. C. Vecchiola, X. Chu, R. Buyya, Aneka: A software platform for .NET-based cloud computing. *High Speed Large Scale Sci. Comput.* **18**, 267–295 (2009)
44. M. Shahrad, J. Balkind, D. Wentzlaff, Architectural implications of function-as-a-service computing, in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, (Association for Computing Machinery, New York, 2019), pp. 1063–1075
45. E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J.E. Gonzalez, R.A. Popa, I. Stoica, D.A. Patterson, Cloud programming simplified: A Berkeley view on serverless computing. arXiv preprint arXiv:1902.03383 (2019), <http://arxiv.org/abs/1902.03383>
46. P. Castro, V. Ishakian, V. Muthusamy, The rise of serverless computing. *Commun. ACM* **62**(12), 44–54 (2019)
47. M. Cusumano, Cloud computing and SaaS as new computing platforms. *Commun. ACM* **53**, 27–29 (2010)
48. C. Paniagua, J. Delsing, Industrial frameworks for internet of things: A survey. *IEEE Syst. J.* **15**, 1149–1159 (2021)
49. kaa: The most flexible IoT platform for your business, <https://www.kaaiot.com>. Last accessed 2021/10/07
50. ThingSpeak: <https://thingspeak.com/>. Last accessed 2021/10/07
51. Q. Deng, M. Goudarzi, R. Buyya, FogBus2: A lightweight and distributed container-based framework for integration of IoT-enabled systems with edge and cloud computing, in *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments*, (ACM, 2021), pp. 1–8
52. Zetta: An API-first Internet of Things (IoT) platform, <https://www.zettajs.org/>. Last accessed 2021/10/08
53. DeviceHive: <https://devicehive.com/>. Last accessed 2021/10/08
54. R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **41**, 23–50 (2011)
55. A. Núñez, J.L. Vázquez-Poletti, A.C. Caminero, G.G. Castañé, J. Carretero, I.M. Llorente, iCanCloud: A flexible and scalable cloud infrastructure simulator. *Int. J. Grid Util. Comput.* **10**, 185–209 (2012)
56. OMNeT++: <https://omnetpp.org/>. Last accessed 2021/10/09
57. S.F. Piraghaj, A.V. Dastjerdi, R.N. Calheiros, R. Buyya, ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers. *Softw. Pract. Exp.* **47**, 505–521 (2017)
58. J. Son, A.V. Dastjerdi, R.N. Calheiros, X. Ji, Y. Yoon, R. Buyya, CloudSimSDN: modeling and simulation of software-defined cloud data centers, in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, (ACM, 2015), pp. 475–484

59. B. Wickremasinghe, R.N. Calheiros, R. Buyya, CloudAnalyst: A CloudSim-based visual modeller for analysing cloud computing environments and applications, in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, (IEEE, 2010), pp. 446–452
60. S.K. Garg, R. Buyya, NetworkCloudSim: Modelling parallel applications in cloud simulations, in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, (IEEE, 2011), pp. 105–113
61. B. Louis, K. Mitra, S. Saguna, C. Åhlund, CloudSimDisk: Energy-aware storage simulation in CloudSim, in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, (IEEE, 2015), pp. 11–15
62. R. Mahmud, S. Pallewatta, M. Goudarzi, R. Buyya, iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. arXiv preprint arXiv:2109.05636 (2021), <http://arxiv.org/abs/2109.05636>
63. H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments. *Softw. Pract. Exp.* **47**, 1275–1296 (2017)
64. C. Sonmez, A. Ozgovde, C. Ersoy, Edgecloudsim: An environment for performance evaluation of edge computing systems. *Trans. Emerg. Telecommun. Technol.* **29**, e3493 (2018)
65. T. Qayyum, A.W. Malik, M.A.K. Khattak, O. Khalid, S.U. Khan, FogNetSim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access* **6**, 63570–63583 (2018)
66. C. Puliafito, D.M. Goncalves, M.M. Lopes, L.L. Martins, E. Madeira, E. Mingozzi, O. Rana, L.F. Bittencourt, MobFogSim: Simulation of mobility and migration for fog computing. *Simul. Model. Pract. Theory* **101**, 102062 (2020)
67. C. Mechalikh, H. Taktak, F. Moussa, PureEdgeSim: A simulation toolkit for performance evaluation of cloud, fog, and pure edge computing environments, in *Proceedings of the International Conference on High Performance Computing Simulation (HPCS)*, (IEEE, 2019), pp. 700–707
68. J. Mass, S.N. Srirama, C. Chang, STEP-ONE: Simulated testbed for edge-fog processes based on the opportunistic network environment simulator. *J. Syst. Softw.* **166**, 110587 (2020)
69. A. Keränen, J. Ott, T. Kärkkäinen, The ONE simulator for DTN protocol evaluation, in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, (ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, 2009), pp. 1–10
70. M. Salama, Y. Elkhatib, G. Blair, IoTNetSim: A modelling and simulation platform for end-to-end IoT services and networking, in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, (ACM, New York, 2019), pp. 251–261

Chapter 2

Recent Advances in Energy-Efficient Resource Management Techniques in Cloud Computing Environments



Niloofer Gholipour, Ehsan Arianyan, and Rajkumar Buyya

Contents

2.1	Introduction	32
2.1.1	Motivation and Contribution	33
2.2	Related Work	35
2.3	Proposed Taxonomy for Energy-Aware Resource Management Solutions in Cloud Environments	38
2.3.1	Goals	39
2.3.2	Workload	39
2.3.3	Resources	40
2.3.4	Dynamism	40
2.4	Virtualization Level	45
2.4.1	Migration	47
2.4.2	Consolidation	50
2.5	Comparing the State of the Art	51
2.6	Future Scope and Conclusion	61
2.6.1	Conclusion	62
	References	63

The original version of the chapter has been revised. A correction to this chapter can be found at
https://doi.org/10.1007/978-3-031-05528-7_16

N. Gholipour

Department of Software and IT Engineering of Ecole De Technology Superior,
University of Quebec, Montreal, Quebec, Canada
e-mail: Niloofar.gholipour.1@ens.etsmtl.ca

E. Arianyan (✉)

IT Faculty, ICT Research Institute, Tehran, Iran
e-mail: ehsan_arianyan@itrc.ac.ir

R. Buyya

CLOUDS Lab, School of Computing and Information Systems, The University of Melbourne,
Parkville, VIC, Australia

2.1 Introduction

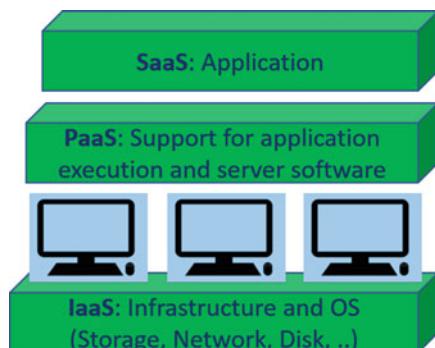
Cloud computing is an internet-based paradigm with the capability of delivering service model including infrastructure, platform, and software as services, which are made available as subscription-based services under the circumstance of a pay-as-you-go model to consumers. In other words, there is a pool of computing resources that consumers can access them on-demand[1]. There are varieties of definitions of cloud computing but the US National Institute of Standards and Technology (NIST) in its definition of cloud computing [2] describes “a “Measured Service” as being one of the five essential characteristics of the cloud computing model. Providing data on measurable capabilities (such as quality of service, security features, availability, and reliability) gives the cloud service customer the opportunity to make informed choices and to gain understanding of the state of the service being delivered. It also gives the cloud service provider the opportunity to present the properties of their cloud services to the cloud service customer.”

Some of the important features of cloud computing are on-demand self-service provisioning, broad network access necessity, availability of resource pools, multi-tenancy, and elasticity.

As depicted in Fig. 2.1, there are three main cloud service models: Infrastructure as a service (IaaS), Platform as a service (PaaS), and Software as a service (SaaS) [3].

In addition to traditional cloud services, there is a new type of cloud service called CaaS that has been introduced recently [4]. A particular example of container management system can be mentioned as Docker that permits developers to define containers for their application and develop, run, and manage their applications without any concern about the underlying infrastructure and required software [5]. Containers share the same host operating system kernel, and they have communication with physical hardware via system call that is much faster than the traditional communication-based hypervisor. Accordingly, they are defined as lightweight virtual environments compared to virtual machines that provide an isolation layer between workloads without the overhead of hypervisor-based virtualization. A Container as a building block of CaaS cloud model suggests an isolated virtual

Fig. 2.1 Cloud service model



environment without any monitoring media such as hypervisor that is existed in traditional systems [4]. Containerization increases the efficiency of resource utilization compared to virtual machines. Google container engine and Amazon Elastic Container Service (Amazon ECS) currently supply this new method.

2.1.1 Motivation and Contribution

With the rapid ever-increasing demand to access diverse information and communication technology services based on the cloud service delivery model, the number of huge energy-hungry cloud data centers is increasing rapidly. Thus, these days energy consumption in cloud environments is a crucial issue. A data center will consume the energy of about 1000TWH in next ten years (2013–2025) [6]. The percentage of energy consumption by the data centers and cooling systems will reach 5% of the total energy consumption in the world. Energy consumption leads to operational cost and environmental implications such as global warming [7].

This key challenge leads to a rethink about the techniques and research strategies to lessen energy consumption as a crucial matter in the cloud environment. To overwhelm this challenge, there are various solutions that researchers have introduced; among them, resource management techniques play a significant role. Nonetheless, energy efficiency is still a challenge for future researchers [8]. Virtualization technique enables cloud providers to create multiple Virtual Machine(VM) instances on a single physical server, or multiple containers on a VM or a physical server which makes it possible to have servers with higher utilization. The dynamic consolidation of both VMs and containers through live migration is an efficient approaches for saving energy in cloud data centers [9]. Although the recent technological developments and paradigms including High-Performance Computing (HPC), containerization, exascale computing, and processing at network edge appear to yield new opportunities for cloud computing, they are also creating new challenges and demands for new approaches and research strategies.

Container technology has emerged thanks to Docker [10] which has boosted in both academia and industry. It provides a way to package an application that can be run with its dependencies and libraries isolated from other applications. Containers arose as a lightweight alternative of VMs that present better microservice architecture supports. The technology of container is strongly supported by PaaS, IaaS, and Internet Service Providers. Traditional hypervisor-based solutions are virtualized at the hardware level, while containerization provides virtualization at the operating system level. The containers interact with each other via system standard calls and they do not have any information about themselves [11]. Although VM technology needs to have an individual operating system for each VM, only one operating system can serve all containers in container technology. So, container technology provides more lightweight virtual systems which makes it possible to utilize system resources such as CPU, RAM, and network bandwidth more efficiently [10]. This happens owing to Linux kernel's cgroups and namespaces which are used by docker. Besides, utilizing container technology considerably

decreases startup time and the expected resources for each image in comparison with VM technology. To exemplify, a container requires 50 milliseconds to start, while a VM is activated in 30–40 seconds [12].

Many Internet companies have embraced this technology and containers have become the de-facto standard for creating, publishing, and running applications. On the other side, there are still impediments and challenges in container-based virtualization demanding to be addressed, including security issues, in particular, during migration, dynamic resource allocation, and energy consumption [13].

Container orchestration appears to yield several possibilities for cloud and application providers to determine how to select, deploy, monitor, and dynamically manage the configuration of containers in the cloud. It is also involved with the management at runtime to support the deploy, run, and maintain states [14]. As a consequence, an especially serious problem to address in this context is the scheduling or placement of containerized applications on the available hosts along with VM consolidation. Our main contributions are as follows:

- Addressing an Energy-aware cloud resource management problem.
- Proposing taxonomies for energy-aware resource management techniques in the cloud environment.
- Studying and reviewing the important issues and parameters in the literature for energy-aware cloud resource management approaches.
- Proposing open research directions related to energy-aware resource management problem in cloud environment.

The aim of this chapter is to express the importance of energy-efficient resource management in cloud environments and present a scientific and taxonomic survey of the relevant recent literature in the period of 2015 through 2021. We propose a novel taxonomy for the subject of energy-efficient resource management in cloud environments and categorize the most recent papers utilizing our proposed taxonomy. In this chapter, we address nine other survey papers in the literature as our related work which are about energy-efficient resource management. We also evaluate the cons and pros of these papers and compare them with our survey based on our proposed taxonomy. In addition, we study and compare 32 articles based on our proposed taxonomy.

Our proposed taxonomy appears to yield a number of distinctions in comparison with other related surveys. One of the most significant superiority is considering the containerization technology and container migration topic in addition to virtualization and VM migration techniques that previous works neglected. Unlike other similar taxonomies that consider the dynamic voltage and frequency scaling (DVFS) technique applied only on Central processing unit(CPU), our taxonomy addresses the DVFS approach applied at two levels: memory and CPU. Other relevant papers consider only arbitrary workloads in their work, while our taxonomy considers different types of workloads, including high-performance computing (HPC), real-time, as well as batch applications. Other differences between this chapter and previous studies can be mentioned as reflecting both passive and active resource types. Finally, our taxonomy considers rack and geographically level consolidations

as well as server-level consolidation. The last important superiority of our chapter is that it surveys the latest state-of-the-art papers in the literature.

The rest of the chapter is organized as follows: Sect. 2.2 reviews the related works, followed by our proposed taxonomy for energy-aware resource management in cloud environments and virtualization in Sect. 2.3 and 2.4, respectively. Comparing and mapping the recent relevant papers based on our proposed taxonomy is fulfilled in Sect. 2.5. Ultimately, future research directions and conclusions are presented in Sects. 2.6.

2.2 Related Work

Energy-efficient resource management approaches in cloud environments are a hot topic which vastly addressed by researchers. Since cloud computing's research has advanced continuously, there is a need for a systematic review to evaluate, update, and join the existing literature. This section summarizes some of the previous works in the literature similar to our work, the result of which is shown in Table 2.1.

The authors in [15] have conducted a survey on energy-aware resource allocation in cloud data centers. They have reviewed some keywords such as virtualization, allocation of VM, energy efficiency, power consumption, as well as cloud computing. They have discussed various kinds of energy-aware system architectures for the cloud and compared energy efficiency in both traditional and virtual data centers. This chapter has further proposed a taxonomy for energy-saving methods in cloud data centers, which were studied in three levels, such as power management, resource management, and thermal management. The researchers have reviewed previous works based on VM allocation algorithms, VM selection algorithms, and Dynamic Voltage Frequency Scaling(DVFS), which conducted to energy saving. Plus, they have shown that the energy-saving approach became possible using renewable energy that plenty of recent research introduced this strategy.

In [16] the authors have presented a brief survey describing primary energy-conserving techniques in the cloud environment. To add, they have classified energy consumption approaches into five categories, including energy-efficient hardware, energy-aware scheduling, consolidation, energy conservation in a cluster of servers, as well as power-efficient networks. Finally, they have evaluated a few papers based on this classification. The researchers further have focused on consolidation techniques in three levels, containing task consolidation, server consolidation, and energy-aware task consolidation.

Researchers in [24] have performed a comprehensive survey on energy-efficient computing, clusters, grids, and clouds. They have reported a number of approaches in the literature which contributed to improve energy efficiency. This chapter has proposed three taxonomies, covering such levels as scheduling, energy-efficient computing, as well as energy-efficient technique at different levels to make data center greener. Plus, [24] studied the energy efficiency of a single system and large-scale cloud data centers, storage systems, and networking.

Table 2.1 Comparison table

Scientists in [19] have reviewed energy-saving strategies in computational clouds. They explored state-of-the-art related to energy efficiency as well as performance administration, vitality for effective data centers, and resource distributions. What is more, they have studied the existing techniques in four stages, including tools, OS, virtualization, and data center. Their proposed taxonomy was divided into static and dynamic power management at the highest level. On the one hand, they have considered DVFS, resource throttling, Dynamic Component Deactivation (DCD), and workload consolidation at the software level in their taxonomy. On the other hand, they have considered Dynamic Performance Scaling (DPS), resource throttling, DVFS, and DCD at the hardware level.

The authors in [22] have carried out a survey on several papers over the last decade respecting energy efficiency techniques for cloud computing applications. They have mentioned some approaches consisting of heuristic algorithms for live VM migration, task scheduling, load balancing. This paper has further discussed various kinds of tools that are applied in order to design energy-efficient schemes. They have analyzed three simulators in this scope, including CloudSim, a popular tool that supports the IaaS layer, GreenCloud, and icanCloud. In [21] scientists have conducted a survey on energy-aware VM consolidation strategies, highlighting the limitations and profits. They have analyzed VM consolidation features such as power consumption, VM consolidation types, and so on. Besides, they have compared state-of-the-art VM consolidation techniques in terms of objectives, Service Level Agreement (SLA) violation, energy cost, number of VM migrations, etc. To add, they have categorized these proposed techniques in a table; however, they have not presented any taxonomy.

The researchers in [23] introduced a survey on energy-efficient algorithms based on VM consolidation for Cloud Computing. In this paper, five state-of-the-art energy-efficient algorithms were compared from architecture, modeling, and metrics points of view; Modified Best Fit Decreasing (MBFD), EcoCloud, Greedy based scheduling Algorithm Minimizing Total Energy (GRANITE), Learning Automata Overload Detection (LOAD), as well as Ant Colony System (ACS). What is more, they have implemented and analyzed these algorithms according to the experimental settings in the CloudSim simulator using the PlanetLab workload. As their results have shown, ACS can achieve the best energy efficiency in most cases as it also has the least number of active hosts.

Another survey which was addressed by Kaur and Bawa [17] has concentrated on energy-aware VM placement and consolidation techniques in cloud environments. Researchers, in this paper, have investigated such approaches as heuristic, constraint, and bin-packing problem, so forth. To exemplify, they have considered some procedures, containing first fit, single-dimensional best fit, volume-based best fit, plus dot product based fit. In the same way, they reviewed several papers in their proposed categories. According to them, some significant issues that should be considered in this area pertains to workload consolidation, Quality of Service (QoS) guarantee, minimizing the number of both migrations and active physical hosts.

Researchers in [20] have briefly reviewed energy-aware task scheduling algorithms in cloud environments. They also have analyzed the detriments and benefits

of existing approaches. One of the algorithms that they considered in their paper can be mentioned as an energy-aware genetic algorithm. Another considered approach can be referred to as an energy-saving task scheduling depending on vacation queue. In addition to these methods, they have studied the DVFS technique. To sum up, they concluded that most state-of-the-art papers do not notice the deadline of a task and the cost of executing a task while making the scheduling decisions.

2.3 Proposed Taxonomy for Energy-Aware Resource Management Solutions in Cloud Environments

In this section, we present our proposed taxonomy for energy-aware resource management solutions in cloud environments, as shown in Fig. 2.2. In our proposed taxonomy, we consider four items at the highest level. The first level pertains to the goals of energy-efficient resource management in cloud environments. As can be seen, the second level goes back to the dynamism of resource management technique. The third level is the considered type of workload, including arbitrary, High-Performance Computing (HPC), batch, and real-time applications. Finally, the fourth level is the type of resources that are classified into active and passive. The details of this taxonomy are described in the following subsection.

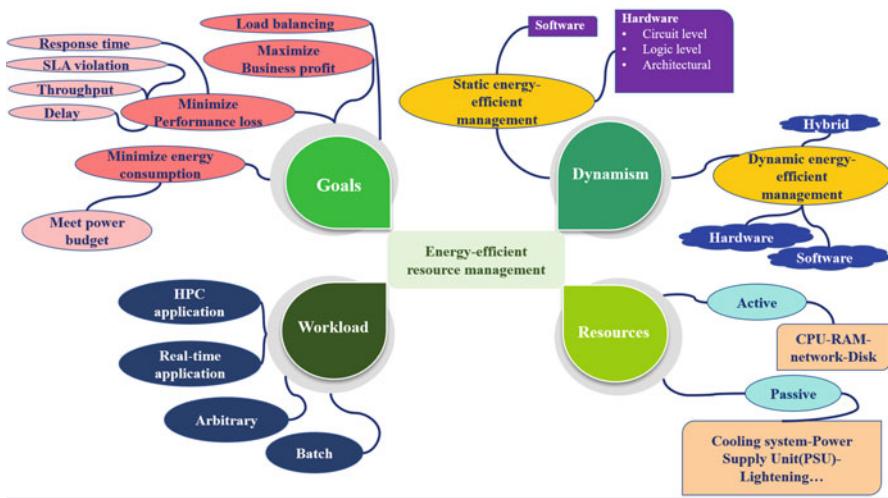


Fig. 2.2 Taxonomy of energy-efficient management solution in cloud environments

2.3.1 Goals

As energy-efficient resource management approaches play a crucial role in cloud data centers these days, there are various kinds of goals in this way. To clarify, Fig. 2.2 summarizes the components of goals. Indeed, we have considered five targets for this component, such as minimizing power consumption, maximizing performance, load balancing, meeting power budget, plus maximizing business profit. We have also regarded four performance metrics, including response time, SLA violation, throughput, and delay. First and foremost, data centers have significant power cost; thereby, it is an essential requirement for data centers' operation to meet the power budget coming from the limitation for power usage and observing this limitation [25].

Due to load imbalance, some of the data center resources may become overloaded or underloaded, which leads to performance degradation and resource wastage. Load balancing conduces to maximize resource utilization and achieve the desired QoS in the cloud by employing optimal resource allocation and workload distribution approaches at both schedule and runtime.

2.3.2 Workload

As a basic definition, the workload is a job that is characterized by release time, worst-case execution time, and deadline. At a high level, the workload is a sequence of jobs and tasks. The computer workload is defined as the amount of work allocated to the system that should be completed in a determined time [4]. A usual system workload comprises tasks and user's requests submitted to the data center. As stated in [26], understanding the workload is far more important than designing new scheduling algorithms. If the systems are not tested using the correct input's workloads, the proposed policies or algorithms' results might not work as expected when applied to real-world scenarios [4]. We consider four workload types in our proposed taxonomy containing arbitrary, batch, HPC, and real-time workload, described as follows:

- **Batch processing:** Theoretically, batch processing is a processing mode when a sequence of jobs are executed on a batch of inputs [27]. Analyzing data on a large scale and batch processing occurs by utilizing data centers and some distributed and computing frameworks such as Map-Reduce and Hadoop [4]. Map-Reduce programming paradigm is the most practical and efficient solution for batch processing of big data [28].
- **HPC:** In the early 1990s, clusters of computers became famous in HPC environments owing to their low cost compared to custom supercomputers and mainframes [29]. Also, HPC computers generally take advantage of open source operating systems such as Linux. In the early 2000s, grid computing was linked to the HPC community as a consequence of need to run parallel

programs even larger than that was normal in grid environments. Grids provide powerful resources operated by independent administrative domains to users [30]. In the late 2000s, cloud computing was quickly growing its adolescent level and reputation, and studies started to appear on the viability of executing HPC applications on remote cloud resources [31, 32]. HPC applications are resource-intensive scientific workflows (in terms of data, computation, and communication) that have usually aimed at Grids and customary HPC platforms like super-computing clusters [33]. Both the size and number of HPC data centers have overgrown in recent years, which conduces to an exponential increase in power drastically [34].

- **Real-time application:** With improved cloud computing infrastructure, real-time computing can be accomplished on cloud infrastructure [35]. In most of the real-time cloud applications, the processing is executed on remote cloud computing nodes. As we meet many real-time systems around us, Cloud's support plays a crucial role in the real-time system [36]. Their application ranges from small mobile phones to huge industrial controls and from a mini pacemaker to larger nuclear plants. A usual real-time, such as financial analysis, distributed databases, or image processing, includes multiple real-time applications or sub-tasks service [37]. Real-time systems are implemented by several simultaneous tasks requesting to access hardware resources [24].

2.3.3 Resources

In cloud computing, resource management plays a key role in the entire system's performance [38]. Ordinarily, a data center comprises four main structural components including network switches, cooling systems, racks, also servers [39, 40]. Each rack consists of several servers, while each server has both a dedicated power unit and a cooling fan. In our proposed taxonomy, we categorize the resources into active and passive types. CPU, RAM, Disk, and network interface are considered as main active resources. Also, cooling system and power supply unit (PSU) are considered as main passive resource type.

2.3.4 Dynamism

Dynamism specifies the dynamicity of the power-aware resource management's techniques in our proposed taxonomy. From dynamism point of view, energy-aware resource management techniques are divided into static power management (SPM) and dynamic power management (DPM). The next subsections present these techniques in detail.

2.3.4.1 Static Power Management (SPM)

The static power management can be executed on both hardware and software levels. Leakage currents in any active circuits cause static power consumption at the hardware level [7]. SPM uses hardware components such as CPU, memory, disk storage, network devices, and power supply unit efficiently. It consists of all applied optimization methods during design time at logic, circuit, and architectural levels that will be explained in the following section.

- **Logic level optimization:** At this level, optimization methods attempt to optimize the power of switching activity in both sequential and combinational circuits. Minimizing the switching capacitance improves the dynamic power consumption straightly by reducing the energy per transition on each logic device [7, 41].
- **Circuit level optimization:** Significant challenges at this optimization level are based on efficient pipelining and interconnections between stages and components. Pipelining technique is regularly used to boost throughput in high-performance designs at the expense of reducing energy efficiency, contributing to increasing area and execution time [41].
- **Architectural level optimization:** Methods include the system's design considering power optimization technique at an architectural level [7]. Power savings are typically accomplished at the architectural level by optimizing the system components' balance to prevent wasting power.[41].

Besides the optimization at the hardware level, considering the SPM at the software level is also essential. Even with robust hardware design, it is crucial to be careful about software design inasmuch as weak design conduces to loss of power and performance, even with perfectly designed hardware. Thus, the code generation, the instructions used in the code, and the order of these instructions must be carefully selected, as they affect performance as well as power consumption.

2.3.4.2 Dynamic Power Management (DPM)

This section describes our taxonomy at the dynamic power management level, as shown in Fig. 2.3. DPM is categorized into three levels, including hardware, software, and hybrid. There are various kinds of optimization methods at both the hardware and software levels [7, 42]. At the hardware level, we can imply techniques such as DVFS, DCD, and sleep states. In addition, the techniques at the software level are classified into virtualization, migration, consolidation, plus containerization. The dynamic power consumption is induced by the high usage of hardware components (such as CPU, storage, and network devices) and the circuits' activity. The main reason enabling dynamic power consumption pertain to both system's components deactivation and tuning the circuit activity. Dynamic power consumption can be reached through different techniques including: (1) diminishing the switching activity, (2) decreasing the physical capacitance that relies on low-

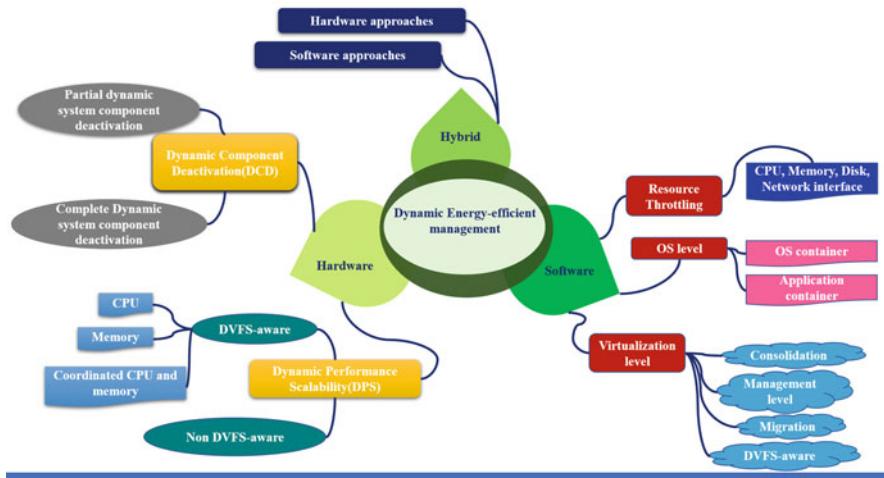


Fig. 2.3 Taxonomy of dynamic resource management solutions in virtualized cloud environments

level design parameters such as transistors' sizes, (3) ebbing the supply voltage, and (4) lessening the clock frequency [7].

DPM improves energy consumption by using knowledge gathered from current resources in the system and the workload of applications running in the system [7, 43]. DPM techniques allow dynamic adjustment of power states to occur based on current system loads. It predicts the best action in the future using the data obtained from the system and according to the system's requirements. DPM techniques are categorized into hardware and software levels. There is another level in our taxonomy, namely hybrid, in which both hardware and software techniques are simultaneously utilized.

1. Hardware-level approaches

DPM techniques applied at the hardware level reconfigure the system dynamically by adopting methodologies to fulfill the requested services with the minimum number of active components or the minimum load on such components [43]. The DPM techniques at a hardware level can optionally turn off the idle system components or reduce the useless ones' performance. It is also possible to exchange some components, containing CPU, between either active or idle modes to save energy. The hardware DPM techniques vary for different hardware components, yet usually, they are splitted into dynamic component deactivation (DCD) and dynamic performance scaling (DPS) [44].

(a) Dynamic component deactivation (DCD)

The techniques in our proposed taxonomy at the DCD level are categorized into both partial dynamic system deactivation (PDSD) and complete dynamic system deactivation (CDS). PDSD techniques are built upon the idea of the clock gating parts of an electronic component. Computer components,

which do not support performance scaling and can be deactivated, need some techniques that leverage the unsustainable workload and disable the idle devices. Nonetheless, CDSD techniques are based on the idea that the components are entirely deactivated during some periods [43].

(b) **Dynamic performance scaling (DPS)**

DPS methods that can be applied in computer components support dynamic adjustment of their performance rationally based on the power consumption and the requested resources [7, 43]. Instead of complete deactivations, just the clock frequency of some components, such as the CPU, is allowed to be decreased or increased along with adjustments of the supply voltage in cases when the resources are not fully utilized. DPS is grouped into both Dynamic Voltage and Frequency Scaling (DVFS) aware techniques and non-DVFS-aware techniques.

DVFS is an effective technique at the system level used for both CPU and memory [4]. This method enables dynamic power management by changing the supply voltage or the processor's operation frequencies and memory. Hence, in our proposed taxonomy, we address DVFS at three levels: DVFS of CPU, DVFS of memory, and DVFS of coordinated CPU and memory.

• **Dynamic voltage and frequency scaling of CPU**

In this technique, the CPU's voltage and frequency are setting up dynamically proportional to the workload [4, 45]. Thus, the CPU frequency is adjusted to help lessen overall energy consumption for executing the tasks until the user's deadline. Ebbing overall energy consumption is the primary purpose of the algorithms. It is crucial to heed that the DVFS technique is not always energy efficient because scaling the CPU frequency might increase execution and CPU idle time.

• **Dynamic voltage and frequency scaling of memory**

In addition to the CPU, the memory consumes considerable energy [22, 46]. In this technique, the system's memory speed is well adjusted according to the memory-intensive workload's peak power consumption. Employing this technique for workloads that are not memory-intensive would lead to performance degradation as well as higher energy consumptions.

• **Dynamic voltage and frequency scaling of coordinated CPU and memory**

Coscale has been introduced by Deng et al. [47] for the first time; it jointly applies DVFS on both CPU and memory concurrently, intending to diminish overall energy consumption. The frequency of each core and memory bus is selected to increase the saving in the overall system's energy. The selected frequencies are not always the lowest ones. One of the most noticeable features of the Coscale can be mentioned as balancing the system and its utilization power. It is done by searching the CPU and memory frequency setting space and fixing the components' voltage according to the picked frequencies.

2. Software-level approaches

Many DPM algorithms can be performed at the hardware level as a part of the circuit [43]. Nevertheless, it is difficult to implement any modification and reconfiguration at the hardware level. Therefore, there are some strong reasons for migration to software-level solutions. In our taxonomy, the techniques in the software level are categorized into resource throttling, OS container level management, application container level management, and virtualization level management which are described in the following sections.

(a) Resource throttling

Resource throttling is beneficial to make cloud computing a great deal more energy efficient. It controls how users are permitted to consume cloud resources in various ways at either hardware or software levels to meet the performance requirements whilst reducing the energy consumption [48]. Several parameters can be throttled in a cloud environment, yet we consider the CPU, memory, disk, plus network interface parameters in our proposed taxonomy. To prevent resources from being overwhelmed by access requests, techniques can be applied to throttle the amount of granted resources.

(b) Operating system (OS) level

In 1996, Intel, Microsoft, and Toshiba published the first version of the Advanced Configuration and power interface (ACPI)[43]. ACPI is an open standard that defines a unified OS, centric device configuration, and power management interface [49]. It further describes an independent platform interface for monitoring, discovery, power management, plus configuration of hardware [50]. It is designed to permit OSs to configure and control each hardware component, dislocating both the predecessor's plug and play (PnP) energy management and the advanced power management (APM). The primary purpose of ACPI is to enable the whole computing system to implement all of the DPM capabilities and efficiently develop the power-managed system. ACPI provides an interface for software developers to adjust the system's power states. It defines some power states that can be applied at runtime, containing processor operational states (P-states), sleep states (S-states), global states (G-states), device states (D-states), and processor idle states (C-states). The two of the most remarkable states in DPM pertain to C-state and P-state [49, 50]. The power management methods in our taxonomy at the OS level are subcategorized into OS container and application container that will be described in the following subsections.

- **OS container** Containerization is a lightweight technology that virtualizes and manages applications has recently been successful in cloud environments [51]. Container technology is currently employed to decrease the difficulty of software deployment and portability of applications in cloud computing infrastructure [52]. In [5], they have proposed Container as a service (CaaS) architecture, consisting of physical hosts, virtual machines (VMs), containers, applications, and their workloads. CaaS can be located between IaaS and

PaaS layers; while, IaaS provides virtualized compute resources, and PaaS provides application runtime services, CaaS sticks these two layers together by providing isolated environments for the deployed applications (or different modules of an application). PaaS has accelerated the development of applications without any requirements to manage underlying infrastructure [4, 40]. Containers can be run on both Physical Machines (PMs) and VMs, and they create an isolation space for the applications and services running on them. Containers are the building blocks of OS-level virtualization that propose an individual virtualization environment, which does not need a monitoring device like a hypervisor. Containerization technology is implemented in large-scale corporations such as Google and Facebook. One of the most significant advantages of containerization compared to virtualization is its software layer compactness, which impacts lighter overhead to the system. Another benefit that can be mentioned for OS containerization is sharing the host kernel.

- **Application container** The application container is dedicated to one process, while the OS container runs a number of processes and services [4]. Application container is a new technology with some advantages such as lightweight, more straightforward configuration and management, and a significant reduction in startup time. As the workloads are unpredictable, containers are provided with auto-scaling that conduces to diminishing of resource wastage and gradual increase in energy consumption. Hence, designing optimal container placement algorithms is the biggest challenge for cloud providers. Containers can be applied fast owing to their low overhead. In CaaS model, applications are normally executed inside containers while containers are placed in virtual machines [5].

3. Hybrid: Software and Hardware

In our proposed taxonomy at DPM level, we have presented a hybrid approach that is a combination of virtualization and OS-level approaches. Hybrid approaches use both hardware and software techniques at the same time. As shown in our comparison tables, some works have concentrated on both techniques; thus, we defined “hybrid” as a new approach in this domain.

2.4 Virtualization Level

In this section, we present virtualization taxonomy, as shown in Fig. 2.4, virtualization is an evolving technology in the IT world [53]. In cloud computing, virtualization strategy is creating a virtual (rather than actual) version of a resource or device, such as a server, an operating system, a storage device, or a network. Virtualization technology is one of the key features of efficient resource management in cloud data centers that can improve hardware efficiency through resource sharing, migration, and consolidation [7]. The virtualization technique makes it possible to abstract the operating system and applications running from the underlying

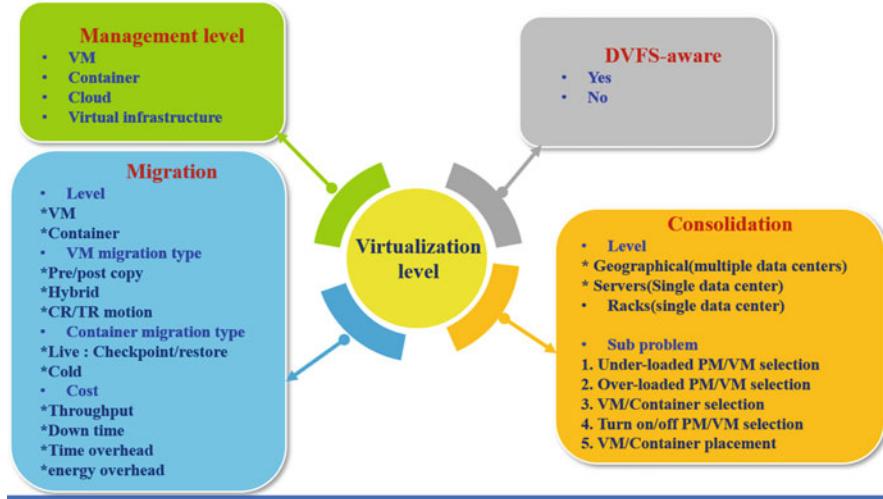


Fig. 2.4 Taxonomy of dynamic energy-efficient resource management in the cloud environment

hardware. This technology decreases the operational costs by sharing physical resources, including CPU, RAM, and I/O, plus reducing the number of physical machines. We split our taxonomy at the virtualization level into four main subjects: (1) whether the solutions in this level are DVFS-aware or not, (2) the management level of the solutions, (3) the considered migration cost and type in the solutions, (4) the consolidation level as well as the considered consolidation subproblems. The DVFS technique scales both the frequency and voltage of processors according to the computation requirements and reduces the energy dissipations. We consider three sub-levels for the management levels of virtualization, consisting of VM management, Virtual infrastructure (VI) management, and cloud management [54]. VM management layer supplies the required procedures for managing VMs on a single PM. The duties of the VI management layer are scheduling and managing VMs on multiple PMs. The cloud management layer provides secure and remote interfaces for monitoring, controlling, and creating virtualized resources. We want to emphasize the importance of this division by highlighting that DVFS and consolidation procedures are performed sequentially and separately in the VM management and cloud management layers, which may have harmful cross-side effects on each other [54]. More precisely, the DVFS governor scales the processor's frequency dynamically in the VM management layer according to the host global CPU load and regardless of the VM local load. The consolidation technique can be applied at the servers, racks, data centers, or VMs levels; furthermore, the remaining nodes can be turned off or put in the low-power mode.

2.4.1 Migration

The notable characteristic of virtualization that makes it possible to manage resources efficiently can be referred to as both virtual machine and container migration and consolidation [46]. In the consolidation technique using VM migration, all the VMs residing on underutilized PMs are transferred to other active PMs. The idle PMs are then turned off or put in the low-power consumption mode to reduce total energy consumption. Three hardware states are carried through migration, including memory, application, and network interface cards (NIC) [55]. In the same way, in container consolidation, the entire containers placed in underloaded VMs are sent to other VMs owing to turning off the underloaded VMs. The VM migration mechanism can be classified into both live (hot) migration and non-live (cold) migration [55]. In non-live migration, a VM is paused first at the source PM and resumes at the destination PM after all the required workloads are migrated. Nevertheless, in live migration, the VM resumes its service during the migration. The optimally utilized nodes consume less energy than either over-utilized or underutilized PMs [56]. Another remarkable profit of VM migration is the possibility of mitigating hot spots in data centers by migrating some loads of over-utilized nodes to other less utilized nodes [55]. More advanced live-migration mechanisms do not require a noticeable pause in virtual machines; also, it replies to the customer's requests during the transition [57]. However, in non-live migration, the response time is too long compared with live migration, and there is no high availability. Generally, migration techniques can be used in order to reach different goals, such as green computing, load balancing, fault-tolerant, and real-time server maintenance [58]. Live-migration approaches that we have considered in our taxonomy are including pre-copy, post-copy, hybrid, as well as CR/TR motion that will be presented in the following subsections.

2.4.1.1 VM Migration Type

1. Pre-copy approach

Pre-copy is one of the common techniques in live-migration mechanism [57]. It is called live owing to the VM does not stop during the migration [59]. First, all of the memory pages are transferred, and then the modified pages are transmitted in each iteration. The pre-copy approach includes warm-up and stop and copy phases [55]. Hypervisor copies just all memory pages from the source node without interrupting the VM, and through this function, the changes on these pages are recorded (in terminology “dirty page”). In the subsequent step, modified pages are sent. The hypervisor suspends the VM in the source node, and in the last iteration, any remaining pages, processor states such as register values are transferred. Finally, the original VM in the source node is discarded, and the VM in the destination node will resume.

2. Post-copy approach

One of the significant drawbacks of the pre-copy approach can be referred to as the warm-up phase is too long, while the post-copy approach outperforms in this regard [55]. The post-copy approach's migration time is far less than the pre-copy mechanism, yet its downtime is beyond pre-copy inasmuch as each memory page is transferred just once [60]. Contrary to pre-copy, post-copy transfers the CPU and device states into the destination node at the first step and commences the destination node's execution in the second step. One of this approach's principal problems is performance degradation when there is a large working set workload. The processes of post-copy approach are as follows [57]:

- The hypervisor stops the VM at the source node.
- At first, the processor states are copied to the destination VM, and this VM starts to work.
- If the VM accesses the page that does not exist in the destination node, the page fault occurs, and the post-copy mechanism handles it. Ultimately, VM's page is transferred to the destination node (on-demand paging).

3. Hybrid approach

This approach is a hybrid of post-copy, and pre-copy methods [55]. Similarly to the pre-copy's first iteration, the whole pages are transferred to the target node in this approach. What is more, this method can avoid performance degradation. Compared with the previous two methods, the performance penalty only after the VM starts on the destination node can be less [55]. The hybrid method accomplishes more memory transition rather than the post-copy one, although it migrates such pages fetched by transporting all memory at the first step. The processes of the hybrid approach are as follows [61]:

- Preparation phase.
- Bounded pre-copy rounds phase: memory pages send in iterations to the target host. There is a maximum number of iterations until this repetitive transition resumes.
- VM state transfer phase: the VM processor state is transferred to the target host.
- VM resume phase: this happens at the target host after the VM's processor state is transferred.
- On-demand paging phase: this phase is similar to the post-copy mechanism. It rises where the VM at the destination host requires transferring the remaining memory pages from the source.

4. CR/TR motion approach

CR/TR motion is a novel procedure in the live VM migration category that stands for Check Pointing/Recovery and Trace/Replay [61]. Previous approaches were useful in a local area network (LAN) environment; nonetheless, they would cause a long period of downtime in a vast area network (WAN) environment [62]. This scheme emerged, leading to a fast and transparent transition for both LAN and WAN environments [63]. CR/TR motion can significantly decrease the migration

downtime and network bandwidth consumption. It transfers an execution log instead of VM memory and repeats the log on the destination node to generate the same state on the target VM [55]. First, it exploits the memory image of the target VM and sends it to the destination node. Following that, the virtual machine monitor (VMM) level mechanism registers the VM's execution and then frequently transfers the log to the destination. The logging mechanism is based on the log/reply system, which confiscates and replays all non-deterministic events that can affect the VM's execution, such as virtual interrupts. The replay mechanism on the destination repeats the VM execution from the transferred log. CR/TR motion begins the stop and copy phase when the log size is smaller than the predefined size [64].

2.4.1.2 Container Migration Type

Let us consider container migration with two general types: live and cold. In the process of cold migration, first, the container stops at the source node, then copies its file system to another node, the destination node. Ultimately, the container at the destination node starts. One of the most notable ways in live migration goes back to checkpoint and restart, which enables it to transmit running container from one physical machine to another without stopping the container. Indeed, the container's file system copies to another server, and the container is restarted on another physical machine from the file [65].

2.4.1.3 Migration Cost

The migration process includes some costs that should be considered [66]. Our proposed taxonomy classifies migration costs into four categories: power overhead, migration time overhead, downtime, as well as throughput. The migration technique increases the utilization of the resources (CPU, network interface), contributing to high-energy consumption [67]. The most energy consumption within the migration occurs by the time the VM states are stored and sent to the target host. Since the target host sends acknowledgment and resource availability checking to the source host for migration commencement declaration, the target host consumes more energy. Besides, there is a time span that two different hosts consume energy for the same VM. The cost of VM migration depends on two factors: the power consumed by network devices and the migrated memory size. Generally, the cost of migration is increased for larger RAM sizes and is decreased by raising the network bandwidth [62]. Due to migration operation not being power-free, the energy consumed during migration must be considered once designing or developing energy-saving techniques [66].

2.4.2 *Consolidation*

The consolidation technique is an act that compacts and combines some units to the integral unit by turning off the underutilized PM or sitting them in the sleep mode. It is formulated as a bin-packing problem [68]. The bin-packing problem is to put several items into a finite number of bins, and the minimal bins are used. In the VM consolidation problem, each VM and host is assumed as an item and a bin, respectively. Virtualization technology is the base of consolidation [46]. This approach maximizes resource utilization and minimizes energy consumption. This survey considers consolidation at three levels including rack, server, and data center (which are geographically distributed).

2.4.2.1 *Consolidation Level*

Consolidation at these levels aims to increase resource utilization and decrease energy consumption. For optimization of the data center's operation, it is essential to consider and minimize all components' energy, including information technology (IT), cooling system, and network equipment [39]. Consolidation of multiple active VMs can be executed on either a single physical server or racks. In rack consolidation, it is also possible to save the cooling system's energy and relevant switches of idle racks by turning them off or putting them in a low-power mode [39, 40].

2.4.2.2 *Consolidation Subproblems*

Virtualization technology makes consolidation of VMs on PMs possible or consolidation of containers on both VMs and PMs. Many researchers have taken advantage of this technique to save the energy consumption in cloud data centers. The methods are categorized according to the subproblems that they explore [4]. Similar to [44], we consider five subproblems for consolidation in our proposed taxonomy:

1. Determination of underloaded PMs/VMs: if the host is considered as an underloaded host, all the VMs/containers residing on this host/VMs should be migrated from it and then host/VM and it should be put in either the low-power or turnoff mode to save energy.
2. Determination of overloaded PMs/VMs: if the host/VM is considered as an overloaded host/VM, some VMs/containers should be migrated from this host/VM to other active or reactivated hosts/VMs to prevent SLA violation.
3. VM/container selection for migration from either overloaded or underloaded hosts/VMs.
4. Determination of physical nodes that should be put in switched ON or OFF modes.
5. Finding suitable placement for migrating VMs/containers on active or reactive hosts/VMs.

2.5 Comparing the State of the Art

In this section, we present a comparison among the state of the arts on energy-efficient cloud resource management solutions taking advantage of our proposed taxonomy presented in previous section. Tables 2.2 and 2.3 present our measurements on selected papers published in 2015 through 2021 at the power management level as well as virtualization level, respectively. Table 2.2 reveals that the selected papers are compared with each other regarding their considered goals, dynamism, and granted workload and resource types. Table 2.3 analyzes the selected papers with each other respecting being DVFS-aware, as well as their considered management level, migration types, plus consolidation solution. used the acronyms shown in Tables 2.2, 2.3, and 2.4 for the considered objects, the migration costs, and considered consolidation subproblems, respectively. Besides, we utilize NM (Not Mentioned) notation in Tables 2.5 and 2.6, where the information is not provided in the reviewed paper.

The authors [69] have proposed an energy-aware combinatorial auction-based model for the resource allocation problem in clouds. They have regarded both MEC and MBP as their goals. Besides, at the dynamism level, they have considered virtualization techniques in the software category of DPM. CPU, RAM, Disk, as well as network are the active resources that they have applied in their proposed heuristic methods. The management level in [69] can be referred to as VM, VI, and cloud. Also, they have considered two distributed datacenters and solved VP and VS problems.

The authors [54] have focused on MPC and MPL as their goals. At the dynamism level, they have used DPM techniques, including DPS and virtualization. Also, they have applied the arbitrary types of workload. They further have considered CPU, RAM, and network bandwidth as cloud resources. Their approach is DVFS-aware, and their considered management levels are VM, VI, and cloud. The type of migration that they have used is pre-copy. Moreover, they have solved consolidation subproblems OPS, UPS, and VP at the server level.

Table 2.2 The notations used for resource management goals

Acronym	Goal
MEC	Minimize energy consumption
MPL	Minimize performance loss
MBP	Maximize business profit
LB	Load balancing

Table 2.3 The notations used for migration costs

Acronym	Migration cost
MTO	Migration time overhead
DT	Down time
EO	Energy overhead
TH	Throughput

Table 2.4 The notations used for consolidation subproblems

Acronym	Consolidation subproblem
OPS/OVS	Overloaded PM/VM selection
UPS/UVS	Underloaded PM/VM selection
VS/CS	VM/container selection
TPS/TVS	Turn on/off PM/VM selection
VP/CP	VM/container placement

The authors [70] have considered MPC and MPL as their goals, as shown in Table 2.2. At the dynamism level, they have adopted the virtualization method and used arbitrary workload. Plus, they have applied active resources such as CPU, RAM, Disk, and network. They have only focused on the cloud management level. For the migration types, they have regarded both MTO and EO as migration costs. They have further utilized the pre-copy migration type and solved consolidation subproblems OPS and VP at the server level.

As can be seen in Table 2.5, the authors [71] have measured MPC and MPL as their goals. At the dynamism level, they have applied the virtualization technique. Additionally, they have applied an arbitrary workload type. In the category of active resources, they have considered CPU, RAM as well as network bandwidth. As further demonstrated, their management level is the cloud level. They have considered both downtime and throughput in their proposed work as migration costs. Furthermore, they have implemented the pre-copy approach and did consolidation subproblems OPS and VS at the server level.

The goals that reflected [72] pertains to MPC and MPL. At the dynamism level, the researchers have applied the virtualization technique via arbitrary workload type. They have used CPU, RAM, and network as active resources in their work in addition to the cooling system as a passive resource. What is more, their management level is in the cloud level. They have merely concentrated on migration time overhead as the migration cost. Moreover, they have employed the pre-copy approach for migration and have solved Vms as one of the consolidation subproblem VS at the server level.

MPC and MPL are considered [73]. The researchers have practiced the virtualization technique at the dynamism level. Moreover, they have used arbitrary workload and focused on CPU, RAM, and network as their resources. They further applied their proposed approach at the cloud level and used the pre-copy migration approach, considering downtime overhead as migration cost. Moreover, they have Addressed OPS,UPS, VMs, and VMP consolidation subproblems at the server level.

The authors [41] have just contemplated MPL object. They have implemented both DPS and virtualization procedures at the dynamism level via arbitrary workload types. Besides, they have considered CPU and RAM as resource types. They have used the DVFS-aware technique at the cloud management level. They more have regarded energy overhead as migration cost, yet they have not mentioned the type of migration. The VMP Consolidation subproblem is done in this paper.

Table 2.5 Comparing the state of the arts in the power management level

Ref.	Year	Goals				DPM				SPM				Workload				Resources							
		MEC	MPL	MBP	LB	DCD	DPS	OS level	Resource	Virtualization	Circuit	Logic	Architecture	Software	Arbitrary	HPC	Batch	Real-time	CPU	RAM	Disk	Network	Cooling system	PSU	Others
		Hardware	Software																						
[69]	2021	✓		✓					✓						✓			✓	✓	✓	✓				
[54]	2017	✓	✓						✓						✓			✓	✓	✓	✓				
[70]	2015	✓							✓						✓			✓	✓	✓	✓				
[71]	2016	✓							✓						✓			✓	✓	✓	✓				
[72]	2016	✓							✓						✓			✓	✓	✓	✓				
[73]	2018	✓							✓						✓			✓	✓	✓	✓				
[41]	2017	✓							✓						✓			✓	✓	✓	✓				
[74]	2018	✓							✓						✓			✓	✓	✓	✓				
[75]	2018	✓							✓						✓			✓	✓	✓	✓				
[76]	2018	✓							✓						✓			✓	✓	✓	✓				
[77]	2018	✓							✓						✓			✓	✓	✓	✓				
[78]	2017	✓							✓						✓			✓	✓	✓	✓				
[78]	2017	✓							✓						✓			✓	✓	✓	✓				
[79]	2017	✓							✓						✓			✓	✓	✓	✓				
[80]	2018	✓							✓						✓			✓	✓	✓	✓				
[81]	2018	✓							✓						✓			✓	✓	✓	✓				
[82]	2018	✓							✓						✓			✓	✓	✓	✓				
[82]	2018	✓							✓						✓			✓	✓	✓	✓				
[83]	2018	✓							✓						✓			✓	✓	✓	✓				

(continued)

Table 2.5 (continued)

Table 2.6 Comparing the state of the art in the virtualization level

Ref.	Year	DVFS-aware	Container	VM	VI	Cloud	Management level			Migration			Consolidation		
							MTO	DT	EO	TH	Technique	Type	Level	Subproblem	
[69]	2021			✓		✓					VM		Geographical	VP-VS	
[54]	2017	✓		✓	✓	✓	✓	✓	✓	✓	Pre-copy	VM	Server	OPS-UPS-VP	
[70]	2015					✓	✓	✓	✓	✓	Pre-copy	VM	Server	UPS-VP	
[71]	2016					✓	✓	✓	✓	✓	Pre-copy	VM	Server	OPS-VS	
[72]	2016					✓	✓	✓	✓	✓	Pre-copy	VM	Server	VS	
[73]	2018					✓	✓	✓	✓	✓	Pre-copy	VM	Server	OPS-UPS-VS-VP	
[41]	2017	✓				✓	✓	✓	✓	✓	NM	VM	Server	VP	
[74]	2018					✓	✓	✓	✓	✓	NM	VM	Server	OPS-UPS-VP	
[75]	2018					✓	✓	✓	✓	✓	Pre-copy	VM	Server	VP	
[76]	2018					✓	✓	✓	✓	✓	Pre-copy	VM	Server	VS-VP	
[77]	2018					✓	✓	✓	✓	✓	NM	VM	Server	OPS-UPS-VP	
[78]	2017	✓				✓	✓	✓	✓	✓	NM	VM	Server	VP	
[97]	2017					✓	✓	✓	✓	✓	NM	VM	Server	VP	
[80]	2018					✓	✓	✓	✓	✓	NM	VM	Geographical	NM	
[81]	2018					✓	✓	✓	✓	✓	Pre-copy	VM	Server	OPS-VS-VP	
[98]	2018					✓	✓	✓	✓	✓	Pre-copy	VM	Server	TPS-VP	
[82]	2018	✓				✓	✓	✓	✓	✓	NM	VM	Server	OPS-VP	
[83]	2018	✓				✓	✓	✓	✓	✓	Pre-copy	VM	Server	VS	
[84]	2017	✓				✓	✓	✓	✓	✓				(continued)	

Table 2.6 (continued)

Ref.	Year	Management level					Migration					Consolidation		
		DVFS-aware	Container	VM	VI	Cloud	MTO	DT	EO	TH	Technique	Type	Level	Subproblem
[85]	2016					✓						VM	Server	TPS
[86]	2017					✓						VM	Geographical	VP
[87]	2018				✓	✓						VM	Geographical	VP
[79]	2017				✓	✓						VM	Geographical	VP
[88]	2017				✓	✓						VM	Geographical	VP
[89]	2020				✓	✓						VM, Container	Server	V-S-CP
[90]	2019				✓	✓						Container	Server	OPM-CS
[91]	2020				✓	✓						Pre-copy	Container	CP
[92]	2018	✓			✓	✓						Pre-copy	Container	OPS-CP
[93]	2017				✓	✓						Pre-copy	Container	CS
[94]	2019				✓	✓						Pre-copy	Container	CP
[95]	2019				✓	✓						Pre-copy, CR	Container, VM	CP
[96]	2020				✓	✓						CR/TR	Container	Geographical CP

Based on our study, the scholars [74] have paid attention to MPC and MPL goals. They have implemented the virtualization technique through an arbitrary workload to verify it at the cloud management level. Besides, they considered active resources containing CPU and RAM. As further demonstrated, they have confronted with OPS, UPS, and VMP consolidation subproblem at server level.

MEC and LB are reflected in [75]. They have come across a solution at the dynamism level by taking advantage of virtualization approach. The authors have not specified their utilized workload; conversely, the active resources they have adopted are CPU, RAM, and network. They have employed their solution not only at VM but also at cloud management level. The type of migration that they have managed pertains to pre-copy, respecting energy overhead as its cost. Another notable part that can be mentioned in this paper is working on VMP consolidation subproblem at the server level.

The goals that are analyzed in [76] goes back to MEC and MPL. Regarding dynamism level, the researchers in this paper deployed the virtualization procedure using the arbitrary workload. Plus, they only focused on active resources such as CPU and RAM. At the migration level, they have applied the pre-copy approach in the cloud management level by considering MTO as cost. In addition, both VMS and VMP consolidation subproblems are addressed in this research work.

MPC and MPL are noted [77]. Similar to previous works, they made use of the virtualization technique utilizing an arbitrary workload prototype. Active resources employed in this research can be referred to as CPU, RAM as well as network. Both VM and cloud management levels are tackled with in this paper. Although they have not mentioned the type of migration, they have considered energy overhead and migration time overhead as its costs. Researchers have also worked on OPS, UPS, and VMP consolidation subproblems at the server level.

The scholars [78] have weighed MEC in addition to MPC as their goal. The approaches at the dynamism level applied by them pertain to DPS and virtualization at both VM and cloud management level. Unlike others, they have utilized batch workload type and have adopted CPU, RAM, and Disk as the active resources in their work. Moreover, the technique addressed in this paper goes back to DVFS-aware. Also, VMP Consolidation subproblem is addressed at the server level.

The authors [79] have just concerned on MEC goal. Like others, they have used the virtualization technique at the dynamism level through the arbitrary workload. CPU and RAM can be mentioned as the active resources that they have deployed. Besides, they have examined their proposed method not only on the VM but also on the cloud management level. Ultimately, they have provided solution for solely the VMP consolidation subproblem.

The researchers [97] have contemplated MEC, MBP, as well as LB goals. At the dynamism level, they have applied the virtualization procedure. Also, they made use of arbitrary workload, yet they have not discussed their utilized resource types. Their management levels are VI and cloud. In contrast to others, their considered consolidation level is geographical.

Like most previous works, [80] have focused on MEC and MPL goals. The researchers in this paper have applied the virtualization technique at the dynamism

level via arbitrary workload. The only active resource that they have used pertains to CPU. Furthermore, the cloud is their management level. They have considered pre-copy type migration and both time and energy overhead for the migration cost. They have solved OPS, VMS, and VMP consolidation subproblems.

According to our review, [81] investigated MEC and MPL goals. They have more implemented the virtualization technique at the dynamism level through the arbitrary workload. Similar to most of the earlier works, they have regarded CPU as the active resource type, and their management level is the cloud level. As further identified, they have not mentioned the migration type, yet they have reflected energy overhead as migration cost. Another significant point addressed in this paper goes back to solving TPS and VMP subproblem at the server level.

Three goals are explored [29], including MEC, MPL, and LB. Moreover, they have applied both DPS and virtualization methods within the arbitrary workload at the dynamism level. The authors have investigated both active and passive resources in their work containing CPU, RAM as an active and cooling system as a passive type. Their proposed approach is DVFS-aware at the cloud management level. Moreover, they have examined OPS and VMP consolidation subproblems.

The authors [82] have studied solely MEC goal in their proposed method. Plus, they have employed the DVFS-aware DPS procedure. Unlike the rest of the papers, this work used real-time workload. Besides, they have applied their proposed approach at both VI and cloud levels.

As we deduced from [83], solely the MPE is regarded in this paper. Plus, they have applied the DVFS technique at the dynamism level, a subset of DPS in our taxonomy. The workload type that they have applied pertains to HPC. They have further utilized both CPU and RAM as active resources.

The scholars [84] have reflected MEC goal. The approach employed in their proposed work goes back to virtualization at the dynamism level, including server consolidation via real-time workload. Another noticeable point in their work is that they have considered not only CPU as an active resource but also RAM. Besides, they have confronted the TPS consolidation.

The MBP and LB goals are probed [85]. The authors have also employed a virtualization approach at the dynamism level, including workload consolidation at geographically distributed data centers through an arbitrary workload.

The goals that are chosen [99] can be referred to as MEC and MPL. Moreover, they have applied the virtualization technique at the dynamism level in the software approach's subset and validated their proposed technique through the arbitrary workload. Unlike the most previous works, they are cared about passive resources such as the cooling system and power supply unit. They have also handled their proposed approach at three management levels consisting of VM, VI as well as the cloud. From the cost point of view, solely an energy overhead was noticed by the authors. As further concluded, the VMP consolidation subproblem is studied by them at the geographically distributed data centers.

MEC and MBP goals are remarked by researchers [87]. The authors explored the DVFS-aware virtualization method within the arbitrary workload in their proposed approach. They have paid attention to both active resources, CPU and RAM, and

passive resources, such as cooling system and co-location interference. In addition, they have confronted with the VMP consolidation subproblem at the geographically distributed data centers.

The scholars [86] are concerned about MEC in addition to MPL goals. They have applied a virtualization approach via arbitrary workload. The active resources operated in their testbed can be mentioned as CPU, RAM, and storage. Three management levels are noticed by them, consisting of VM, VI, as well as the cloud. Another significant point in their proposed work is that they studied VMP consolidation subproblem in geographically distributed data centers.

The authors [88] have addressed MEC and LB goals. They have also implemented virtualization techniques by the arbitrary workload. They have further focused solely on active resources, including CPU, RAM, and storage.

The authors [89] have focused on both MEC and MPL goals. Besides, their proposed approach is at the virtualization level, a subset of software category in dynamic energy-efficient management. Indeed, they have considered joint VM and container consolidation and verified it by the arbitrary workload in the CloudSim simulator. Their considered type migration was pre-copy, plus energy, as well as time overhead, as their migration cost. As further investigated, they have only concentrated on an active resource, CPU. What is more, Cloud and VM are their management level, and the VMs subproblem is solved in this paper at the server level.

From the goals perspective, MEC and MPL are remarked by Xu and Buyya [90]. They proposed Brownout, which dynamically deactivates or activates optional microservices or containers to handle over- loads and reduce power consumption within a datacenter which belongs to the software level. As more explored, the authors have examined container solely in their proposed approach. Besides, they have chosen the cloud management level. The OPS and TPS consolidation subproblems that are fulfilled in this paper.

The MPL goal is noted [91]. The term of the cost that is estimated in this paper is based on VM's provisioning from the request moment until deprovisioning request. Nonetheless, for static nodes, the cost is measured according to the workload's total scheduling time. They solely contemplated container management in their proposed technique through the Kubernetes platform via batch workload. The active resources that they have applied in their work can be referred to as memory as well as CPU.

Another study that have met CaaS environment goes back to [92] that measured the MEC and LB goals. The authors have proposed a renewable energy-aware multi-indexed job classification and scheduling scheme using CaaS for data centers. Thus, they have addressed CP and OPS as a subset of the consolidation problem. Plus, they have verified their proposed scheme utilizing extensive simulations for 3 geo-distributed data centers having 200 heterogeneous servers via Google workload traces. They have noted both passive and active resources including cooling system, CPU, and network.

As we perceived from [93] the MPL goal including, improving the utilization rate of resources and enhancing the user experience are the targets that the author covered. They have proposed a live container migration algorithm named Gray-

Markov prediction model to foresee the probability that dirty pages are modified again. Further, the type of migration is pre-copy, and lessening both downtime, iteration as well as total copy time are considered as migration costs in their method.

The authors [94], have focused on consolidation technique in containerized datacenters. They have investigated a new algorithm for container allocation concerning both MEC and MPL goals. Besides, they have proposed a mathematical model which estimates the energy consumption of containers executed in VMs. The proposed migration technique is performed only if the migration cost capable of recovering to save energy. They have further applied CPU and RAM as active resources. The consolidation subproblem that they have considered is CP with considering of EO as migration cost at the server level.

The goals which are addressed [95] go back to both MEC and MBP. The main contribution of this work is that they have developed an experimental setup to execute and assess the performance of the Linux Container Hypervisor (LXD) and checkpoint/restore (CR) container-based migration methods and comparing it with a pre-copy VM migration scheme. Their applied workload is real-time while considering CPU and RAM as active resources in their proposed technique. They have adopted LXD/CR mechanism for container migration. The CP subproblem has addressed by Bhardwaj and Krishna [95] at the server level.

The goal that is targeted [96] can be referred to as MPL. The authors of this paper have applied the virtualization method at the OS level. Further, they considered both CPU and memory as actives resources to evaluate their proposed approach at container level management. They have concerned about DT as well as TH cost in CR/TR migration. Plus, they have considered the CP subproblem at the geographically distributed data centers.

It can be inferred from Table 2.5 that not only minimizing power consumption but also the performance loss were the main aims reflected in the state of the arts for resource management in cloud environments. Nonetheless, few papers consisting of, [75, 97, 98], as well as [85, 92, 96], and [88] regard the load-balancing goal along with declining power consumption and performance loss. Also, [69, 83, 85, 87, 97], and [95] counted maximizing business profit. As further demonstrated, one of the most remarkable points goes back to the researcher's attention to DPM techniques as an offspring of its superiority compared with SPM techniques. What is more, only [96] and [95] utilized resource management techniques at the OS level. further, none of the papers employed hybrid simultaneous hardware and software DPM techniques except [54]. Another significant case that can be mentioned as considering only arbitrary workload type by most of the researchers in their work. Indeed, they neglected batch, real-time, as well as HPC workload types except [78] and [91] papers that noticed batch workload type, plus [82, 84] and [95] papers that marked real-time workload. Ultimately, solely the authors in [83] validated their proposed approach through HPC workload type. Moreover, it can be inferred from Table 2.5 that most of the recently selected papers only analyze the power consumed over active resources such as CPU, memory, network, and disk storage. While, most of the state of the arts regarded the power consumed in passive peripherals. The researchers in [72, 86, 98], and [87] considered the power consumed in the cooling

systems. To add, [86] considered only passive resources, including cooling systems and PSU, and [87] focused on both the cooling system and other passive peripherals.

As can be seen in Table 2.6, a few papers are addressed DVFS-aware approach in their proposed cloud resource management solutions. Further, most of the articles introduced resource management solutions only at the cloud level without considering either VM or VI level. By contrast, just [54] and [87] regarded all three management levels, including VM, VI, as well as cloud. It can also be deduced from Table 2.6 that different migration costs are considered in the selected papers. As more displayed, all the studied articles applied the pre-copy migration procedure in their solutions except [95] and [96] applied CR/TR approach. Additionally, [79, 80, 86–88, 92, 96], and [69] addressed consolidation technique at the geographical level, yet the rest of the papers concentrated only on the server level. As further explored, [89–95], and [96] remarked container level management along with container consolidation in their work; among them merely [89] applied joint VM and container migration approach, but others employed VM and container migration separately.

2.6 Future Scope and Conclusion

This section presents some research issues and challenges concerning energy-efficient resource management methods in the cloud environments. Although notable progress has been accomplished in applying containerization to the cloud computing systems and the adaptive management of resources and applications is widely developed; there are still many research gaps and challenges in this area needed to be further investigated as discussed below.

- **Multiple system resources:** Due to the broad admission of multi-core CPUs, developing energy-efficient resource management approaches plays a crucial role in leveraging such architectures. To optimize a data center’s operation, it is critical to reflect and lessen all energy elements consumption, including the cooling system and power supply units, as passive resources. To add, RAM, disk storage, and network equipment as active resources are usually overlooked by researchers. From the perspective of active resources, current works mostly focus on CPU. More resource types, like memory, network, storage, and GPU need to be regarded as parameters to create more comprehensive resource management.
- **Rack consolidation and geographically distributed data center:** Many big data analysis applications involve analyzing a large volume of data generated in a geographical-distributed data center. Besides, plenty of data-intensive applications, such as social networks, involve large data sets in multiple geographically distributed cloud data centers. As a case in point, Facebook receives terabytes of text, image, and video data every day from users worldwide. Another noticeable future research direction goes back to the exploration of cloud environment geographically distributed data centers and rack consolidation in addition to

server and VM consolidation to make it possible to provide more reliable services in greener data centers.

- **System workload:** Most of the current papers applied arbitrary workloads in their study; conversely, this is a crucial issue to consider other workload types in addition to arbitrary workloads containing the batch, HPC, plus real-time application workloads.
- **Security and privacy:** Over the years, the ever-increasing growth of cloud data centers utilized by famous corporations such as Google, Facebook, and Microsoft can result in rise of new different administrative and security. So, addressing the security concerns which are become more and more complicated by development of new containerized services, such as Distributed Denial of Service (DDoS), has become an important issue to be considered in future research directions.
- **Cognitive approach contributing to Joint VM and container consolidation:** Container consolidation is an evolving technology which has a great deal better performance than VM consolidation in the light of energy consumption and performance loss. Besides, the research in [89] has justified that joint VM and container consolidation outperforms individual VM or container consolidation approaches, regarding energy consumption and QoS. By contrast, applying artificial intelligence, or machine learning, to make a cognitive decision for simultaneous migration of both VM and container is a hot research topic.
- **Container security:** There are some levels for container ecosystem security including image, registry, orchestrator, container, and host OS. For instance, container technologies like Docker and Kubernetes accelerate the development and deployment of application; hence, their security issues play a notable role in software development and cloud industries. The research in this field can be directed in two levels including protecting a container from the security attacks of its applications and protecting a physical server from the security attacks of its containers.

2.6.1 Conclusion

In this chapter, we proposed a holistic taxonomy for cloud resource management and then we surveyed 32 recent articles in the literature related to energy-efficient resource management procedures based on the proposed taxonomy. Since the research has dramatically advanced in the field of resource management techniques in cloud computing, there is a demand for a systematic review to evaluate, update, and combine the existing literature. On the one hand, the requests to access cloud services are ever-increasing. On the other hand, cloud services are hosted on massive data centers consuming enormous electrical power. Thereby, efficient energy-aware resource management has become a matter of great concern in cloud environments from single server to data centers and Clouds. This chapter centered on the energy-aware resource management problem in cloud environments and proposed a novel taxonomy and solutions classification. More precisely, first we proposed a holistic taxonomy for energy-aware resource management in cloud

environment. Then, we surveyed 32 articles in the literature related to energy-efficient resource management and categorized them according to their proposed solution based on a scientific comparison. Eventually, we opened up new research directions by gap analysis and suggesting major drawbacks in the current literature on energy-aware cloud resource management. This chapter rose up to four research questions and answered them, including: (1) The definition of energy-aware cloud resource management. (2) The taxonomy of power-aware resource management solutions. (3) The parameters considered in the literature for energy-aware resource management in cloud environments. (4) The open research directions for energy-aware cloud resource management.

This chapter appears to yield several innovations. One of the most remarkable novelties pertains to presenting a scientific taxonomic survey of recent literature over seven years, 2015–2021. This chapter categorized recent research progressions across four main categories: dynamism, workload types, resources, as well as goals. In the light of comparison, we classified articles according to their suggested solution for energy-aware cloud resource management. Moreover, the workload part characterizes literature regarding the considered workload types in the proposed solution. Considered resources revealed the type and variety of taken into consideration resources in the literature. Ultimately, the goal level categorized the literature regarding their main objectives. Further, the dynamism level itself was divided into static and dynamic power management levels. Three subcategories were chosen for dynamic solutions such as hardware, software, plus hybrid levels. Another item belongs to software, which was divided into three subcategories: operating system, virtualization, as well as resource throttling levels. DVFS-aware, management level, migration type and technique, and consolidation were the subcategories of virtualization. Another noticeable novelty can be referred to as regarding container migration in addition to VM migration in the determined period. To add, this chapter categorized the selected papers according to the proposed taxonomy and provided a scientific comparison. Finally, it opened up new research directions by gap analysis and suggesting major drawbacks in the current literature on energy-aware cloud resource management.

Acknowledgments We thank Professor Rodrigo N Calheiros and Kiarash Geraili for their suggestions on improving the manuscript.

References

1. M.L. Badger, T. Grance, R. Patt-Corner, J.M. Voas, *Cloud Computing Synopsis and Recommendations*. (National Institute of Standards & Technology, Gaithersburg, 2012)
2. P. Mell, T. Grance, et al., *The NIST Definition of Cloud Computing* (2011)
3. A. Khosravi, R. Buyya, Energy and carbon footprint-aware management of geo-distributed cloud data centers: a taxonomy state of the art, and future directions, in *Sustainable Development: Concepts, Methodologies, Tools, and Applications* (IGI Global, Pennsylvania, 2018), pp. 1456–1475

4. S.F. Piraghaj, Energy-efficient management of resources in enterprise and container-based clouds. PhD, University of Melbourne, Melbourne, Australia, 2016
5. S.F. Piraghaj, A.V. Dastjerdi, R.N. Calheiros, R. Buyya, Containercloudsim: An environment for modeling and simulation of containers in cloud data centers. *Softw. Pract. Exp.* **47**(4), 505–521 (2017)
6. X. You, Y. Li, M. Zheng, C. Zhu, L. Yu, A survey and taxonomy of energy efficiency relevant surveys in cloud-related environments. *IEEE Access* **5**, 14066–14078 (2017)
7. A. Beloglazov, R. Buyya, Y.C. Lee, A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, in *Advances in Computers*, vol. 82 (Elsevier, Amsterdam, 2011), pp. 47–111
8. R. Buyya, S.N. Srivama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L.M. Vaquero, M.A. Netto, et al., A manifesto for future generation cloud computing: research directions for the next decade. *ACM Comput. Surveys (CSUR)* **51**(5), 1–38 (2018)
9. R. Yadav, W. Zhang, K. Li, C. Liu, A.A. Laghari, Managing overloaded hosts for energy-efficiency in cloud data centers. *Cluster Comput.* **24**, 1–15 (2021)
10. D. Merkel, Docker: lightweight Linux containers for consistent development and deployment. *Linux J.* **2014**(239), 2 (2014)
11. N. Gholipour, N. Shoaebi, E. Arianyan, An energy-aware dynamic resource management technique using deep q-learning algorithm and joint VM and container consolidation approach for green computing in cloud data centers, in *International Symposium on Distributed Computing and Artificial Intelligence* (Springer, Berlin, 2020), pp. 227–233
12. S. Sultan, I. Ahmad, T. Dimitriou, Container security: issues, challenges, and the road ahead. *IEEE Access* **7**, 52976–52996 (2019)
13. A. Bhardwaj, C.R. Krishna, Virtualization in cloud computing: Moving from hypervisor to containerization—a survey. *Arab. J. Sci. Eng.* **46**, 1–17 (2021)
14. E. Casalicchio, S. Iannucci, The state-of-the-art in container technologies: Application, orchestration and security. *Concurrency Comput. Pract. Exp.* **32**(17), e5668 (2020)
15. N. Akhter, M. Othman, Energy aware resource allocation of cloud data center: review and open issues. *Cluster Comput.* **19**(3), 1163–1182 (2016)
16. S. Singh, A. Swaroop, A. Kumar, et al., A survey on techniques to achieve energy efficiency in cloud computing, in *2016 International Conference on Computing, Communication and Automation (ICCCA)* (IEEE, Piscataway, 2016), pp. 1281–1285
17. S. Kaur, S. Bawa, A review on energy aware VM placement and consolidation techniques, in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3 (IEEE, Piscataway, 2016), pp. 1–7
18. M. Zakarya, L. Gillam, An energy aware cost recovery approach for virtual machine migration, in *International Conference on the Economics of Grids, Clouds, Systems, and Services* (Springer, Berlin, 2016), pp. 175–190
19. Q. Shaheen, M. Shiraz, S. Khan, R. Majeed, M. Guizani, N. Khan, A.M. Aseere, Towards energy saving in computational clouds: taxonomy review, and open challenges. *IEEE Access* **6**, 29407–29418 (2018)
20. D. Hazra, A. Roy, S. Midya, K. Majumder, Energy aware task scheduling algorithms in cloud environment: A survey, in *Smart Computing and Informatics* (Springer, Berlin, 2018), pp. 631–639
21. N. Hamdi, W. Chainbi, A survey on energy aware VM consolidation strategies. *Sustain. Comput. Inf. Syst.* **23**, 80–87 (2019)
22. S. Puhan, D. Panda, B.K. Mishra, Energy efficiency for cloud computing applications: A survey on the recent trends and future scopes, in *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)* (IEEE, Piscataway, 2020), pp. 1–6
23. Q. Zhou, M. Xu, S.S. Gill, C. Gao, W. Tian, C. Xu, R. Buyya, Energy efficient algorithms based on VM consolidation for cloud computing: Comparisons and evaluations,” in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)* (IEEE, Piscataway, 2020), pp. 489–498

24. M. Zakarya, L. Gillam, Energy efficient computing, clusters, grids and clouds: a taxonomy and survey. *Sustain. Comput. Inf. Syst.* **14**, 13–33 (2017)
25. X. Zhan, S. Reda, Power budgeting techniques for data centers. *IEEE Trans. Comput.* **64**(8), 2267–2278 (2015). <https://doi.org/10.1109/TC.2014.2357810>
26. D.G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation* (Cambridge University Press, Cambridge, 2015)
27. S. Khan, K.A. Shakil, M. Alam, Big data computing using cloud-based technologies, challenges and future perspectives (2017). Preprint arXiv:1712.05233
28. P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, K. Tzoumas, Apache flink: stream and batch processing in a single engine. *Bull. IEEE Comput. Soc. Tech. Comm. Data Eng.* **36**(4), 28–38 (2015)
29. M.A. Netto, R.N. Calheiros, E.R. Rodrigues, R.L. Cunha, R. Buyya, HPC cloud for scientific and business applications: Taxonomy vision, and research challenges. *ACM Comput. Surv.* **51**(1), 1–29 (2018)
30. I. Foster, C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure* (Elsevier, Amsterdam, 2003)
31. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing. *Commun. ACM* **53**(4), 50–58 (2010)
32. R. Buyya, C. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: vision, hype, and reality for start [connect central controller using star topology network delivering computing as the 5th utility. *Futu. Gener. Comput. Syst.* **25**, 599–616 (2009)
33. E.K. Lee, H. Viswanathan, D. Pompili, Proactive thermal-aware resource management in virtualized HPC cloud datacenters. *IEEE Trans. Cloud Comput.* **5**(2), 234–248 (2015)
34. Q. Fang, J. Wang, Q. Gong, M. Song, Thermal-aware energy management of an HPC data center via two-time-scale control. *IEEE Trans. Ind. Inf.* **13**(5), 2260–2269 (2017)
35. S. Malik, F. Huet, Adaptive fault tolerance in real time cloud computing, in *2011 IEEE World Congress on Services* (IEEE, Piscataway, 2011), pp. 280–287
36. J.A. Stankovic, Misconceptions about real-time computing: a serious problem for next-generation systems. *Computer* **21**(10), 10–19 (1988)
37. K.H. Kim, A. Beloglazov, R. Buyya, Power-aware provisioning of cloud resources for real-time services, in *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science* (2009), pp. 1–6
38. R. Sudeepa, H. Guruprasad, Resource allocation in cloud computing. *Int. J. Modern Commun. Technol. Res.* **2**(4), 265–808 (2014)
39. S. Esfandiarpoor, A. Pahlavan, M. Goudarzi, Structure-aware online virtual machine consolidation for datacenter energy improvement in cloud computing. *Comput. Electr. Eng.* **42**, 74–89 (2015)
40. C. Pahl, Containerization and the PaaS cloud. *IEEE Cloud Comput.* **2**(3), 24–31 (2015)
41. P. Arroba, J.M. Moya, J.L. Ayala, R. Buyya, Dynamic voltage and frequency scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers. *Concurrency Comput. Pract. Exp.* **29**(10), e4067 (2017)
42. V.M. Raj, R. Shriram, Power management in virtualized datacenter—a survey. *J. Netw. Comput. Appl.* **69**, 117–133 (2016)
43. A. Al-Dulaimy, W. Itani, A. Zekri, R. Zantout, Power management in virtualized data centers: state of the art. *J. Cloud Comput.* **5**(1), 6 (2016)
44. A. Beloglazov, Energy-efficient management of virtual machines in data centers for cloud computing, Ph.D. Dissertation, 2013
45. H. David, C. Fallin, E. Gorbatov, U.R. Hanebutte, O. Mutlu, Memory power management via dynamic voltage/frequency scaling, in *Proceedings of the 8th ACM International Conference on Autonomic Computing* (2011), pp. 31–40
46. T. Kaur, I. Chana, Energy efficiency techniques in cloud computing: a survey and taxonomy. *ACM Comput. Surv.* **48**(2), 1–46 (2015)
47. Q. Deng, D. Meisner, A. Bhattacharjee, T.F. Wenisch, R. Bianchini, Coscale: Coordinating CPU and memory system DVFS in server systems, in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture* (IEEE, Piscataway, 2012), pp. 143–154

48. A. Mahajan¹, A. Ganpati, A study of energy efficiency techniques in cloud computing. *Int. J. Comput. Sci. Mobile Comput.* **3**(8), 707–712 (2014)
49. F.D. Rossi, M.G. Xavier, C.A. De Rose, R.N. Calheiros, R. Buyya, Ecco: performance-aware energy-efficient cloud data center orchestration. *J. Netw. Comput. Appl.* **78**, 83–96 (2017)
50. M.G. Xavier, F.D. Rossi, C.A. De Rose, R.N. Calheiros, D.G. Gomes, Modeling and simulation of global and sleep states in ACPI-compliant energy-efficient cloud environments. *Concurrency Comput. Pract. Exp.* **29**(4), e3839 (2017)
51. C. Pahl, A. Brogi, J. Soldani, P. Jamshidi, Cloud container technologies: a state-of-the-art review. *IEEE Trans. Cloud Comput.* **7**, 677–692 (2017)
52. I. Jimenez, C. Maltzahn, A. Moody, K. Mohror, J. Lofstead, R. Arpacı-Dusseau, A. Arpacı-Dusseau, The role of container technology in reproducible computer systems research, in *2015 IEEE International Conference on Cloud Engineering* (IEEE, Piscataway, 2015), pp. 379–385
53. R. Kumar, S. Charu, An importance of using virtualization technology in cloud computing. *Global J. Comput. Technol.* **1**(2), 56–60 (2015)
54. E. Arianyan, H. Taheri, V. Khoshdel, Novel fuzzy multi objective DVFS-aware consolidation heuristics for energy and SLA efficient resource management in cloud data centers. *J. Netw. Comput. Appl.* **78**, 43–61 (2017)
55. H. Yamada, Survey on mechanisms for live virtual machine migration and its improvements. *Inf. Media Technol.* **11**, 101–115 (2016)
56. N.J. Kansal, I. Chana, Energy-aware virtual machine migration for cloud computing-a firefly optimization approach. *J. Grid Comput.* **14**(2), 327–345 (2016)
57. A.-Y. Son, E.-N. Huh, Migration method for seamless service in cloud computing: Survey and research challenges, in *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)* (IEEE, Piscataway, 2016), pp. 404–409
58. M. Liaqat, S. Ninoriya, J. Shuja, R.W. Ahmad, A. Gani, Virtual machine migration enabled cloud resource management: A challenging task (2016). Preprint arXiv:1601.03854
59. H. Liu, H. Jin, C.-Z. Xu, X. Liao, Performance and energy modeling for live migration of virtual machines. *Cluster Comput.* **16**(2), 249–264 (2013)
60. S. Akram, S. Ghaleb, S. Ba, V. Siva, Survey study of virtual machine migration techniques in cloud computing. *Int. J. Comput. Appl.* **177**, 18–22 (2017)
61. S. Sharma, M. Chawla, A technical review for efficient virtual machine migration, in *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies* (IEEE, Piscataway, 2013), pp. 20–25
62. H. Liu, H. Jin, X. Liao, C. Yu, C.-Z. Xu, Live virtual machine migration via asynchronous replication and state synchronization. *IEEE Trans. Parallel Distrib. Syst.* **22**(12), 1986–1999 (2011)
63. D. Kapil, E. S. Pilli, R.C. Joshi, Live virtual machine migration techniques: Survey and research challenges, in *2013 3rd IEEE International Advance Computing Conference (IACC)* (IEEE, Piscataway, 2013), pp. 963–969
64. G.W. Dunlap, S.T. King, S. Cinar, M.A. Basrai, P.M. Chen, Revirt: enabling intrusion analysis through virtual-machine logging and replay. *ACM SIGOPS Oper. Syst. Rev.* **36**(SI), 211–224 (2002)
65. S. Gursharan, and P. Singh. A taxonomy and survey on container migration techniques in cloud computing. *Sustainable Development Through Engineering Innovations*, **113**, 419–429 (2021)
66. T. Chaabouni, M. Khemakhem, Energy management strategy in cloud computing: A perspective study. *J. Supercomput.* **74**(12), 6569–6597 (2018)
67. M.C. Silva Filho, C.C. Monteiro, P.R. Inácio, M.M. Freire, Approaches for optimizing virtual machine placement and migration in cloud environments: a survey. *J. Parallel Distrib. Comput.* **111**, 222–250 (2018)
68. F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, H. Tenhunen, Utilization prediction aware vm consolidation approach for green cloud computing, in *2015 IEEE 8th International Conference on Cloud Computing* (IEEE, Piscataway, 2015), pp. 381–388
69. M. Gamsız, A.H. Özer, An energy-aware combinatorial virtual machine allocation and placement model for green cloud computing. *IEEE Access* **9**, 18625–18648 (2021)

70. E. Arianyan, H. Taheri, S. Sharifian, Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers. *Comput. Electr. Eng.* **47**, 222–240 (2015)
71. E. Arianyan, H. Taheri, S. Sharifian, Multi target dynamic VM consolidation in cloud data centers using genetic algorithm. *J. Inf. Sci. Eng.* **32**(6), 1575–1593 (2016)
72. E. Arianyan, H. Taheri, S. Sharifian, Novel heuristics for consolidation of virtual machines in cloud data centers using multicriteria resource management solutions. *J. Supercomput.* **72**(2), 688–717 (2016)
73. E. Arianyan, H. Taheri, S. Sharifian, M. Tarighi, New six-phase on-line resource management process for energy and SLA efficient consolidation in cloud data centers. *Int. Arab J. Inf. Technol.* **15**(1), 10–20 (2018)
74. A. Aryania, H.S. Aghdasi, L.M. Khanli, Energy-aware virtual machine consolidation algorithm based on ant colony system. *J. Grid Comput.* **16**(3), 477–491 (2018)
75. T.H. Duong-Ba, T. Nguyen, B. Bose, T.T. Tran, A dynamic virtual machine placement and migration scheme for data centers. *IEEE Trans. Serv. Comput.* **14**, 329–341 (2018)
76. H. Li, W. Li, H. Wang, J. Wang, An optimization of virtual machine selection and placement by using memory content similarity for server consolidation in cloud. *Futur. Gener. Comput. Syst.* **84**, 98–107 (2018)
77. S. Mustafa, K. Bilal, S.U.R. Malik, S.A. Madani, Sla-aware energy efficient resource management for cloud environments. *IEEE Access* **6**, 15004–15020 (2018)
78. F. Teng, L. Yu, T. Li, D. Deng, F. Magoulès, Energy efficiency of VM consolidation in IaaS clouds. *J. Supercomput.* **73**(2), 782–809 (2017)
79. F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, V.C. Leung, Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm. *IEEE Syst. J.* **12**(2), 1688–1699 (2017)
80. H. Wang, H. Tianfield, Energy-aware dynamic virtual machine consolidation for cloud datacenters. *IEEE Access* **6**, 15259–15273 (2018)
81. M. Zakarya, An extended energy-aware cost recovery approach for virtual machine migration. *IEEE Syst. J.* **13**(2), 1466–1477 (2018)
82. G.L. Stavrinides, H.D. Karatza, Energy-aware scheduling of real-time workflow applications in clouds utilizing DVFS and approximate computations, in *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)* (IEEE, Piscataway, 2018), pp. 33–40
83. A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, L. Benini, Scheduling-based power capping in high performance computing systems. *Sustain. Comput. Inf. Syst.* **19**, 1–13 (2018)
84. S. Mazumdar, M. Pranzo, Power efficient server consolidation for cloud data center. *Futur. Gener. Comput. Syst.* **70**, 4–16 (2017)
85. A. Forestiero, C. Mastrianni, M. Meo, G. Papuzzo, M. Sheikhalishahi, Hierarchical approach for efficient workload management in geo-distributed data centers. *IEEE Trans. Green Commun. Netw.* **1**(1), 97–111 (2016)
86. A. Khosravi, A. Nadjaran Toosi, R. Buyya, Online virtual machine migration for renewable energy usage maximization in geographically distributed cloud data centers. *Concurrency Comput. Pract. Exp.* **29**(18), e4125 (2017)
87. N. Hogade, S. Pasricha, H.J. Siegel, A.A. Maciejewski, M.A. Oxley, E. Jonardi, Minimizing energy costs for geographically distributed heterogeneous data centers. *IEEE Trans. Sustain. Comput.* **3**(4), 318–331 (2018)
88. H. Teyeb, N.B. Hadj-Alouane, S. Tata, A. Balma, Optimal dynamic placement of virtual machines in geographically distributed cloud data centers. *Int. J. Cooperative Inf. Syst.* **26**(03), 1750001 (2017)
89. N. Gholipour, E. Arianyan, R. Buyya, A novel energy-aware resource management technique using joint VM and container consolidation approach for green computing in cloud data centers. *Simul. Model. Pract. Theory* **104**, 102–127 (2020)
90. M. Xu, R. Buyya, Brownoutcon: a software system based on brownout and containers for energy-efficient cloud computing. *J. Syst. Softw.* **155**, 91–103 (2019)

91. M. Rodriguez, R. Buyya, Container orchestration with cost-efficient autoscaling in cloud computing environments, in *Handbook of Research on Multimedia Cyber Security* (IGI Global, Pennsylvania, 2020), pp. 190–213
92. N. Kumar, G.S. Aujla, S. Garg, K. Kaur, R. Ranjan, S.K. Garg, Renewable energy-based multi-indexed job classification and container management scheme for sustainability of cloud data centers. *IEEE Trans. Ind. Inf.* **15**(5), 2947–2957 (2018)
93. H. Nie, P. Li, H. Xu, L. Dong, J. Song, R. Wang, Research on optimized pre-copy algorithm of live container migration in cloud environment, in *International Symposium on Parallel Architecture, Algorithm and Programming* (Springer, Berlin, 2017), pp. 554–565
94. A.A. Khan, M. Zakarya, R. Buyya, R. Khan, M. Khan, O. Rana, An energy and performance aware consolidation technique for containerized datacenters. *IEEE Trans. Cloud Comput.* **9**, 1305–1322 (2019)
95. A. Bhardwaj, C.R. Krishna, A container-based technique to improve virtual machine migration in cloud computing. *IETE J. Res.* **68**(1), 401–416 (2019)
96. P.S. Junior, D. Miorandi, G. Pierre, Stateful container migration in geo-distributed environments, in *CloudCom 2020-12th IEEE International Conference on Cloud Computing Technology and Science* (2020)
97. A.N. Toosi, C. Qu, M.D. de Assunção, R. Buyya, Renewable-aware geographical load balancing of web applications for sustainable data centers. *J. Netw. Comput. Appl.* **83**, 155–168 (2017)
98. M.A. Oxley E. Jonardi, S. Pasricha, A.A. Maciejewski, H.J. Siegel, P.J. Burns, G.A. Koenig, Rate-based thermal, power, and co-location aware resource management for heterogeneous data centers. *J. Parallel Distribut. Comput.* **112**, 126–139 (2018)
99. A. Khosravi, L.L. Andrew, R. Buyya, Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed cloud data centers. *IEEE Trans. Sustain. Comput.* **2**(2), 183–196 (2017)

Chapter 3

Multi-objective Dynamic Virtual Machine Consolidation Algorithm for Cloud Data Centers with Highly Energy Proportional Servers and Heterogeneous Workload



Md Anit Khan, Andrew P. Paplinski, Abdul Malik Khan, Manzur Murshed, and Rajkumar Buyya

Contents

3.1	Introduction	70
3.2	Related Work	75
3.3	Modelling Stochastic VM Release Time, Notations Used and Important Concepts	76
3.3.1	Modelling Stochastic VM Release Time, PM Release Time and Notations Used	76
3.3.2	Modelling Resource Utilization and Constraints	77
3.3.3	Modelling Energy Consumption	82
3.3.4	Objective Functions	82
3.4	Proposed Solution	83
3.4.1	Two Phases of SRTDVMC Algorithm	84
3.4.2	Characteristics of Proposed Algorithm	88
3.5	Performance Evaluation	89
3.5.1	Experimental Setup	89
3.5.2	Performance Metrics and Workload Data	90
3.6	Simulation Results Analysis	92
3.6.1	Energy Consumption	92
3.6.2	VM Migration	96
3.7	Observations	100
3.8	Conclusions and Future Work	102
3.8.1	Conclusions	102
3.8.2	Future Work	103
	References	104

M. A. Khan · A. P. Paplinski · A. M. Khan

Faculty of Information Technology, Monash University, Clayton, VIC, Australia

e-mail: andrew.paplinski@monash.edu; malik.khan@monash.edu

M. Murshed

School of Information Technology, Federation University Australia, Churchill, VIC, Australia
e-mail: manzur.murshed@federation.edu.au

R. Buyya (✉)

CLOUDS Lab, School of Computing and Information Systems, The University of Melbourne,
Melbourne, VIC, Australia
e-mail: rbuyya@unimelb.edu.au

3.1 Introduction

Data Center (DC) construction has increased 47% in 2017, resulting in consumption of 3% of the world's energy [1] – equivalent to the energy consumption by the airline industry [2]. Research on DCs of the United States has revealed that in 2014, the sum of energy usage by all DCs in the United States was 70 billion kWh – accounting for 1.8% of the country's total energy usage [3]. More importantly, the trend of energy consumption of DCs highlights that energy consumption is rising every year and is expected to increase by 4% from 2014 to 2020. The energy consumption by the DCs of the US data centers is estimated to reach up to 73 billion kWh in 2020. Subsequently, countries across the world have come forward to address the challenge of increasing energy consumption by DC [4]. Joint Research Centre (JRC) of European Commission has formed the Code of Conduct for energy efficiency in CDC with an aim to inform and encourage DC owners and operators to effectively level off the energy consumption [5]. Such recent literature highlights great significance of energy consumption minimization of CDC and its relevance to present day.

In 2018, JRC has proposed a detailed guideline in relation to the best practices to limit the energy consumption of DCs [6]. One of the practices strongly advised by JRC is called VM Consolidation (VMC). The essence of VMC is to consolidate VMs in minimum possible number of PMs, so that the number of unused PMs having no VMs can be maximized and turn them into lower energy consumption state, such as sleep state or turned off state. Therefore, energy consumption can be reduced regardless of the types of PMs and regardless of the types of energy sources used to power those PMs. The data representing VMRT (i.e., the lifespan of a VM) of real VMs resided in Nectar Cloud [7] highlighted through Figs. 3.1, 3.2, and 3.3 exhibit that VMRT varies from one VM to another, whereas existing DVMC algorithms, such as [8–23], either mentioned that homogeneous VMRT has been used for experiments or have not articulated any assumption made on VMRT. Through experimenting with homogenous VMRT (i.e., all VMs are assigned with tasks of equal length in terms of task finishing time) using CloudSim [24, 25] as used by respective researchers, we have obtained similar results as presented in respective literature. However, in real scenario, CDC consists of VMs with heterogeneous VMRT. In other words, VMs are assigned with tasks of unequal lengths in terms of task finishing time.

It is critical to note that VMRT has strong impact on CDC energy consumption. Larger VMRT increases CDC energy consumption, since the total resource utilization by a VM grows as the lifespan of that VM grows. Hence, without considering heterogeneous VMRT, the overall estimation about the impact of any DVMC algorithm on CDC energy consumption and subsequent QoS would be less accurate. In contrast with traditional DVMC algorithms which only consider homogeneous VMRT, *RTDVMC* takes the heterogeneous VMRT into consideration [26]. Nevertheless, several limitations exist with *RTDVMC* as elucidated in the following:

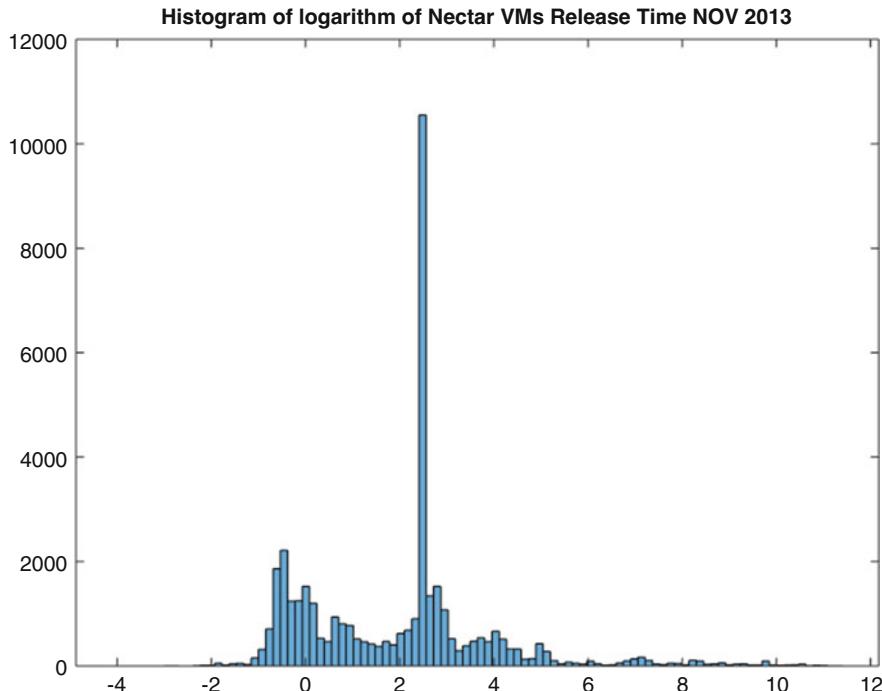


Fig. 3.1 Histogram of logarithm of release time (second) of VMs created in Nectar Cloud in November 2013

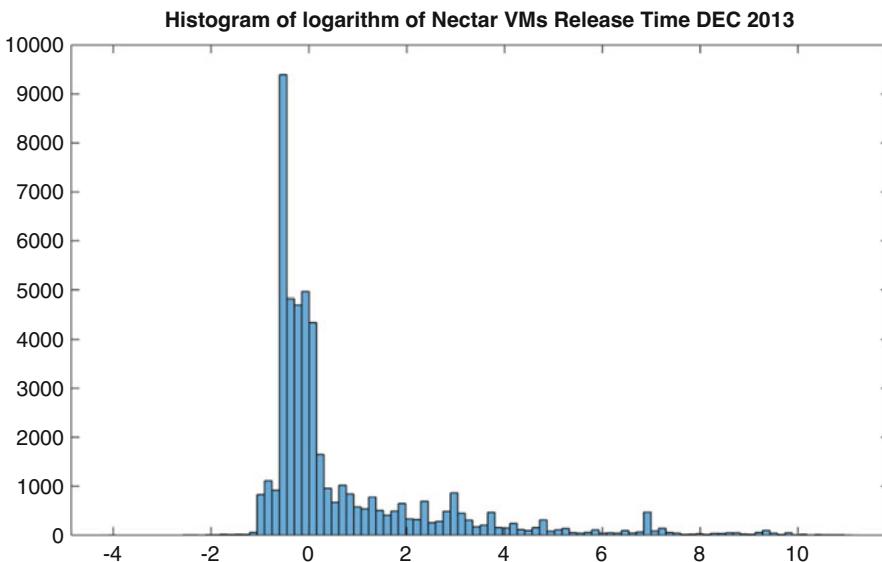


Fig. 3.2 Histogram of logarithm of release time (second) of VMs created in Nectar Cloud in December 2013

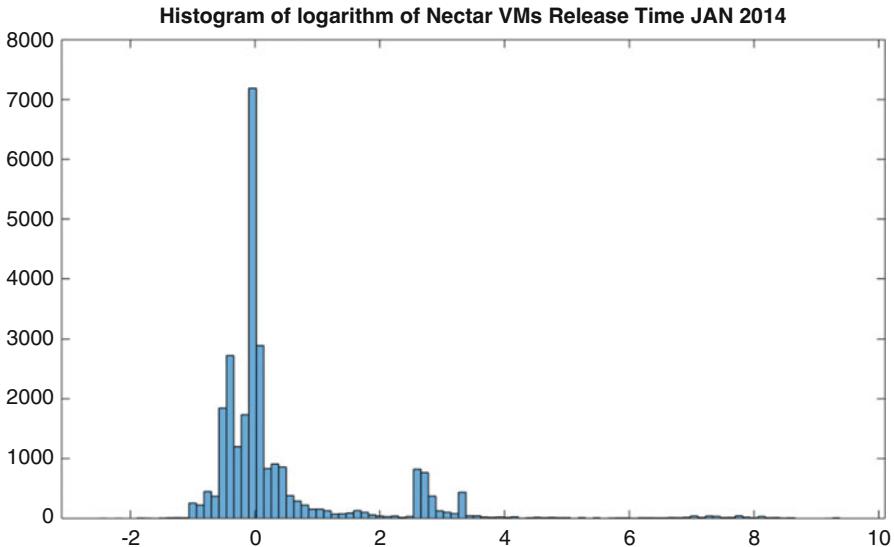


Fig. 3.3 Histogram of logarithm of release time (second) of VMs created in Nectar Cloud in January 2014

- To the best of our level of understanding of such existing DVMC algorithms as [8–23, 27], including *RTDVMC* [26] are developed based on the fundamental underlying assumption that optimal energy efficiency is achievable at maximum PM resource utilization. The basis of such claim against these existing DVMC algorithms is that these algorithms use legacy PMs, such as HP ProLiant ML110 G4 [28] and HP ProLiant ML110 G5 [29] for performance validation. From Fig. 3.4, we can see that for those legacy PMs the energy efficiency (i.e., Ratio of Power to Throughput) is as high as its incurred utilization. In stark contrast, for modern highly energy proportional PMs, such as Dell PowerEdgeR940 (Intel Xeon Platinum 8180, 112 cores → 25,000 MHz, 384 GB) [30], HP ProLiant DL560 Gen10 (Intel Xeon Platinum 8180, 112 cores → 25,000 MHz, 384 GB) [31], and HP ProLiant ML350 Gen10 (Intel Xeon Platinum 8180, 28 cores → 25,000 MHz, 192 GB) [32], energy efficiency rather drops beyond 70% utilization intervals, as delineated in Fig. 3.4. The underlying reason is that, while the throughput increases uniformly with the increase of load or utilization (Fig. 3.5), the power consumption of modern PMs beyond 70% utilization level rises such drastically (Fig. 3.6) that it exceeds the respective linear increase of throughput. Consequently, the ratio of throughput to power consumption, translated as energy efficiency, drops [33]. As such, several researchers [34, 35] argue that consolidation towards maximum increase of PM resource utilization does not feature the optimal minimization of CDC energy consumption with new generation of highly energy proportional PMs. Based on our literature study [8–23, 27], no DVMC algorithm including *RTDVMC* is found to address this issue.

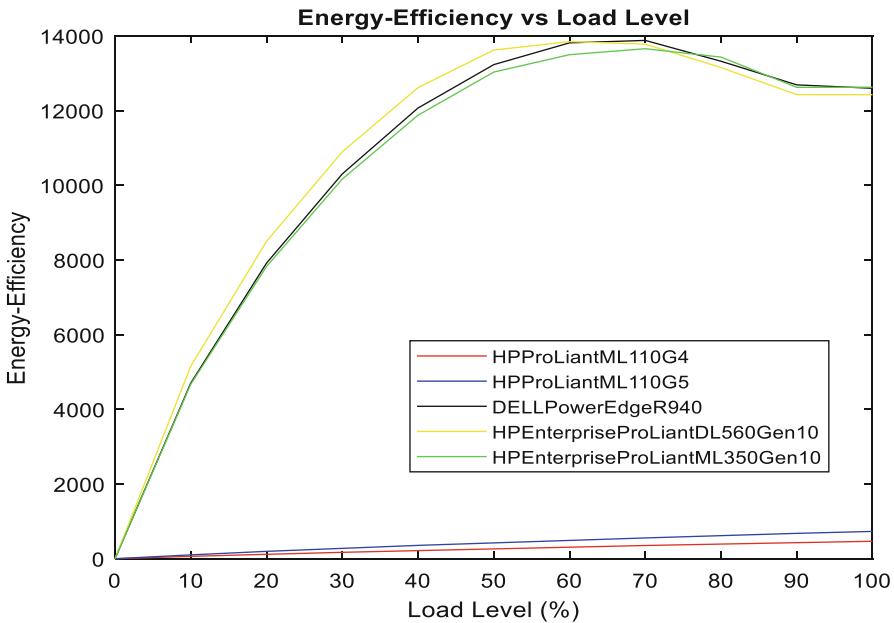


Fig. 3.4 Change of energy efficiency with varying load level for various PMs

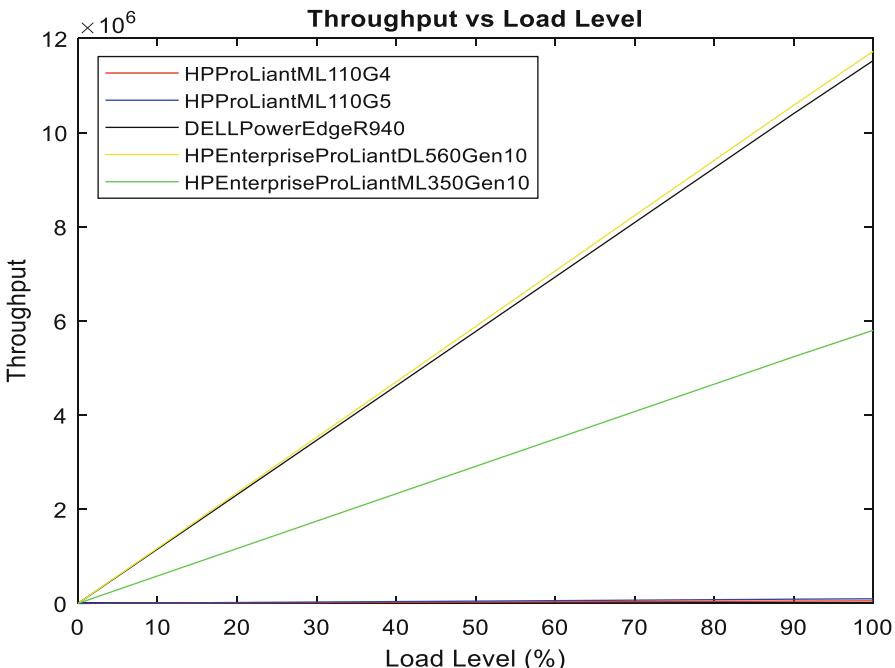


Fig. 3.5 Change of throughput with varying load level for various PMs

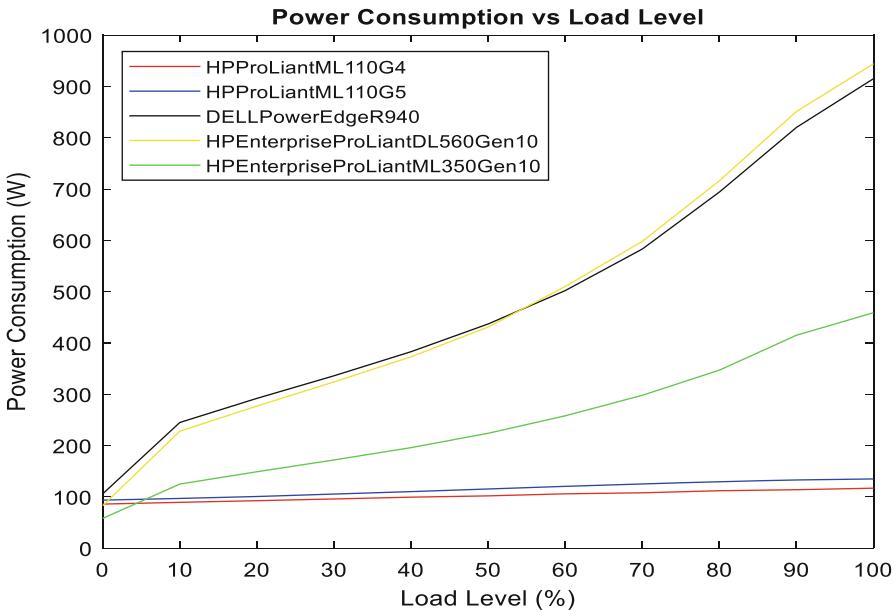


Fig. 3.6 Change of power consumption with varying load level for various PMs

- Preventing Quality of Service (QoS) degradation in CDC due to excessive VM migration is as much important as it is to minimize CDC energy consumption. However, higher energy consumption minimization by *RTDVMC* comes at a cost of excessive increase of VM migration.
- While, performance of *RTDVMC* has strong correlation with the value of VMRT, instead of VMRT values of real Cloud workload, only simulated VMRT values have been used. The complex behavior and interaction of VMs have impact on VMRT values, which cannot be reflected in simulated VMRT values.
- *RTDVMC* consolidates VMs primarily on the value of VMRT with an assumption that VMRT would be precisely known in advance, whereas, in reality VMRT would not be strictly accurate in many scenarios.

With respect to above-mentioned challenges, our contributions in this paper are briefly outlined in the following.

- A novel heuristic DVMC algorithm, namely, *Stochastic Release Time Based DVMC (SRTDVMC)* algorithm, has been proposed, which is robust to uphold optimal performance in terms of minimizing energy consumption regardless of the potential change in underlying PMs' energy efficiency characteristics.
- One crucial goal of experiment is to predict behavior of an algorithm in the real world [36]. For performance evaluation, VMRT values extracted from real Cloud, namely, Nectar Cloud, has been used, which assists to obtain a stronger performance prediction under real Cloud scenarios.

- While minimization of VM migration reduces network overhead and subsequent energy consumption by networking equipment, it is intrinsic in VM consolidation. *SRTDVMC* is multi-objective, which concomitantly addresses two confronting goals: minimizing energy consumption and minimizing VM migration.
- *SRTDVMC* eliminates the restriction of strictly accurate VMRT through the conversion of VMRT into respective Stochastic VMRT (SVMRT).

The organization of the rest of the paper is as follows. In Sect. 3.2, related literature is highlighted. Stochastic Release Time and various notations used in the algorithm are elucidated in Sect. 3.3. Next, in Sect. 3.4, we have brought forth the proposed algorithm, followed by elucidation of core components and characteristics of *SRTDVMC*. In Sects. 3.5 and 3.6, the experimental setup and performance evaluation of the proposed algorithm would be articulated. In Sect. 3.7, we have elucidated our critical observations extracted from empirical evaluations. Finally, in Sect. 3.8, we have summarized our research with future research directions and motivation.

3.2 Related Work

In depth review and classification of VMC algorithms have been broadly discussed in [37]. VMC is a NP-hard problem and can be broadly classified into two groups: Static VM Consolidation (SVMC) [38–40] and DVMC. SVMC algorithm provides the solution of energy-efficient initial VM placement [38]. However, as time progresses, the initial VM placement solution loses the efficiency with the change in resource demand and the resource availability. In case of increase in resource demand, VM(s) might be required to be migrated out into other suitable PM(s), while the decrease in resource demand widens up the room to host additional VM(s) and, hence, presents the opportunity to minimize the CDC energy consumption further by consolidating more VMs in lesser number of PMs than before. Contrast to SVMC, dynamic nature of DVMC algorithm contributes in further energy consumption minimization, as it captures the opportunity arisen from varying resource demand and resource availability and iteratively consolidate VMs whenever consolidation opportunity arises.

DVMC algorithms can be broadly classified into two groups: Centralized DVMC [41–43] and Distributed DVMC [27, 44]. Major DVMC algorithms found in the literature are centralized DVMC and only a few Distributed DVMC [27, 44] has been proposed. In [27], authors have presented their distributed DVMC algorithm for a P2P network oriented CDC. According to [27, 44], the growing number of PMs becomes a bottleneck for CDVMC at the time of selecting a destination for any migrating VM, since the asymptotic time complexity of the centralized DVMC algorithm is proportional to the number of PMs in the CDC, whereas the number of potential PMs to choose from for a migrating VM is relatively small in distributed

DVMC. Thus, distributed DVMC is more scalable for CDC with huge number of PMs. However, distributed DVMC has message passing overhead, as every PM must update its present resource availability to all of its neighbors. Message passing increases network overhead, decreases network throughput, and increases network-related energy consumption. Both centralized and distributed DVMC algorithms can be classified into two groups: Static Threshold-Based DVMC algorithm [8, 11, 26] and Adaptive Threshold-Based DVMC algorithm [12, 23]. The later increases VM migration. Our objective is to minimize both energy consumption and VM migration. Hence, in this paper, we have opted to Static Threshold-based DVMC approach.

Elucidated earlier in Sect. 3.1, performance of existing VMC algorithms lacks in terms of minimizing CDC energy consumption with the rise of highly energy proportional PMs. PEAS [39] and EPACT [45] as SVMC algorithms address this issue. However, on account of being an SVMC algorithm, those algorithms do not continue to dynamically consolidate VMs whenever opportunity arises, and hence, the solution would lose optimality over time. Based on our literature study [8–23, 27], no existing DVMC algorithm is designed considering energy-efficiency characteristics of highly energy proportional state-of-the-art PMs in their bedrocks. As such, we have brought forth *SRTDVMC* which takes the dynamics of modern PMs' energy-efficiency characteristics with respect to varying load level into account and limits both VM migration and CDC energy consumption.

In our proposed *SRTDVMC* algorithm, stochastic release time is one of the key distinctive features used in VMC decision process. To ensure easy understanding of our proposed *SRTDVMC* algorithm, we have first explained the key terms used in the proposed algorithm in following section.

3.3 Modelling Stochastic VM Release Time, Notations Used and Important Concepts

3.3.1 Modelling Stochastic VM Release Time, PM Release Time and Notations Used

Workload finishing time or lifetime of a VM is referred to as VMRT. Prior receiving any service, negotiation of service related conditions including service expiry date followed by an acceptance of the contract takes place between Cloud Service Providers (CSPs) and CSUs, interpreted as Service Level Agreement (SLA). For many VMs, VMRT is equal to the contract of service period between the respective CSU (i.e., VM owner) and the CSP as agreed during SLA. Before the contract is expired, both CSUs and CSPs might agree/disagree to renew, extend or early termination of the contract and respective VMRT would be updated accordingly. Some web applications hosted in CDC remains unremoved for a very long period. Estimated VMRT of such VMs would be large values referring to the time when the

respective contract of service between CSU and CSP would expire as agreed prior the service during SLA. If renewal or early termination of contract of service takes place, VMRT would be readjusted accordingly.

For some applications, VMRT corresponds to QoS and potential resource demand. Resource demand may change with the variation in number of users, causing creation of additional VMs and later deletion of such VMs. At the time of SLA, a SAAS provider/PAAS user must consider the potential number of application users and mention it to PAAS provider so that by taking the potential number of end users and corresponding resource demand into account a certain standard of QoS can be upheld. Pattern of changing resource demand over time derived from past data can also be utilized to recognize the change of resource demand in future [46]. Considering the change in resource demand over time and demanded QoS, PAAS provider/IAAS user can estimate resource/VM release time, which would be proffered to IAAS provider.

In many cases, the prior estimated VMRT might not turn out as strictly accurate in future. Hence, to reflect the reality further closely, we propose to embody a stochastic version of VMRT, referred to as Stochastic VM Release Time (SVMRT) in *SRTDVMC*. SVMRT can be calculated from (3.1). In Table 3.1, we have articulated the meaning of the notations used in this paper.

$$S_{V_j} = (1 + \alpha \cdot Y) \cdot T_{V_j} \quad (3.1)$$

Apart from *VMRT*, two more crucial terms noteworthy explaining is *PM Release Time (PMRT)* and *Stochastic PMRT (SPMRT)*. *PMRT* refers to the time when a PM can be either shut down or put into a sleep state that would consume no energy, or lower amount of energy compared to its active state. A PM can be shut down or put into sleep state, if it has either no VM hosted on it, or none of its hosted VMs is in the active state. Since *SVMRT* refers to the maximum time until which the VM would be in the active state, hence *SPMRT* of a PM P_i denoted by S_{P_i} refers to the maximum *SVMRT* value among all the *VMRT* values of VMs that are hosted in that PM, as articulated in (3.2).

$$S_{P_i} = \max(S_{V^i}) \quad (3.2)$$

3.3.2 Modelling Resource Utilization and Constraints

DVMC algorithm migrates VMs from one PM to another PM, so that VMs would be placed in minimum number of PMs and thus increase resource utilization and energy efficiency. VMs hosted in a PM utilize the resources of that PM. Therefore, resource utilization of a PM corresponds to the total resource demand by all the VMs hosted in that PM. Let U_i^k denote utilization of Resource type, R_k of PM, P_i . Hence, the equation for calculating U_i^k [47] is as follows:

Table 3.1 Notations used

Notations	Meaning
$\ \cdot \ $	Cardinality of a set
$P = \{P_i\}_{ P }$	The set of $ P $ PMs
$V = \{V_j\}_{ V }$	The set of $ V $ VMs
$V^i = \left\{ V_j^i \right\}_{ V^i }$	Set of VMs hosted in PM, P_i
T_{V_j}	$VMRT$ of VM, V_j
$T_{V_j^i}$	$VMRT$ of VM, V_j^i
$T_{V^i} = \{T_{V_j^i}\}$	The set of $VMRT$ of VMs hosted in PM, P_i
Y	Random variable ranging $[-1, +1]$
α	Maximum deviation of $VMRT$
S_{V_j}	$SVMR$ of VM, V_j
$S_{V_j^i}$	$SVMR$ of VM, V_j^i hosted in PM, P_i
$S_{V^i} = \{S_{V_j^i}\}$	The set of $SVMR$ of VMs hosted in PM, P_i
S_{P_i}	$SPMR$ of PM, P_i , $S_{P_i} = \max(S_{V^i})$
$x = \{x_{i,j}\}_{ P \times V }$	Placement matrix
$R = \{R_k\}_{ R }$	The set of different types of resources
D_k^j	Demand of resource, R_k by VM, V_j
U_j^k	Utilization of resource, R_k of PM, P_i
C_j^k	Capacity of PM, P_i in terms of resource, R_k
θ_{MAX}	Maximum threshold
$OP = \{P_o\}_{ OP }$	The set of O -UPMs
$V^o = \left\{ V_q^o \right\}_{ V^o }$	The set of VMs hosted in an O -UPM, P_o
D_g^k	Demand of resource, R_k by VM, V_g^o
U_o^k	Utilization of resource, R_k of PM, P_o
C_o^k	Capacity of PM, P_o in terms of resource, R_k
$NOP = \{P_x\}_{ NOP }$	The set of non-over-utilized PMs, which are neither O -UPMs nor in sleep mode or switched off
$SP = \{P_s\}_{ SP }$	The set of PMs that are in sleep mode or switched off

P_d	Destination PM
vmsToMigrate	The set VMs to migrate out into new PMs
$= V_l _{\text{vmsToMigrate}}$	
destinationPMs	The set of new destination PMs for migrating VMs from source O-UPMs
$= P_d _{\text{destinationPMs}}$	
candidateSources	The set of PMs from which a PM would be selected as U-UPM
$= P_c _{\text{candidateSources}}$	
$V^c = \{V_n^c\}_{V^c}$	The set of VMs hosted in PM, P_c
$\text{candidateDestinations}$	The set of PMs from which a PM would be selected to host migrating VM of an U-UPM
$= P_m _{\text{candidateDestinations}}$	
Destinations	The set of new destination PMs for migrating VMs of source U-UPMs
E_i	Energy consumption by PM, P_i
E_{CDC}	Energy consumption by the CDC
\bar{E}_{CDC}	Mean energy consumption by the CDC
f_1	First objective function of SRTDVMC
f_2	Second objective function of SRTDVMC
$\text{Nectar} = \{\text{Nectar}_{NT}\}_{NT}^{13}$	Set of different months of Nectar Cloud
$=\{\text{Nov 2013}, \text{Dec 2013}, \text{Jan 2014}\}$	
$PLab = \{PLab_{PL}\}_{PL}^{14}$	Set of different days of PlanetLab Cloud
$=\{6 \text{ March}, 9 \text{ March}, 9 \text{ April}, 20 \text{ April}\}$	
$\bar{E}^S = \left\{ \bar{E}_{NT, PL}^S \right\}_{ Nectar \cdot PLab }$	Set of mean energy consumption values for SRTDVMC algorithm under different workload combinations of Nectar and PlanetLab
$\bar{E}^R = \left\{ \bar{E}_{NT, PL}^R \right\}_{ Nectar \cdot PLab }$	Set of mean energy consumption values for RTDVMC algorithm under different workload combinations of Nectar and PlanetLab
$np = \{1, 2, \dots, Nectar \cdot PLab \}$	Index to denote np th smallest element of \bar{E}^S , \bar{E}^R , $\bar{\psi}^S$ and $\bar{\psi}^R$
a_{np}	Weights related to Shapiro-Wilk Normality Test
$\frac{E_{(np)}^S}{\bar{E}^S}$	np th smallest element of \bar{E}^S
$\frac{E_{(np)}^R}{\bar{E}^R}$	np th smallest element of \bar{E}^R

(continued)

Table 3.1 (continued)

Notations	Meaning
$\overline{\overline{E}}^S$	Mean of \overline{E}^S
$\overline{\overline{E}}^R$	Mean of \overline{E}^R
ψ	Total number of VM migration
$\overline{\psi}$	Mean total number of VM migration
$\overline{\psi}^S = \left\{ \overline{\psi}_{NT, PL}^S \right\}_{ Nectar \times PLab }$	Set of mean number of VM migration for SRTDVMC algorithm under different workload of Nectar and PlanetLab
$\overline{\psi}^R = \left\{ \overline{\psi}_{NT, PL}^R \right\}_{ Nectar \times PLab }$	Set of mean number of VM migration for RTDVMC algorithm under different workload of Nectar and PlanetLab
$X^{\overline{E}} = \left\{ X_{NT, PL}^{\overline{E}} \right\}_{ Nectar \times PLab }$	Minimization of \overline{E}_{CDC} by SRTDVMC compared to RTDVMC under different workload combination of Nectar and PlanetLab
$X^{\overline{\psi}} = \left\{ X_{NT, PL}^{\overline{\psi}} \right\}_{ Nectar \times PLab }$	Minimization of $\overline{\psi}$ by SRTDVMC compared to RTDVMC under different workload combination of Nectar and PlanetLab
SW_E^S	Test statistics for the S-W normality test with \overline{E}^S
SW_E^R	Test statistics for the S-W normality test with \overline{E}^R
$SW_{\overline{E}}^S$	Test statistics for the S-W normality test with $\overline{\psi}^S$
$SW_{\overline{E}}^R$	Test statistics for the S-W normality test with $\overline{\psi}^R$
$t_{\overline{X}^E}$	Test statistic for the <i>t-test</i> with data samples of \overline{X}^E
$t_{\overline{X}^{\overline{\psi}}}$	Test statistic for the <i>t-test</i> with data samples of $\overline{X}^{\overline{\psi}}$
$\hat{\sigma}_{\overline{X}^E}$	Standard deviation of \overline{X}^E
$\hat{\sigma}_{\overline{X}^{\overline{\psi}}}$	Standard deviation of $\overline{X}^{\overline{\psi}}$
\overline{X}^E	Mean of $X^{\overline{E}}$
$\overline{X}^{\overline{\psi}}$	Mean of $X^{\overline{\psi}}$

$$U_i^k = \sum_{j=1}^{|V|} D_j^k \cdot x_{i,j} \quad (3.3)$$

where D_j^k denotes Demand of Resource type, R_k by VM, V_j , $x_{i,j}$ denotes the element of placement matrix, x and value of $x_{i,j}$ is determined as follows:

$$x_{i,j} = \begin{cases} 1, & \text{if } V_j \text{ is placed in } P_i \\ 0, & \text{otherwise} \end{cases} \quad (3.4)$$

Since, Resource Capacity of a PM, P_i is fixed and it cannot provide additional resources to its hosted VMs than its capacity, Therefore, a VM, V_j can only be placed in a PM, P_i if the amount of available resources in P_i is adequate to meet the resource demand of V_j . Hence, V_j can only be placed in P_i if the following equation is satisfied [47]:

$$C_i^k - U_i^k \geq D_j^k \quad (3.5)$$

In other words, a PM, P_i cannot host a VM, V_j if the available resource of P_i is lesser than the resource demand of V_j . We denote such constraint presented in (3.5) as *Resource Constraint (RC)*.

At times, workload of VMs could rise very high resulting in steep resource utilization of the hosting PM. We denote such PM with heavy resource utilization as *Over-utilized PM (O-UPM)* and use a threshold, referred to as *Maximum Threshold*, θ_{\max} to distinguish whether a PM is *Over-utilized* or not. Let us denote OP (3.6) as a set of *O-UPMs*.

$$OP = \left\{ P_i \mid U_i^k \geq \theta_{\max} \text{ for any } R_k \in R \text{ and } 1 \leq i \leq |P| \right\} \quad (3.6)$$

If VM(s) were not migrated out of an *O-UPM*, then SLA violation would unfold. In order to avoid causing SLA violation, during destination PM selection for a migrating VM, it is essential to ensure that hosting the migrating VM would not turn the destination into an *O-UPM*, which we have modelled through the following equation:

$$D_j^k + U_i^k < \theta_{\max} \quad (3.7)$$

We refer such constraint presented in (3.7) as *Maximum Utilization Threshold Constraint (MUTC)*.

3.3.3 Modelling Energy Consumption

Most of the existing VM consolidation algorithms have mentioned that energy consumption of a PM is primarily dominated by its CPU utilization [8, 47]. Hence, our energy consumption model is a function of CPU utilization (3.8), where, E_i denotes the energy consumption by PM, P_i .

$$E_i = f(U_i^{\text{CPU}}) \quad (3.8)$$

Based on (3.8), we can determine the total energy consumption of the CDC through (3.9), where E_{CDC} denotes the total energy consumption of the CDC.

$$E_{\text{CDC}} = \sum_{i=1}^{|P|} E_i \quad (3.9)$$

In order to relate closely to the real energy consumption by PMs, we have opted to draw energy consumption benchmark results of three different types of PMs presented in Table 3.2.

3.3.4 Objective Functions

DVMC algorithms aim to minimize CDC energy consumption through migrating VM(s) out of lower utilized PMs and placing those VM(s) in relatively higher utilized PM(s). As such, the first objective function, f_1 of SRTDVMC has been characterized through (3.10).

$$f_1 = \min(\bar{E}_{\text{CDC}}) \quad (3.10)$$

One downfall of DVMC is that energy consumption minimization through VM consolidation cannot be achieved without VM migration, which itself deteriorates

Table 3.2 Energy consumption values of contemporary servers at different load level

	Energy consumption (kW) at different percentage of load level										
	Sleep	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Dell Power Edge R940	106	245	292	336	383	437	502	583	694	820	915
HP ProLiant DL560 Gen10	82.9	228	277	324	373	431	510	598	716	851	944
HP ProLiant ML350 Gen10	58.1	125	149	172	196	224	258	298	347	415	459

QoS as well as raises network overhead leading towards increased SLA violation and increased energy consumption by networking equipment of CDC. Therefore, restricting VM migration in CDC is no less important than lowering CDC energy consumption. As such, (3.11) characterizes the second objective function, f_2 of *SRTDVMC*.

$$f_2 = \min (\bar{\psi}) \quad (3.11)$$

It is worthwhile to note that f_1 and f_2 are two confronting objective functions. Value of f_1 can only be minimized through migrating VM(s) from lower utilized PM(s) to relatively higher utilized PMs, which increases ψ , whereas, increased ψ negatively affects f_2 . Hence, it is more challenging to design a DVMC algorithm, which can show better performance in terms of both f_1 and f_2 . In the following section, our proposed multi-objective heuristic DVMC algorithm, namely, *SRTDVMC* has been presented, which aims to optimize f_1 and f_2 .

3.4 Proposed Solution

At the outset of the paper, in Sect. 3.1, we have elucidated the limitations of *RTDVMC*. To address those limitations, we have proposed our *SRTDVMC* algorithm presented in Algorithm 3.1. The meaning of notations used in *SRTDVMC* algorithm has been presented earlier in Table 3.1.

Algorithm 3.1: The SRTDVMC algorithm

Input: P, V, R, SP

Output: VM Placement

The first phase: O-UPMs

```

1: for each  $P_i$  in  $P$  do
2:   if (3.6) is satisfied then
3:      $OP \leftarrow \{P_i \cup OP\}$ 
4:   end if
5: end for           // end of for loop from Line 1 to 5
6: for each  $P_o$  in  $OP$  do
7:   migratingVMs  $\leftarrow$  Invoke the VSO algorithm with  $P_o$ 
8:   vmsToMigrate  $\leftarrow \{migratingVMs \cup vmsToMigrate\}
9: end for           // end of for loop from Line 6 to 9
10: Sort vmsToMigrate in the order of decreasing SVMRT
11:  $NOP \leftarrow \{P - OP - SP\}$ 
12: for each  $V_l$  in sorted vmsToMigrate do
13:    $P_d \leftarrow$  Invoke the DPSVO algorithm with  $V_l$  and  $NOP$ 
14:   destinationPMs  $\leftarrow \{P_d \cup destinationPMs\}$ 
15:   if  $P_d$  is in  $SP$  then
16:      $SP \leftarrow \{SP - P_d\}$ 
17:     for each  $R_k$  in  $R$  do$ 
```

```

18:            $U_d^k \leftarrow U_d^k + D_q^k$ 
19:            $C_d^k \leftarrow C_d^k - D_q^k$ 
20:       end for // end of for loop from Line 17 to 20
21:        $NOP \leftarrow \{P_d \cup NOP\}$ 
22:   end if // end of if at Line 15
23: end for // end of for loop from Line 12 to 23
The second phase: U-UPMs
24:  $candidateSources \leftarrow \{P - OP - SP - destinationPMs\}$ 
25:  $candidateDestinations \leftarrow \{P - OP - SP\}$ 
26: Sort  $candidateSources$  in the order of increasing  $SPMRT$ 
27: for each  $P_c$  in sorted  $candidateSources$  do
28:    $candidateDestinations \leftarrow \{candidateDestinations - P_c\}$ 
29:    $destinations \leftarrow$  Invoke the DPSVU algorithm with  $P_c$ 
      and  $candidateDestinations$ 
30:    $candidateSources \leftarrow \{candidateSources - destinations\}$ 
31: end for // end of for loop from Line 27 to 31

```

3.4.1 Two Phases of SRTDVMC Algorithm

In this section, we have explained our proposed *SRTDVMC* algorithm. Resource demand of VMs may change over time, resulting variation of resource utilization in host PMs. When resource demand of hosted VM(s) grow higher over time, the hosting PM might fall short of adequate resources to prevent potential performance degradation, translated as SLA violation. Again, hosted VMs resource demand may decline over time, bringing forth a gap in the hosting PM to accommodate additional VMs from other low utilized PMs, leading towards potential opportunity to reduce the number of active PMs and subsequent reduction of CDC energy consumption. Based on this principal, *SRTDVMC* algorithm works in two phases – the first phase and the second phase.

In the *first phase*, VMs from *O-UPMs* are migrated out to control SLA violations (Line 1–23 of Algorithm 3.1) and in the *second phase*, VMs from *U-UPMs* are migrated out to consolidate in lesser number of active PMs (Line 24–31 of Algorithm 3.1). In the following sections, we have comprehensibly discussed each of these phases and its components.

3.4.1.1 The First Phase O-UPMs

As expressed in (3.6), for any resource type (i.e., CPU, RAM and Bandwidth), if a PM's resource utilization is found as equal or greater than θ_{max} , then the PM is denoted as *O-UPM* (Line 1–5 of Algorithm 3.1). Next, VM(s) are selected from all *O-UPMs* to migrate out into new destination PMs (Line 7 of Algorithm 3.1). The *VSO* algorithm (Algorithm 3.2) proffers to the VM(s), which are attempted to migrate out from an *O-UPM*. In the following section, we have explained the *VSO* algorithm.

- **VM Selection from *O-UPMs***

VMs of an *O-UPM* are first sorted in decreasing order of VMRT (Line 1 of Algorithm 3.2). The first VM from the sorted list of VMs is first selected and checked whether the respective PM's utilization drops lower than θ_{\max} . If yes, then the respective PM is no more *O-UPM* and VM selection process stops (Line 2–3 of Algorithm 3.2). Otherwise, the second VM from the sorted list of VMs is selected and then the third VM and so forth, until the PM's utilization is decreased below θ_{\max} (Line 3–10 of Algorithm 3.2). The rationale of selecting VM(s) with largest VMRT is to minimize the active duration of source *O-UPM*, which might aid in minimization of energy consumption.

Algorithm 3.2: The VM Selection from *O-UPM* (*VSO*) Algorithm

Input: The *O-UPM*, P_o .

Output: List of VMs to be migrated out from the given *O-UPM*

```

1: Sort  $V^o$ , the set of VMs of  $P_o$  in order of decreasing VMRT
2:  $q \leftarrow 1$ 
3: while  $q \leq |V^o|$  and  $\theta_{MAX} < U_o^k$  for any  $R_k$  in  $R$  do
4:    $migratingVMs \leftarrow \{V_q^o \cup migratingVMs\}$ 
5:   for each  $R_k$  in  $R$  do
6:      $U_o^k \leftarrow U_o^k - D_q^k$ 
7:      $C_o^k \leftarrow C_o^k + D_q^k$ 
8:   end for
9:    $q \leftarrow q + 1$ 
10: end while
11: return  $migratingVMs$ 
```

- **Destination PM Selection for Migrating VMs from *O-UPMs***

SRTDVMC algorithm develops a set, denoted as *vmsToMigrate*, comprised of all migrating VMs from *O-UPMs* (Line 6–9 of Algorithm 3.1), as these VMs of *vmsToMigrate* are sorted in decreasing order of VMRT (Line 10 of Algorithm 3.1). The migrating VMs are attempted to host in PMs, which are neither *O-UPMs*, nor in sleep state or turned off state. Such set of PMs, which are not *O-UPMs* and not in either turned off or sleep state is referred to as *NOP* (Line 11 of Algorithm 3.1). *SRTDVMC* algorithm then keeps invoking *DPSVO* algorithm, presented as Algorithm 3.3 to determine the destination PM for each of the migrating VMs of *vmsToMigrate* starting from the largest VM to the smallest VM in terms of VMRT (Line 12–23 of Algorithm 3.1).

Algorithm 3.3: The Destination PM Selection for VM of *O-UPM* (*DPSVO*) Algorithm

Input: The VM V_j to be migrated out from an *O-UPM*

Input: *NOP*

Output: The new destination PM for the given migrating VM

```

1: Sort NOP in the order of increasing SPMRT
2: for each  $P_x$  in sorted NOP do
3:   suitable  $\leftarrow$  Invoke the PST algorithm with  $P_x$  and  $V_j$ 
```

```

4:   if suitable is true then
5:     return  $P_x$ 
6:   end if
7: end for
8: return most energy-efficient  $P_s$ , which satisfies (3.5) and (3.7)

```

Algorithm 3.4: The PM Suitability Test (*PST*) Algorithm

Input: PM, VM

Output: A decision whether the given PM can host the given VM

```

1: if RC (3.5) and MUTC (3.7) are satisfied for all  $R_k$  in  $R$  then
2:   return true
3: end if
4: return false

```

In order to determine a PM from NOP as destination host for a migrating VM of $vmsToMigrate$, the *DPSVO* algorithm first sorts the PMs of NOP in increasing order of *SPMRT* (Line 1 of Algorithm 3.3). The smallest PM in terms of *SPMRT* from the sorted NOP is first checked whether it is suitable to accommodate the migrating VM or not (Line 2 of Algorithm 3.3). The *PST* algorithm presented in Algorithm 3.4 is invoked to check the suitability of a PM as a potential destination PM (Line 3 of Algorithm 3.3). A PM is considered suitable, if RC (3.5) and MUTC (3.7) constraints are not violated (Line 1–4 of Algorithm 3.4). If that PM is found as suitable as per *PST* algorithm, then it is selected as the new destination PM for the migrating VM and the destination PM selection process ends (Line 4–6 of Algorithm 3.3). In case the PM is found as unsuitable, suitability of the second smallest PM in terms of *SPMRT* from the sorted NOP is checked and then the third smallest PM and so forth until a suitable PM is found (Line 2–7 of Algorithm 3.3). If no PM from NOP can accommodate that particular migrating VM, then the most energy-efficient and suitable PM from the set of PMs, which are in either sleep or turned-off state, referred to as SP is awoken and selected as destination PM (Line 8 of Algorithm 3.3). If the destination PM is selected from SP , then that PM is removed from the set SP , since it is no more in sleep or turned-off state and its utilization and capacity values across all resource types are adjusted (Line 15–20 of Algorithm 3.1). Furthermore, the PM is added in the set NOP , so that it can be considered as a potential destination PM for following migrating VMs of $vmsToMigrate$ (Line 21–22 of Algorithm 3.1).

3.4.1.2 The Second Phase U-UPMs

Under-Utilized PMs (U-UPMs) refers to the set of those PMs, which had not been determined as *O-UPMs* in the first phase of *SRTDVMC* and which do not belong to SP . After determining the destination PMs for VMs of *O-UPMs*, the second phase of *SRTDVMC* is commenced when VMs from *U-UPMs* are migrated out, followed by strategic destination PM selection for those migrating VMs with an aim to lower

the number of active PMs so that CDC energy consumption can be minimized (Line 24–31 of Algorithm 3.1).

- **Source PM Selection from $O\text{-UPMs}$**

In the second phase, *SRTDVMC* first rounds up a set of $U\text{-UPMs}$, namely, *candidateSources* – the set representing potential source $U\text{-UPM}(s)$ from which VM(s) would be attempted to migrate out. The set of PMs, which were identified as $O\text{-UPMs}$ in the first phase, referred to as OP along with the set of PMs, which hosted migrating VMs of $O\text{-UPMs}$ are excluded from *candidateSources* to avoid repeated handling of PMs from OP and to inhibit re-migration of those VMs, which had been migrated out once in the first phase (Line 24 of Algorithm 3.1). The PMs of *candidateSources* are then sorted in the increasing order of *SPMRT* (Line 26 of Algorithm 3.1). All PMs of sorted *candidateSources* starting from the smallest PM to the largest PM in terms of *SPMRT* is sequentially selected as source $U\text{-UPM}$. However, the set of $U\text{-UPMs}$ denoted by *destinations*, which are determined as the new destination PM(s) for migrating VM(s) from a source $U\text{-UPM}$ is excluded from *candidateSources* and hence, those new destination $U\text{-UPMs}$ cannot become source $U\text{-UPM}$, which prevents repeated migration of same VMs (Line 30 of Algorithm 3.1).

- **Migrating VM and Destination PM Selection**

SRTDVMC algorithm invokes the *DPSVU* algorithm (Algorithm 3.5) to select migrating VMs from $U\text{-UPMs}$ and corresponding new destination PMs. Once a $U\text{-UPM}$ from sorted *candidateSources* is selected as source $U\text{-UPM}$, the hosted VMs in that source $U\text{-UPM}$ is sorted in decreasing order of *VMRT* (Line 1 of Algorithm 3.5). The VMs starting from the largest to the smallest in terms of *VMRT* are attempted to migrate out (Line 2 of Algorithm 3.5). The reason of selecting VMs in descending order of *VMRT* is that migrating out the largest VM can reduce the *SPMRT* of the source PM leading towards energy consumption minimization. If for any VM, a suitable new destination $U\text{-UPM}$ cannot be found, the migrating VM(s) selection from a source $U\text{-UPM}$ terminates (Line 18–20 inside of Line 2–21 from Algorithm 3.5). In the following, we have discussed the process of determining the new destination PM for such migrating VM.

Algorithm 3.5: The Destination PMs Selection for VMs of $U\text{-UPMs}$ (*DPSVU*) Algorithm

Input: An $U\text{-UPM}$, P_c from the set of *candidateSources*

Input: *candidateDestinations*

Output: List of new destination PMs to host migrating VMs

- 1: Sort V^c , the set of VMs of P_c in order of decreasing *VMRT*
- 2: **for** each V_n^c in sorted V^c **do**
- 3: $P_d \leftarrow \text{null}$
- 4: Sort *candidateDestinations* in order of increasing *SPMRT*
- 5: **for** each P_m in sorted *candidateDestinations* **do**
- 6: suitable \leftarrow Invoke the *PST* algorithm with P_m and V_n^c
- 7: **if** suitable is **true**

```

8:     energyDrop ← Energy drop in  $P_c$  without  $V_n^c$ 
9:     energyRise ← Energy rise of  $P_m$  for hosting  $V_n^c$ 
10:    netEnergyGain ← EnergyDrop - EnergyRise
11:    if NetEnergyGain > 0
12:         $P_d \leftarrow P_m$ 
13:        hostList ← { $P_d \cup hostList$ }
14:        break loop
15:    end if
16:    end if
17: end for
18: if  $P_d$  is null then
19:     break loop
20: end if
21: end for
22: return hostList

```

In order to select the destination PM for a migrating VM of *U-UPM*, *SRTDVMC* algorithm first creates a set of potential destination PMs, referred to as *candidateDestinations*. The PMs of *SP*, *OP* and the source *U-UPMs* hosting the migrating VMs are excluded from *candidateDestinations*, since a source PM cannot be the new destination PM of its own VMs and to avoid increasing the likelihood of turning the PMs from *OP* into *O-UPMs* again (Line 25, 27 and 28 of Algorithm 3.1). The *DPSVU* algorithm (Algorithm 3.5) is then invoked to select the destination PM from *candidateDestinations* (Line 29 of Algorithm 3.1). The PMs of *candidateDestinations* are first sorted in increasing order of *SPMRT* (Line 4 of Algorithm 3.5) and then the suitability of these PMs from sorted *candidateDestinations* are sequentially checked starting from the smallest to the largest PM in terms of *SPMRT* (Line 5–6 of Algorithm 3.5). If a PM is found as suitable satisfying both *RC* (3.5) and *MUTC* (3.7) constraints as per *PST* Algorithm (Algorithm 3.4), then net energy gain for the potential VM migration is estimated from the difference between reduced energy consumption of source *U-UPM* and increased energy consumption of new destination *U-UPM*. If net energy gain is found as positive, then that PM is selected as the new destination PM (Line 7–17 of Algorithm 3.5). In the following section, we have discussed the characteristics of *SRTDVMC* algorithm.

3.4.2 Characteristics of Proposed Algorithm

SRTDVMC attempts to fit the largest VM in terms of *VMRT* from the smallest PM in terms of *SPMRT* into the next smallest possible PM. Such consolidation approach shortens the *SPMRT* of source PM without raising the *SPMRT* of destination PM, resulting into decreased energy consumption. Additionally, selecting the next smallest possible PM as destination PM ensures that the PM is accomplishing largest possible jobs before moving into sleep state or turned off state. Consequently, remaining workload for the existing active PMs becomes lower, which aids in

energy consumption minimization. Furthermore, lesser remaining workload for existing active PMs increases the likelihood that upcoming workload can be served by these active PMs without turning on PMs, which are in lower energy consumption state, for instance, sleep or turned off state. Hence, energy consumption minimization is complemented.

One critical aspect of *SRTDVMC* is that both rise of energy in potential destination PM (i.e., cost) and drop of energy in potential source PM (i.e., benefit) is checked prior any potential VM migration. VMs from *U-UPMs* are migrated only if the net energy gain (i.e., energy drop – energy rise) is positive, which limits the number of VM migrations and improves QoS without compromising energy efficiency. Hence, *SRTDVMC* can concurrently satisfy both objective functions (3.10) and (3.11). Furthermore, *SRTDVMC* smartly selects destination PMs ensuring that the increased energy consumption of potential destination *U-UPM* does not outweigh the reduced energy consumption of potential source *U-UPM*. It aids to uphold the energy efficiency of the solution regardless of the drastic rise of state-of-the-art PMs' energy consumption causing declined energy efficiency at utilization level beyond 70%. Thus, *SRTDVMC* encounters the lack of energy-efficiency issue in the presence of state-of-the-art PMs as experienced with existing DVMC algorithms. As a result, *SRTDVMC* is robust against underlying PMs' change of energy-efficient characteristics with varying load.

3.5 Performance Evaluation

Figures 3.1, 3.2, and 3.3 representing the diverse range of VMRT of Nectar Cloud reveal the heterogeneous nature of real Cloud workloads in terms of finishing time. To the best of our knowledge, none of the existing DVMC algorithms except *RTDVMC* is designed considering heterogeneous VMRT in their bedrock assuming all jobs finish at the same time, which is unrealistic. Consequently, these traditional DVMC algorithms cannot provide optimal solution for heterogeneous VMRT. As such, we have compared the performance of *SRTDVMC* with *RTDVMC*, since both are developed considering heterogeneous VMRT in their bedrocks.

3.5.1 Experimental Setup

Performance of *RTDVMC* [26] has been evaluated through CloudSim [24]. Since, performance of *SRTDVMC* has been compared with *RTDVMC*, therefore, we have modelled and simulated a cloud environment in CloudSim [24], which we have used to simulate *SRTDVMC* algorithm under different workload scenarios. For fair comparison, both algorithms have been simulated using same environment with respect to the characteristics of CDC, VM, PM and energy module. The simulated CDC consists of 800 heterogeneous PMs. Three different modern generation of

PMs, such as Dell PowerEdgeR940 (Intel Xeon Platinum 8180, 112 cores → 25,000 MHz, 384 GB) [30], HP ProLiant DL560 Gen10 (Intel Xeon Platinum 8180, 112 cores → 25,000 MHz, 384 GB) [31], and HP ProLiant ML350 Gen10 (Intel Xeon Platinum 8180, 28 cores → 25,000 MHz, 192 GB) [32] have been used. Each server is provided with 1 GB/s network bandwidth. The energy consumption characteristics of these servers with varying workload is articulated in Table 3.2.

The characteristics of different VM types match with the VMs used by *RTDVMC* and correspond to Amazon EC2 instance types [48]. However, the difference between the simulated VMs and Amazon EC2 instance types is that the simulated VMs are single-core, which is explained by the fact that the workload data used for the simulations come from single-core VMs. Since, the single-core is used, the amount of RAM is divided by the number of cores for each VM type: High-CPU Medium Instance (2500 MIPS, 0.85 GB); Extra Large Instance (2000 MIPS, 3.75 GB); Small Instance (1000 MIPS, 1.7 GB); and Micro Instance (500 MIPS, 613 MB).

Lifetime of a VM V_j , aka VMRT of a VM V_j , denoted by T_{V_j} can be different from one VM to another (i.e., heterogeneous). For further accurate estimation of the performance of both *SRTDVMC* and *RTDVMC* algorithms under real Cloud scenario, T_{V_j} values are drawn from VMRT traces of a real Cloud, namely, Nectar Cloud. Nectar Cloud consists of over thousands of VMs across multiple data centers located in eight different cities of Australia [7]. For *SRTDVMC* algorithm, T_{V_j} is converted into *SVMRT*, S_{V_j} as per (3.1), using 0.05 as the value of α and a uniformly distributed random variable ranging $[-1, +1]$ as X . For further clarity, maximum deviation of T_{V_j} from S_{V_j} is $\pm 5\%$. At the outset, VMs are provided with the resources defined by the VM types. However, during the lifetime, VMs utilize less resources according to the workload data, widening opportunities for dynamic consolidation. The workload data also reflects traces of real Cloud workload traffic, originated as part of the CoMon project, a monitoring infrastructure for PlanetLab [49]. For both *RTDVMC* and *SRTDVMC*, upper utilization threshold, θ_{\max} is considered as 80%. With every workload scenario, a DVMC algorithm has been run twice to generate mean CDC energy consumption and mean total number of VM migration by that DVMC algorithm under such workload scenario. Each time, the simulation has been run until 24 h CloudSim simulation clock time.

3.5.2 Performance Metrics and Workload Data

3.5.2.1 Performance Metrics

SRTDVMC is a multi-objective DVMC algorithm, which aims to minimize the CDC energy consumption and VM migrations. Hence, performance of *SRTDVMC* and *RTDVMC* algorithms have been measured and compared in terms E_{CDC} and ψ . Expressed in (3.9), E_{CDC} is the sum of energy consumption of all the PMs, while each PM's energy consumption is derived from Table 3.2 according to its current

Table 3.3 Characteristics of PlanetLab Data (CPU Utilization)

Day	Number of VMs	Mean	St. dev.	Quartile 1 (%)	Quartile 2 (%)	Quartile 3 (%)
6 March	898	11.44	16.83	2	5	13
9 March	1061	10.70	15.57	2	4	13
9 April	1358	11.12	15.09	2	6	15
20 April	1033	10.43	15.21	2	4	12

CPU utilization. The second metric, ψ is the number of VM migrations initiated during the VM placement adaptation.

3.5.2.2 Workload Data

In order to make a simulation-based evaluation applicable in real world, it is crucial to use workload traces from a real system in experiments [12]. Therefore, the performance of *RTDVMC* and other DVMC algorithms have been measured with real Cloud workload traffic traces representing time varying resource utilization. Real workload data is provided as part of the CoMon project, a monitoring infrastructure for PlanetLab [49]. Data of CPU usage of thousands of VMs has been collected every 5 min, while these VMs had been hosted in PMs spread globally across 500 locations. Both algorithms have been tested with the PlanetLab workload data of four different days: 6 March, 9 March, 9 April, and 20 April featuring different sets of varying resource demand over time. The characteristics of different PlanetLab workload data is articulated in Table 3.3.

For each workload, the associated VMs' release time or workload finishing time have been drawn from monthly VMRT traces of real Cloud, namely, Nectar Cloud. Traces of VMs created in Nectar Cloud over a month along with respective release time of those VMs constitutes the monthly VMRT data. The latest available VMRT data of three different months: November 2013, December 2013, and January 2014 have been used for experiments. To explain more, a single day's PlanetLab workload data is tested with Nectar VMRT data of three different months offering diverse VMRT distributions, so that the impact of heterogeneous workload finishing time or release time can be analyzed. Histogram of different months of Nectar VMRT data has been articulated through Figs. 3.1, 3.2, and 3.3. The number of VMs in Nectar VMRT data of a month is greater than the number of VMs in the PlanetLab workload data of a day. Therefore, a uniformly distributed random variable has been used to select a smaller set of VMs from monthly Nectar data to match the number of VMs of the daily PlanetLab data. Uniformly distributed random variable proffers the smaller set of VMs with similar VMRT distribution present in the monthly Nectar data.

3.6 Simulation Results Analysis

SRTDVMC and *RTDVMC* have been simulated under different workload scenarios. Four different days of PlanetLab workload data has been randomly selected (i.e., 6 March, 9 March, 9 April, and 20 April). PlanetLab workload data of every single day featuring varying resource demand over time has then been blended with three diverse sets of VMRT data originated from three different months of Nectar Cloud Data (i.e., Nectar Nov, Nectar Dec, and Nectar Jan) featuring heterogeneous VMRT. Thus, from a single set of daily PlanetLab workload data, three diverse sets of workload data are produced featuring time variant resource demand and diverse workload finishing time, which matches with real Cloud. Both algorithms are reiterated over multiple times for each set of time variant workload representing a unique combination of PlanetLab and Nectar Cloud data, to produce corresponding \bar{E}_{CDC} and \bar{y} .

3.6.1 Energy Consumption

Values of \bar{E}^R and \bar{E}^S representing mean CDC energy consumption by *RTDVMC* and *SRTDVMC*, respectively are highlighted in Fig. 3.7. Let $X^{\bar{E}}$ (3.12) denote the set representing difference between mean energy consumption by *RTDVMC* and mean energy consumption by *SRTDVMC* for different workload scenarios. In other words, $X_{NT, PL}^{\bar{E}}$ represents the minimization of mean energy consumption proffered by *SRTDVMC* compared to *RTDVMC* for diverse workloads, as articulated in Table 3.4.

$$X^{\bar{E}} = \left\{ X_{NT, PL}^{\bar{E}} \right\}_{|Nectar| \cdot |PLab|} = \left\{ \bar{E}_{NT, PL}^R - \bar{E}_{NT, PL}^S \right\}_{|Nectar| \cdot |PLab|} \quad (3.12)$$

From experimental results, as portrayed in Fig. 3.7 and Table 3.4, we can observe that *SRTDVMC* significantly reduces CDC energy consumption compared to existing DVMC algorithm. However, one might reject the superiority of *SRTDVMC* over existing DVMC algorithm based on the argument that no proof of statistical significance has been provided. To address such arguments, in the following section, we have presented diverse statistical testing.

3.6.1.1 Normality Testing

Parametric tests are reported as more powerful than non-parametric tests. Assumption of parametric tests is that data samples are normally distributed. Therefore, prior parametric tests, normality testing is executed. The capability to accurately figure out if a data sample has come from a non-normal distribution, referred to as

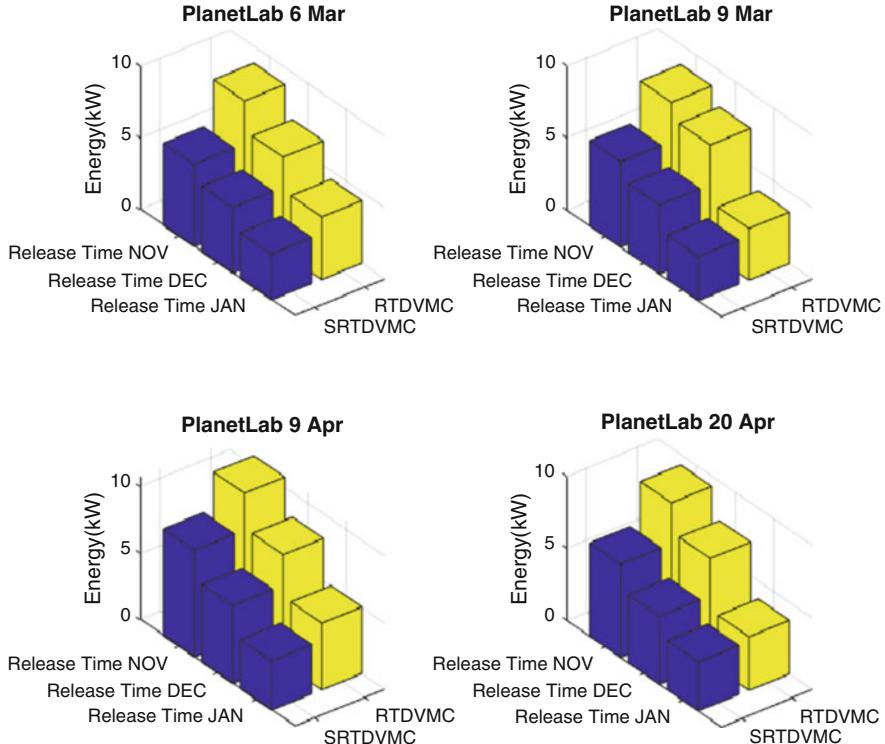


Fig. 3.7 Mean energy consumption of *SRTDVMC* vs *RTDVMC*

Table 3.4 Minimization of mean CDC energy consumption (kW) by *SRTDVMC* compared to *RTDVMC*

		PlanetLab (CPU utilization distribution)			
		6 March	9 March	9 April	20 April
Nectar VMRT	Nov	3.01	2.71	2.75	2.85
	Dec	2.09	2.77	2.3	2.59
	Jan	1.16	0.52	1.3	0.32

power, is the most widespread measure of the strength of a normality test [50]. Chi-square test for normality is not as powerful and unsuitable for small data samples. Kolmogorov-Smirnov (K-S) test is reported to have low power to test normality [51] and has high sensitivity issue with extreme values, which is handled by Lilliefors correction [52]. The S-W normality test is regarded as more powerful than the K-S test even after the Lilliefors correction [53] and recommend as the best option for testing the normality of data [50].

Test statistics for the S-W normality test with \bar{E}^R and \bar{E}^S , referred to as $SW_{\bar{E}}^R$ and $SW_{\bar{E}}^S$, respectively can be calculated through (3.13)–(3.16) [54, 55]. a_{np} weights are

Table 3.5 Mean CDC energy consumption for *RTDVMC*

np	1	2	3	4	5	6	7	8	9	10	11	12
$\bar{E}_{(np)}^R$	3.58	3.72	4.37	5	6.68	7.3	7.53	8.14	8.62	8.75	9.31	10.76

Table 3.6 Mean CDC energy consumption for *SRTDVMC*

np	1	2	3	4	5	6	7	8	9	10	11	12
$\bar{E}_{(np)}^S$	3.06	3.21	3.4	3.7	4.59	4.71	4.76	5.74	5.84	5.92	6.46	8.01

available in Shapiro-Wilk Table [56]. Different $\bar{E}_{(np)}^R$ and $\bar{E}_{(np)}^S$ values are presented in Tables 3.5 and 3.6.

$$SW_E^R = \left(\sum_{np=1}^{|np|} a_{np} \cdot (\bar{E}_{(|np|+1-np)}^R - \bar{E}_{(np)}^R) \right)^2 / \sum_{w=1}^{|w|} (\bar{E}_{(np)}^R - \bar{E}^R)^2 \quad (3.13)$$

$$SW_E^S = \left(\sum_{np=1}^{|np|} a_{np} \cdot (\bar{E}_{(|np|+1-np)}^S - \bar{E}_{(np)}^S) \right)^2 / \sum_{w=1}^{|w|} (\bar{E}_{(np)}^S - \bar{E}^S)^2 \quad (3.14)$$

$$\bar{E}^R = \sum_{np=1}^{|np|} \bar{E}_{(np)}^R / (|np|) \quad (3.15)$$

$$\bar{E}^S = \sum_{np=1}^{|np|} \bar{E}_{(np)}^S / (|np|) \quad (3.16)$$

For the S-W normality tests with data samples of \bar{E}^R and \bar{E}^S , the Null Hypothesis is that elements of \bar{E}^R and \bar{E}^S are normally distributed. We have utilized the software collected from [57] to perform the S-W normality test. For distribution of \bar{E}^R and \bar{E}^S , corresponding p values are found as 0.54 and 0.65 respectively, which are greater than critical value, α as 0.05. Hence, no strong evidence could be found to reject the Null Hypothesis that elements of \bar{E}^R and \bar{E}^S have come from normal distribution.

3.6.1.2 Parametric Hypothesis Testing and Test Error

Results for normality testing through the S-W normality test have suggested that elements of \bar{E}^R and \bar{E}^S follow normal distribution, which meets the prior condition of parametric tests. Positive numeric value of every element of $X^{\bar{E}}$ (3.12) articulated in Table 3.4 refers to the fact that energy consumption by *SRTDVMC* is numerically lower compared to *RTDVMC* for different workload scenarios as

presented in experiments. We hence aim to perform parametric hypothesis test to find whether simulation output samples, $X_{NT, PL}^{\bar{E}}$ (3.12) featuring difference between two corresponding means of *RTDVMC* and *SRTDVMC* associated to a unique combination of Nectar and PlanetLab workload is statistically significant. Our sample size is less than 30 and means of no more than two DVMC algorithms (i.e., *SRTDVMC* and *RTDVMC*) would be compared. Therefore, among different parametric tests we opt to use the *t-test*, instead of *Z-test*, *F-test*, and *ANOVA*. Based on the data samples, the *t-tests* can be classified into three groups: One sample, Two Independent Samples and Paired Samples *t-test*. For a specific combination of *NL* and *PL*, corresponding $\bar{E}_{NT, PL}^R$ and $\bar{E}_{NT, PL}^S$ has a relationship, as $\bar{E}_{NT, PL}^R$ and $\bar{E}_{NT, PL}^S$ represent \bar{E}_{CDC} for *RTDVMC* and *SRTDVMC* respectively, under a particular workload distribution scenario. Therefore, the paired two tail *t-test* is performed.

The null hypothesis with the *t-test* is that mean CDC energy consumption with *RTDVMC*, \bar{E}^R and mean CDC energy consumption with *SRTDVMC*, \bar{E}^S are same, while the alternative hypothesis is that $\bar{E}^S < \bar{E}^R$. Utilizing (3.17)–(3.19), the test statistic for the *t-test*, denoted by $t_{\bar{X}^{\bar{E}}}$ is found as 2.13 and the corresponding *p* value is found as 7.10693×10^{-6} , which is lower than critical value, α as 0.05. For clear understanding of the interpretation of the *t-test* result, we have first explained the performed the *t-test* in more details in the following.

$$t_{\bar{X}^{\bar{E}}} = \left(\left(\bar{X}^{\bar{E}} - 0 \right) / \left(\hat{\sigma}_{\bar{X}^{\bar{E}}} \right) \right) \quad (3.17)$$

$$\bar{X}^{\bar{E}} = \left(\left(\sum_{np=1}^{|Nectar| \cdot |PLab|} \left(X_{np}^{\bar{E}} \right) \right) / (|Nectar| \cdot |PLab|) \right) \quad (3.18)$$

$$\hat{\sigma}_{\bar{X}^{\bar{E}}} = \sqrt{\frac{\left(\sum \left(X_{np}^{\bar{E}} - \bar{X}^{\bar{E}} \right)^2 / ((|Nectar| \cdot |PLab|) - 1) \right)}{(|Nectar| \cdot |PLab|)}} \quad (3.19)$$

Previously through S-W normality test results, we have shown that distributions of \bar{E}^R and \bar{E}^S are normal distributions. It is critical to note that, if we subtract two corresponding elements of two different normal distributions, then the resulting distribution is a normal distribution. Hence, elements of $X^{\bar{E}}$ (3.12) featuring difference between two corresponding means of *RTDVMC* and *SRTDVMC* are normally distributed. Since elements of $X^{\bar{E}}$ are normally distributed, therefore, the distribution of their means, denoted by $\bar{X}^{\bar{E}}$ (3.18), is also normally distributed. Now, assuming the null hypothesis true that \bar{E}^R and \bar{E}^S are same, the mean of the

distribution of $\bar{X}^{\bar{E}}$ is 0. Nonetheless, utilizing (3.18) with our experimental results articulated in Table 3.4, value of $\bar{X}^{\bar{E}}$ has been found as 2.03 which mismatches with the value of the mean as 0 assuming the null hypothesis is true. As a result, we have then estimated the probability of $\bar{X}^{\bar{E}}$ to be 2.03, given the null hypothesis is true. In order to determine such probability, utilizing (3.17), we have calculated that how many standard deviations far away is our experimental mean (i.e., 2.03) from the distribution mean (i.e., 0), aka the test statistic for the *t-test*.

The test statistic for the *t-test*, denoted by $t_{\bar{X}^{\bar{E}}}$ is found as 2.13 and the corresponding probability is found as 7.10693×10^{-6} . Now, 7.10693×10^{-6} is less than 0.05. In other words, the outcome of the *t-test* shows that if the null hypothesis is true that mean of distribution of $\bar{X}^{\bar{E}}$ is 0, there remains less than 5% chance for $\bar{X}^{\bar{E}}$ to be 2.03. According to the rule of the *t-test*, we can then argue that despite such low probability, since we still have received $\bar{X}^{\bar{E}}$ as 2.03, therefore, the null hypothesis itself cannot be true. So, we retain the alternative hypothesis as true. If $\bar{X}^{\bar{E}}$ (i.e., the mean of minimization of mean CDC energy consumption by *SRTDVMC* compared to *RTDVMC*) was not significant from the perspective of such inferential statistics as the *t-test*, then respective *p* value would not have been found as lower than 0.05, which gives strong evidence to reject the null hypothesis itself. Hence, we have provided evidence through utilizing inferential statistics that the performance improvement through *SRTDVMC* compared to *RTDVMC* in terms of mean CDC energy consumption is statistically significant.

Errors related to the *t-test* can be classified into two groups: Type I error and Type II error. Type I error refers to the total probability of falsely rejecting the null hypothesis while it was true, and Type II error refers to the total probability of falsely rejecting alternative hypothesis while it was true. The null hypothesis has been rejected based on the corresponding probability value of 7.10693×10^{-6} . Hence, the probability is 7.10693×10^{-6} that we have rejected the null hypothesis while it was true, aka Type I error. In the following section, the simulation results in relation to VM migration has been presented.

3.6.2 VM Migration

VM consolidation is applied to regulate CDC energy consumption. However, one major downside of VM consolidation is that VM consolidation is impossible without VM migration, while, increased VM migration increases network overhead. *SRTDVMC* being a multi-objective DVMC algorithm is designed to minimize CDC energy consumption without incurring increased VM migration. In Fig. 3.8, we have illustrated mean total number of VM migrations with *RTDVMC* and *SRTDVMC*, denoted by $\bar{\psi}^R$ and $\bar{\psi}^S$, respectively. Let $X^{\bar{\psi}}$ (3.20) denotes the set representing difference of mean total number of VM migrations between *RTDVMC* and *SRTDVMC* under different workload scenario, as articulated in Table 3.7.

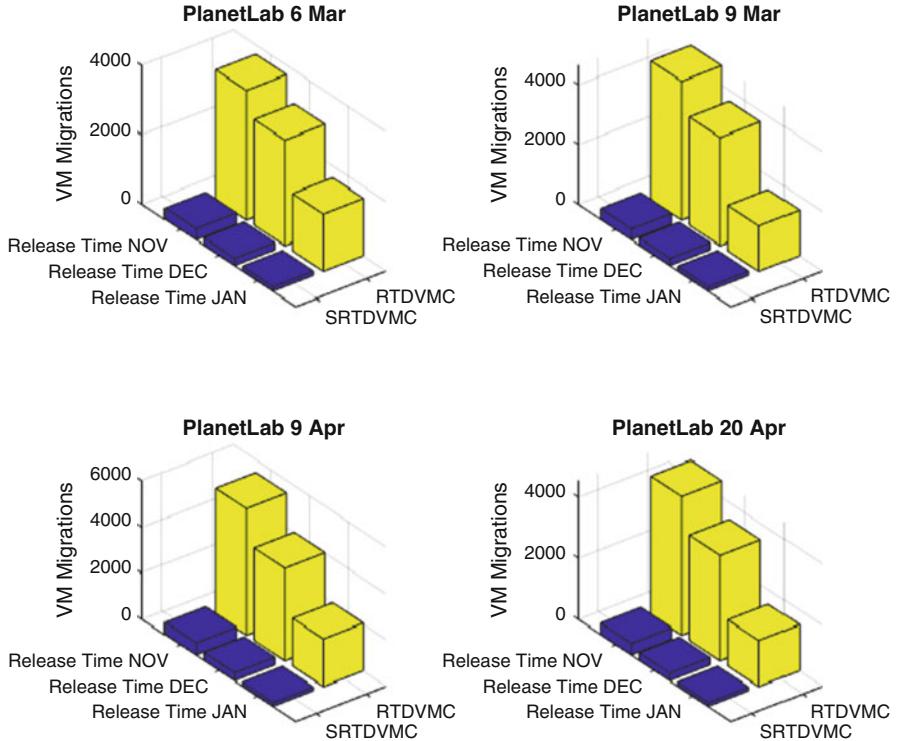


Fig. 3.8 Mean total number of VM migration of SRTDVMC vs RTDVMC

Table 3.7 Minimization of mean number of VM migration by SRTDVMC compared to RTDVMC

		PlanetLab (CPU utilization distribution)			
		6 March	9 March	9 April	20 April
Nectar VMRT	Nov	3416	4279	4951	4162
	Dec	2838	3385	3667	3206
	Jan	1531	1381	1908	1390

$$X^{\bar{\psi}} = \left\{ X_{NT, PL}^{\bar{\psi}} \right\}_{|Nectar| \cdot |PLab|} = \left\{ \bar{\psi}_{NT, PL}^R - \bar{\psi}_{NT, PL}^S \right\}_{|Nectar| \cdot |PLab|} \quad (3.20)$$

From Table 3.7, we can observe that SRTDVMC outperforms RTDVMC in terms of mean of total number of VM migration. One might argue that such improvement is merely a random event and the results are not statistically significant. To rebut such argument, we have performed the *t-test* on experimental results to check if results are statistically significant or not. One critical point to note that the *t*-

Table 3.8 Mean of total number of VM migration for *RTDVMC*

np	1	2	3	4	5	6	7	8	9	10	11	12
$\bar{\psi}_{(np)}^R$	1511	1524	1654	2066	3046	3444	3619	3738	3999	4535	4639	5472

Table 3.9 Mean of total number of VM migration for *SRTDVMC*

np	1	2	3	4	5	6	7	8	9	10	11	12
$\bar{\psi}_{(np)}^S$	120	123	143	157	208	234	238	322	332	360	373	521

test cannot prove statistical significance of target data, if data is not normally distributed. To address that issue, normality testing is required to prove that the data representing mean minimization of CDC energy consumption obtained through *SRTDVMC* compared to existing literature as portrayed in Fig. 3.8 is normally distributed. In the following section, we have discussed normality testing performed on our experimental results.

3.6.2.1 Normality Testing

We have applied the S-W normality test with set of mean of total number of VM migration by *RTDVMC* and *SRTDVMC*, denoted by $\bar{\psi}^R$ and $\bar{\psi}^S$, respectively. Test statistics for the S-W normality test with $\bar{\psi}^R$ and $\bar{\psi}^S$, referred to as SW_{ψ}^R and SW_{ψ}^S , respectively can be calculated through (3.21)–(3.24) [54, 55]. a_{np} weights are available in Shapiro-Wilk Table [56]. Different $\bar{\psi}_{(np)}^R$ and $\bar{\psi}_{(np)}^S$ values are presented in Tables 3.8 and 3.9.

$$SW_{\psi}^R = \left(\sum_{np=1}^{|np|} a_{np} \cdot (\bar{\psi}_{(|np|+1-np)}^R - \bar{\psi}_{(np)}^R) \right)^2 / \sum_{np=1}^{|np|} (\bar{\psi}_{(np)}^R - \bar{\psi}^R)^2 \quad (3.21)$$

$$SW_{\psi}^S = \left(\sum_{np=1}^{|np|} a_{np} \cdot (\bar{\psi}_{(|np|+1-np)}^S - \bar{\psi}_{(np)}^S) \right)^2 / \sum_{np=1}^{|np|} (\bar{\psi}_{(np)}^S - \bar{\psi}^S)^2 \quad (3.22)$$

$$\bar{\psi}^R = \sum_{np=1}^{|np|} \bar{\psi}_{(np)}^R / (|np|) \quad (3.23)$$

$$\bar{\psi}^S = \sum_{np=1}^{|np|} \bar{\psi}_{(np)}^S / (|np|) \quad (3.24)$$

The null hypothesis for the S-W normality tests with data samples of $\bar{\psi}^R$ and $\bar{\psi}^S$ is that data is normally distributed. For distribution of $\bar{\psi}^R$ and $\bar{\psi}^S$, corresponding p values are found as 0.42 and 0.37 respectively, which are greater than critical value, α as 0.05. Hence, no strong evidence could be found to reject Null Hypothesis that elements of $\bar{\psi}^R$ and $\bar{\psi}^S$ have come from normal distribution. As such, in the following section we have proceeded with the *t-test* to verify if the reduction of VM migration by *SRTDVMC* compared to *RTDVMC* as obtained through experiments is statistically significant.

3.6.2.2 Parametric Hypothesis Testing and Test Error

Previously, we have explained the reason of choosing the two sample paired *t-test*. We aim to prove that minimization of mean of total number of VM migration by *SRTDVMC* compared to *RTDVMC* is statistically significant. As null hypothesis, we assume that the opposite is true. Hence, the null hypothesis is that mean of total number of VM migration, $\bar{\psi}^R$ and mean of total number of VM migration with *SRTDVMC*, $\bar{\psi}^S$ are same, while the alternative hypothesis is that $\bar{\psi}^S < \bar{\psi}^R$. Utilizing (3.25)–(3.27), the test statistic for the *t-test*, denoted by $t_{\bar{X}\bar{\psi}}$ is found as 2.48 and the corresponding p value is found as 1.64×10^{-6} , which is lower than critical value, α as 0.05. In the following, we have elaborately discussed the interpretation of the *t-test* result.

$$t_{\bar{X}\bar{\psi}} = \left(\left(\bar{X}^{\bar{\psi}} - 0 \right) / \left(\hat{\sigma}_{\bar{X}\bar{\psi}} \right) \right) \quad (3.25)$$

$$\bar{X}^{\bar{\psi}} = \left(\left(\sum_{np=1}^{|Nectar| \cdot |PLab|} \left(X_{np}^{\bar{\psi}} \right) \right) / (|Nectar| \cdot |PLab|) \right) \quad (3.26)$$

$$\hat{\sigma}_{\bar{X}\bar{\psi}} = \sqrt{\frac{\left(\sum \left(X_{np}^{\bar{\psi}} - \bar{X}^{\bar{\psi}} \right)^2 / ((|Nectar| \cdot |PLab|) - 1) \right)}{(|Nectar| \cdot |PLab|)}} \quad (3.27)$$

In the earlier section, we have shown that such distributions as $\bar{\psi}^R$ and $\bar{\psi}^S$ are normal distributions. As a result, elements of $X^{\bar{\psi}}$ (3.20) are normally distributed. Furthermore, since elements of $X^{\bar{\psi}}$ are normally distributed, therefore, distribution of their means, denoted by $\bar{X}^{\bar{\psi}}$ (3.26) is also normally distributed. Now, under the given null hypothesis that $\bar{\psi}^R$ and $\bar{\psi}^S$ are same, the mean of the distribution $\bar{X}^{\bar{\psi}}$ is 0. Applying the results articulated in Table 3.7, into (3.26), $\bar{X}^{\bar{\psi}}$ is found as 3010. We have then determined the probability of $\bar{X}^{\bar{\psi}}$ to be 3010, under the scenario that null

hypothesis is true (i.e., \bar{X}^{ψ} is 0). To determine that probability, we have estimated the distance between our experimental mean (i.e., 3010) and the distribution mean (i.e., 0) in terms of standard deviation, aka the test statistic for the *t-test*, $t_{\bar{X}^{\psi}}$ (3.25).

$t_{\bar{X}^{\psi}}$ is found as 2.48 and the corresponding probability is found as 1.64×10^{-6} . Now, 1.64×10^{-6} is less than 0.05. In other words, the outcome of the *t-test* shows that given the null hypothesis is true, there is less than 5% chance of \bar{X}^{ψ} to be 3010. Nevertheless, despite such low probability, since we still have received \bar{X}^{ψ} as 3010, therefore, we can rebut that the null hypothesis itself is not true. Hence, we retain the alternative hypothesis as true. If the mean of minimization of mean CDC energy consumption by *SRTDVMC* compared to *RTDVMC* was insignificant, then respective *p* value would not have been found as lower than 0.05. Thus, through the *t-test* results we have provided evidence that the reduction of VM migration by *SRTDVMC* compared to *RTDVMC* is statically significant. Since, the null hypothesis has been rejected based on the estimated probability of 1.64×10^{-6} , therefore, the probability of false rejection of null hypothesis while it was true, aka Type I error is 1.64×10^{-6} , which is very low. Experimental results also reveal several critical aspects as discussed in the following section.

3.7 Observations

Observation 1: From experiments results portrayed in Figs. 3.7 and 3.8, we can observe that such traditional DVMC algorithm as *RTDVMC* lacks in performance compared to *SRTDVMC* with the presence of state-of-the-art highly energy proportional PMs. Performance lacking by traditional DVMC algorithm is attributed to its flawed working principal that maximum energy efficiency is attainable through maximum load on PMs.

Observation 2: DVMC algorithm reduces energy consumption through VM migration, which detrimentally affects QoS. As such, concomitant minimization of energy consumption and VM migration are confronting objectives. Hence, developing a DVMC algorithm, which optimizes energy efficiency without increasing the number of VM migration is a much greater challenge than designing an algorithm that merely focuses on the former aspect and ignores the later aspect. *SRTDVMC* being designed to optimize both aspects, only migrates a VM if the respective benefit is greater than the corresponding cost. Our research outcome, as illustrated in Figs. 3.7 and 3.8, substantiates the success of such strategic VM consolidation technique of *SRTDVMC* in both aspects.

Observation 3: Experimental results reveal that energy consumption and VM migration correspond to VMRT. The number of PMs has not been altered. PlanetLab workload data of every single day has been combined with three different VMRT distributions of Nectar Cloud. Hence, from one single set of PlanetLab data of a day, three different sets of workload data have been created representing same number of VMs and same varying resource demand, but different VMRT distributions and performances of both algorithms change with the change in VMRT distributions. The underlying reason is that considering the entire lifetime, aggregated resource consumption by a VM with a relatively large VMRT is likely to be greater than the aggregated resource consumption by a VM with smaller VMRT. In addition, longer VMRT provides more time resulting into more likelihood of VM migrations.

Observation 4: Our research has highlighted that a correlation exists between energy consumption and VMRT. Existence of such correlation has also been found as true between number of VM migrations and VMRT. To elucidate further, one common pattern with both *SRTDVMC* and *RTDVMC* is unfolded that for any day's PlanetLab data, VMRT distribution of November 2013 displays the highest energy consumption and highest number of VM migration, while VMRT distribution of January 2014 proffers to the lowest energy consumption and lowest number of VM migration (Figs. 3.7 and 3.8). The answer lies within VMRT distributions of these months (Figs. 3.1, 3.2, and 3.3). Considering the sum of VMRT of all VMs, VMRT distribution of November 2013 consists of total 8343 days, which is the highest among all months, while VMRT distribution of January 2014 consists of total 727 days, which is the lowest among all months. Since, November 2013 represents the highest total workload duration resulting into highest total resource utilization; therefore, maximum energy consumption is observed for November 2013. Due to maximum duration of total workload existence, number of VM migrations is also found as maximum with November 2013. Similarly, since January 2014 features the lowest total workload duration resulting into lowest total resource utilization, hence, minimum energy consumption is observed for January 2014. Minimum duration of workload existence results into lowest number of VM migration for January 2014.

Observation 5: Experimental results in Figs. 3.7 and 3.8 also present the fact that energy consumption and VM migration are affected by the change in resource demand. Nectar VMRT data of one single month is blended with four different days of PlanetLab data resulting into four different sets of workloads, representing same VMRT, but different sets of resource demand and the performance of both algorithms change due to the variation in resource demand. The reason can be explained through Eqs. (3.8) and (3.9), showing that energy consumption is a function of CPU utilization, while the utilization refers to the sum the resource demand (3.3). Hence, energy consumption is affected by the change in resource demand. The reason of observing the change in total VM migrations with the change in resource demand is that further opportunities of VM consolidation arises as the resource demand changes. DVMC algorithm keeps capitalizing such consolidation

opportunities through further VM migration and hence, number of VM migration changes with the change in resource demand.

Observation 6: Furthermore, energy consumption and VM migration are associated with number of VMs. PlanetLab 9 April features the highest number of VMs, while PlanetLab 6 March holds the lowest number of VMs, which reflects on energy consumption (Fig. 3.7) and VM migration (Fig. 3.8). Higher the number of VMs, higher is the energy consumption and VM migration. The reason is that more VMs consume more resources, resulting into higher energy consumption and more VMs would normally contribute to a greater number of VM migrations. In the following section, we have summarized our research.

3.8 Conclusions and Future Work

3.8.1 Conclusions

While correlation exists between VMRT and energy consumption, traditional DVMC algorithms except *RTDVMC* do not consider heterogenous VMRT in VM consolidation decision process. Furthermore, existing algorithms consolidate VMs in as few PMs as possible based on the premise that optimal energy efficiency can be achieved with maximum load on PM. However, for state-of-the-art PMs, energy efficiency rather drops above 70% load level. Combining lack of consideration of heterogeneous VMRT and ignoring changed energy-efficient characteristics of underlying PMs, existing DVMC algorithms lack in performance in the context of real Cloud scenario with heterogeneous VMRT and state-of-the-art PMs.

RTDVMC considers heterogeneous VMRT. However, issues with *RTDVMC* are twofold – first, it does not take the changed energy-efficiency characteristics of modern PMs into account and second, it only aims to minimize energy consumption without considering VM migration minimization. VM migration, nonetheless, increases network overload causing degraded QoS and increased energy consumption by networking equipment. VM migration being an unavoidable part of VM consolidation, minimizing both energy consumption and VM migrations at the same time are confronting objectives. As such, in this paper, we have brought forth a novel multi-objective DVMC algorithm, namely, *SRTDVMC*, which aims to reduce VM migrations without compromising energy efficiency. Consideration of heterogeneous VMRT values in VM consolidation decision process enables *SRTDVMC* to be more energy efficient. On top of that, contrast to *RTDVMC*, *SRTDVMC* incorporates consideration both benefit and cost prior any VM migration. As a result, it is robust against the changed energy-efficiency characteristics of underlying PMs and can reduce VM migration without compromising energy-efficiency compared to *RTDVMC*.

Performance of *SRTDVMC* has been tested through the most popular Cloud-based simulation tool, namely, CloudSim, in the context of hundreds of different cutting-edge PMs and thousands of VMs representing heterogenous VMRT of real Nectar Cloud, as the assigned workload reflects real Cloud workload obtained from PlanetLab. The empirical outcome exhibits the superiority of *SRTDVMC* over *RTDVMC* in both metrics – CDC energy consumption and VM migration. Three key elements are extracted from our research. First, based on our experiments, VMRT impacts on both aspects – energy consumption and VM migration, and hence, DVMC algorithms are needed to be developed considering the presence of heterogeneous VMRT. Second, such working principal of existing algorithms that maximum energy efficiency is achievable at maximum load on PM is found as false for state-of-the-art PMs, resulting into performance inefficiencies. Our proposed *SRTDVMC* algorithm addresses this issue. Third, simulation results show that if corresponding cost and benefit are considered prior VM migration, then concomitant optimization of both aspects – reduction of energy consumption and VM migration – can be achieved. In the following section, we have suggested several future research pathways to further improve the energy-efficient management of CDC.

3.8.2 Future Work

Accurate estimation of VMRT information plays an important role in minimizing energy consumption through release time-based DVMC algorithm. To explain further, if input VMRT value (i.e., VMRT value given as input in the system) is greater/lower than the true VMRT value (i.e., the authentic time when the VM would truly be removed from CDC), then such decisions as source PM selection, migrating VMs selection and destination PM selection taken on the basis of VMRT value would also be inaccurate, resulting into inefficient performance. Diverse research pathways can be examined in this regard. Advanced machine learning based techniques, neural networks and so forth can be embodied in automated release time-based DVMC algorithms to generate *PVMRT*. Each technique would consist its own set of advantages and disadvantages. It would be an interesting research problem to measure the change in system performance with the change in accuracy of input VMRT.

Different categories of DVMC algorithms can be found from literature, each comes with its own set of advantages and disadvantages. Scope of such diverse heuristic and meta-heuristic DVMC algorithms is yet to be explored for heterogeneous workload coupled with heterogeneous VMRT and state-of-the-art highly energy proportional PMs. As part of future work, we aim to explore that research domain.

References

1. M. Smolaks, *Google Data Center Chief: Go Green Because Your Customers Care* (Data Centre Dynamics Ltd., London, 2014) Available from: <http://www.datacenterdynamics.com/content-tracks/design-build/google-data-center-chief-go-green-because-your-customers-care/91942.fullarticle>
2. DatacenterDynamics: 3M: The future of data centers will depend on cooling technologies [A detailed look at immersion cooling, with a little help from DCD] (2018). Available from: <http://www.datacenterdynamics.com/content-tracks/power-cooling/3m-the-future-of-data-centers-will-depend-on-cooling-technologies/99977.article>
3. A. Shehabi, S.J. Smith, N. Horner, I. Azevedo, R. Brown, J. Koomey, et al., *United States Data Center Energy Usage Report* (Lawrence Berkley National Laboratory, Berkley, 2016)
4. M. Avgerinou, P. Bertoldi, L. Castellazzi, Trends in data centre energy consumption under the European code of conduct for data centre energy efficiency. *Energies* **10**(10), 1470 (2017)
5. European Commission: Code of conduct for energy efficiency in data centres (2016). Available from: <https://ec.europa.eu/jrc/en/energy-efficiency/code-conduct/datacentres>
6. A. Mark, B. Paolo, B. John, N. Liam, R. Andre, T. Robert, *2018 Best Practice Guidelines for the EU Code of Conduct on Data Centre Energy Efficiency* (European Union, 2018)
7. NECTAR CLOUD 2018. Available from: <https://nectar.org.au/research-cloud/>
8. F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, et al., Using ant colony system to consolidate VMs for green cloud computing. *IEEE Trans. Serv. Comput.* **8**(2), 187–198 (2015)
9. F. Farahnakian, R. Bahsoon, P. Liljeberg, T. Pahikkala, (eds.), Self-adaptive resource management system in IaaS clouds, in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD), June 27–July 2 2016*, (IEEE, 2016)
10. F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N.T. Hieu, H. Tenhunen, Energy-aware VM consolidation in cloud data centers using utilization prediction model. *IEEE Trans. Cloud Comput.* **7**(2), 524–536 (2019)
11. A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Futur. Gener. Comput. Syst.* **28**(5), 755–768 (2012)
12. A. Beloglazov, R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Comput.* **24**(13), 1397–1420 (2012)
13. M.H. Fathi, L.M. Khanli, Consolidating VMs in green cloud computing using harmony search algorithm, in *Proceedings of the 2018 International Conference on Internet and e-Business*, (ACM, Singapore, 2018), pp. 146–151
14. A. Mosa, R. Sakellariou, (eds.), Virtual machine consolidation for cloud data centers using parameter-based adaptive allocation, in *Proceedings of the Fifth European Conference on the Engineering of Computer-Based Systems*, (ACM, 2017)
15. D. Alsadie, E.J. Alzahrani, N. Sohrabi, Z. Tari, A.Y. Zomaya, (eds.), DTFA: A dynamic threshold-based fuzzy approach for power-efficient VM consolidation, in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), 1–3 Nov 2018*, (IEEE, 2018)
16. K. Ajmera, T.K. Tewari, (eds.), Greening the cloud through power-aware virtual machine allocation, in *2018 Eleventh International Conference on Contemporary Computing (IC3), 2–4 Aug 2018*, (IEEE, 2018)
17. Y. Liu, X. Sun, W. Wei, W. Jing, Enhancing energy-efficient and QoS dynamic virtual machine consolidation method in cloud environment. *IEEE Access* **6**, 31224–31235 (2018)
18. H. Wang, H. Tianfield, Energy-aware dynamic virtual machine consolidation for cloud datacenters. *IEEE Access* **6**, 15259–15273 (2018)

19. Y. Chang, C. Gu, F. Luo, (eds.), Energy efficient virtual machine consolidation in cloud datacenters, in *2017 4th International Conference on Systems and Informatics (ICSAI), 11–13 Nov 2017*, (IEEE, 2017)
20. S.B. Shaw, J.P. Kumar, A.K. Singh, (eds.), Energy-performance trade-off through restricted virtual machine consolidation in cloud data center, in *2017 International Conference on Intelligent Computing and Control (I2C2), 23–24 June 2017*, (IEEE, 2017)
21. D. Alsadie, Z. Tari, E.J. Alzahrani, A.Y. Zomaya, (eds.), LIFE: A predictive approach for VM placement in cloud environments, in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), 30 Oct–1 Nov 2017*, (IEEE, 2017)
22. M.A.H. Monil, A.D. Malony, (eds.), QoS-aware virtual machine consolidation in cloud datacenter, in *2017 IEEE International Conference on Cloud Engineering (IC2E), 4–7 Apr 2017*, (IEEE, 2017)
23. E. Arianyan, (ed.), Multi objective consolidation of virtual machines for green computing in Cloud data centers, in *2016 8th International Symposium on Telecommunications (IST), 27–28 Sept 2016*, (IEEE, 2016)
24. R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **41**(1), 23–50 (2011)
25. The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, *CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services [Internet]* (Cloud Computing and Distributed Systems (CLOUDS) Laboratory Department of Computer Science and Software Engineering The University of Melbourne, Melbourne, n.d.) Available from: <http://cloudbus.org/cloudsim/>
26. M.A. Khan, A. Paplinski, A.M. Khan, M. Murshed, R. Buyya, Exploiting user provided information in dynamic consolidation of virtual machines to minimize energy consumption of cloud data centers, in *Third International Conference on Fog and Mobile Edge Computing (FMEC), 23–26 Apr 2018, Barcelona, Spain*, (IEEE, 2018)
27. S.S. Masoumzadeh, H. Hlavacs, A gossip-based dynamic virtual machine consolidation strategy for large-scale cloud data centers, in *Proceedings of the Third International Workshop on Adaptive Resource Management and Scheduling for Cloud Computing, Chicago, IL, USA*, (ACM, 2016), pp. 28–34
28. Standard Performance Evaluation Corporation, *SPECpower_ssj2008* (Standard Performance Evaluation Corporation, 2011) Available from: http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00338.html
29. Standard Performance Evaluation Corporation, *SPECpower_ssj2008* (Standard Performance Evaluation Corporation, 2011) Available from: http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00339.html
30. Standard Performance Evaluation Corporation, *SPECpower_ssj2008* (Standard Performance Evaluation Corporation, 2017) Available from: https://www.spec.org/power_ssj2008/results/res2017q4/power_ssj2008-20171010-00789.html
31. Standard Performance Evaluation Corporation, *SPECpower_ssj2008* (Standard Performance Evaluation Corporation, 2017) Available from: https://www.spec.org/power_ssj2008/results/res2017q4/power_ssj2008-20171010-00790.html
32. Standard Performance Evaluation Corporation, *SPECpower_ssj2008* (Standard Performance Evaluation Corporation, 2017) Available from: https://www.spec.org/power_ssj2008/results/res2017q4/power_ssj2008-20171009-00787.html
33. J. von Kistowski, H. Block, J. Beckett, K.-D. Lange, J.A. Arnold, S. Kounev, (eds.), Analysis of the influences on server power consumption and energy efficiency for CPU-intensive workloads, in *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, (ACM, 2015)
34. J. von Kistowski, J. Beckett, K.-D. Lange, H. Block, J.A. Arnold, S. Kounev, (eds.), Energy efficiency of hierarchical server load distribution strategies, in *2015 IEEE 23rd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, (IEEE, 2015)

35. D. Wong, M. Annavaram, (eds.), Implications of high energy proportional servers on cluster-wide energy proportionality, in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, (IEEE, 2014)
36. J. Gustedt, E. Jeannot, M. Quinson, Experimental methodologies for large-scale systems: A survey. *Parallel Process. Lett.* **19**(3), 399–418 (2009)
37. M.A. Khan, A. Paplinski, A.M. Khan, M. Murshed, R. Buyya, *Dynamic Virtual Machine Consolidation Algorithms for Energy-Efficient Cloud Resource Management: A Review*, Sustainable Cloud and Energy Services (Springer, 2018), pp. 135–165
38. H. Shen, L. Chen, CompVM: A complementary VM allocation mechanism for cloud systems. *IEEE ACM Trans. Netw.* **26**(3), 1348–1361 (2018)
39. D. Wong, (ed.), Peak efficiency aware scheduling for highly energy proportional servers, in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, (IEEE, 2016)
40. R. Nasim, A.J. Kassler, (eds.), A robust Tabu Search heuristic for VM consolidation under demand uncertainty in virtualized datacenters, in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, (IEEE, 2017)
41. F. Ahamed, S. Shahrestani, B. Javadi, (eds.), Security aware and energy-efficient virtual machine consolidation in cloud computing systems, in *2016 IEEE Trustcom/BigDataSE/ISPA, 23–26 Aug 2016*, (IEEE, 2016)
42. D. Deng, K. He, Y. Chen, (eds.), Dynamic virtual machine consolidation for improving energy efficiency in cloud data centers, in *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS), 17–19 Aug 2016*, (IEEE, 2016)
43. G.B. Fioccola, P. Donadio, R. Canonico, G. Ventre, (eds.), Dynamic routing and virtual machine consolidation in green clouds, in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 12–15 Dec 2016*, (IEEE, 2016)
44. M. Khelghatdoust, V. Gramoli, D. Sun, (eds.), GLAP: Distributed dynamic workload consolidation through gossip-based learning, in *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, (IEEE, 2016)
45. A. Pahlevan, Y.M. Qureshi, M. Zapater, A. Bartolini, D. Rossi, L. Benini, et al., (eds.), Energy proportionality in near-threshold computing servers and cloud data centers: Consolidating or not? in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, (IEEE, 2018)
46. X. Peng, B. Pernici, M. Vitali, (eds.) *Virtual Machine Profiling for Analyzing Resource Usage of Applications* (Springer International Publishing, Cham, 2018)
47. M.H. Ferdaus, M. Murshed, R.N. Calheiros, R. Buyya, (eds.), Virtual machine consolidation in cloud data centers using ACO metaheuristic, in *European Conference on Parallel Processing*, (Springer, 2014)
48. Amazon Web Services: Amazon EC2 instance types (2017). Available from: <https://aws.amazon.com/ec2/instance-types/>
49. K. Park, V.S. Pai, CoMon: A mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Oper. Syst. Rev.* **40**(1), 65–74 (2006)
50. A. Ghasemi, S. Zahediasl, Normality tests for statistical analysis: A guide for non-statisticians. *Int. J. Endocrinol. Metab.* **10**(2), 486–489 (2012)
51. H.C. Thode, *Testing for Normality* (CRC Press, Boca Raton, 2002) 368 p
52. J. Peat, B. Barton, *Medical Statistics: A Guide to Data Analysis and Critical Appraisal* (John Wiley & Sons, 2008)
53. D.J. Steinskog, D.B. Tjøstheim, N.G. Kvamstø, A cautionary note on the use of the Kolmogorov–Smirnov test for normality. *Mon. Weather Rev.* **135**(3), 1151–1157 (2007)
54. Z. Charles, Real statistics using Excel [Internet]. [cited 2018]. Available from: <http://www.real-statistics.com/tests-normality-and-symmetry/statistical-tests-normality-symmetry/shapiro-wilk-test/>
55. M.E. Clapham, 9: Shapiro-Wilk test [video on the Internet] (2016) [updated 18 Jan 2016]. Available from: <https://www.youtube.com/watch?v=dRAqSsgkCUC>
56. Z. Charles, Real statistics using Excel [Internet]. [cited 2018]. Available from: <http://www.real-statistics.com/statistics-tables/shapiro-wilk-table/>
57. Z. Charles, Real statistics using Excel [Internet]. [cited 2019]. Available from: <http://www.real-statistics.com/free-download/real-statistics-software/>

Chapter 4

Energy-Efficient Resource Management of Virtual Machine in Cloud Infrastructure



H. Priyanka and Mary Cherian

Contents

4.1	Introduction	107
4.2	Background	109
4.3	Proposed Work.....	110
4.3.1	Design of Cloud Environment	110
4.3.2	Selection of Workload	111
4.3.3	Allocation Policy	112
4.3.4	Framework Monitor	116
4.3.5	Migration of Virtual Instances	118
4.3.6	VM Selection Policy.....	119
4.4	Result and Analysis	121
4.4.1	Experimental Results	121
4.4.2	Analysis of Complexity of Time	129
4.5	Conclusion and Future Work.....	129
	References	130

4.1 Introduction

Cloud computing has evolved as a computational tool for large enterprises because of the advancement in the Internet and virtualization technologies. Cloud computing can be defined “as a type of distributed system which consists of a collection of dynamically sourced resources with interconnected and virtualized computers” [1]. There is a huge demand for the cloud services in large industries. To meet the customer requirements, huge number of servers are needed at the data center. This consumes more energy and emission of carbon dioxide also increases. Cloud computing possesses some obstacles such as reliability, management of resources, security, efficiency, and power consumption [2]. One of the resource management

H. Priyanka (✉)

CSE R&D Centre, Dr. Ambedkar Institute of Technology, Bengaluru, Karnataka, India

M. Cherian

Department of CSE, Dr. Ambedkar Institute of Technology, Bengaluru, Karnataka, India

challenges is scheduling of tasks. Task scheduling refers to allocating cloudlets (tasks) to the resources which are accessible, this will improve the performance and maximize resource utilization [3, 4].

We have proposed a conceptual framework to reduce power consumption and improve the utilization of infrastructural resources of the data center. The framework efficiently assists in provisioning the resources and minimizing the migrations of a virtual instance.

The framework is designed with the help of a hybrid data center with highly configured infrastructural resources [5] for the experimental evaluation of our proposed algorithms. It monitors the processes of cloud environments dynamically, collects, and updates all the characteristics of data center frequently. In the proposed framework, the task scheduling initially receives the workloads with the help of the user's demand. It considers the configuration of Virtual Machines (VMs) and tasks by using the Expected Computation Time (ECT) matrix to generate an initial population.

The Genetically Enhanced and Shuffled Frog Leaping Algorithm (GESFLA) is proposed to select the optimal VMs to schedule the tasks and allocate them into Physical Machines (PMs). The proposed data center monitoring system checks the host utilization and observes the state and mode of it. If a host is in high or low loaded state by over or under-utilization of tasks in an active mode, the VMs are migrated from these hosts to imbalanced or idle hosts. The proposed algorithm executes applications through efficient resource management. Also, the proposed GESFLA reduces power consumption, improves the utilization of cloud infrastructures with minimum migration of VMs. This proposed algorithm performs better than the Genetic Algorithm and Particle Swarm Optimization (GAPSO) algorithm [6].

The virtualization of computer systems and use of servers are increased, thereby leading to increased power consumption and bandwidth demand in the data centers. For the migration of the VMs, transfer time also increases. This raises the data center's costs, carbon dioxide emissions and leads to global warming.

To address these problems, Genetic Algorithm (GA) [7] and SFLA [8] are chosen because GA generates uniform initial population and it can be applied for optimization problems. It is helpful as it gives a better solution. The GA cannot be applied for large optimization problems as it selects random crossover point for transferring of VMs and convergence time is also more. In our proposed method, we have combined SFLA with GA because of following advantages of SFLA:

- (i) The searching capability is faster.
- (ii) It is robust because it does the shuffling of frog memplexes and results in good solution.
- (iii) It has early convergence criteria.

The GA and SFLA are combined to get better results. The genetic algorithm convergence is slow, which can be overcome by applying SFLA. The GESFLA has following advantages in comparison with GAPSO.

- (i) Efficient optimization of task scheduling is carried out on the server.
- (ii) GAPSO algorithm takes relatively large computation time.
- (iii) As it takes more time to terminate, VM migration time is also increased and the resources are not utilized efficiently.

The research objective is stated below. The various phases of GESFLA are explained in Sect. 4.4.

The goals and objectives of research are as follows:

- (i) The applications running inside the virtual machine are tracked, and the load should be stored uniformly between all the virtual machines.
- (ii) To reduce the execution time of the task and transfer time of the VM, by tracking the usage of cloud resources of applications (tasks) running within VMs.
- (iii) To reduce Cloud data center power consumption by efficient scheduling of resources, VM migration allows cloud operators to achieve various resource management goals, such as green computing, load balancing, and maintenance of real-time servers.

This chapter is organized as follows: Section 4.2 elaborates the background of existing works. Section 4.3 explains the proposed methodology GESFLA and its different phases and algorithms. Section 4.4 evaluates the experimental results. Finally, in Sect. 4.5, this chapter is concluded and also future work is discussed.

4.2 Background

Multi-server energy consumption strategy aims at reducing the power usage of the data center by using the least amount of resources that are available in the PMs while switching off inactive PMs at the data center. Man et al. [9] and Ajiro et al. [10] used different methods, such as “First Fit,” “Best Fit,” to solve VMs allocation problem. A global optimum solution is obtained from the heuristic approaches. “Best fit heuristic solution” is inaccessible because it takes more convergence time, and the proposed work has one more limitation based on a “single objective function.” Beloglazov et al. [11] recommended a “Modified Best Fit Decreasing (MBFD) algorithm” by arranging the ascending order of PMs and descending order of VMs based on their processing capability. In the “First Fit Decreasing (FFD),” VM allocation on PMs is done after VMs and PMs are sorted. The drawbacks are: Distinct allocation goal for VMs and MBFD is not flexible as huge requested VMs get influenced at the data center. The VM allocation issue is solved by using different types of algorithms. Many researchers employed bio-inspired and evolution-inspired algorithms for the cloud data center for allocation of VMs, such as “Genetic Algorithm (GA) [12],” “Particle swarm optimization (PSO),” etc. Xiong et al. [13] used PSO to resolve the issue of allocating energy-efficient VMs. The authors considered a single VM type as it is the major drawback of this research.

“Ant colony”-based VM is proposed for VM allocation by the cloud data centers by Gao et al. [14]. The authors used only a single VM and PM combination which is the drawback of their work. Wang et al. [15] recommended the use of PSO to locate efficient VMs in a data center. The drawback of this research is that the allocations of VMs change the particle velocity that requires more iterations and gives an inaccurate result. “Particle Swarm Optimization (PSO) algorithm” [16] is derived from bird flocking, its a bio-inspired algorithm [17]. It is through PSO that each particle in the swarm gives a solution with four dimensions, its present location, the position, its speed, and the location found among all the particles in the population. Its location is set in the search area based on the position its neighborhood has reached. The limitations of PSO algorithms suffer from the partial optimism that triggers less accurate velocity and distance control. PSO is unable to address particle issues in the field of energy consumption. Sharma et al. proposed “Genetic algorithm and Particle Swarm Optimization (GAPSO)” [6, 8], hybridization of GA and PSO results in increasing the fitness value of the parent chromosomes, and thus allocation of VMs to the PMs is achieved in lesser time. The limitation of GAPSO algorithm takes more convergence time.

4.3 Proposed Work

In this section, the proposed method is discussed with different algorithms. Figure 4.1 shows the different phases of GESFLA. To efficiently carry out the process of migration, VM allocation and VM placement, the GESFLA is designed with various phases as follows:

1. Design of Cloud environment
2. Selection of workload
3. Allocation policy
4. Framework monitor
5. Migration of virtual instances
6. VM selection policy

4.3.1 *Design of Cloud Environment*

This phase designs the data center environment with the help of CloudSim simulator. The basic configuration of servers like types of a host, speed of processors (MIPS), number of processing elements, size of memory, bandwidth for I/O data transmission, the capacity of storage, and number of servers are placed to create a power data center. In the power data center environment, we can instantiate the virtual resources to fulfill the user’s requirements. As per the user’s requirements,

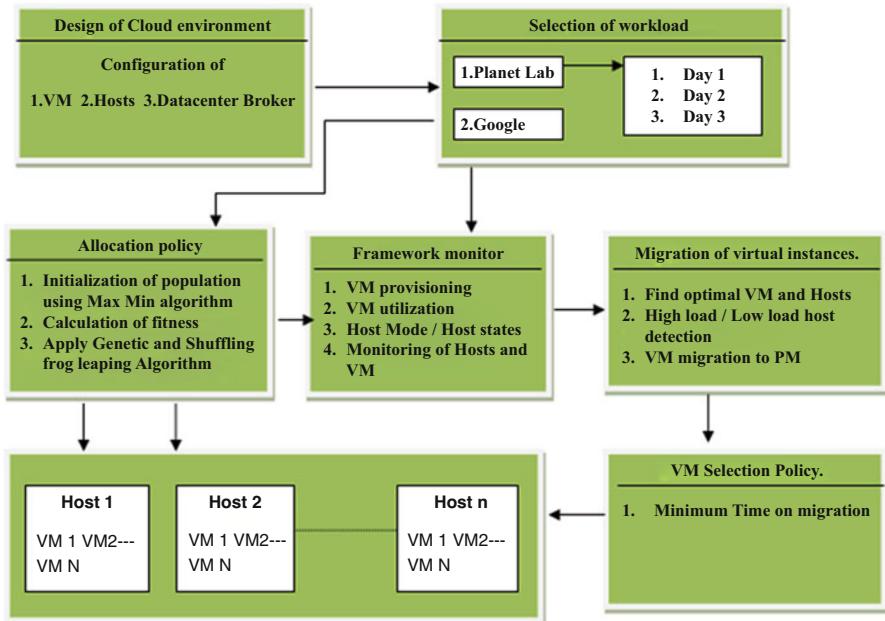


Fig. 4.1 Workflow of GESFLA

the VMs are configured and data center is initialized to experiment with our proposed and existing model.

4.3.2 Selection of Workload

(a) PlanetLab

The PlanetLab's realistic traces of VM utilization were collected for 3 days, it has 288 readings. The PlanetLab workload traces are converted into cloudlet format and are used as per user's requirements for experimental purpose and it consists of 10 days task details respectively.

(b) Google Datasets

A new “2019” trace [18] has been released by Google, which contains comprehensive Borg work scheduling data from eight different compute cluster of Google. The Google Computing Clusters (Borg cells) contains data of May 2019. The latest trace contents are an extension of the 2011 trace data [19, 20].

4.3.3 Allocation Policy

It consists of the following processes:

- (i) Adaptive selection of VM.
- (ii) Scheduling of Tasks.
- (iii) Identification of suitable hosts based on their states and modes.
- (iv) Assigning the VMs to the hosts. The processes are done by combining genetically enhanced and shuffled frog leaping algorithm as follows.

4.3.3.1 Genetic Algorithm

There are three basic genetic operations. They are selection, crossover, and mutation.

- (i) The population is generated from GA, in which parents are selected randomly.
- (ii) The parents are cross-bred to produce offspring in the crossover.
- (iii) The children will be altered as per the policy of mutation.

In genetic algorithm, the solution obtained is called individuals, and the generations are considered as variants of an algorithm. Apart from the basic three operations of the genetic algorithm, it has some preliminary and repetitive operations such as:

- (i) Initial production of the population.
- (ii) Fitness function.

(i) Initial Production of the Population

The initial population includes individuals in which chromosome sizes are specified. In this method, the initial population is considered for having a better solution from “MIN-MIN” [19] and “MAX-MIN” [20] scheduling methods.

Meta Task (MT): In the data center, the series of tasks will be assigned to available resources. $MT = \{t_1, t_2, t_3, \dots, t_n\}$.

“Min-Min Algorithm: It is based on the concept of assigning task having minimum completion time (MCT) for execution on the resource” [19]. The Min-Min algorithm is divided into two steps:

- (i) The completion time of each task is calculated on each resource of the meta task.
- (ii) The job with the shortest processing time is chosen and allocated to the respective resource, and the selected task is deleted from the meta task. This method continues until all the meta tasks are mapped.

Algorithm 4.1: Min-Min Algorithm

Input: Submitted task

Output: Completion time

Stage 1: The minimum completion time of each task is calculated.

1. The tasks (t_i) are submitted to meta task (MT).
2. Assign resources R_j .
3. Calculation of completion time $CT_{ij} = ET_{ij} + r_j$ is carried out.
4. Step 1 is ended.
5. Step 2 is ended.

Stage 2: Allocation of the resource to task t_i with a least completion time.

6. In the MT, find the task t_i with minimum completion time and the estimated resource.
7. Check for the task t_i , which has a minimum completion time and assign to the resource R_j .
8. From MT, delete task t_i .
9. Ready time of resource r_j is updated.
10. Unmapped tasks' completion time is upgraded in MT.
11. Until all activities have been mapped in meta task (MT), repeat 6–10 steps.
12. Step 6 ends.

“Max-Min Method. It is built on the concept of assigning a task having minimum completion time (MCT)” [20]. For resource execution, it must have the maximum completion time (fastest resource). This algorithm is divided into two steps.

- (i) The completion time of each task is calculated on each resource of the meta task.
- (ii) The job with the longest processing time is chosen and allocated to the respective resource, and the selected task is deleted from the meta task. This method continues until all the meta tasks are mapped.

Minimum Completion Time (MCT): Assigning each task to the available host so that the request can be completed as quickly as possible, which will yield the fastest result, which means evaluating the resource availability before allocating the job. Minimum completion time can be calculated by adding task t_i execution time, known as ET_{ij} , and resource ready time or start time, known as r_j . This can be expressed as:

$$MCT = ET_{ij} + r_j.$$

Algorithm 4.2: Max-Min Algorithm

Input: Submitted task

Output: Completion time

Stage 1: The minimum completion time of each task is calculated.

1. The tasks (t_i) are submitted to meta task (MT).
2. Assign resources R_j .
3. Calculation of completion time $CT_{ij} = ET_{ij} + r_j$ is carried out

4. Step 1 is ended.
5. Step 2 is ended.

Stage 2: Allocation of the resource to task t_i with a maximum completion time.

6. In the MT, find the task t_i with minimum completion time and the estimated resource.
7. Check for the task t_i , which has a minimum completion time and assign to the resource R_j .
8. From MT, delete task t_i .
9. Ready time of resource r_j is updated.
10. Unmapped tasks' completion time is upgraded in MT.
11. Until all activities have been mapped in meta task (MT), repeat 6–10 steps.
12. The loop of Step 6 is ended.

(ii) Fitness Function

The fitness value plays a key role in determining the individuals to create the next generation. In this work, the execution time (makespan) is considered, in which the fitness value and the fitness function are obtained through Max-Min method. To enhance resource management in the cloud environments, the utilization of resources is increased, the makespan of the task is minimized and the energy consumption of the host is also reduced. In this proposed algorithm, the GA operators, the selection, crossover, and mutation operations are applied. A set of solutions are produced; these solutions are considered to be the initial population of SFLA, and the solution which is generated will become the next population to GA. These operations are performed several times, and all the solutions are modified when the SFLA terminates the operations. In this algorithm, the early convergence is avoided by comparing the generated children with their parents. If the fitness value is improved in comparison with parents, then the children must replace the parents or it will be terminated. Based on the generation of the initial population, the fitness values are evaluated. The termination condition is verified for each iteration of the algorithm and the optimal solutions will be generated. Otherwise, the operators of GA and SFLA algorithms are applied respectively to the individuals.

4.3.3.2 Shuffled Frog Leaping Algorithm

“SFLA” [21] is a meta-heuristic strategy of optimization. It is focused on analyzing, imitating, and predicting a group of frog’s behavior while looking for the position with the maximum quantity of food available. It is used to resolve several non-linear, non-differentiable, and multi-model optimization challenges. SFLA has been introduced mainly to solve the problems that occur in the field of engineering. The most distinguished advantage of SFLA is its quick convergence speed. The SFLA incorporates the advantages of both the “Particle Swarm Optimization” and the “Memetic Algorithm” (MA) [22] influenced by social behavior.

SFLA is a random search algorithm based on population, influenced by nature memetics. In the SFLA, a feasible solution population is represented by a community of frogs which is divided into multiple groups called memeplexes. Every frog executes a local search in the memeplexes. The memeplexes are allowed to combine after a specified range of memetics evolution steps, and new memeplexes are created through a shuffling phase. The shuffling processes and local search continues until the convergence constraints are fulfilled.

Algorithm 4.3: Proposed GESFLA (Genetically Enhanced Shuffling Frog Leaping Algorithm)

Input: GenesList = List of VMs and List of Cloudlets (tasks)

Output: Scheduling of tasks to VM

1. *Create an initial population*
2. *Initialize parent 1 and parent 2*
3. *Calculate fitness value of both parent 1 and parent 2*
4. *Set fitness value to population list*
5. *Population list = genesList and fitnessMakespan*
6. *Apply selection operator of Genetic*
7. *Call method 1*
8. *Apply mutation operator*
9. *Calculate the fitness of offspring produced*
10. *If(fitness of child > fitness of parent)*
11. *Add the offspring into the population list*
12. *End if*
13. *Apply mutation operator*
14. *Check the fitness value of the mutated population with parent's*
15. *If(population.fitness makespan > mutationfitness)*
16. *Mutated population parent key value to be replaced*
17. *End if*
18. *Apply SFL Algorithm*
19. *Initialize frog_population list*
20. *Assign population of genetic to frog_population list*
21. *Select the population with a minimum fitness value*
22. *Create the frog list*
23. *Generate the virtual population and memeplex*
24. *Compute performance of memeplexes*
25. *Partitioning frog population into sub-memeplexes*
26. *Shuffle the memeplexes*
27. *Calculate the number of frogs for each submemeplex*
28. *Calculate the memetic evolution within each memeplex*
29. *Find execution time and makespan of sub-memeplex*
30. *Schedule the tasks to VMs*
31. *Calculate execution time and completion time of each memeplex*

Method 1: Selection Operator of Genetic

Input: Initial population

Population list = number of task, number of VM's

Pop size = number of parents

Output: fittest chromosome

1. Assign pop size
2. While initial population size <pop size
3. Do
4. Create pop size random integers
5. Compute fitness
6. Spin the roulette wheel = pop size
7. If then
8. Choose the chromosome
9. End if
10. End while

4.3.4 Framework Monitor

It verifies and examines the behavior of the data center. The host usage and resource management for the user's service level agreement are retained, so that it should not exceed the threshold limit. In case the hosts are in the state of high or low loaded [23, 24], it begins to move the VMs from the current host to another appropriate host. This constantly monitors the host states, configurations, and data center activities.

Consider number of PMs = “n” and number of VMs = “m” to execute the tasks in a data center, the use of PM (PM_u) will be evaluated based on the allocated number of VMs in it as shown in Eq. (4.1).

$$\sum_{i=1}^n CVM_{ij} PM_u = CPM_j \quad (4.1)$$

i = 1 to m, j = 1 to n.

Where CVM_{ij} is computing power of CPU (MIPS*Pes) of *i*th Virtual Machine (VM_{ij}) on *j*th PM.

CPM_j is the processing capability of the *j*th PM (PM_j) CPU (MIPS*Pes).

The PM in “active” mode has the following threshold limit to define the computational resource states.

- (i) Upper threshold limit of PM >75% – High state.
- (ii) 60% < threshold limit of PM ≤ 75% – Balanced state.
- (iii) 40% ≤ threshold limit of PM ≤ 60% – Imbalanced state.
- (iv) Lower threshold limit of PM <40% – Low state.

Algorithm 4.4: VM Allocation**Input:** List of VMs, List of PMs**Output:** Allocation of VM to PM

1. *Check for host state: active or idle*
2. *If (host== idle)*
3. *sleep*
4. *else*
5. *check for suitability for placing the VM*
6. *if (Host utilization=Over utilized || Host utilization== Underutilized)*
7. *Host utilization = over utilized*
8. *Go to step 10*
9. *Host utilization = under utilized*
11. *Go to step 22*
12. *End if*
13. *If (hostUtilization> 0.75)*
14. *Host = highloaded;*
15. *host.setHostStates(“High”);*
16. *host.setHostUtilization(hostUtilization);*
17. *End if*
18. *else if (hostUtilization<= 0.75 &&hostUtilization> 0.60)*
19. *Host =BalancedHost*
20. *host.setHostStates(“Balanced”);*
21. *host.setHostUtilization(hostUtilization);*
22. *End if*
23. *else if (hostUtilization<= 0.60 &&hostUtilization>0.40)*
24. *Host = ImbalancedHost;*
25. *host.setHostStates(“Imbalanced”);*
26. *host.setHostUtilization(hostUtilization);*
27. *End if*
28. *else if (hostUtilization<= 0.40 &&hostUtilization> 0.0)*
29. *host.setHostStates(“Low ”);*
30. *host.setHostUtilization(hostUtilization);*
31. *LowLoadedHost.add(host);*
32. *End*
33. *else*
34. *host.setHostUtilization(hostUtilization);*
35. *host.setHostMode(“Sleep ”);*
36. *End*
37. *Allocate VMs to PMs*
38. *End*

4.3.5 Migration of Virtual Instances

VM migration [25] plays a key role in virtualization, allowing for quick migration of a VM from one physical host to another. VM migration can prove helpful in different situations like (i) balancing the load, (ii) maintenance of DC, (iii) energy management, and (iv) the failures of VMs.

4.3.5.1 High Loaded Host Detection

If a host's CPU consumption drops below the threshold, then all VMs must be transferred from the current host to another host and to minimize idle power usage. The current host should be switched to the sleep mode. If the consumption crosses the higher limits of the threshold, it is required to transfer such VMs from the existing host to minimize resource usage and to avoid SLA violations [26].

4.3.5.2 Low Loaded Host Detection

In this case, the system has a list of underloaded hosts [27]. Next, the system needs to verify if VMs can be transferred to the other servers. Therefore, the proposed algorithm checks the possibility of reorganizing all VMs to other active hosts before commencing the live migrations. A host must fulfill three conditions for accepting any VM:

1. It should not be under pressure: Once the host is in the migration process, it cannot be approached for another migration.
2. It should have sufficient resources for VM: In this situation, the capacity of the CPU, memory, and bandwidth are considered, it should not exceed the capacity of the PM.
3. The VM should not be overloaded: After moving the VMs to an imbalanced or idle host and if migration overloads the imbalanced host, the migration cycle will be aborted to the specific host.

If the other active hosts accept all VMs, the host will be shifted to sleep mode and its VMs will be included in the transfer list; or else the host will remain operational. This cycle is repeated iteratively, for all under loaded hosts.

Algorithm 4.5: VM Placement

Input: List of VMs

Output: Placing the VM to PM

1. Create VM migration list
2. Sort VM according to their MIPS
3. Find the host for the VM migration

```

4. if (migratedHost != null)
5. go to step 9
6. migrate.put("host," migratedVm);
7. End if
8. return
9. Get host list
10. for (Host host : getHostList())
11. if (host.isSuitableForVm(vm))
12. Calculate host utilization
13. utilization=((vm.getMips*vm.getNumberOfPes())/host.getTotalMips())
14. result = host.vmCreate(vm);
15. if (utilization >= 0.75)
16. if (utilization != 0 && utilization > 0.75)
17. Migration of VM to the host
18. Get VM id and host Id
19. Calculate host utilization
20. End if
21. End if
22. else
23. allocatedHost = host;
24. host.setHostUtilization(utilization);
25. Get the allocation of VM Id to the host
26. Get allocated host Id
27. return allocated host;
28. End for
29. Get the Vms to Migrate FromUnderUtilized Host
30. for (Host HOST : host)
31. HOST.getId() & HOST.getHostStates());
32. Get the VM migrated list
33. for (Vmvm : HOST.getVmList())
34. if (!vm.isInMigration())
35. vmsToMigrate.add(vm)
36. End if
37. return vmsToMigrate;
38. End for

```

4.3.6 VM Selection Policy

When a server is overloaded, the next step is to select the VM's and transfer from the server. In this section, VM selection policy is discussed. After the VM is chosen for migration, the server is tested for over-loaded condition. If the server is still deemed

as overloaded, then again the VM selection policy will be implemented. The VM selection policy is discussed in Sect. 4.3.6.1.

4.3.6.1 Minimum Time on Migration (MTM)

The MTM strategy migrates a VM (VM) that takes less amount of time to accomplish a migration in comparison with other VMs assigned to the server. “The transfer time is measured as the amount of the RAM used by the VM, separated by the available network bandwidth of the server” [28]. V_k is a collection of VMs assigned to server “ k .” The MTM strategy discovers a VM (b) that matches the conditions shown in Eq. (4.2).

$$\frac{\text{RAM}_u(b)/\text{NET}_i}{\text{vm} \in V_k | \forall b \in V_k} \quad (4.2)$$

Where,

$\text{RAM}_u(b)$ = percentage of RAM presently used by the VM (b)

NET_i = spare bandwidth for the server k

The transfer time is calculated using the formula given below. The total bandwidth available for the Host is 1 GB. Half of the network bandwidth is used for communication among the VMs and another half is used for the VM migration.

1. VMs Transfer Time

The transfer time of the VM on the PM is calculated using the formula as follows:

$$\text{Transfer time} = \sum_{i=1}^n \frac{1}{2} * \frac{\text{VM}_{\text{RAM}}}{\text{HostBW}}$$

Where,

Transfer Time: This refers to the time that needed to migrate the task to the candidate VM.

VM_{RAM} = Total available ram of VM.

HostBW = Total network bandwidth = 1 GB/s.

2. Utilization Ratio

The resource utilization of each host is calculated using the formula as follows:

$$\text{Host utilization ratio} = (\text{VM requested MIPS}) / (\text{Total Host MIPS})$$

$$\text{Memory utilization ratio} = (\text{VM requested RAM}) / (\text{Host Total RAM})$$

3. Reduction of Power Consumption

In our model, the number of VMs = N , number of PMs = M , and set of resources = R . The Y_j variable checks if the PM_j is operational and the X_{ij} variable checks if VM_i assigned to PM_j. Our goal is to minimize a data center's energy consumption, based on the formula as shown in Eq. (4.3).

$$P_j = \left(P_j^{\text{active}} - P_j^{\text{idle}} \right) \times U_j^P + P_j^{\text{idle}} \quad (4.3)$$

Where, U_j^P is the usage of the CPU ($U_j^P \in [0, 1]$), and P_j^{active} and P_j^{idle} are the average power values while the jth PM is active and idle.

The consumption of total energy is shown in Eq. (4.4).

$$\text{Min} \sum_{j=1}^M P_j^{\text{PM}} = \sum_{j=1}^M Y_j \left(P_j^{\text{active}} - P_j^{\text{idle}} \right) \times \sum_{i=1}^N (X_{ij} \cdot R_{i,1}^{\text{vm}}) + P_j^{\text{idle}} \quad (4.4)$$

Where,

$R_{i,1}^{\text{vm}}$ -is a set of CPUs that are needed by VM_i

Virtual Machines (VMs) = N

Physical Machines (PMs) = M,

The total number of resources = R.

The Y_j variable determines whether PM_j is operational

The X_{ij} variable determines whether VM_i has been allocated to PM_j.

In the VM formula, R_i , VM is a range of CPUs that VM_i requires.

4.4 Result and Analysis

The suggested algorithm is designed to be implemented through the conceptual framework. Minimizing resource usage and optimizing the transfer time is the primary focus of our proposed algorithm.

4.4.1 Experimental Results

Experimental Setup: In heterogeneous environment, we have used various combinations of VMs and PMs. Table 4.1 shows the different types of PMs and VMs combinations for conducting detailed experiments. To check the proposed algorithm results, we have used the Cloudsim simulator [29]. Java language is used to implement our proposed GESFL algorithm.

The simulation is carried out using PlanetLab [28] datasets, which is a part of the CoMon [28] initiative. CPU usage data was taken from 500 different

Table 4.1 Configuration of simulation

Factor	Value
Host configuration	HP ProLiant ML110 G4 servers MIPS = 1860 PES = 2 RAM = 4096 HP ProLiant ML110 G5 servers MIPS = 2660 PES = 2 RAM = 4096 BW = 1 Gbit/s STORAGE = 1 GB
Number of hosts	800 400 = HP ProLiant ML110 G4 servers 400 = HP ProLiant ML110 G5 servers
VM configuration	1. Extra-Large [MIPS = 2500, PES = 1, RAM = 1740] 2. Medium [MIPS = 2000, PES = 1, RAM = 1740] 3. Small [$\text{MIPS} = 2000, \text{PES} = 1, \text{RAM} = 870$] 4. Micro [$\text{MIPS} = 500, \text{PES} = 1, \text{RAM} = 613$] BW = 1 Gbit/s = for all 4 types of VM.
Google cluster datasets	29 days
PlanetLab-workload	10 days

locations around the world which are running on the server from thousands of VMs. Therefore, every trace has $(24 \times 60)/5 = 288$ entries. CPU usage was taken in intervals of 5 min. The server with more cores are chosen mainly to simulate a large number of servers and to analyse the impact of consolidating VM. It is advantageous to simulate less powerful CPUs, as fewer tasks are needed for server overloading. Therefore, to check the resource management algorithms designed for multi-core processor architectures, dual-core CPUs are sufficient.

A dataset of Google Cluster Trace is published by Google in May 2019 and contains about 30 days of cell results. A collection of multiple machines sharing an eight cluster management system is defined by each cell. Each trace job involves one or more tasks that may contain many processes to be run on a single machine for each task. The trace data includes the following tables: Table of Machine Events, Table of Machine Attributes, Table of Instance Events, Table of Collection Events, and Table of Instance Usage.

In instance usage table, the *cpu_usage_distribution* and *tail_cpu_usage_distrib* vectors provide detailed information about the distribution of CPU consumption with 5 min intervals. These details are used to carry out the simulation.

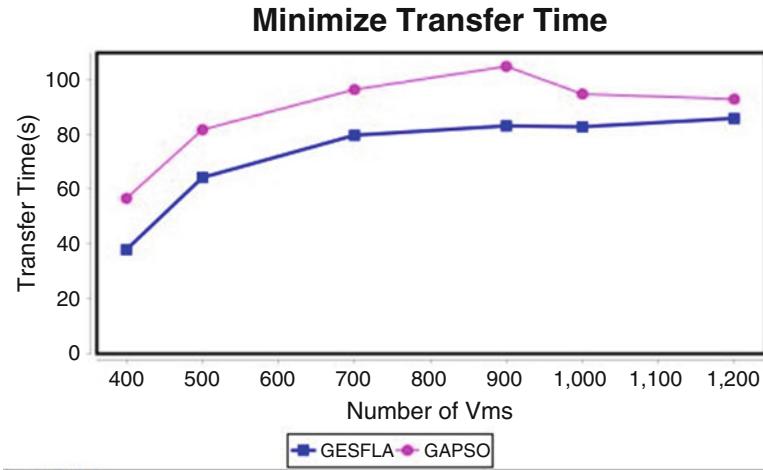


Fig. 4.2 VM transfer time (Google datasets)

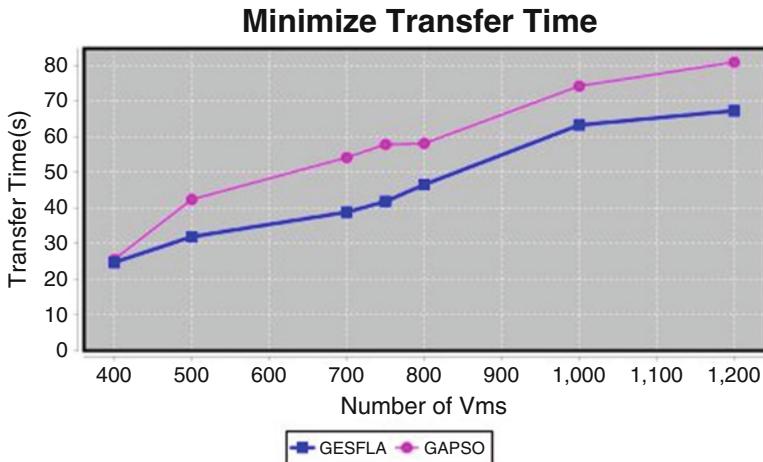


Fig. 4.3 VM transfer time (PlanetLab)

Further, the GESFL algorithm transfer time is shown in Figs. 4.2 and 4.3 is relatively smaller in comparison with GAPSO. If the load is balanced equally among VMs, then the CPU is also utilized efficiently as shown in Figs. 4.4 and 4.5. The energy usage is depicted in Figs. 4.6 and 4.7.

To determine the efficiency of our proposed GESFL algorithm, various combinations of VMs and PMs are compared with GAPSO in the context of resource utilization, time of migration, and energy consumption. GESFLA's migration time is calculated for the Google datasets and PlanetLab datasets as shown in Table 4.2.

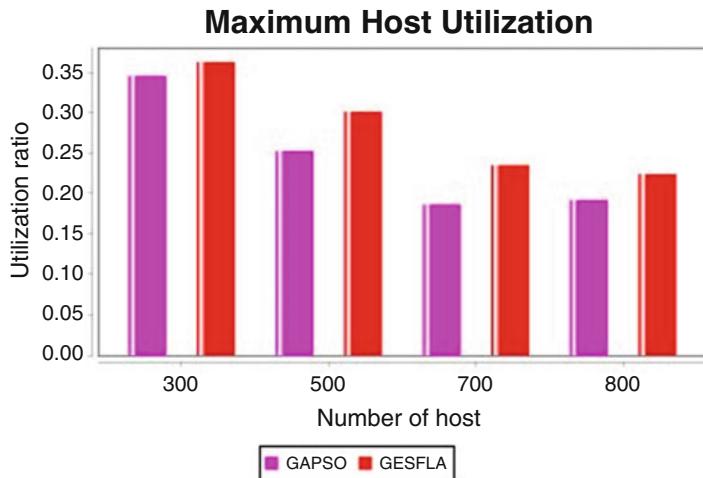


Fig. 4.4 Utilization of the host (Google datasets)

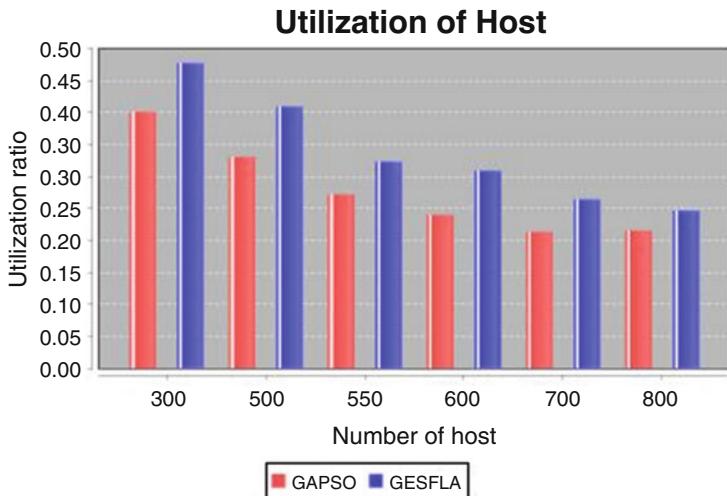


Fig. 4.5 Host utilization (PlanetLab)

The improvement in the maximum utilization of our suggested algorithm is calculated as shown in Table 4.3. The algorithm provides better outcomes for power usage over the GAPSO algorithm, due to the usage of the VM transfer strategy to turn off under loaded PMs at the data center. Thus, the CPU usage of PMs is improved by reducing the number of VMs in data center. The migration of VMs is built on fixed time intervals in data center. For every 60 s, we have received the utilization status of every used PM and several VMs that are assigned to underused PMs. Next, we can turn off the underused PMs in data center by transferring the

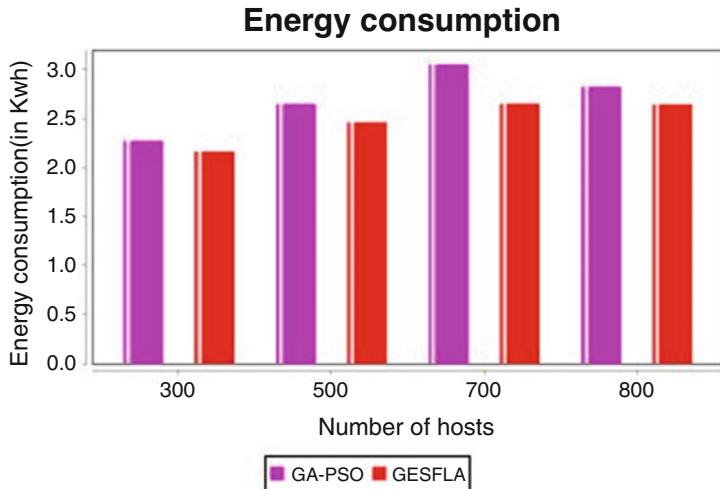


Fig. 4.6 Power consumption (Google datasets)

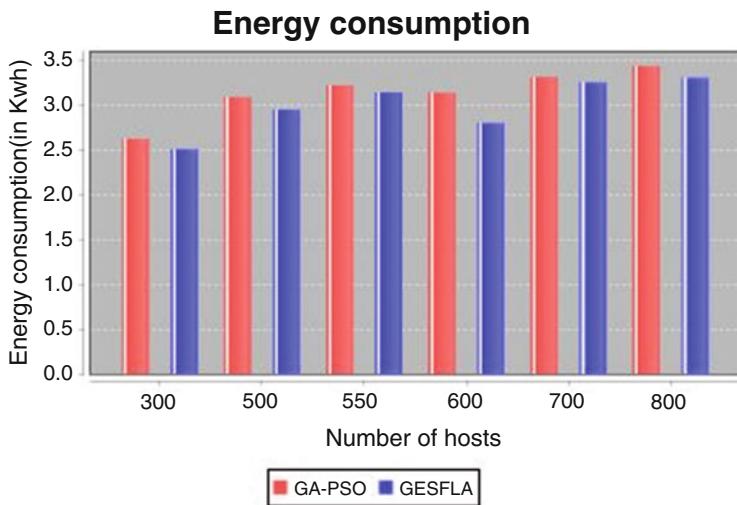


Fig. 4.7 Power consumption (PlanetLab)

VMs using the migration policy discussed in Sect. 3.6.1. Table 4.4 depicts the improvement of energy consumption of the hosts.

Since the fewer number of PMs and VMs are used, the transfer time of the proposed GESFL algorithm significantly gives better results than the GAPSO (PMs) configuration. As more PMs are switched off, data center's expense reduces. The hybrid model suggested by GESFLA has increased chromosomal fitness. The children obtained by the crossover and mutation operations are of better quality

Table 4.2 Improvement in transfer time

Number of VMs	Minimizing the transfer time Google datasets		Minimizing the transfer time Planetlab datasets		Improvement (%) in transfer time of GESFLA with GAPSO	
	GAPSO (Existing)	GESFLA (Proposed)	Improvement (%) in transfer time of GESFLA with GAPSO	GAPSO	GESFLA	
400	56.53	37.79	33	25.42	24.58	3
500	82.60	63.12	23	42.28	32.56	22
700	96.28	83.203	13	54.06	47.71	12
1000	95.65	78.668	17	74.19	62.19	16
1200	77.80	60.46	22	80.88	70.22	14

Table 4.3 Improvement in CPU utilization of host

No of Hosts	Maximum CPU utilization of the host (Google workload)		Maximum utilization of the host (PlanetLab workload)		Improvement (%) in the host usage of GESFLA with GAPSO
	GAPSO (Existing)	GESFLA (Proposed)	GAPSO	GESFLA	
300	0.345	0.3629	5	0.401	0.478
500	0.41	0.328	20	0.252	0.301
700	0.1865	0.235	20	0.213	0.264
800	0.191	0.223	14	0.215	0.246

Table 4.4 Improvement in energy consumption

No of Host	Minimizing energy consumption (Google workload)		Minimizing energy consumption (PlanetLab workload)		Improvement (%) of GESFLA with GAPSO		Improvement (%) of GESFLA with GAPSPO	
	GAPSO (Existing)	GESFLA (Proposed)	Improvement (%) of GESFLA with GAPSO	GAPSO	GESFLA	GESFLA	GAPSPO	
300	2.64	2.45	3	2.62	2.50	4		
500	2.26	2.16	4	3.09	2.95	5		
700	3.05	2.64	13	3.31	3.25	12		
800	2.82	2.65	6	3.43	3.30	3		

and thus have achieved the optimal solution for allocating VMs. This helps in maximizing the usage of the resource by reducing execution time and transfer time.

The efficiency of GESFLA is increased because mutation and crossover operators of genetic algorithm helps to generate quality of good children. The children produced are grouped into memplexes using SFLA. The memplexes does the shuffling of children produced and the local search of availability of resources. This takes very less convergence time, which reduces the transfer time of VMs from one host to another.

It is necessary to consider the power consumption because it is a crucial factor for data center's [30] and also for the service providers. Minimizing power consumption [31] helps to reduce the cost of data center. We are considering under loaded hosts and enabling the host to switch to the balanced state by shutting down the under loaded host. The shuffling process of SFLA helps VMs to find the underloaded hosts and it is beneficial in maintaining the load among the hosts.

4.4.2 Analysis of Complexity of Time

The proposed GESFLA processing time is dependent upon GA and SFLA algorithms.

Consider,

“k” = number of tasks

“x” = number of PMs

“y” = number of VMs and

“s” = size of the generation' and

“C” = crossover rate of each generation

GA's time complexity is focused on the distinct operations of generation, crossover, and mutation. SFLA focuses on Frog generation, memplex number, an update of the position, estimation of memplex fitness, Frog's location, return the missing VMs, and delete duplicate VMs for specified intervals. GAPSO-based migration of VM needs $O(n \log n)$ processing time. $GESFLA = O(GA) + O(SFLA)$. As a result, the time complexity of $GESFLA = O(\log n)$ is equal to $O(n: k, s, m) + n \log n$ computation.

4.5 Conclusion and Future Work

We have discussed different phases of our proposed GESFLA algorithm. The algorithm is tested for Google cluster trace datasets as well as real-time workload traces from the PlanetLab. The suggested approach used threshold values for VM migration and load on the server is reduced by frequent monitoring of all the VMs. The experimental results indicate that GESFLA framework efficiency is better than the GAPSO algorithm. The proposed algorithm increases the performance of data

center by minimizing the transfer time by 17%, maximizing resource utilization around 14–16% and energy consumption is reduced in comparison with the existing algorithm by 6%. The limitation of our proposed algorithm is that we have listed only four combinations of VMs and PMs which can be enhanced with more combinations. In the future, the algorithm can be applied for actual data centers in real-time.

References

1. H. Priyanka, Analytics of application resource utilization within the virtual machine. *Int. J. Sci. Res.* **5**(4), 1690–1693 (2016)
2. H. Hajj, W. El-Hajj, M. Dabbagh, T.R. Arabi, An algorithm centric energy-aware design methodology. *IEEE Trans. Very Large Scale Integr. Syst.* **22**(11), 2431–2435 (2014)
3. F. Ramezani, J. Lu, F.K. Hussain, Task-based system load balancing in cloud computing using particle swarm optimization. *Int. J. Parallel Prog.*, Springer **42**, 739–754 (2014)
4. L. Guo, S. Zhao, S. Shen, C. Jiang, Task scheduling optimization in cloud computing based on a heuristic algorithm. *J. Networks* **7**(3), 547–553 (2012)
5. H. Priyanka, M. Cherian, The challenges in virtual machine live migration and resource management. *Int. J. Eng. Res. Technol.* **8**(11), 5 (2020)
6. N.K. Sharma, G. Ram Mohana Reddy, Multi-objective energy efficient virtual machines allocation at the cloud data center. *IEEE Trans. Serv. Comput.* **12**(1), 158–171 (2019)
7. Y. Ge, G. Wei, GA-based task scheduler for the cloud computing systems, in *Proceedings of the International Conference on Web Information Systems and Mining (WISM '10)*, vol. 2, (IEEE, 2010), pp. 181–186
8. G. Giftson Samuel, C. Christober Asi Rajan, Hybrid: Particle Swarm Optimization–Genetic Algorithm and Particle Swarm Optimization–Shuffled Frog Leaping Algorithm for long-term generator maintenance scheduling. *Int. J. Elect. Power Energy Syst.*, Elsevier **65**, 432–442 (2015)
9. E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: A survey, in *Approximation Algorithms for NP-Hard Problems*, ed. by D. S. Hochbaum, (PWS Publishing Co, Boston, 1997), pp. 46–93
10. Y. Ajiro, A. Tanaka, Improving packing algorithms for server consolidation, in *Proceedings of the 33rd International Computer Measurement Group Conference, December 2–7, 2007, San Diego, CA, USA*, (DBLP, 2007), pp. 399–406
11. A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Trans. Parallel Distrib. Syst.* **24**(7), 1366–1379 (2013)
12. K. Dasgupta, B. Mandal, P. Dutta, J.K. Mandal, S. Dam, A genetic algorithm (GA) based load balancing strategy for cloud computing. *Procedia Technol.* **10**, 340–347 (2013)
13. A.-P. Xiong, C.-X. Xu, Energy efficient multiresource allocation of virtual machine based on PSO in cloud data center. *Math. Probl. Eng.* **2014**, 1–8 (2014)
14. W.-T. Wen, C.-D. Wang, D.-S. Wu, Y.-Y. Xie, An ACO-based scheduling strategy on load balancing in cloud computing environment, in *Ninth IEEE International Conference on Frontier of Computer Science and Technology*, vol. 6, (IEEE, 2015), pp. 364–369
15. S. Wang, Z. Liu, Z. Zheng, Q. Sun, F. Yang, Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers, in *Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, (IEEE, 2013), pp. 102–109
16. H. Xu, B. Yang, W. Qi, E. Ahene, A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery. *KSII Trans. Internet Inf. Syst.* **10**(3), 976–994 (2016)

17. S. Chitra, B. Madhusudhanan, G.R. Sakthidharan, P. Saravanan, Local minima jump PSO for workflow scheduling in cloud computing environments, in *Advances in Computer Science and its Applications. Lecture Notes in Electrical Engineering*, vol. 279, (Springer, 2014), pp. 1225–1234
18. C. Reiss, J. Wilkes, J.L. Hellerstein, Google cluster-usage traces: Format+ schema. Technical report at <https://github.com/google/clusterdata>, Google, Mountain View, CA, USA, Revised 2014-11-17 for version 2.2, Nov. 2011
19. Y. Mao, X. Chen, X. Li, Max-Min task scheduling algorithm for load balance in cloud computing, in *Proceedings of International Conference on Computer Science and Information Technology. Advances in Intelligent Systems and Computing*, ed. by S. Patnaik, X. Li, vol. 255, (Springer, New Delhi, 2012)
20. B. Santhosh, D.H. Manjaiah, A hybrid AvgTask-Min and Max-Min algorithm for scheduling tasks in cloud computing, in *International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil*, (IEEE, 2015), pp. 325–328. <https://doi.org/10.1109/ICCICCT.2015.7475298>
21. J. Wilkes, More Google cluster data. Google research blog (Nov. 2011). Posted at <http://googleresearch.blogspot.com/2011/11/more-googlecluster-data.html>
22. J. Wilkes, Google cluster-usage traces v3. Technical report at <https://github.com/google/cluster-data>, Google, Mountain View, CA, USA, Nov. 2019
23. H. Priyanka, M. Cherian, Efficient utilization of resources of virtual machines through monitoring the cloud data center, in *International Conference on Communication, Computing and Electronics Systems. Lecture Notes in Electrical Engineering*, ed. by V. Bindhu, J. Chen, J. Tavares, vol. 637, (Springer, Singapore, 2020), pp. 645–653
24. H. Priyanka, M. Cherian, Novel approach to virtual machine migration in cloud computing environment – A survey. *Int. J. Sci. Rep.* **7**(1), 81–84 (2018)
25. L. Weining, F. Ta, Live migration of virtual machine based on recovering system and CPU scheduling, in *6th IEEE joint International Information Technology and Artificial Intelligence Conference, Piscataway, NJ, USA*, (IEEE, 2009), pp. 303–305
26. S.B. Melhem, A. Agarwal, N. Goel, M. Zaman, Markov prediction model for host load detection and VM placement in live migration. *IEEE Access* **6**, 7190–7205 (2017)
27. A. Kishor, R. Niyogi, Multi-objective load balancing in distributed computing environment: An evolutionary computing approach, in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, (Association for Computing Machinery, Brno, 2020), pp. 170–172
28. A. Beloglazov, R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers, in *Concurrency and Computation: Practice and Experience*, vol. 24, No. 13, (Wiley Press, New York, 2011). <https://doi.org/10.1002/cpe.1867>
29. R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. de Rose, R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **41**(1), 23–50 (2011)
30. J. Singh, J. Chen, Optimizing energy consumption for cloud computing: A cluster and migration based approach (CMBA), in *Proceedings of the 2019 3rd International Conference on Computer Science and Artificial Intelligence*, vol. 22, No. 11, (Association for Computing Machinery (ACM), 2019), pp. 28–32
31. S. Rahman, A. Gupta, M. Tornatore, B. Mukherjee, Dynamic workload migration over backbone network to minimize data center electricity cost. *IEEE Trans. Green Commun. Netw.* **2**, 570–579 (2018)

Chapter 5

Dynamic Resource-Efficient Scheduling in Data Stream Management Systems Deployed on Computing Clouds



Xunyun Liu, Yufei Lin, and Rajkumar Buyya

Contents

5.1	Introduction	133
5.2	Background	136
5.3	Dynamic Resource-Efficient Scheduling	138
5.3.1	Problem Formulation	139
5.3.2	Heuristic-Based Scheduling Algorithm	141
5.3.3	Complexity Analysis	144
5.4	Implementation of D-Storm Prototype	144
5.5	Performance Evaluation	147
5.5.1	Experiment Setup	147
5.5.2	Evaluation of Applicability	151
5.5.3	Evaluation of Cost Efficiency	156
5.5.4	Evaluation of Scheduling Overhead	157
5.6	Related Work	158
5.7	Conclusions and Future Work	160
	References	161

5.1 Introduction

Big data processing has gained increasing popularity by proposing new paradigms and infrastructure for handling big-volume and high-velocity data, which greatly extends the capacity of traditional data management systems. High velocity, as one of the core characteristics of big data, requires processing inputs in real-time to exploit their volatile value. For example, the statistics of share market are collected and analysed in real-time to avoid investment losses, while the ongoing user posts

X. Liu · Y. Lin (✉)
Academy of Military Science, Beijing, P.R. China
e-mail: linyufei@nudt.edu.cn

R. Buyya
Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Parkville, VIC, Australia
e-mail: rbuyya@unimelb.edu.au

on a social media website are aggregated continuously to suggest trending topics on hype. As the Internet started to connect everything, the generation of high-velocity data is also contributed by a variety of IoT (Internet-of-Things) driven applications such as smart cities [1], RFID (Radio Frequency Identification) systems [2] and sensor networks [3]. To cater for the real-time processing requirement, stream processing emerges as a new in-memory paradigm that handles continuous, unbounded inputs on a record-by-record basis. Such once-at-a-time processing model performs independent computations on a smallish window of data upon arrival, delivering incremental results with merely sub-second processing latencies.

Despite the diversity of real-time use cases, most of the streaming applications in existence are built on top of a Data Stream Management System (DSMS) to reap the benefits of better programmability and manageability. Apache Storm, for example, is a state-of-the-art distributed DSMS that provides a unified data stream management model for semantic consistency and supports the use of an imperative programming language to express user logic. It also offers user-transparent fault tolerance, horizontal scalability and state management by providing the abstraction of streaming primitives and simplifying the coordination of distributed resources at the middleware level.

Scheduling of streaming applications is one of the many tasks that should be transparently handled by the DSMSs [4]. As the deployment platform of DSMS shifts from a homogeneous on-premise cluster to an elastic cloud resource pool, new challenges have arisen in the scheduling process to enable fast processing of high-velocity data with minimum resource consumption. First, the infrastructural layer can be composed of heterogeneous instance types ranging from Shared Core to High-CPU and High-memory machines,¹ each equipped with different computing power to suit the diverse resource needs of different streaming operators. This could also be motivated by economic considerations when users have made a long-term commitment to a few specific instance configurations, which makes it more economically viable to build a heterogeneous cluster with reserved instances at hand. In these cases, the assumption of homogeneous resources becomes invalid, and the node differences must be captured in the scheduling process to avoid resource contention. Additionally, the distance of task communication needs to be optimised at runtime to improve application performance. In stream processing systems, intra-node communication (i.e. information exchange between streaming tasks within a single node) is much faster than inter-node communication as the former does not involve cumbersome processes of data (de)serialisation, (un)marshalling and network transmission [4, 5]. Therefore, it is up to the dynamic scheduling process to convert as much inter-node communication as possible into intra-node communication. Last but not least, the dynamics of real-time applications lead to unpredictable stream data generation, requiring the processing system to be able to manage elastic resources according to the current workload and improve cost efficiency at runtime.

¹ <https://cloud.google.com/compute/docs/machine-types>.

Therefore, to maximise application performance and reduce the resource footprints, it is of crucial importance for the DSMS to schedule streaming applications as compact as possible, in a manner that fewer computing and network resources are consumed to achieve the same performance target. This motivates the needs of resource-aware scheduling, which matches the resource demands of streaming tasks to the capacity of distributed nodes. However, the default schedulers adopted in the state-of-the-art DSMSs, including Storm, are resource agnostic. Without capturing the differences of task resource consumptions, they follow a simple round-robin process to scatter the application tasks over the cluster, thus inevitably leading to over/under-utilisation and causing execution inefficiency. A few dynamic schedulers have also been proposed recently to reduce the network traffics and improve the maximum application throughput at runtime [6–12]. However, they all share the load-balancing principle that distributes the workload as evenly as possible across participating nodes, thus ignoring the need of resource consolidation when the input is small. Also, without application isolation, the scheduling result may suffer from severe performance degradation when multiple applications are submitted to the same cluster and end up competing for the computation and network resources on every single node.

To fill in this gap, Peng et al. [13] proposed a resource-aware scheduler that schedules streaming applications based on the resource profiles submitted by users at compile time. But the problem is only partially tackled for the following reasons:

1. The resource consumption of each task is statically configured within the application, which suggests that it is agnostic to the actual application workload and will remain unchanged during the whole lifecycle of the streaming application. However, the resource consumption of a streaming task is known to be correlated to the input workload, and the latter may be subject to unforeseeable fluctuations due to the real-time streaming nature.
2. The scheduler only executes once during the initial application deployment, making it impossible to adapt the scheduling plan to runtime changes. Its implementation is static, which tackles the scheduling problem as a one-time item packing process, so it only works on unassigned tasks brought by new application submissions or worker failures.

In this chapter, we propose a dynamic resource-efficient scheduling algorithm to tackle the problem as a bin-packing variant. We also implement a prototype named D-Storm to validate the efficacy and efficiency of the proposed algorithm. D-Storm does not require users to statically specify the resource needs of streaming applications; instead, it models the resource consumption of each task at runtime by monitoring the volume of incoming workload. Secondly, D-Storm is a dynamic scheduler that repeats its bin-packing policy with a customisable scheduling interval, which means that it can free under-utilised nodes whenever possible to save resource costs.

This chapter is a significant extension of our previous work [14]. The main **contributions** reported in this chapter are summarised as follows:

- We formulate the scheduling problem as a bin-packing variant using a fine-grained resource model to describe requirements and availability. To the best of our knowledge, this work is the first of its kind to dynamically schedule streaming applications based on bin-packing formulations.
- We design a greedy algorithm to solve the bin-packing problem, which generalises the classical *First Fit Decreasing* (FFD) heuristic to allocate multi-dimensional resources. The algorithm is capable of reducing the amount of inter-node communication as well as minimising the resource footprints used by the streaming applications.
- We implement the prototype on Storm and conduct extensive experiments in a heterogeneous cloud environment. The evaluation involving realistic applications such as Twitter Sentiment Analysis demonstrates the superiority of our approach compared to the existing static resource-aware scheduler and the default scheduler.

It is worth noting that though our D-Storm prototype has been implemented as an extended framework on Storm, it is not bundled with this specific platform. The fact that D-Storm is loosely coupled with the existing Storm modules and the design of external configuration make it viable to be generalised to other operator-based data stream management systems as well.

The remainder of this chapter is organised as follows: we introduce Apache Storm as a background system in Sect. 5.2 to explain the scheduling problem. Then, we formulate the scheduling problem, present the heuristic-based algorithm and provide an overview of the proposed framework in Sects. 5.3 and 5.4. The performance evaluation is presented in Sect. 5.5, followed by the related work and conclusions in Sects. 5.6 and 5.7, respectively.

5.2 Background

This section introduces Apache Storm, explains the concept of scheduling and uses Storm as an example to illustrate the scheduling process in the state-of-the-art DSMSs. Apache Storm is a real-time stream computation framework built for processing high-velocity data, which has attracted attention from both academia and industry over the recent years. Though its core implementation is written in Clojure, Storm does provide programming supports for multiple high-level languages such as Java and Python through the use of Thrift interfaces. Being fast, horizontally scalable, fault tolerant and easy to operate, Storm is considered by many as the counterpart of Hadoop in the real-time computation field.

Storm also resembles Hadoop from the structural point of view—there is a *Nimbus node* acting as the master to distribute jobs across the cluster and manage the subsequent computations; while the rests are the *worker nodes* with the *worker processes* running on them to carry out the streaming logic in JVMs. Each worker node has a *Supervisor* daemon to start/stop worker processes as per Nimbus's assignment. Zookeeper, a distributed hierarchical key-value store, is used

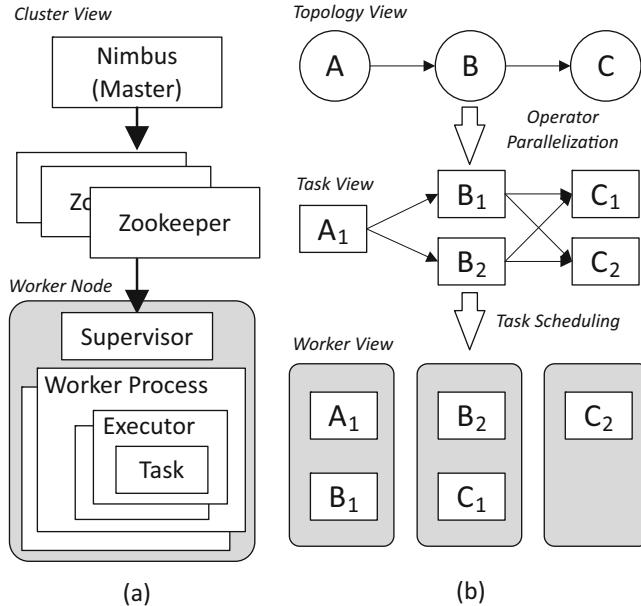


Fig. 5.1 The structural view and logical view of a Storm cluster, in which the process of task scheduling is illustrated with a three-operator application

to coordinate the Storm cluster by serving as a communication channel between the Nimbus and Supervisors. We refer to Fig. 5.1a for the structural view of a Storm cluster.

From the programming perspective, Storm has its unique data model and terminology. A *tuple* is an ordered list of named elements (each element is a key-value pair referred to as a *field*) and a *stream* is an unbounded sequence of tuples. The streaming logic of a particular application is represented by its *topology*, which is a Directed Acyclic Graph (DAG) of *operators* standing on continuous input streams. There are two types of operators: *spouts* act as data sources by pulling streams from the outside world for computation, while the others are *bolts* that encapsulate certain user-defined processing logic such as functions, filtering and aggregations.

When it comes to execution, an operator is parallelised into one or more *tasks* to split the input streams and concurrently perform the computation. Each task applies the same streaming logic to its portion of inputs which are determined by the associated *grouping policy*. Figure 5.1b illustrates this process as operator parallelisation. In order to make efficient use of the underlying distributed resources, Storm distributes tasks over different worker nodes in thread containers named *executors*. Executors are the minimal schedulable entities of Storm that are spawned by the worker process to run one or more tasks of the same bolt/spout sequentially, and Storm has a default setting to run one executor per task. The assignment of

all executors of the topology to the worker processes available in the cluster is called *scheduling*. Without loss of generality, in this work, we assume each executor contains a single task so that executor scheduling can be interpreted as a process of task scheduling.

Since version 0.9.2, Storm implements inter-node communications with Netty² to enable low-latency network transmission with asynchronous, event-driven I/O operations. However, the data to be transferred still needs to be serialised, then hit the transfer buffer, the socket interface and the network card at both sides of communication for delivery. By contrast, intra-node communication does not involve any network transfer and is conveyed by the message queues backed by LMAX Disruptor,³ which significantly improves the performance as tuples are deposited directly from the Executor send buffer to the Executor receive buffer.

As compared to newer stream processing systems such as Apache Flink⁴ and Apache Beam,⁵ Storm does not support lossless migration of operator states during the scheduling process, which means users need to implement that functionality at the application layer to handle the relocation of operator internal states. The default scheduler in Storm is implemented as a part of Nimbus function that endeavours to distribute the same number of tasks over the participating worker nodes, where a round-robin process is adopted for this purpose. However, as pointed out in Sect. 5.1, such a simple scheduling policy may lead to over/under resource utilisation.

On the other hand, the scheduler proposed by Peng et al. [13] is the most relevant to our work and is widely adopted in the Storm community because it is resource-aware, bin-packing-related and readily available within the standard Storm release. However, it can only partially tackle the problem of over/under resource utilisation due to the limitation of being static in nature and requiring users to input the correct resource configuration prior to execution. In Sect. 5.5, we conduct a thorough comparison of our approach and Peng et al.'s work with performance evaluation in a real environment.

5.3 Dynamic Resource-Efficient Scheduling

The dynamic resource-efficient scheduling exhibits the following characteristics: (1) each task has a set of resource requirements that are constantly changing along with the amount of inputs being processed; (2) each machine (worker node) has a set of available resources for accommodating tasks that are assigned to it; and (3) the scheduling algorithm is executed on-demand to take into account any runtime changes in resource requirements and availability.

² <https://netty.io/>.

³ <https://lmax-exchange.github.io/disruptor/>.

⁴ <https://flink.apache.org/>.

⁵ <https://beam.apache.org/>.

5.3.1 Problem Formulation

For each round of scheduling, the essence of the problem is to find a mapping of tasks to worker nodes such that the communicating tasks are packed as compact as possible. In addition, the resource constraints need to be met—the resource requirements of the allocated tasks should not exceed the resource availability in each worker node. Since the compact assignment of tasks also leads to reducing the number of used machines, we model the scheduling problem as a variant of the bin-packing problem and formulate it using the symbols illustrated in Table 5.1.

In this work, the resource consumptions and availability are examined in two dimensions—CPU and memory. Though memory resources can be intuitively measured in terms of megabytes, the quantification of CPU resources is usually vague and imprecise due to the diversity of CPU architectures and implementations. Therefore, following the convention in literature [13], we specify the amount of CPU resources with a point-based system, where 100 points are given to represent the full capacity of a Standard Compute Unit (SCU). The concept of SCU is similar to the EC2 Compute Unit (ECU) introduced by Amazon Web Services (AWS). It is then the responsibility of the IaaS provider to define the computing power of an SCU, so that developers can compare the CPU capacity of different instance types with consistency and predictability regardless of the hardware heterogeneity presented in the infrastructure. As a relative measure, the definition of an SCU can be updated through benchmarks and tests after introducing new hardware to the data centre infrastructure.

Table 5.1 Symbols used for dynamic resource-efficient scheduling

Symbol	Description
n	The number of tasks to be assigned
τ_i	Task i , $i \in \{1, \dots, n\}$
m	The number of available worker nodes in the cluster
v_i	Worker node i , $i \in \{1, \dots, m\}$
$W_c^{v_i}$	CPU capacity of v_i , measured in a point-based system, $i \in \{1, \dots, m\}$
$W_m^{v_i}$	Memory capacity of v_i , measured in Mega Bytes (MB), $i \in \{1, \dots, m\}$
$\omega_c^{\tau_i}$	Total CPU requirement of τ_i in points, $i \in \{1, \dots, n\}$
$\omega_m^{\tau_i}$	Total memory requirement of τ_i in Mega Bytes (MB), $i \in \{1, \dots, n\}$
$\rho_c^{\tau_i}$	Unit CPU requirement for τ_i to process a single tuple, $i \in \{1, \dots, n\}$
$\rho_m^{\tau_i}$	Unit memory requirement for τ_i to process a single tuple, $i \in \{1, \dots, n\}$
ξ_{τ_i, τ_j}	The size of data stream transmitting from τ_i to τ_j , $i, j \in \{1, \dots, n\}, i \neq j$
Θ_{τ_i}	The set of upstream tasks for τ_i , $i \in \{1, \dots, n\}$
Φ_{τ_i}	The set of downstream tasks for τ_i , $i \in \{1, \dots, n\}$
χ	The volume of inter-node traffic within the cluster
V_{used}	The set of used worker nodes in the cluster
m_{used}	The number of used worker nodes in the cluster

In this chapter, we assume that the IaaS cloud provider has followed the example of Amazon to create a vCPU as a hyperthread of an Intel Xeon core,⁶ where 1 SCU is defined as the CPU capacity of a vCPU. Therefore, every single core in the provisioned virtual machine is allocated with 100 points. A multi-core instance can get a capacity of $num_of_cores * 100$ points, and a task that accounts for $p\%$ CPU usages reported by the monitoring system has a resource demand of p points.

As reported in [15], task τ_i 's CPU and memory resource requirements can be linearly modelled with regard to the size of the current inputs, which are illustrated in Eq. (5.1).

$$\begin{aligned}\omega_c^{\tau_i} &= \left(\sum_{\tau_j \in \Theta_{\tau_i}} \xi_{\tau_j, \tau_i} \right) * \rho_c^{\tau_i} \\ \omega_m^{\tau_i} &= \left(\sum_{\tau_j \in \Theta_{\tau_i}} \xi_{\tau_j, \tau_i} \right) * \rho_m^{\tau_i}.\end{aligned}\quad (5.1)$$

Note that i and j in Eq. (5.1) are just two generic subscripts that represent certain values within a range defined in Table 5.1. Therefore, ξ_{τ_j, τ_i} has a similar meaning of ξ_{τ_i, τ_j} that denotes the size of data stream transmitting from the former task to the latter.

Having modelled the resource consumption at runtime, each task is considered as an item of multidimensional volumes that needs to be allocated to a particular machine during the scheduling process. Given a set of m machines (bins) with CPU capacity $W_c^{v_i}$ and memory capacity $W_m^{v_i}$ ($i \in \{1, \dots, m\}$), and a list of n tasks (items) $\tau_1, \tau_2, \dots, \tau_n$ with their CPU demands and memory demands denoted as $\omega_c^{\tau_i}, \omega_m^{\tau_i}$ ($i \in \{1, 2, \dots, n\}$), the problem is formulated as follows:

$$\begin{aligned}&\text{minimise} \quad \kappa(\boldsymbol{\xi}, \mathbf{x}) = \sum_{i, j \in \{1, \dots, n\}} \xi_{\tau_i, \tau_j} \left(1 - \sum_{k \in \{1, \dots, m\}} x_{i, k} * x_{j, k} \right) \\ &\text{subject to} \quad \sum_{k=1}^m x_{i, k} = 1, \quad i = 1, \dots, n, \\ & \quad \sum_{i=1}^n \omega_c^{\tau_i} x_{i, k} \leq W_c^{v_k} \quad k = 1, \dots, m, \\ & \quad \sum_{i=1}^n \omega_m^{\tau_i} x_{i, k} \leq W_m^{v_k} \quad k = 1, \dots, m,\end{aligned}\quad (5.2)$$

where \mathbf{x} is the control variable that stores the task placement in a binary form: $x_{i, k} = 1$ if and only if task τ_i is assigned to machine v_k .

⁶ <https://aws.amazon.com/ec2/instance-types/>.

Through the formulation, we quantify the compactness of scheduling by counting the total amount of inter-node communication resulted from the assignment plan, with the optimisation target being reducing this number to its minimal.

Specifically, the expression $(1 - \sum_{k \in \{1, \dots, m\}} x_{i,k} * x_{j,k})$ is a toggle switch that yields either 0 or 1 depending on whether task τ_i and τ_j are assigned to the same node. If yes, the result of $(1 - \sum_{k \in \{1, \dots, m\}} x_{i,k} * x_{j,k})$ becomes 0 which eliminates the size of the data stream ξ_{τ_i, τ_j} to make sure that only inter-node communication is counted in our objective function.

There are three constraints formulated in Eq. (5.2): (1) each task shall be assigned to one and only one node during the scheduling process; (2) the CPU resource availability of each node must not be exceeded by the accrued CPU requirements of the allocated tasks; and (3) the memory availability of each node must not be exceeded by the accrued memory requirements of the allocated tasks.

Also, Eq. (5.3) shows that x can be used to reason the number of used worker nodes as the result of scheduling:

$$\begin{aligned} V_{\text{used}} &= \left\{ v_j \mid \sum_{i \in \{1, \dots, n\}} x_{i,j} > 0, j \in \{1, \dots, m\} \right\} \\ m_{\text{used}} &= |V_{\text{used}}| \end{aligned} \quad (5.3)$$

5.3.2 Heuristic-Based Scheduling Algorithm

The classical bin-packing problem has proved to be NP-Hard [16], and so does the scheduling of streaming applications [13]. There could be a massive amount of tasks involved in each single assignment, so it is computationally infeasible to find the optimal solution in polynomial time. Besides, streaming applications are known for their strict Quality of Service (QoS) constraints on processing time [17], so the efficiency of scheduling is even more important than the result optimality to prevent the violation of the real-time requirement. Therefore, we opt for greedy heuristics rather than exact algorithms such as bin completion [18] and branch-and-price [19], which have exponential time complexity.

The proposed algorithm is a generalisation of the classical *First Fit Decreasing* (FFD) heuristic. FFD is essentially a greedy algorithm that sorts the items in decreasing order (normally by their size) and then sequentially allocates them into the first bin with sufficient remaining space. However, in order to apply FFD in our multidimensional bin-packing problem, the standard bin-packing procedure has to be generalised in three aspects as shown in Algorithm 1.

Firstly, all the available machines are arranged in descending order by their resource availability so that the more powerful ones get utilised first for task placement. This step is to ensure that the FFD heuristic has a better chance to convey more task communications within the same machine, thus reducing the cumbersome

Algorithm 1 The multidimensional FFD heuristic scheduling algorithm

Input: A task set $\tau = \{\tau_1, \tau_2, \dots, \tau_n\}$ to be assigned
Output: A machine set $v = \{v_1, v_2, \dots, v_{m_{\text{used}}}\}$ with each machine hosting a disjoint subset of τ , where m_{used} is the number of used machines

```

1 Sort available nodes in descending order by their resource availability as defined in Eq. (5.4)
2  $m_{\text{used}} \leftarrow 0$ 
3 while there are tasks remaining in  $\tau$  to be placed do
4   Start a new machine  $v_m$  from the sorted list;
5   if there are no available nodes then
6     | return Failure
7   end
8   Increase  $m_{\text{used}}$  by 1
9   while there are tasks that fit into machine  $v_m$  do
10    | foreach  $\tau \in \tau$  do
11      |   | Calculate  $\varphi(\tau_i, v_m)$  according to Eq. (5.5)
12    | end
13    | Sort all viable tasks based on their priority
14    | Place the task with the highest  $\varphi(\tau_i, v_m)$  into machine  $v_m$ 
15    | Remove the assigned task from  $\tau$ 
16    | Update the remaining capacity of machine  $v_m$ 
17  end
18 end
19 return  $v$ 
```

serialisation and de-serialisation procedures that would have been necessary for network transmissions. Since the considered machine characteristics—CPU and memory are measured in different metrics, we define a resource availability function that holistically combines these two dimensions and returns a scalar for each node, as shown in Eq. (5.4).

$$\varphi(v_i) = \min \left\{ \frac{n W_c^{v_i}}{\sum_{j \in \{1, \dots, n\}} \omega_c^{\tau_j}}, \frac{n W_m^{v_i}}{\sum_{j \in \{1, \dots, n\}} \omega_m^{\tau_j}} \right\}. \quad (5.4)$$

Secondly, the evaluation of the task priority function is dynamic and runtime-aware, considering not only the task communication pattern but also the node to which it attempts to assign. We denote the attempted node as v_m , then the task priority function $\varphi(\tau_i, v_m)$ can be formulated as a fraction—the greater the resulting value, the higher priority τ_i will have to be assigned into v_m .

The numerator of this fraction quantifies the increase of intra-node communications as the benefit of assigning τ_i to v_m . It is a weighted sum of two terms, which are namely: (1) the amount of newly introduced intra-node communication if τ_i is assigned to v_m , and (2) the amount of potential intra-node communication that τ_i can bring to v_m in the subsequent task assignments. It is worth noting that we stop counting the second term in the numerator when the node v_m is about to be filled up, so tasks capable of bringing more potential intra-node communications will be left for other nodes with more available resources.

On the other hand, the denominator of this fraction depicts the resource costs that v_m spends on accommodating τ_i . Inspired by the Dominant Resource Fairness (DRF) approach used in Apache Mesos [20], we evaluate the resource costs of τ_i in terms of its usages of critical resource, where “critical resource” is defined as the most scarce and demanding resource type (either being CPU or memory) at the current state. Therefore, tasks occupying less critical resources will be preferred in the priority calculation, and the resulting resource usages are more likely to be balanced across different resource types.

After introducing the rationales behind our priority design, the mathematical formulation of $\varrho(\tau_i, v_m)$ is given as follows:

$$\begin{aligned}
 \varrho_1(\tau_i, v_m) &= \sum_{j \in \{1, \dots, n\}} x_{j, v_m} (\xi_{\tau_i, \tau_j} + \xi_{\tau_j, \tau_i}) \\
 \varrho_2(\tau_i, v_m) &= \sum_{j \in \Phi_{\tau_i}} \left(1 - \sum_{k \in \{1, \dots, m\}} x_{j, k} \right) \xi_{\tau_i, \tau_j} \\
 &\quad + \sum_{j \in \Theta_{\tau_i}} \left(1 - \sum_{k \in \{1, \dots, m\}} x_{j, k} \right) \xi_{\tau_j, \tau_i} \\
 \mathfrak{R}_{v_m} &= \max \left\{ \frac{\sum_{j \in \{1, \dots, n\}} \omega_c^{\tau_j} x_{j, v_m}}{W_c^{v_m}}, \frac{\sum_{j \in \{1, \dots, n\}} \omega_m^{\tau_j} x_{j, v_m}}{W_m^{v_m}} \right\} \\
 \varrho_3(\tau_i, v_m) &= \mathfrak{R}_{v_m}^{x_{\tau_i, v_m}=1} - \mathfrak{R}_{v_m}^{x_{\tau_i, v_m}=0} \\
 \varrho(\tau_i, v_m) &= \begin{cases} \frac{\alpha \varrho_1(\tau_i, v_m) + \beta \varrho_2(\tau_i, v_m)}{\varrho_3(\tau_i, v_m)} & \mathfrak{R}_{v_m} \leq D_{\text{Threshold}} \\ \frac{\alpha \varrho_1(\tau_i, v_m)}{\varrho_3(\tau_i, v_m)} & \text{otherwise;} \end{cases} \tag{5.5}
 \end{aligned}$$

In Eq. (5.5), $\varrho_1(\tau_i, v_m)$ represents the sum of introduced intra-node communication if τ_i is assigned to v_m , while $\varrho_2(\tau_i, v_m)$ denotes the sum of communications that τ_i has with an unassigned peer, which effectively translates to the potential intra-node communication gains in the subsequent task assignments. After that, \mathfrak{R}_{v_m} represents the current usage of critical resources in v_m by the percentage measurement, and $\varrho_3(\tau_i, v_m)$ calculates the difference of \mathfrak{R}_{v_m} after and before the assignment to reflect the resource costs of v_m accommodating τ_i . In the end, $\varrho(\tau_i, v_m)$ is defined as a comprehensive fraction of the benefits and costs relating to this assignment. In Eq. (5.5), α and β are the weight parameters that determine the relative importance of the two independent terms in the numerator, and $D_{\text{Threshold}}$ is the threshold parameter that indicates when the node resources should be considered nearly depleted.

Designing $\varrho(\tau_i, v_m)$ in this way makes sure that the packing priority of the remaining tasks is dynamically updated after each assignment, and those tasks sharing a large volume of communication are prioritised to be packed into the same node. This is in contrast to the classical FFD heuristics that first sort the items in terms of their priority and then proceed to the packing process strictly following the pre-defined order.

Finally, our algorithm implements the FFD heuristic from a bin-centric perspective, which opens only one machine at a time to accept task assignment. The algorithm keeps filling the open node with new tasks until its remaining capacity is depleted, thus satisfying the resource constraints stated in Eq. (5.2).

5.3.3 Complexity Analysis

We analyse the workflow of Algorithm 1 to identify its complexity in the worst case. Line 1 of the algorithm requires at most quasilinear time $O(m\log(m))$ to finish, while the internal while loop from Line 9 to Line 17 will be repeated for at most n times to be either complete or failed. Diving into this loop, we find that the calculation of $\varrho(\tau_i, v_m)$ at Line 11 consumes linear time of n , and the sorting at Line 13 takes at most $O(n\log(n))$ time to complete. Therefore, the whole algorithm has the worst case complexity of $O(m\log(m) + n^2\log(n))$.

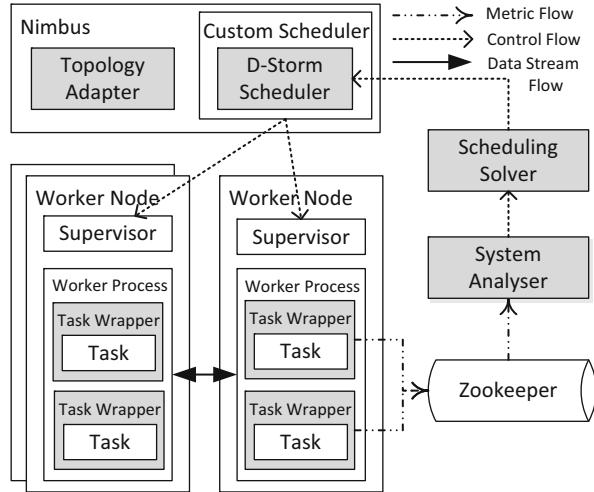
5.4 Implementation of D-Storm Prototype

A prototype called D-Storm has been implemented to demonstrate dynamic resource-efficient scheduling, which incorporates the following new features into the standard Storm framework:

- It tracks streaming tasks at runtime to obtain their resource usages and the volumes of inbound/outbound communications. This information is critical for making scheduling decisions that avoid resource contention and minimise inter-node communication.
- It endeavours to pack streaming tasks as compact as possible without causing resource contention, which effectively translates to the reduction of resource footprints while satisfying the performance requirements of streaming applications.
- It automatically reschedules the application whenever a performance issue is spotted or possible task consolidation is identified.

To implement these new features, D-Storm extends the standard Storm release with several loosely coupled modules, thus constituting a MAPE-K (Monitoring, Analysis, Planning, Execution and Knowledge) framework as shown in Fig. 5.2. This architectural concept was first introduced by IBM to design autonomic

Fig. 5.2 The extended D-Storm architecture on top of the standard Storm release, where the newly introduced modules are highlighted in grey



systems with self-capabilities, such as self-managing, self-healing [21] and self-adapting [22]. In this work, the proposed MAPE-K framework incorporates self-adaptivity and runtime-awareness into D-Storm, allowing it to tackle any performance degradation or mismatch between resource requirements and availability at runtime.

The MAPE-K loop in D-Storm is essentially a feedback control process that considers the current system metrics while making scheduling decisions. Based on the level at which the metrics of interest are collected, the monitoring system generally reports three categories of information—application metrics, task metrics and OS (Operating System) metrics.

The application metrics, such as the topology throughput and complete latency,⁷ are obtained through the built-in Storm RESTful API and used as a coarse-grained interpretation of the application performance. The volume of incoming workloads is also monitored outside the application in order to examine the system's sustainability under the current workload.

The task metrics, on the other hand, depict the resource usages of different tasks and their communication patterns within the DSMS. Acquiring this information requires some custom changes to the Storm core, so we introduce the *Task Wrapper* as a middle layer between the current task and executor abstractions. Each task wrapper encapsulates a single task following the decorator pattern, with monitoring logic transparently inserted into the task execution. Specifically, it obtains the CPU usages in the *execute* method by making use of the *ThreadMXBean* class, and it logs the communication traffics among tasks using a custom metric consumer which is registered in the topology building process.

⁷ Complete latency: the average time a tuple tree takes to be completely processed by the topology.

Apart from higher level metrics, Collectd,⁸ a lightweight monitoring daemon, is installed on each worker node to collect statistics on the operating system level. These include CPU utilisation, memory usage and network interface access on every worker node. It is worth noting that due to the dynamic nature of stream processing, the collected metrics on communication and resource utilisation are all subject to non-negligible instantaneous fluctuations. Therefore, the monitor modules average the metric readings over an observation window and periodically report the results to Zookeeper for persistence.

The analysing phase in the MAPE-K loop is carried out by the *System Analyser* module, which is implemented as a boundary checker on the collected metrics to determine whether they represent a normal system state. There are two possible abnormal states defined by the comparison of the application and OS metrics. (1) Unsatisfactory performance—the monitored application throughput is lower than the volume of incoming workloads, or the monitored complete latency breaches the maximum constraint articulated in the Quality of Service (QoS). (2) Consolidation required—the majority of worker nodes exhibit resource utilisations below the consolidation threshold, and the monitored topology throughput closely matches the volume of incoming workloads. Note that for the sake of system stability, we do not alter the scheduling plan only because the resulting resource utilisation is high. Instead, we define abnormal states as strong indicators that the current scheduling plan needs to be updated to adapt to the ongoing system changes.

The *Scheduling Solver* comes into play when it receives the signal from the system analyser reporting the abnormal system states, with all the collected metrics passed on to it for planning possible amendments. It updates the model inputs with the retrieved metrics and then conducts scheduling calculation using the algorithm elaborated in Sect. 5.3. The passive design of invocation makes sure that the scheduler solver does not execute more frequently than the pre-defined scheduling interval, and this value should be fine-tuned to strike a balance between the system stability and agility.

Once a new scheduling plan is made, the executor in the MAPE-K loop—*D-Storm Scheduler* takes the responsibility to put the new plan into effect. From a practical perspective, it is a jar file placed on the Nimbus node which implements the *IScheduler* interface to leverage the scheduling APIs provided by Storm. The result of assignment is then cross-validated with the application metrics retrieved from the RESTful API to confirm the success of re-scheduling.

The Knowledge component of the MAPE-K loop is an abstract module that represents the data and logic shared among the monitoring, analysing, planning and execution functions. For the ease of implementation, D-Storm incorporates the scheduling knowledge into the actual components shown in Fig. 5.2, which includes background information on topology structures and user requirements, as well as the intelligent scheduling algorithm based on which the self-adaptation activities take place.

⁸ <https://collectd.org/>.

In order to keep our D-Storm scheduling framework user-transparent to the application developers, we also supply a *Topology Adapter* module in the Storm core that masks the changes made for task profiling. When the topology is built for submission, the adapter automatically registers the metric consumer and encapsulates tasks in task wrappers with logic to probe resource usages and monitor communication volumes. In addition, developers can specify the scheduling parameters through this module, which proves to be an elegant way to satisfy the diverse needs of different streaming scenarios.

5.5 Performance Evaluation

In this section, we evaluate the D-Storm prototype using both synthetic and realistic streaming applications. The proposed heuristic-based scheduling algorithm is compared against the static resource-aware algorithm [13], as well as the round-robin algorithm used in the default Storm scheduler.

Specifically, the performance evaluation focuses on answering the following independent research questions:

- Whether D-Storm applies to a variety of streaming applications with diverse topology structures, communication patterns and resource consumption behaviours. Whether it successfully reduces the total amount of inter-node communication and improves the application performance in terms of latency. (Section 5.5.2)
- How much resource cost is incurred by D-Storm to handle various volumes of workload? (Section 5.5.3)
- How long does it take for D-Storm to schedule relatively large streaming applications? (Section 5.5.4)

5.5.1 Experiment Setup

Our experiment platform is set up on the Nectar Cloud,⁹ comprising 1 Nimbus node, 1 Zookeeper node, 1 Kestrel¹⁰ node and 12 worker nodes. The whole cluster is located in the availability zone of National Computational Infrastructure (NCI) to avoid cross data centre traffic, and there are various types of resources present to constitute a heterogeneous cluster. Specifically, the 12 worker nodes are evenly created from three different instance flavours, which are (1) m2.large (4 VCPUs, 12 GB memory and 110 GB disk space); (2) m2.medium (2 VCPUs, 6 GB memory

⁹ <https://nectar.org.au/research-cloud/>.

¹⁰ <https://github.com/twitter-archive/kestrel>.

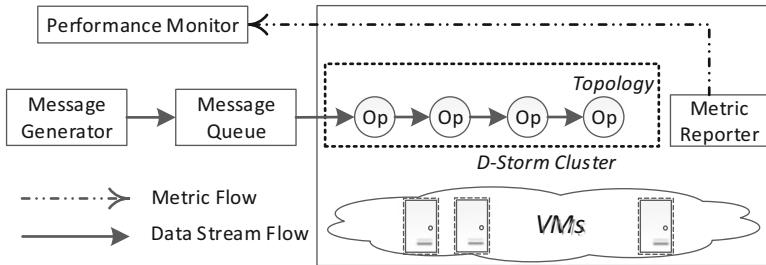


Fig. 5.3 The profiling environment used for controlling the input load. The solid lines denote the generated data stream flow, and the dashed lines represent the flow of performance metrics

and 30 GB disk space) and (3) m2.small (1 vCPUs, 4 GB memory and 30 GB disk space). On the other hand, the managing and coordination nodes are all spawned from the m2.medium flavour. Note that we denote the used instance types as “large”, “medium” and “small” hereafter for the convenience of presentation.

As for the software stack, all the participating nodes are configured with Ubuntu 16.04 and Oracle JDK 8, update 121. The version of Apache Storm on which we build our D-Storm extension is v1.0.2, and the comparable approaches—the static resource-aware scheduler and the default scheduler are directly extracted from this release.

In order to evaluate the performance of D-Storm under different sizes of workload, we have set up a profiling environment that allows us to adjust the size of the input stream with finer-grained control. Figure 5.3 illustrates the components of the profiling environment from a workflow perspective.

The *Message Generator* reads a local file of tweets and generates a profiling stream to the Kestrel node using its message push API. The workload file contains 159,620 tweets in JSON format that were collected from a time span of 24/03/2014 to 14/04/2014 utilising the Twitter’s Streaming API.¹¹ To avoid shortage of tweets during evaluation, the Message Generator loops over the workload file repeatedly after loading it into memory, and the size of the generated data stream is externally configurable. The *Message Queue* running on the Kestrel node implements a Kestrel queue to cache any message that has been received but not pulled by the streaming application. It serves as a message buffer between the message generator and the streaming application to avoid choking either side of them in the case of mismatch.

The *D-Storm cluster* runs the D-Storm prototype as well as the streaming application, where different scheduling algorithms are evaluated for efficiency. The *Metric Reporter* is responsible for probing the application performance, i.e. throughput and latency, and reporting the volume of inter-node communication in the forms of the number of tuples transferred and the volume of data streams conveyed in the network. Finally, the *Performance Monitor* is introduced to examine whether the application is sustainably processing the profiling input and if the

¹¹ <https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data>.

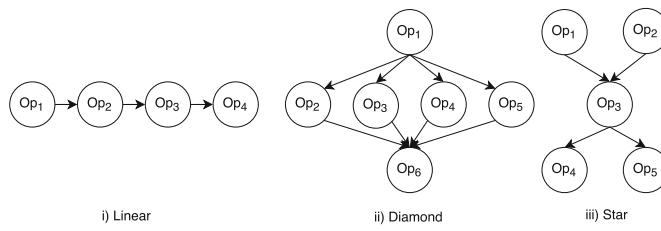
application performance has satisfied the pre-defined Quality of Service (QoS), such as processing 5000 tuples per second with the processing latency no higher than 500 ms.

5.5.1.1 Test Applications

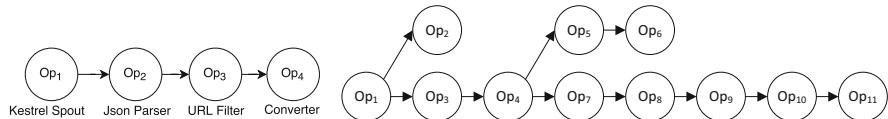
The evaluation includes five test applications—three synthetically made and two drawn from real-world streaming use cases. The acknowledgement mechanism is turned on for all these applications to achieve reliable message processing, which guarantees that each incoming tuple will be processed at least once and the complete latency is automatically tracked by the Storm framework. We also set the Storm configuration *MaxSpoutPending*¹² to 10,000, so that the resulting complete latency of different applications can be reasonably compared in the same context.

Synthetic Applications (Micro-Benchmark) the three synthetic applications are collectively referred to as micro-benchmark. They are designed to reflect various topological patterns as well as mimic different types of streaming applications, such as CPU bound, I/O bound and parallelism bound computations.

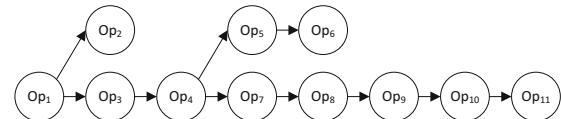
As shown in Fig. 5.4a, the micro-benchmark covers three common topological structures—Linear, Diamond and Star, corresponding to operators having (1) one-input-one-output, (2) multiple-outputs or multiple-inputs and (3) multiple-inputs-multiple-outputs, respectively. In addition, the synthetic operators used in the



(a) The topologies of the micro-benchmark synthetic application



(b) The topology of the URL-converter application



(c) The topology of the twitter sentiment analysis application

Fig. 5.4 The topologies of test applications. (a) Is synthetically designed while the rest two are drawn from realistic use cases

¹² The maximum number of unacknowledged tuples that are allowed to be pending on a spout task at any given time.

Table 5.2 The configurations of synthetic operators in the micro-benchmark

Symbol	Configuration description
C_s	The CPU load of each synthetic operator
S_s	The selectivity ^a of each synthetic operator
T_s	The number of tasks that each synthetic operator has, also referred to as operator parallelism

^a Selectivity is the number of tuples emitted per tuple consumed; e.g. selectivity = 2 means the operator emits 2 tuples for every 1 consumed

micro-benchmark can be configured in several ways to mimic different application types, which are summarised in Table 5.2.

From the implementation point of view, the configuration items listed in Table 5.2 have a significant impact on the operator execution. Specifically, C_s determines how many times the method of random number generation *Math.random()* is invoked by the operator upon any tuple receipt (or tuple sending for topology spout), with $C_s = 1$ representing 100 invocations. Therefore, the higher C_s is set, the larger CPU load the operator will have. Besides, S_s determines the selectivity of this operator as well as the size of internal communication stream within the application, while T_s indicates the operator parallelism which is the number of tasks spawned from this particular operator.

URL-Converter it is selected as a representative of memory-bound applications. Social media websites, such as Twitter and Google, make intensive use of short links for the convenience of sharing. However, these short links eventually need to be interpreted by the service provider to be accessible on the Internet. The URL-Converter is a prototype interpreter, which extracts short Uniform Resource Locators (URLs) from the incoming tweets and replaces them with complete URL addresses in real-time. As depicted in Fig. 5.4b, there are four operators concatenated in tandem: Op_1 (Kestrel Spout) pulls the tweet data from the Kestrel queue as a stream of JSON strings; Op_2 (Json Parser) parses the JSON string for drawing the main message body; Op_3 (URL Filter) identifies the short URLs from the tweet content; and Op_4 (Converter) completes the URL conversion with the help of the remote service. This application results in significant memory usages, as it caches a map of short and complete URLs in memory to identify the trending pages from the statistics and prevent checking the remote database again upon receiving the same short URL.

Twitter Sentiment Analysis (TSA) the second realistic application is adapted from a comprehensive data mining use case—analysing the sentiment of tweet contents by word parsing and scoring. Figure 5.4c shows that there are 11 operators constituting a tree-like topology, with the sentimental score calculated using AFFINN—a list of words associated with pre-defined sentiment values. We refer to [23] for more details of this analysis process and [15] for the application implementation.

5.5.1.2 Parameter Selection and Evaluation Methodology

In our evaluation, the metric collection window is set to 1 minute and the scheduling interval is set to 10 minutes. These values are empirically determined for D-Storm to avoid overshooting and mitigate the fluctuation of metric observations on the selected test applications. As for the heuristic parameters, we configure $D_{\text{Threshold}}$ to 80%, α to 10, and β to 1, putting more emphasis on the immediate gain of each task assignment rather than the potential benefits. Additionally, the latency constraint of each application is set to 500 ms, which represents a typical real-time requirement for streaming use cases.

For all conducted experiments, we deployed the test application using the same approach recommended by the Storm community.¹³ Specifically, the number of worker processes is set to one per machine and the number of executors is configured to be the same as the number of tasks, thereby eliminating unnecessary inter-process communications. Once the test application is deployed, we supply the profiling stream to a given volume and only collect performance results after the application is stabilised.

Also, the performance of the static resource-aware scheduler largely depends on the accuracy of resource profile. As the scheduler required that users submit the static resource profile at compile time [13], we conducted pilot run on test applications, probing their up-to-date resource profile and leading to a fair comparison between the dynamic and static resource-aware schedulers. In particular, we utilised the *LoggingMetricsConsumer*¹⁴ from the *storm-metrics* package to probe the amount of memory/CPU resources being consumed by each operator, and we associate each pilot run with a particular application setting, so that the resource profile can be properly updated whenever the application configuration is changed.

5.5.2 Evaluation of Applicability

In this evaluation, we ran both the synthetic and realistic applications under the same scenario that a given size of profiling stream needs to be processed within the latency constraint. Different schedulers are compared in two major aspects: (1) the amount of inter-node communications resulted from the task placement, and (2) the complete latency of the application in milliseconds.

To better examine the applicability of D-storm, we configure micro-benchmark to exhibit different patterns of resource consumption. These include CPU intensive (varying C_s), I/O intensive (varying S_s) and parallelism intensive (varying T_s). In addition, we alter the volume of profiling stream (P_s) for all the applications to test the scheduler performance under different workload pressures. Table 5.3 lists

¹³ <https://storm.apache.org/documentation/FAQ.html>.

¹⁴ <https://storm.apache.org/releases/1.0.2/javadocs/org/apache/storm/metric/LoggingMetricsConsumer.html>.

Table 5.3 Evaluated configurations and their values (defaults in bold)

Configuration	Value
C_s (for micro-benchmark only)	10, 20, 30, 40
S_s (for micro-benchmark only)	1, 1.333, 1.666, 2
T_s (for micro-benchmark only)	4, 8, 12, 16
P_s (all applications)	2500, 5000, 7500, 10,000

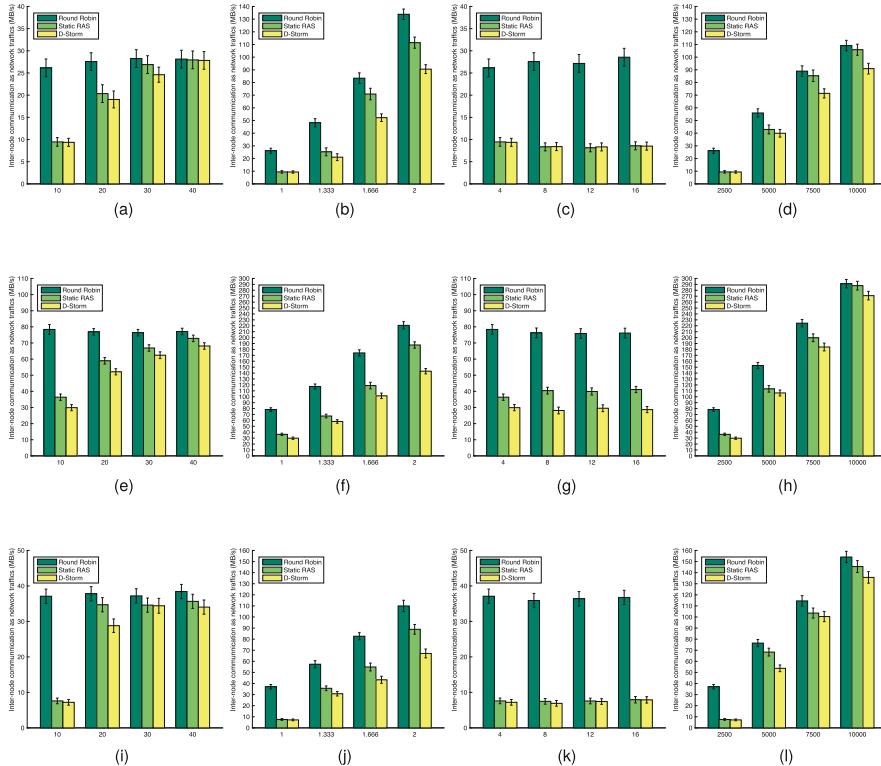


Fig. 5.5 The change of the inter-node communication when varying the configurations of the micro-benchmark. We repeated each experiment for 10 times to show the standard deviation of the results. In the legend, RAS stands for the Resource-Aware scheduler. (a) Varying C_s (Synthetic linear). (b) Varying S_s (Synthetic linear). (c) Varying T_s (Synthetic linear). (d) Varying P_s (Synthetic linear). (e) Varying C_s (Synthetic diamond). (f) Varying S_s (Synthetic diamond). (g) Varying T_s (Synthetic diamond). (h) Varying P_s (Synthetic diamond). (i) Varying C_s (Synthetic star). (j) Varying S_s (Synthetic star). (k) Varying T_s (Synthetic star). (l) Varying P_s (Synthetic star)

the evaluated values for the application configurations, where the default values are highlighted in bold. Note that when one configuration is altered, the others are set to their default value for fair comparison.

Figure 5.5 presents the changes of inter-node communication while altering the micro-benchmark configurations listed in Table 5.3. We find that our D-Storm

prototype always performs at least as well as the static counterpart and often results in significant communication reduction as compared to the default scheduler.

Specifically, a study of Fig. 5.5a, e and i reveals that D-Storm performs slightly better than, or at least similarly to the static Resource-Aware Scheduler (RAS) when applied to CPU-intensive use cases. In most instances, D-Storm achieves the similar communication reduction as the static RAS, with the difference also reaching as high as 17% when C_s is set to 20 for the Star topology. We interpret this performance similarity in that all streaming tasks are created homogeneously by their implementation, which makes the job easy for the static method to probe accurate resource profiles through a pilot run. Moreover, as the scheduling of CPU-intensive application reduces to a typical bin-packing problem, both resource-aware approaches performed well in the beginning by utilising the large nodes first for assignment. On average, they saved 63.9%, 57.7% and 80.1% inter-node traffic compared to the default scheduler in the linear, diamond and star topology, respectively.

This also explains the variation trend we see in the figures: as the applications become increasingly computational-intensive, the performance gain of being resource-aware is gradually reduced (from on average 67.2% less to almost identical). This is because the streaming tasks are forced to spread out to other nodes as they become more resource-demanding, thus introducing new inter-node communications within the cluster.

However, it is worth noting that the communication reduction brought by the static RAS is based on the correct resource profile provided by the pilot run. If this information were not specified correctly, the static resource-aware scheduler would lead to undesirable scheduling results, causing over-utilisation and impairing the system stability.

Figure 5.5b, f and j, on the other hand, showcase the communication changes as the selectivity configuration is altered, which creates heterogeneous and intensive communications on the tailing edges of the topology. The results demonstrate that D-Storm outperforms the static RAS in terms of the communication reduction by on average 15.8%, 17.4% and 16.2% for the linear, diamond and star topology, respectively. This is credited to the fact that D-Storm is able to take runtime communications into account during the decision-making process. By contrast, the existing resource-aware scheduler can only optimise inter-node communication based on the number of task connections, which contains only coarse-grained information and does not reflect the actual communication pattern. We also find out that the amount of inter-node communication increases rapidly along with the growing selectivity. Especially, the four-operator linear topology exhibits 5.1, 11.8 and 9.7 times network traffic increase under all the three schedulers when the selectivity configuration doubles, which proves that the internal communication has been magnified exponentially by the concatenated selectivity settings.

Figure 5.5c, g and k compare the three schedulers in parallelism intensive test cases. The analysis of results discovers that the amount of inter-node communication is relatively insensitive to the variations of the parallelism settings. We found it is because a single worker node can accommodate more tasks for execution, as

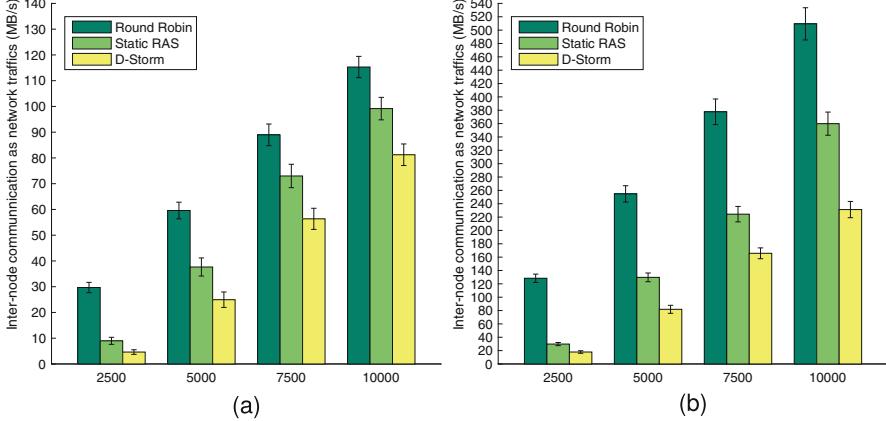


Fig. 5.6 The change of the inter-node communication when varying the input size of realistic applications. We repeated each experiment for 10 times to show the standard deviation of the results. (a) Varying P_s (URL-converter). (b) Varying P_s (TSA)

the resource requirement of each streaming task reduces effectively in inverse proportion to the parallelism increase. However, it is also worth reporting that in Fig. 5.5g the static RAS performs noticeably worse than D-Storm, causing an average of 10.4 MB/s more network traffic in the four test cases. We interpret this result as the static scheduler having overestimated the resource requirement for Op_6 , which results in the use of more worker nodes in the cluster. As a matter of fact, the additional overhead of thread scheduling and context switching increases along with the parallelism setting, which would be hard for the static RAS to estimate prior to the actual execution.

Finally, we evaluate the communication changes as the test application handles different volumes of workload. The results of micro-benchmark are shown in Fig. 5.5d, h and l, while the results of the realistic applications are presented in Fig. 5.6. Specifically, D-Storm and the static RAS performed similarly when applied to the micro-benchmark, which demonstrates that the static method works well on applications with homogeneous operators. On the other hand, D-Storm achieved much better performance than its static counterpart in the realistic applications—Fig. 5.6 shows that D-Storm improves the communication reduction by 14.7%, 21.3%, 18.7% and 15.5% in the URL-Converter, and by 9.3%, 18.8%, 15.5% and 25.3% in the Twitter Sentiment Analysis when P_s is varied from 2500 to 10,000. Part of this performance improvement is credited to D-Storm being able to handle uneven load distributions, which is a common problem in realistic applications due to the hash function based stream routing. As a contrast, the static scheduler configures resource profile at the operator-level, deeming the spawned streaming tasks homogeneous in all aspects. Consequently, the increasingly unbalanced workload distribution among the same-operator tasks is ignored, which leads to the performance degradation of the static scheduler.

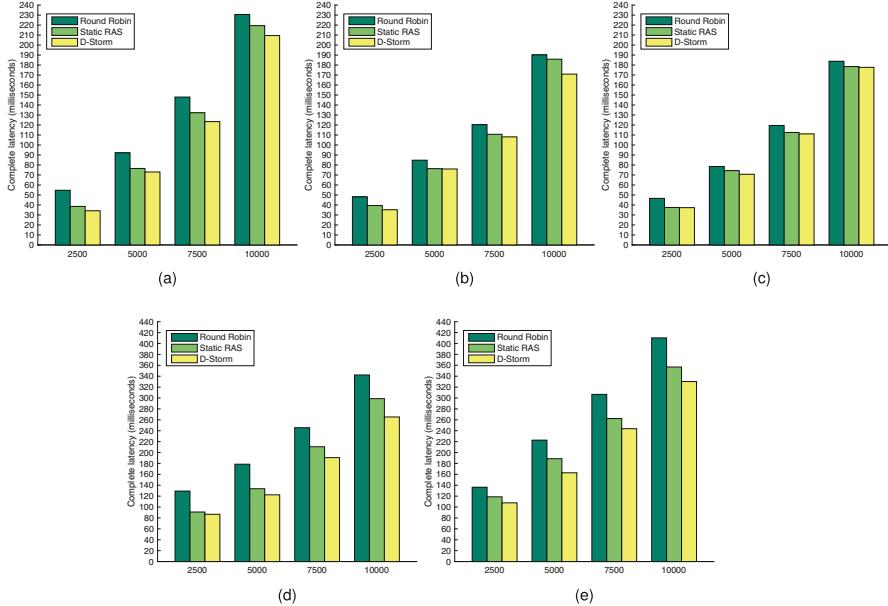


Fig. 5.7 The application complete latency under different volumes of profiling streams. Each result is an average of statistics collected in a time window of 10 minutes, so the error bar is omitted as the standard deviation of latency is negligible for stabilised applications. **(a)** Varying P_s (Linear). **(b)** Varying P_s (Diamond). **(c)** Varying P_s (Star). **(d)** Varying P_s (URL-converter). **(e)** Varying P_s (TSA)

We also collected the metric of complete latency to examine the application responsiveness while using different schedulers. As observed in Fig. 5.7, the complete latency is strongly affected by the combination of two factors—the size of the profiling stream and the volume of inter-node communications. First of all, the higher the application throughput, the higher the complete latency is likely to be yield. If we calculate an average complete latency for these three schedulers, we can find out the results for the linear, diamond, star, URL-Converter and Twitter Sentiment Analysis have increased to 5.2, 4.4, 4.3, 2.9 and 3.1 times the original values, respectively. It also shows that more resources are required for the Twitter Sentiment Analysis to handle higher throughput without violating the given latency constraint, as the complete latency has reached as high as 410.4 milliseconds in our evaluation.

Besides, we notice that reducing the inter-node communication is also beneficial to improving the application responsiveness. As shown in Fig. 5.7d, e, packing communicating tasks onto fewer worker nodes allows D-Storm to reduce the communication latency to on average 72.7 and 78% of that of the default scheduler in the URL-Converter and Twitter Sentiment Analysis, respectively. These results confirm the fact that conducting communication on networks is much more expensive than

inter-thread messaging, as the later avoids data serialisation and network delay through the use of a low-latency, high-throughput message queue in memory.

5.5.3 Evaluation of Cost Efficiency

Modelling the scheduling problem as a bin-packing variant offers the possibility to consolidate tasks into fewer nodes when the volume of incoming workload decreases. In this evaluation, we examine the minimal resources required to process the given workload without violating the latency constraint. Specifically, D-Strom scheduler is applied to the test applications, with the size of input stream (P_s) varied from 10,000 tuples/second to 2500 tuples/second. To intuitively illustrate the cost of resource usages, we associate each worker node created in the Nectar cloud with the pricing model in the AWS Sydney Region.¹⁵ In particular, a small instance is billed at \$0.0292 per hour, a medium instance costs \$0.0584 per hour and a large instance charges \$0.1168 per hour.

All five test applications introduced in Sect. 5.5.1.1 are included in this evaluation, in which the synthetic topologies have configured their settings to the default values. As shown in Fig. 5.8, the cost of resources used by the D-Storm scheduler steadily reduces when the input load decreases. Specifically, the diamond topology is the most resource-consuming synthetic application in the micro-benchmark, utilising 4 large nodes and 4 medium nodes to handle the profiling stream at 10,000

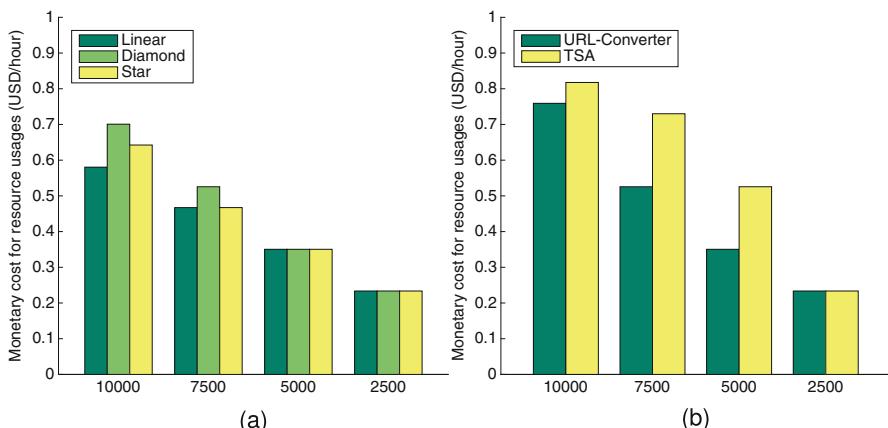


Fig. 5.8 Cost efficiency analysis of D-Storm scheduler as the input load decreases. The pricing model in the AWS Sydney region is used to calculate the resource usage cost. **(a)** Decreasing P_s (Micro-benchmark). **(b)** Decreasing P_s (Realistic apps)

¹⁵ <https://aws.amazon.com/ec2/pricing/>.

tuples/second. Such resource configuration also demonstrates that D-Storm avoids using smaller instances for scheduling unless the larger nodes are all depleted, which helps minimise the inter-node communication and improve the application latency. As the volume of the profiling stream drops from 10,000 tuples/second to 2500 tuples/second, the resource consolidation is triggered and the usage cost of the diamond, linear and star topology reduces to 33.3%, 40.2%, 36.4% of the original values, respectively.

This same trend is also observed in the evaluation of realistic applications, with consolidation resulting in 69.2 and 71.43% cost reduction for the URL-Converter and Twitter Sentiment Analysis, respectively. In particular, Twitter Sentiment Analysis only requires two large instances to handle the reduced workload whereas it used to occupy the whole cluster for processing.

However, the comparable schedulers such as the static resource-aware scheduler and the default Storm scheduler lack the ability to consolidate tasks when necessary. In these test scenarios, they would occupy the same amount of resources even if the input load dropped to only one-quarter of the previous amount, which results in under-utilisation and significant resource waste.

5.5.4 Evaluation of Scheduling Overhead

We also examine the time required for D-Storm to calculate a viable scheduling plan using Algorithm 1, as compared to that of the static RAS scheduler and the default Storm scheduler. In this case, the synthetic applications are evaluated with various parallelism settings, as well as the realistic applications under the default size of the profiling stream.

Specifically, we utilised the *java.lang.System.nanoTime* method to probe the elapsed time of scheduling in nanosecond precision. To overcome the fluctuation of results, we repeated the clocking procedure for 5 times and present the average values in Table 5.4.

Studying Table 5.4, we find that the default Storm scheduler is the fastest among all three comparable schedulers, which takes less than 3 milliseconds to run the round-robin strategy for all test applications. Its performance is also relatively insensitive to the increasing parallelism configuration, as there are no task sorting or comparison involved in the scheduling process.

On the other hand, the static resource-aware scheduler usually takes 3–6 milliseconds to run its greedy algorithm. Compared to the default round-robin scheduler, it consumes roughly twice the time of the former to make sure that the number of communication connections across different worker nodes is minimised.

In contrast, the algorithm proposed in D-Storm is the slowest among the three, as it requires dynamically re-sorting all the remaining tasks by their updated priority after each single task assignment. However, considering the fact that the absolute value of the time consumption is at the millisecond level, and the analysis in

Table 5.4 Time consumed in creating schedules by different strategies (unit: milliseconds)

Schedulers	Test cases			
	Linear topology			
	$T_s = 4$	$T_s = 8$	$T_s = 12$	$T_s = 16$
D-Storm	15.49	18.07	26.52	32.29
Static scheduler	3.01	3.91	4.25	4.51
Default scheduler	1.20	1.64	1.98	2.04
Schedulers	Test cases			
	Diamond topology			
	$T_s = 4$	$T_s = 8$	$T_s = 12$	$T_s = 16$
D-Storm	19.08	22.90	35.89	39.64
Static scheduler	3.29	3.62	5.78	6.01
Default scheduler	1.77	1.51	2.84	2.83
Schedulers	Test cases			
	Star topology			
	$T_s = 4$	$T_s = 8$	$T_s = 12$	$T_s = 16$
D-Storm	13.80	23.66	28.71	32.59
Static scheduler	3.11	5.38	5.76	5.27
Default scheduler	1.36	1.78	2.17	2.47
Schedulers	Test cases			
	Realistic applications			
	URL-converter	TSA		
D-Storm	18.17	42.25		
Static scheduler	5.56	5.91		
Default scheduler	1.55	2.99		

Sect. 5.3.3 has shown that the algorithm is at worst in quadratic time complexity, we conclude our solution is still efficient and scalable to deal with large problem instances from the real world.

5.6 Related Work

Scheduling of streaming applications has attracted close attention from both big data researchers and practitioners. This section conducts a multifaceted comparison between the proposed D-Storm prototype and the most related schedulers in various aspects, as summarised in Table 5.5.

Aniello et al. pioneered dynamic scheduling algorithms in the stream processing context [12]. They developed a heuristic-based algorithm that prioritises the placement of communicating tasks, thus reducing the amount of inter-node communication. The proposed solution is self-adaptive, which includes a task monitor to collect metrics at runtime and conducts threshold-based re-scheduling

Table 5.5 Related work comparison

Aspects	Related works								Our work
	[12]	[13]	[9]	[11]	[24]	[25]	[26]	[6]	
Dynamic	Y	N	Y	Y	Y	Y	N	Y	Y
Resource-aware	N	Y	N	N	Y	Y	Y	N	Y
Communication-aware	Y	N	Y	Y	N	N	Y	Y	Y
Self-adaptive	Y	N	Y	Y	N	N	N	Y	Y
User-transparent	N	N	Y	Y	N	N	N	N	Y
Cost-efficient	N	Y	N	N	Y	Y	N	N	Y

for performance improvement. However, the task monitor is not transparently set up at the middleware level and the algorithm is unaware of the resource demands of each task being scheduled. It also lacks the ability to consolidate tasks into fewer nodes for improving cost efficiency.

By modelling the task scheduling as a graph partitioning problem, Fisher et al. [9] demonstrated that the METIS software is also applicable to the scheduling of stream processing applications, which achieves better results on load balancing and further reduction of inter-node communication as compared to Aniello's work [12]. However, their work is also not aware of resource demand and availability, let alone reducing the resource footprints with regard to the varying input load.

Xu et al. proposed another dynamic scheduler that is not only communication-aware but also user-transparent [11]. The proposed algorithm reduces inter-node traffic through iterative tuning and mitigates the resource contention by passively rebalancing the workload distribution. However, it does not model the resource consumption and availability for each task and node, thus lacking the ability to prevent resource contention from happening in the first place.

Sun et al. investigated energy-efficient scheduling by modelling the mathematical relationship between energy consumption, response time and resource utilisation [24]. They also studied reliability-oriented scheduling to trade-off between competing objectives like better fault tolerance and lower response time [25]. But the algorithms proposed in these two papers require modifying the application topology to merge operators on non-critical paths. A similar technique is also seen in Li's work [6], which adjusts the number of tasks for each operator to mitigate performance bottleneck at runtime. Nevertheless, bundling scheduling with topology adjustment sacrifices the user transparency and impairs the applicability of the approach.

Cardellini et al. [27] proposed a distributed QoS-aware scheduler that aims at placing the streaming applications as close as possible to the data sources and final consumers. Differently, D-Storm makes scheduling decisions out of the resource-saving perspective and regards the minimisation of network communication as its first-class citizen. Papageorgiou et al. [28] proposed a deployment model for stream processing applications to optimise the application-external interactions with other Internet-of-Things entities such as databases or users, while our work

focuses entirely on reducing network traffic among streaming operators. Schneider et al. [29] proposed an elastic scheduling algorithm for the ordered streaming runtime to minimise thread synchronisation, global data and access locks, allowing any thread to execute any operator while maintaining the tuple order in operator communication. Their work focuses on finding the right number of threads to be used without seeking input from programmers, while D-Storm aims to distribute working threads with regard to their resource consumption and communication patterns. Shukla et al. [30] proposed a model-based approach to offer reliable estimates of the required resource allocation, which makes use of a priori knowledge of the applications to provide predictable scheduling behaviour. In comparison, our work does not require building application performance models beforehand and the resource profile of each task is obtained and updated during runtime. The same authors also studied robust means to respond to dynamism in the input rates and task behaviour and provided rapid task migration across VMs [31]. Since Storm does not support lossless migration of operator states and state management is not the main focus of our contribution, we have implemented a simple state management mechanism in test applications utilising Redis in-memory store to support relocation of the operator internal state.

The static resource-aware scheduler proposed by Peng et al. [13] has been introduced in Sects. 5.1 and 5.2. The main limitation of their work, as well as [26, 32], is that the runtime changes to the resource consumptions and availability are not taken into consideration during the scheduling process.

5.7 Conclusions and Future Work

In this chapter, we proposed a resource-efficient algorithm for scheduling streaming applications in Data Stream Management Systems and implemented a prototype scheduler named D-Storm to validate its effectiveness. It tackles new scheduling challenges introduced by the deployment migration to computing clouds, including node heterogeneity, network complexity and the need of workload-oriented task consolidation. D-Storm tracks each streaming task at runtime to collect its resource usages and communication pattern, and then it formulates a multidimensional bin-packing problem in the scheduling process to pack communicating tasks as compact as possible while respecting the resource constraints. The compact scheduling strategy leads to the reduction of inter-node communication and resource costs, as well as reducing the processing latency to improve the application responsiveness. Our new algorithm overcomes the limitation of the static resource-aware scheduler, offering the ability to adjust the scheduling plan to the runtime changes while remaining sheer transparent to the upper-level application logic.

As for future work, we are currently studying the impact of sudden workload change to the scheduler performance, as well as improving the design of D-Storm scheduler with a backpressure mechanism in order to avoid oscillation

and over-adjustment. We also plan to investigate the use of meta-heuristics to find a better solution for the scheduling problem. Genetic algorithms, simulated annealing and tabu search are among the list of candidates that require further investigation. In addition, we would like to conduct scheduling research in an Edge and Fog environment, where heterogeneous and geographically distributed resources are typically connected using wireless technologies that are subject to network constraints such as limited bandwidth, longer communication delays and unpredictable interruptions [4]. The proposed scheduling algorithm should take the networking dimension into consideration to generate an efficient scheduling strategy, which avoids over-utilising the power-limited Edge devices and puts a large volume of task communications over links with higher network performance.

Acknowledgments This work is supported by Australian Research Council (ARC) and China Scholarship Council (CSC). This chapter is a substantial extension of a preliminary conference paper [14], with an improved algorithm design, new test applications and new experimental results collected from a heterogeneous cloud environment.

References

1. N. Mitton, S. Papavassiliou, A. Puliafito, K.S. Trivedi, Combining cloud and sensors in a smart city environment. *EURASIP J. Wirel. Commun. Netw.* **2012**(1), 247–256 (2012)
2. A. Puliafito, A. Cucinotta, A.L. Minnolo, A. Zaia, Making the Internet of things a reality: The WhereX solution, in *The Internet of Things* (Springer, New York, 2010), pp. 99–108
3. A. Cuzzocrea, G. Fortino, O. Rana, Managing data and processes in cloud-enabled large-scale sensor networks: State-of-the-art and future research directions, in *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, ser. CCGRID '13 (IEEE, Piscataway, 2013), pp. 583–588
4. X. Liu, R. Buyya, Resource management and scheduling in distributed stream processing systems: a taxonomy, review, and future directions. *ACM Comput. Surv.* **53**(3) (2020). <https://doi.org/10.1145/3355399>
5. X. Liu, R. Buyya, Performance-oriented deployment of streaming applications on cloud. *IEEE Trans. Big Data* **5**(1), 46–59 (2019)
6. C. Li, J. Zhang, Y. Luo, Real-time scheduling based on optimized topology and communication traffic in distributed real-time computation platform of storm. *J. Netw. Comput. Appl.* **87**(3), 100–115 (2017)
7. D. Sun, R. Huang, A stable online scheduling strategy for real-time stream computing over fluctuating big data streams. *IEEE Access* **4**, 8593–8607 (2016)
8. L. Cheng, T. Li, Efficient data redistribution to speedup big data analytics in large systems, in *Proceedings of the 23rd IEEE International Conference on High Performance Computing*, ser. HiPC '16, (2016), pp. 91–100
9. L. Fischer, A. Bernstein, Workload scheduling in distributed stream processors using graph partitioning, in *Proceedings of the 2015 IEEE International Conference on Big Data* (IEEE, Piscataway, 2015), pp. 124–133
10. A. Chatzistergiou, S.D. Viglas, Fast heuristics for near-optimal task allocation in data stream processing over clusters, in *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, ser. CIKM '14 (ACM Press, New York, 2014), pp. 1579–1588

11. J. Xu, Z. Chen, J. Tang, S. Su, T-storm: Traffic-aware online scheduling in storm, in *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems*, ser. ICDCS '14 (IEEE, Piscataway, 2014), pp. 535–544
12. L. Aniello, R. Baldoni, L. Querzoni, Adaptive online scheduling in storm, in *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems*, ser. DEBS '13 (ACM Press, New York, 2013), pp. 207–218
13. B. Peng, M. Hosseini, Z. Hong, R. Farivar, R. Campbell, R-storm: Resource-aware scheduling in storm, in *Proceedings of the 16th Annual Conference on Middleware*, ser. Middleware '15 (ACM Press, New York, 2015), pp. 149–161
14. X. Liu, R. Buyya, D-storm: Dynamic resource-efficient scheduling of stream processing applications, in *Proceedings of the IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)* (IEEE, Piscataway, 2017), pp. 485–492
15. X. Liu, R. Buyya, Performance-oriented deployment of streaming applications on cloud. *IEEE Trans. Big Data* **14**(8), 1–14 (2017)
16. E.G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, D. Vigo, Bin packing approximation algorithms: Survey and classification, in *Handbook of Combinatorial Optimization*, ed. by P.M. Pardalos, D.-Z. Du, R.L. Graham (Springer, New York, 2013), pp. 455–531
17. R. Tolosana-Calasanz, J. Á. Bañares, C. Pham, O.F. Rana, Resource management for bursty streams on multi-tenancy cloud environments. *Futur. Gener. Comput. Syst.* **55**, 444–459 (2016)
18. A.S. Fukunaga, R.E. Korf, Bin-completion algorithms for multicontainer packing and covering problems, in *Proceedings of the 2005 International Joint Conference on Artificial Intelligence*, ser. IJCAI '05 (ACM Press, New York, 2005), pp. 117–124
19. J. Desrosiers, M.E. Lübbecke, Branch-price-and-cut algorithms, in *Wiley Encyclopedia of Operations Research and Management Science* (Wiley, Hoboken, 2011), pp. 1–18
20. A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, I. Stoica, Dominant resource fairness : Fair allocation of multiple resource types, in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI '11 (2011), pp. 323–336
21. S. Caton, O. Rana, towards autonomic management for cloud services based upon volunteered resources. *Concurrency Comput. Pract. Exp.* **24**(9), 992–1014 (2012)
22. J.O. Kephart, D.M. Chess, The vision of autonomic computing. *Computer* **36**(1), 41–50 (2003)
23. F.A. Nielsen, A new ANEW: Evaluation of a word list for sentiment analysis in microblogs, in *Proceedings of the ESWC2011 Workshop on ‘Making Sense of Microposts’: Big Things Come in Small Packages* (Springer, Berlin, 2011), pp. 93–98
24. D. Sun, G. Zhang, S. Yang, W. Zheng, S.U. Khan, K. Li, Re-stream: real-time and energy-efficient resource scheduling in big data stream computing environments. *Inf. Sci.* **319**, 92–112 (2015)
25. D. Sun, G. Zhang, C. Wu, K. Li, W. Zheng, Building a fault tolerant framework with deadline guarantee in big data stream computing environments. *J. Comput. Syst. Sci.* **89**, 4–23 (2017)
26. T. Li, J. Tang, J. Xu, Performance modeling and predictive scheduling for distributed stream data processing. *IEEE Trans. Big Data* **7790**(99), 1–12 (2016)
27. V. Cardellini, V. Grassi, F.L. Presti, M. Nardelli, On QOS-aware scheduling of data stream applications over fog computing infrastructures, in *Proceedings of the IEEE Symposium on Computers and Communication* (IEEE, Piscataway, 2015), pp. 271–276
28. A. Papageorgiou, E. Poormohammady, B. Cheng, Edge-computing-aware deployment of stream processing tasks based on topology-external information: Model, algorithms, and a storm-based prototype, in *Proceedings of the 5th IEEE International Congress on Big Data* (IEEE, Piscataway, 2016), pp. 259–266
29. S. Schneider, K.-L. Wu, Low-synchronization, mostly lock-free, elastic scheduling for streaming runtimes, in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation* (ACM Press, New York, 2017), pp. 648–661
30. A. Shukla, Y. Simmhan, Model-driven scheduling for distributed stream processing systems. *J. Parallel Distrib. Comput.* **117**, 98–114 (2018)

31. A. Shukla, Y. Simmhan, Toward reliable and rapid elasticity for streaming dataflows on clouds, in *Proceedings of the 38th IEEE International Conference on Distributed Computing Systems* (IEEE, Piscataway, 2018), pp. 1096–1106
32. P. Smirnov, M. Melnik, D. Nasonov, Performance-aware scheduling of streaming applications using genetic algorithm. *Procedia Comput. Sci.* **108**(6), 2240–2249 (2017)

Chapter 6

Serverless Platforms on the Edge: A Performance Analysis



Hamza Javed, Adel N. Toosi , and Mohammad S. Aslanpour

Contents

6.1	Introduction	165
6.2	Background	167
6.2.1	Motivation	167
6.2.2	History of Serverless	168
6.2.3	Serverless at the Edge	170
6.3	Related Works	170
6.3.1	AWS Lambda	173
6.3.2	AWS Greengrass	173
6.3.3	Azure Functions	173
6.3.4	Apache OpenWhisk	174
6.3.5	OpenFaaS	174
6.4	Performance Evaluation	174
6.4.1	Experimental Setup	175
6.4.2	Results	177
6.5	Discussions	181
6.6	Conclusions and Future Work	182
	Bibliography	182

6.1 Introduction

In recent years, we have seen a major increase in new developments in cloud computing due the requirement of handling data at a massive scale. New technologies such as Internet of Things (IoT) and edge devices such as smartphones have created a large influx of data and network traffic that has to be processed at scale [27, 33]. This has given rise to new technologies and computational architectures being introduced into cloud computing in order to maintain the Quality of Service (QoS) of the applications providing these services. In the past few

H. Javed · A. N. Toosi · M. S. Aslanpour

Faculty of Information Technology, Monash University, Clayton, VIC, Australia

e-mail: hamza.javed@monash.edu; adel.n.toosi@monash.edu;

mohammad.aslanpour@monash.edu

years, deployment architectures have changed from monolithic [4] to microservice architectures that aid the developers to build large applications in a scalable manner. Microservice architectures demanded the need of Container-as-a-Service (CaaS) [13] that provided always-on servers with the ability to scale out easily. However, serverless computing provides a different approach to managing servers in the form of Function-as-a-Service (FaaS) [7] that proposes the elimination of always-on servers and transition of the services to an on-use basis.

Serverless architecture uses the approach of using ephemeral containers that can be launched instantaneously on request and be stopped upon completion of computation [22]. This requires the developers to break down their application into multiple functions as the basic units of computation and these functions are hosted in separate containers that are launched upon request and are destroyed when the function execution is over. This architecture provides the opportunity to reduce the resource usage of containers when they are idle and are not receiving requests and also reduces the cost of always-running servers [20, 24]. Instead, the cost of the infrastructure is only accumulated for the time the function was actually executed. The serverless computing model removes the need for infrastructure management, and this provides a great advantage over monolithic and microservice deployment models.

The serverless computing model provides a great opportunity for edge computing, which consists of an event-driven architecture and is subject to varying workloads [25, 31]. Processing data on the edge of the network provides a solution for limited network bandwidth and latency problems as well as data privacy concerns by executing code closer to devices and end users [12]. This compute model can provide support for latency-sensitive workloads [10] and help adhere to compliance and privacy regulations. Serverless computing also offers reliability and the ability to scale with no associated management overhead in an edge environment. A serverless architecture for edge computing, what we call it *serverless edge computing*, can provide the benefit of faster deployment, a smaller footprint and increased performance [5].

This book chapter focuses on the performance evaluation of serverless platforms of both public cloud and open-source contributors in an attempt to analyze the viability of serverless platforms in an edge environment. In order to emulate the edge environment, we aim to set up the serverless platforms on ARM based single-board computers (SBCs), namely, Raspberry Pis, and we evaluate the performance of ARM architecture-compatible serverless platforms on these devices. For small-scale infrastructures that are set up in edge environments, single-board computers are commonly used. By simulating a similar environment, this study aims to provide insights on the performance of serverless platform on resource-constrained devices as this research does not exist in the present literature. The analyzed open-source serverless platforms are OpenFaaS,¹ Apache OpenWhisk,² and AWS

¹ OpenFaaS (2020), <https://docs.openfaas.com>.

² Apache OpenWhisk (2020), <https://openwhisk.apache.org/documentation.html>.

Greengrass.³ We compare their performance with serverless computing services of AWS Lambda⁴ and Azure Functions⁵ in an attempt to research the suitability of a serverless architecture on the edge. We set up our experiments by configuring these serverless platforms on a cluster of four Raspberry Pis and subjected them to varying amounts of workloads and analyzed the differences in the performance metrics such as response time and success rate for various types of workloads, e.g., CPU-intensive, memory-intensive, and disk-intensive functions.

The organization of this book chapter is as follows: Sect. 6.2 provides a background and motivation on the topic. Section 6.3 focuses on the related works that have been carried out in this research area and describes serverless computing platforms that we have selected for this study. Section 6.4 discusses our performance evaluation methodology, experimental setup, and the analysis of the results. Section 6.5 presents the discussion on our findings. Finally, Sect. 6.6 provides a conclusion to our work and outlines future work.

6.2 Background

6.2.1 Motivation

Imagine a course-grained application with several dependencies between its services which is deployed on the cloud. The incoming workload is naturally highly dynamic, so the application requires varied amount of computing resources to respond to the dynamism over the time, so-called auto-scaling [1]. This scenario applies to a great number of web applications hosted in the cloud [3].

The first deployment solution appears resource over-provisioning: providing resources, i.e., VMs, for the application as much as the required resources for the maximum expected incoming workload. This solution will greatly guarantee Quality of Service (QoS), but resource wastage is highly likely. Inefficiency also appears more seriously when hypervisor-based machines, i.e., VMs, with large footprint are intended to host the application.

Another solution for efficiently deploying monolithic applications is to adopting an auto-scaler to dynamically respond to the workload changes by adding or removing replicas of the application, i.e., adding or removing VMs [3]. This can narrow the gap, but auto-scaling large monolithic applications with several interconnected services will not potentially be a sufficiently smart and quick action. It is not smart, as potentially only certain parts of the application may need auto-scaling, not the entire application. A monolithic application is composed of several services tightly coupled with each other and cannot be scaled individually easily. The solution is not quick also, as heavyweight hypervisor-based virtualization tends

³ AWS Greengrass (2020), <https://docs.aws.amazon.com/greengrass/>.

⁴ AWS lambda (2020), <https://docs.aws.amazon.com/lambda/index.html>.

⁵ Microsoft Azure Functions (2020), <https://docs.microsoft.com/en-us/azure/azure-functions>.

to suffer from long boot-up times for newly provisioned VMs, sacrificing QoS. Add to auto-scaling challenges the considerable complexity for deploying auto-scalers that will not be practical for non-expert users.

Given the motivational scenario, this question is raised: “how we can re-design cloud platforms to properly respond to the dynamic nature of the application?” Recently, serverless computing has attracted great attention in dealing with such challenges. In the following, a historical view of serverless computing will be presented.

6.2.2 *History of Serverless*

In a nutshell, “serverless attended the reunion between technological advances such as microservices, containerization, and the idea of effortless auto-scaling and pure pay per use model [2].” From the DevOps perspective, let us fork the advances in development (e.g., coding) and deployment (e.g., installation and maintenance) sides for applications and merge them into a united idea as serverless.

Development advances: monolithic applications, despite easy deployment, are not easily scaled. Hence, service-oriented architectures for applications appeared and advanced toward today’s microservices that reshape a monolithic application into loosely coupled services that can scale individually. Furthermore, the idea of fully decomposed application arose, which assumes each individual microservice can live by itself, under the name of a function. This led to Function-as-a-Services (FaaS) [2] wherein only an individual service is developed for a single unit of task. FaaS also decouples the application from their state and runs stateless functions. Note that if the state is still required for certain applications while staying in FaaS, the state can be preserved separately by storage services. With this advancement, the question is how to take advantage of FaaS in deployment side? and how to bring them in reality?

Deployment advances: given the large footprint of VMs, the containerization [2] came into the picture. This means that, instead of abstracting the underlying resource from a hypervisor in order for launching an isolated computing resource as VM, one can employ the OS kernel to launch a semi-isolated container. This idea came from small footprint requirements for computing resources to efficiently provision resources for certain applications. With this in mind and the advances in development side, it is obvious that containerization will definitely suit FaaS, where single units of computation are intended to be provisioned and scaled.

One step is left to serverless. Such advances helped decoupling monolithic applications to fine-grained FaaS and help provisioning lightweight resources, i.e., containers instead of VMs for making FaaS a reality. However, the challenges of auto-scaling and maintenance are still remained unsolved and poses the question that how to benefit from such advances for efficient execution of applications?

That was a problem until Amazon introduced Lambda [2] for running functions in terms of FaaS that run on ephemeral short-lived lightweight virtualization

(container) technology called Firecracker. The auto-scaling and maintenance is left to the serverless platform. Precisely, serverless platforms enable developers to write their applications in terms of functions in any language of interest. Then, the platform is responsible for provisioning resources per each request (invocation) to the function. Simplistically, once a request is made to the function, the serverless platform will run a new container, and after completing the task, the container is terminated or stays alive for a short period of time for reusability. Functions are intended to run ephemerally and for short-running tasks. Note that the idea of FaaS and serverless appeared under the same banner historically and occasionally are used interchangeably. An added bonus with serverless offered by cloud providers is also to realize a pure pay per use pricing model wherein one just pays for the actual execution time of tasks. In conventional cloud pricing models, customers pay for the duration of time the resources are provisioned, regardless of being actually used or not.

This provides a huge advantage over other infrastructures such as Platform-as-a-Service (PaaS) or Container-as-a-Service (CaaS) as servers do not need to be running constantly in the background accumulating costs. Functions start within milliseconds and process individual requests. If there are several simultaneous requests for your function, then the system will create as many copies of the function, which will be managed by the container orchestration system, as needed to meet demand. When demand drops, the application automatically scales down. Dynamic scaling is a benefit of FaaS and is cost-efficient as well because providers only charge for the resources that are used, not idle time [21]. When running on premise, this dynamic nature can also increase platform density, allowing more workloads to run and optimize resource consumption. An event-driven service that needs horizontal scaling can work well as a function, as well as RESTful applications [8]. Figure 6.1 shows a schematic serverless architecture.

Given the emergence of AWS Lambda serverless in 2014 as the pioneer, several companies and individual users employed this platform in practice. Other IT companies such as Microsoft (Azure Functions), Google (Google Cloud Functions)

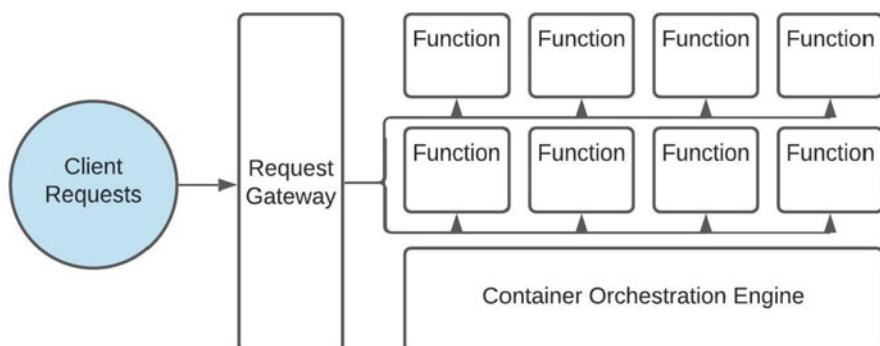


Fig. 6.1 Serverless architecture

IBM (Open Whisk), etc. also attended the market and offered their own serverless platforms. Open-source platforms also came to the picture including OpenFaaS, Kubeless, etc.

6.2.3 *Serverless at the Edge*

Given the focus area of this research—serverless edge computing—this question is raised that “why will serverless be good practice for edge computing?”

Edge computing is intended to bring computation closer to data sources and of the biggest stakeholders of edge will be IoT applications [2]. Let us analyze how IoT applications work. Typically, a group of sensors are located in a particular area, e.g., street, farm, factory, body, etc., in order for collecting and sending data to a computing node for execution. The occurrence of such events is highly variable. The execution also would involve short-running tasks such as analyzing if a body temperature is higher than a certain degree or if the moisture content of the soil is at a certain level. Add to these characteristics of IoT applications that they will potentially face energy preservation challenges of edge nodes as well. An edge computing platform for several IoT use cases is deemed to perform on low-power devices such as SBCs that demands certain considerations for energy saving.

Obviously, IoT applications with event-driven nature and short-running tasks characteristics will match with serverless platforms functionality. Serverless tends to avoid always-on deployment of applications and instead tends provision containers per requests and to terminate after execution. Hence, this will be a huge step toward energy saving on edge nodes as well.

Theoretically, serverless appears practical for adaptation in edge. This idea has attracted cloud serverless providers to re-design their platform to support edge-specific requirements as well. AWS Lambda was once again of the dominant in this move and AWS is now offering relative services such as Greengrass. In academia, researchers also made effort to assess the feasibility of serverless at the edge for different IoT use cases, which will be elaborated in the next section.

6.3 Related Works

Function-as-a-Service (FaaS) is an event-driven computing execution model that runs in stateless containers and those functions manage server-side logic and state through the use of services [9, 29]. It allows developers to develop, deploy, and manage those application services as functions without having to maintain their own infrastructure. FaaS provides developers with the flexibility to develop event-driven applications without managing servers. Serverless infrastructure has the ability to scale to zero, i.e., when there are no requests, the servers are stopped and are only

running when they are needed [11]. Serverless computing is a relatively new area and has received a lot of attention in recent years [16, 30]. This is due to the potential the serverless computing architecture offers and the need for being validated by research. The work by Van Eyk et al. [30] provides a detailed description and explanation of the serverless computing architecture along with the evolution of cloud computing that led to the rise of serverless computing. Lee et al. [16] performed a performance analysis and comparison of the serverless computing services from major cloud providers such as AWS, Azure, GCP, and IBM Cloud and analyzed the performance of these services in a production environment.

The work done by Mohanty et al. [18] researched multiple open-source serverless platforms and comprehensively compared the features and architecture of each platform in order to identify the most suitable platform for production environments. They further elaborated on their research by also comparing the performance of these platforms when running on a Kubernetes cluster. Another study done by Palade et al. [23] is related to our work as they focus on the performance of open-source serverless computing platform at the edge. The serverless platforms that were compared in that study are Kubeless, OpenFaaS, Knative, and Apache OpenWhisk. However, their study does not take into account the computational limitations of edge devices as the platforms are set up on Desktop grade machines and are used to compute the data that is being generated by edge devices. Their study also evaluates qualitative features of these platforms, such as programming language support, container orchestration engine support, monitoring support and CLI interfaces, and the quantitative evaluation, which measures response time, throughput, and success rate, and does not include resource-intensive tasks. This book chapter provides insights into how a serverless architecture on the edge looks like when we use resource-constrained devices such as Raspberry Pis.

Baresi et al. [5] discuss the adoption of serverless architectures on the edge and propose a new serverless platform that can be suitable specifically for an edge environment. Lloyd et al. [17] discuss the performance of AWS Lambda and Azure Functions and compare it across multiple metrics including latency and the effect of warm and cold starts on serverless functions. Shillaker et al. [28] evaluate the performance of OpenWhisk and evaluate the response times at varying levels of concurrency. However, none of the works analyzed the performance of AWS Lambda function running on the edge using AWS Greengrass and none of the works explicitly compare the performance of both open-source and public cloud serverless platforms that is a significant deciding factor in the viability of serverless platforms on the edge. Moreover, to the best of our knowledge, this is the first work that evaluates the performance of open-source serverless platforms that are set up on resource-constrained edge devices like Raspberry Pis.

In this book chapter, we compare the performance of *AWS Lambda*, *AWS Greengrass* (running local lambda functions), *Azure Functions*, *Apache OpenWhisk*, and *OpenFaaS*. In the following subsections, we briefly describe each of these severless platforms. Table 6.1 also provides the qualitative evaluation of the various serverless platforms that we analyzed in this book chapter.

Table 6.1 Qualitative evaluation of serverless platforms

	OpenFaaS	OpenWhisk	AWS lambda	AWS greengrass	Azure functions
Characteristics	Allows the development of serverless functions in multiple languages and natively supports Docker. Easy deployment. Auto-scaling according to demand. Portable as it runs on existing hardware and can be deployed on both public and private cloud.	Writes functional logic called actions which can be scheduled and triggered via HTTP. Highly scalable and resilient.	Lambda runs your code on high-availability compute infrastructure and performs all the administration of the server and operating system maintenance. Capacity provisioning and automatic scaling. Code and security patch deployment. Code monitoring and logging. Fault tolerance.	Runs AWS lambda functions on the device to respond quickly to local events, interact with local resources, and process data to minimize the cost of transmitting data to the cloud. Support local publish/subscribe messaging between components. Highly secure.	Easily develop and run massively parallel real-time analytics on multiple streams of data—including IoT—using Azure Stream Analytics. With no infrastructure to manage, process data on demand, scale instantly and only pay per job.
Programming languages supported	NodeJS Python, Java, Ruby, PHP, and C#	NodeJS, Go, Java, Scala, PHP, Python, Ruby, and Swift	Java, Go, PowerShell, NodeJS, C#, Python, and Ruby	Java, NodeJS, Python	Java, Python, TypeScript, F#, C#, Powershell, and JavaScript
Intended infrastructure	Kubernetes, docker swarm, extendable to other orchestrators, public cloud supported	No orchestrator required, kubernetes supported, public cloud supported	NA	Docker compose	NA
Virtualization	Docker	Docker	Micro-VM/firecracker	Micro-VM/firecracker	Docker
Triggers	HTTP, faas-cli	HTTP, OW-CLI	S3, SNS, DynamoDB, CloudWatch, config rules, API gateway, Greengrass.	MQTT event, AWS greengrass CLI	Azure event hub and azure storage, web triggering, trigger types, and scheduled types.
Billing	NA	NA	Price according to the memory allocation. \$0.00001667 per GB-second.	Monthly price per device. \$0.16 per month	Price according to the memory allocation. \$0.000016 per GB-second.
Limitations	Does not provide authentication	Open whisk does not have many options for triggering actions since it is a bit difficult to integrate them. Also, there is a problem in the execution of the concurrent systems.	Maximum function runtime is 15 minutes, and the default timeout is 3 seconds which makes it unsuitable for long-running workloads. The payload for each invocation of a Lambda function is limited to 6 MB, and memory is limited to just under 3 GB.	Lack of programming language support currently. Steep learning curve. Limited support for integration with other AWS services.	Maximum function runtime is 10 minutes. Memory usage limit is 1.5 GB. This memory is shared among all functions in the application.
Communication pattern	Request/reply	Request/reply	Request/reply	Publish/subscribe and request/reply	Request/reply
State	Stateless	Stateless	Stateless	Stateless	Stateless

6.3.1 AWS Lambda

Amazon Web Services (AWS) offers their serverless computing service called Lambda. AWS Lambda was the pioneer of public cloud serverless computing services as AWS was the first major cloud computing provider to provide a serverless compute service. AWS Lambda follows the FaaS architecture and allows the developers to focus on their application and not worry about managing infrastructure [32]. AWS Lambda supports multiple programming languages such as JAVA, Go, Python, Ruby, etc., and the cost model is based on the computational expenses of the function execution, i.e., the cost is only accumulated for the amount of time that the function is executed. AWS Lambda also provides its Lambda@Edge service which utilizes the AWS Edge Locations in an attempt to reduce latency for its customers, and the functions are executed closer to the application which results in lower response times. AWS allows Lambda functions to be run on local edge devices using AWS Greengrass service which is discussed in the next section.

6.3.2 AWS Greengrass

AWS Greengrass is a service provided by AWS that allows its customers to run local compute, messaging, and multiple AWS services on a local edge device. AWS Greengrass allows the developer to set up a Greengrass core device that acts as the primary point of connection between the edge setup and the cloud servers. Other devices can either be added as worker nodes for computation or as “Things” that are devices that are able to communicate with the core device. AWS Greengrass primarily uses MQTT [19] protocol to communicate between the core devices and the IoT things that are registered in the AWS Console. Greengrass is relevant to our study because AWS allows compute services such as AWS Lambda to be run on edge devices in order for the computation to be closer to the application and user. We use ARM devices to set up our serverless platforms. Luckily, Greengrass natively supports ARM architecture and allows the Greengrass Core to be set up on a Raspberry Pi along with providing the ability to add multiple Raspberry Pi devices as worker nodes for the core device.

6.3.3 Azure Functions

Microsoft Azure provides Azure Functions as its serverless compute service and follows the same paradigm of serverless computing as discussed previously. The underlying infrastructure is managed by Azure Function providing the developer to focus on their code and building their applications following the serverless development paradigm [15]. Azure Functions use authentication to protect system

from intruders and unauthorized access. Azure Functions is a relatively new service, so there are fewer studies that have analyzed its performance and compared it to other public cloud serverless offerings.

6.3.4 Apache OpenWhisk

Apache OpenWhisk is an open-source serverless platform currently in incubation. IBM Cloud has built their IBM Cloud Functions on top of Apache OpenWhisk utilizing the power of OpenWhisk's services. OpenWhisk has multiple components and relatively consumes more resources due to the added components [26]. For the purposes of this research, we will be utilizing the Lean OpenWhisk branch of the Apache OpenWhisk platform as it is less resource intensive and more suitable for edge environments compared to its full version. Lean OpenWhisk supports the same number of languages as its master branch but is not natively compatible with the ARM architecture. OpenWhisk can be run on Kubernetes or Docker compose [14]. For the purposes of this study, we will be setting up Lean OpenWhisk on Docker compose as it consumes less system resources. In order for it to be compatible, custom Docker images of the Lean OpenWhisk platform have to be created for the ARM architecture as it currently only supports x64 and x86 architectures.

6.3.5 OpenFaaS

OpenFaaS is a lightweight open-source serverless platform that provides native compatibility with the ARM architectures and is easy to set up and start running production workloads. OpenFaaS is compatible with multiple system architectures and can be run on top of Kubernetes and Docker Swarm. OpenFaaS follows the same serverless compute paradigm as the other serverless platforms and provides services for resource and performance monitoring and supports multiple programming such as Python, JavaScript, Ruby, etc. *Faasd* is a variant of OpenFaaS that utilizes Containerd as the container orchestration system and is lighter than OpenFaaS. However, faasd does not support setup on multiple Raspberry Pi devices in a cluster which is why we use OpenFaaS in our experiments.

6.4 Performance Evaluation

We analyze the performance metrics including response times and success rate for the requests measured for each of the aforementioned platforms for the sake of comparison. The response times consist of the time it takes a request to reach the server, the time to execute the function on the server, and the time it takes

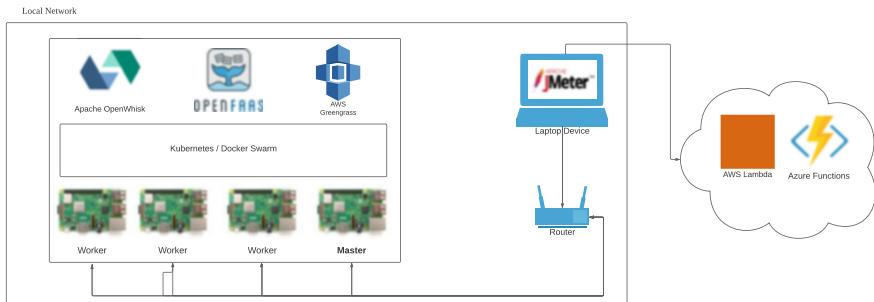


Fig. 6.2 Deployment setup

the response to be delivered back to the client. Success rate measures as the percentage of requests that were executed successfully. As displayed in Fig. 6.2, the local functions are run on a cluster of 4 Raspberry Pis, whereas calls are made to OpenFaas, OpenWhisk, and AWS Greengrass in our serverless setup on the local network and also to the cloud functions in AWS Lambda and Azure Functions from the same test machine. This allows us to compare the performance of running local functions as compared to cloud functions which will be a good deciding factor on the viability of running serverless workloads on the edge or running the workloads on the cloud.

6.4.1 Experimental Setup

We run the experiments on a cluster of four Raspberry Pis, with our requests being generated by a test machine running JMeter on the same local network. For the purpose of accuracy, the experiments are run on an isolated network so that there is no other network overhead that interferes with the network. These tests have been carried in the city of Melbourne, Australia. The deployment of each platform is done individually in order to make sure no other services are consuming the system resources. This is especially important due to the limited resources available on the edge devices and compute power plays an essential role in our results. We use Raspberry Pi Model 3B+ devices running Raspberry Pi OS which is a Linux distribution designed specifically for Raspberry Pi devices.

6.4.1.1 Testbed Setup

Each Raspberry Pi has a 1.4 GHz 64-bit quad-core ARM processor and 1 GB RAM. Unless stated otherwise, we used the default deployment settings for each platform,

Table 6.2 Ping results from the test machine

Destination	Ping
Raspberry pi cluster (local network)	4 ms
AWS Sydney region (ap-southeast-2)	42 ms
Azure Sydney region (Australia east)	51 ms

for example, Docker Swarm as the container orchestration system for OpenFaas.⁶ We use Apache’s Lean OpenWhisk offering that utilizes no Kafka, Zookeeper, and no Invokers as separate entities which is suitable for the ARM architecture. The Lean OpenWhisk branch of the Apache OpenWhisk project does not provide compatibility with the ARM architecture of the Raspberry Pi. Each component of OpenWhisk runs as a Docker container and these Docker images have been configured for the ARM architecture in this GitHub repository.⁷ We created the ARM compatible Docker images to set up OpenWhisk on each Raspberry Pi in the cluster. This setup does not support multiple Raspberry Pi devices in a cluster, so we use an NGINX Load Balancer to distribute our HTTP requests to the Raspberry Pi cluster.

We use AWS Greengrass to set up our cluster which involves setting up a Greengrass Group on the AWS Management Console and an AWS Greengrass IoT Core on the master Raspberry Pi. The other Raspberry Pi devices are registered as non-core devices and communicate with the core over localhost. The region that the remote Lambda functions are running on Greengrass is set as *ap-southeast-2*, i.e., *Asia Pacific (Sydney) region*. For our AWS Lambda setup, the region is set as *ap-southeast-2*, i.e., *Asia Pacific (Sydney, New South Wales) region* in order to maintain consistency and minimize network latency. For our Azure Functions setup, the region is set to *Australia East* which is also the *New South Wales region*. Azure does offer the Australia South-East region that is the Victoria region, but for the purposes of comparison with AWS Lambda, the Australia East region is selected. Table 6.2 shows the average ping results from the test machine to the AWS and Azure servers in Sydney and also the ping results to the Raspberry Pis on the local cluster. This provides us with the time associated with network latency in our experiments.

We use Apache JMeter version 5.3 to generate HTTP requests that invoke the functions deployed on each platform. We run JMeter on a Windows Machine that has a quad-core Core i7 CPU and 8GB RAM. This machine is set up on the same local network as the Raspberry Pi Cluster in order to minimize latency. The JMeter tool is set up to send 1000 requests with different levels of concurrency (5, 10, and 15). These concurrent requests affect the number of simultaneous requests received by each platform. These concurrency levels were decided based on the compute power available to the platforms, and each experiment is replicated 3 times in

⁶ OpenFaas has switched to Kubernetes as the officially supported orchestration system. This update was provided by OpenFaas after the experiments were conducted in this book chapter.

⁷ <https://github.com/kpavel/incubator-openwhisk>.

order to maintain statistical accuracy. Specifically, for AWS Greengrass, we use the JMeter MQTT plugin which extends the JMeter capability to send MQTT requests instead of HTTP requests as AWS Greengrass requires the user to send MQTT requests.

6.4.1.2 Test Functions

The three types of functions that we use to test our serverless platforms are CPU-intensive function, memory-intensive functions, and disk-intensive functions. We use JavaScript for our functions as it is supported by all of our test platforms. The workloads designed for testing these platforms are representative of the various types of workloads that are generated in an edge environment. Due to the lack of research on serverless platforms' performance on resource-constrained devices, this workload has been designed to test how each platform performs when subjected to an emulated workload. In order to simulate a CPU-intensive function, we multiply two square matrices of 128 by 128 dimensions, which has a complexity of $O(n^3)$. The execution time of this function varies on the computational power available to the compiler. For the memory-intensive function, we utilize the *memoize()* function of JavaScript that allows us to store the expensive results of our functions in the cache and retrieve them quickly from the cache if the same result occurs [6].

To simulate a disk-intensive workload, we created a function to unzip 10 zipped files of size 2.5 MB onto the disk. This utilizes both read and write operations for the function. The disk-intensive function requires an extensive setup especially for the cloud functions that we are executing using AWS Lambda and Azure Functions. This is because these services are event-driven and make use of ephemeral containers that are stateless and do not have access to file system that can be shared across all other functions. While a temporary file system for the individual function is included, this ephemeral storage is not intended for durable storage which is why need to attach a shared file system for the functions to access shared data. For AWS Lambda, we created an EFS File System on the AWS Console and provided read/write permissions for Lambda. Then we added the EFS File System in the Lambda configuration to as Local Mount Path. Azure Functions follows a similar procedure to AWS Lambda. We used Azure Blob Storage to provide file access to our Azure Function. For the other platforms, we placed the zip files on each Raspberry Pi in the cluster as the devices do not share a file system. In future, we would like to test these platforms on actual IoT application workload traces in order to gain better insight on the performance of each serverless platform.

6.4.2 Results

In order to compare the performance of each platform, we measure the response times of each request under different levels of load. The number of concurrent

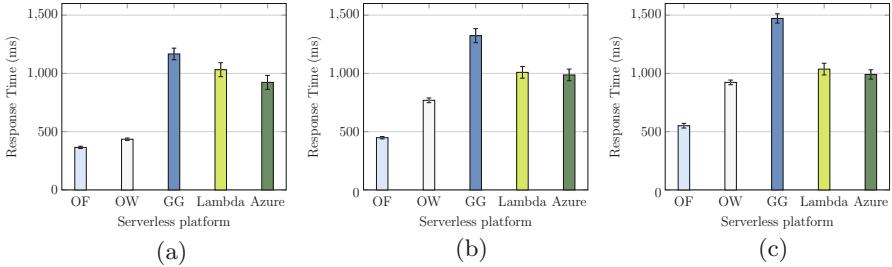


Fig. 6.3 Median response time with standard error bars on CPU-intensive tasks for OpenFaaS (OF), OpenWhisk (OW), AWS Greengrass (GG), AWS Lambda (Lambda), and Azure Functions (Azure) with a various number of concurrent users. **(a)** 5 concurrent users. **(b)** 10 concurrent users. **(c)** 15 concurrent users

requests directly affects the performance of platforms which alludes to the number of function instances being created in order to handle the requests. The function instances consume the system resources, and we want to measure the effect of this on the performance of each platform. Our main aim is to find out the performance issues for each platform and how it handles different types of serverless workloads, i.e., CPU-intensive, memory-intensive, and disk-intensive functions.

6.4.2.1 CPU-Intensive Functions

Figure 6.3 shows the median response times for each platform for the 1000 requests that we sent under different concurrency. These response times include the time taken to send the request, the time taken to execute the function, and the time taken to receive the result to our test machine. This is a synchronous request, so it waits for the function to be executed and then receives the response. In each of the results, the lowest response time recorded is for the OpenFaaS platform. We observe that the lowest response time is when we send 5 concurrent requests to OpenFaaS with values around 400 ms. OpenWhisk performs fairly similarly with values around 450ms. These response times increase as we increase the number of concurrent requests to 10 where OpenWhisk is impacted considerably more than OpenFaaS as the response time increases to 750 ms. OpenFaaS response time rises to 448 ms when dealing with 10 concurrent requests. The response times for OpenWhisk rise even more when subject to 15 concurrent requests as the response time is 923 ms. As compared to OpenFaaS, the response time remains fairly stable when we increase the concurrent requests.

As shown in Table 6.3, OpenWhisk records the worst success rate in our experiments for CPU-intensive workloads. We recorded 8 failed requests out of 1000 as a result of an increase in the system's resource consumption. OpenFaaS records a perfect success rate with no failed requests deducing that it handles CPU-intensive workloads very well under various loads. These results show that

Table 6.3 Request success rate for platforms under different workloads

	OpenFaaS	OpenWhisk	AWS greengrass	AWS lambda	Azure functions
CPU	100%	99.1%	100%	100%	100%
Memory	99.5%	99.3%	100%	100%	100%
Disk	99.9%	99.6%	100%	100%	100%

performance of OpenWhisk decreases significantly as we increase the number of concurrent requests and the reason for that is because OpenWhisk consumes more compute power to run its components as compared to OpenFaaS which is more lightweight. If we analyze the response times for AWS Greengrass, we can compare that the rise in latency is not that significant when we increase the number of concurrent requests accordingly. However, AWS Greengrass records the highest response times out of all the platforms, due to the time it takes for the request to reach the AWS server to trigger the function on the core device. AWS Lambda and Azure Functions perform very similarly for each level of concurrency and show very slight changes in latency which is predictable as these platforms do not have as limited resources as compared to the other platforms and provide highest level of scalability. For our CPU-intensive functions, AWS Greengrass, AWS Lambda, Azure Functions, and OpenFaaS do not have a single failed request as compared to OpenWhisk.

6.4.2.2 Memory-Intensive Functions

Figure 6.4 demonstrates the results for the performance of each serverless platform for memory-intensive functions. The response times were recorded for each platform until varying concurrent requests in order to observe the effect it has on the performance. The lowest response times, in each of the tests for memory-intensive functions, were for Apache OpenWhisk that performed considerably better than the other platforms, and the results were quite stable with only a gradual increase in response time when we increased the number of concurrent requests. The lowest median response time for OpenWhisk is 416 ms for 5 concurrent requests, and it reaches a maximum of 621 ms for 15 concurrent requests. OpenFaaS, however, demonstrated higher response times than OpenWhisk for memory-intensive functions, but the response time varied less between different concurrent requests which proposes that this increase in latency is less likely to be caused by OpenFaaS ability to scale and is more likely attributed to memory management for function execution in the platform.

The highest response times were for AWS Greengrass that maintained its values between 1400 and 1500 ms between varying concurrent requests. We observed that AWS Greengrass took more time for memory-intensive functions than CPU-intensive functions but is affected less by change in concurrent results. In fact, there is only a 13 ms difference between the results we send 5 concurrent requests and 10 concurrent requests, and Greengrass only witnesses a slight increase in latency

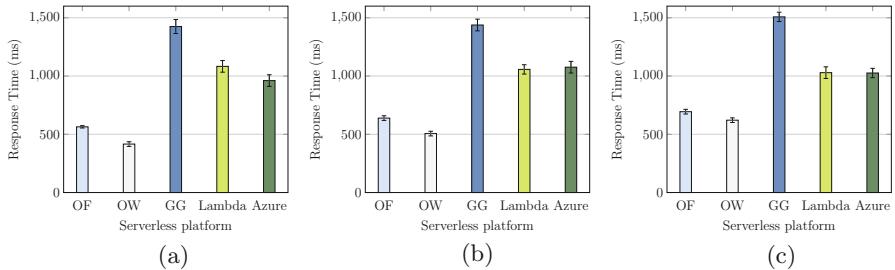


Fig. 6.4 Median response time with standard error bars on memory-intensive tasks for OpenFaas (OF), OpenWhisk (OW), AWS Greengrass (GG), AWS Lambda (Lambda), and Azure Functions (Azure) with a various number of concurrent users. **(a)** 5 concurrent users. **(b)** 10 concurrent users. **(c)** 15 concurrent users

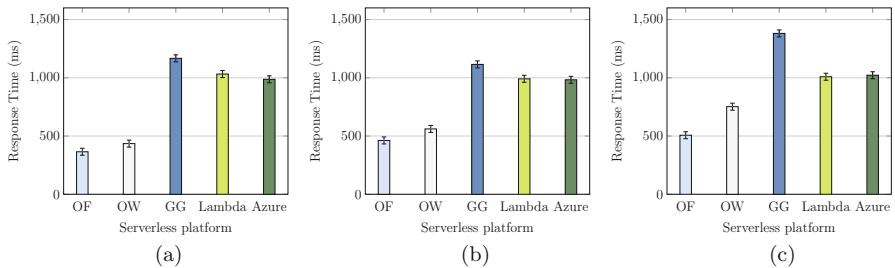


Fig. 6.5 Median response time with standard error bars on disk-intensive tasks for OpenFaas (OF), OpenWhisk (OW), AWS Greengrass (GG), AWS Lambda (Lambda), and Azure Functions (Azure) with a various number of concurrent users. **(a)** 5 concurrent users. **(b)** 10 concurrent users. **(c)** 15 concurrent users

when we increase the concurrent requests to 15. Both AWS Lambda and Azure Functions perform very similarly with little changes in latency when we increase the number of concurrent requests. Both of the platforms provided similar results to our CPU-intensive tasks. This is attributed to both of the cloud serverless offerings' ability to scale indefinitely and also due to the higher computational power available to the functions. However, as the functions are not being executed locally and need to communicate with the cloud servers, the latency is much higher as compared to the platforms that are set up locally.

6.4.2.3 Disk-Intensive Functions

Figure 6.5 shows the results for the evaluation of each platform for disk-intensive functions. The lowest response times for this experiment are recorded by OpenFaas and the platform scaling very well when we increased the number of concurrent requests. The response time only increased by 50 ms between 10 and 15 concurrent requests, which demonstrates the new function replicas were able to handle the increased load easily. However, we noticed that OpenWhisk recorded higher

response times as compared to OpenFaaS, and when we increased the load from 10 to 15 concurrent requests, it caused an increase in response time of 200 ms. The success rates for OpenFaaS and OpenWhisk were also very acceptable as we recorded very few failed requests as compared to the other function types. AWS Greengrass showed similar trends to OpenWhisk as the response times for 5 and 10 concurrent requests are very similar but an increase in load to 15 requests caused the latency to increase by 220 ms. This can be attributed to the creation of new function replicas in both cases of OpenWhisk and AWS Greengrass. AWS Lambda and Azure Functions both displayed similar results for this experiment, as we did not notice any changes in latency caused due to an increase in load or the function type. The file storages for both platforms were configured on the same server as the functions, which shows us that there is minimum latency for disk read and write operations. The results for the success rates for AWS Greengrass, AWS Lambda, and Azure Functions were the same as the previous experiments as they recorded perfect success rates with no failed requests.

6.5 Discussions

Our experimental results show that OpenFaaS performs considerably better on the edge than the other platforms. We can attribute this to the lean architecture of the platform which suits very favorably to edge devices and consumes fewer resources which is very suitable for edge devices. Along with that, it provides native support for the ARM architecture, along with the ability to scale easily by adding more Raspberry Pi devices. However, the results vary for high loads especially for CPU-intensive and memory-intensive workloads. We found out that the performance of Apache OpenWhisk is inferior to OpenFaaS in our setup as OpenWhisk does not provide full support for the ARM architecture and the lean version restricts its performance when handling increased workloads. This performance bottleneck may be attributed the high memory usage of the platform that leads to resource starvation for new containers. Furthermore, the ability to add more worker nodes with ease is an integral part of edge computing setup, which OpenWhisk does not support natively.

AWS Greengrass performed very consistently across our experiments, but the increased response times are a considerable factor when making the decision to deploy this platform on the edge. The cause of these increased response times is the added latency of the requested being routed to the AWS cloud servers. The increased latency for CPU-intensive tasks also suggests that the functions take longer to compute due to resource starvation. AWS Greengrass seamlessly extends AWS to edge devices, so they can act locally on the data they generate, while still using the cloud for management, analytics, and durable storage. It enables the connected devices to run AWS Lambda functions, keep device data in sync, and communicate with other devices securely. AWS Greengrass provides support for ARM architecture and also provides the ability to scale out by adding more

worker devices. AWS Greengrass provides an opportunity for further research to be conducted by testing its performance in an isolated edge without connectivity to the cloud.

If we want consistent performance regardless of the amount of workload, the cloud serverless offerings of AWS Lambda and Azure Functions provide a stable service and can scale according to the increased load. This demonstrates that for workloads that require increased computational capacity, along with the ability to handle workloads at an increased scale, the cloud serverless offerings are very suitable. However, the overall latency to cloud is higher than the edge devices as experiments show and running serverless platforms such as OpenFaaS on the edge provides promising prospects to meet the stringent latency requirements of emerging real-time applications such as autonomous vehicles. Nevertheless, our work provides the groundwork for comprehensively benchmarking the performance of serverless platforms on the edge, specifically on devices with ARM architecture.

6.6 Conclusions and Future Work

Performing serverless computations on the edge proposes numerous advantages in an event-driven architecture, allowing us to generate more data on the edge without having to worry about managing its applications. This book chapter analyzed the performance of serverless platforms on the edge, specifically on ARM architecture edge devices such as Raspberry Pis. We performed comprehensive performance tests on the compatible serverless platforms of OpenFaaS, Apache OpenWhisk, and AWS Lambda functions running locally in the AWS Greengrass platform on edge devices and compared their performance to cloud serverless offerings of AWS Lambda and Azure Functions to research the viability of setting up a serverless architecture in an edge environment. Based on that, the results demonstrated that OpenFaaS is the most suitable platform for an edge setup as it provides a lightweight architecture with support for simple and rapid scaling. We used the metrics of response time and success rate for each platform to compare the performance on how each platform would cope in an edge setup. AWS Greengrass provides a promising opportunity in this environment for further research due to its native support for ARM architecture and its development support from AWS. For future research, we aim to analyze other suitable serverless platforms and further research the viability of serverless architecture on the edge.

Bibliography

1. M.S. Aslanpour, A.N. Toosi, R. Gaire, M.A. Cheema, Auto-scaling of web applications in clouds: a tail latency evaluation, in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)* (2020), pp. 186–195. <https://doi.org/10.1109/UCC48980.2020.00037>

2. M.S. Aslanpour, A.N. Toosi, C. Cicconetti, B. Javadi, P. Sbarski, D. Taibi, M. Assuncao, S.S. Gill, R. Gaire, S. Dustdar, Serverless edge computing: vision and challenges, in *2021 Australasian Computer Science Week Multiconference, ACSW'21* (Association for Computing Machinery, New York, 2021). <https://doi.org/10.1145/3437378.3444367>
3. M.S. Aslanpour, A.N. Toosi, J. Taheri, R. Gaire, AutoScaleSim: a simulation toolkit for auto-scaling web applications in clouds. *Simul. Model. Pract. Theory* **108**, 102245 (2021). <https://doi.org/10.1016/j.simpat.2020.102245>.
4. I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, et al., Serverless computing: current trends and open problems, in *Research Advances in Cloud Computing* (Springer, Berlin, 2017), pp. 1–20
5. L. Baresi, D.F. Mendonça, Towards a serverless platform for edge computing, in *2019 IEEE International Conference on Fog Computing (ICFC)* (IEEE, Piscataway, 2019), pp. 1–10
6. D. Berube, Speeding up function calls with memoize, in *Practical Ruby Gems* (Apress, New York, 2007), pp. 215–220
7. P. Castro, V. Ishakian, V. Muthusamy, A. Slominski, Serverless programming (function as a service), in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (IEEE, Piscataway, 2017), pp. 2658–2659
8. R. Chard, T.J. Skluzacek, Z. Li, Y. Babuji, A. Woodard, B. Blaiszik, S. Tuecke, I. Foster, K. Chard, Serverless supercomputing: high performance function as a service for science (2019, preprint). arXiv:1908.04907
9. S. Eismann, J. Scheuner, E. van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. Abad, A. Iosup, Serverless applications: why, when, and how? *IEEE Softw.* **38**, 32–39 (2020). <http://dx.doi.org/10.1109/MS.2020.3023302>
10. A. Glikson, S. Nastic, S. Dustdar, Deviceless edge computing: extending serverless computing to the edge of the network, in *Proceedings of the 10th ACM International Systems and Storage Conference* (2017), p. 1
11. A. Hall, U. Ramachandran, An execution model for serverless functions at the edge, in *Proceedings of the International Conference on Internet of Things Design and Implementation* (2019), pp. 225–236
12. J.M. Hellerstein, J. Faleiro, J.E. Gonzalez, J. Schleier-Smith, V. Sreekanti, A. Tumanov, C. Wu, Serverless computing: one step forward, two steps back (2018, preprint). <https://doi.org/10.48550/arXiv.1812.03651>
13. M.K. Hussein, M.H. Mousa, M.A. Alqarni, A placement architecture for a container as a service (CaaS) in a cloud environment. *J. Cloud Comput.* **8**(1), 7 (2019)
14. A. Kuntsevich, P. Nasirifard, H.A. Jacobsen, A distributed analysis and benchmarking framework for Apache openwhisk serverless platform, in *Proceedings of the 19th International Middleware Conference (Posters)* (2018), pp. 3–4
15. A. Kurniawan, W. Lau, Introduction to azure functions, in *Practical Azure Functions* (Springer, Berlin, 2019), pp. 1–21
16. H. Lee, K. Satyam, G. Fox, Evaluation of production serverless computing environments, in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (IEEE, Piscataway, 2018), pp. 442–450
17. W. Lloyd, S. Ramesh, S. Chinthalapati, L. Ly, S. Pallickara, Serverless computing: an investigation of factors influencing microservice performance, in *2018 IEEE International Conference on Cloud Engineering (IC2E)* (IEEE, Piscataway, 2018), pp. 159–169
18. S.K. Mohanty, G. Premsankar, M. Di Francesco, et al., An evaluation of open source serverless computing frameworks, in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (2018), pp. 115–120
19. MQTT: Mqtt protocol docs (2020). <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>
20. S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, R. Prodan, A serverless real-time data analytics platform for edge computing. *IEEE Internet Comput.* **21**(4), 64–71 (2017)

21. H.D. Nguyen, C. Zhang, Z. Xiao, A.A. Chien, Real-time serverless: enabling application performance guarantees, in *Proceedings of the 5th International Workshop on Serverless Computing* (2019), pp. 1–6
22. J. Nupponen, D. Taibi, Serverless: what it is, what to do and what not to do, in 2020 IEEE International Conference on Software Architecture Companion (ICSA-C) (IEEE, Piscataway, 2020), pp. 49–50
23. A. Palade, A. Kazmi, S. Clarke, An evaluation of open source serverless computing frameworks support at the edge, in 2019 IEEE World Congress on Services (SERVICES), vol. 2642 (IEEE, Piscataway, 2019), pp. 206–211
24. D. Pinto, J.P. Dias, H.S. Ferreira, Dynamic allocation of serverless functions in IoT environments, in 2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC) (IEEE, Piscataway, 2018), pp. 1–8
25. G. Premsankar, M. Di Francesco, T. Taleb, Edge computing for the internet of things: a case study. *IEEE Internet Things J.* **5**(2), 1275–1284 (2018)
26. S. Quevedo, F. Merchán, R. Rivadeneira, F.X. Dominguez, Evaluating apache openwhisk-FaaS, in 2019 IEEE Fourth Ecuador Technical Chapters Meeting (ETCM) (IEEE, Piscataway, 2019), pp. 1–5
27. W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges. *IEEE Internet Things J.* **3**(5), 637–646 (2016)
28. S. Shillaker, A provider-friendly serverless framework for latency-critical applications, in 12th *Eurosys Doctoral Workshop* (2018), p. 71
29. D. Taibi, N. El Ioini, C. Pahl, J. Niederkofler, Patterns for serverless functions (function-as-a-service): a multivocal literature review, in *CLOSER 2020 - Proceedings of the 10th International Conference on Cloud Computing and Services Science*, ed. by D. Ferguson, M. Helfert, C. Pahl, vol. 1 (SCITEPRESS, 2020), pp. 181–192. <https://doi.org/10.5220/0009578501810192>
30. E. Van Eyk, L. Toader, S. Talluri, L. Versluis, A. Uță, A. Iosup, Serverless is more: from PaaS to present cloud computing. *IEEE Internet Comput.* **22**(5), 8–17 (2018)
31. B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, D.S. Nikolopoulos, Challenges and opportunities in edge computing, in 2016 IEEE International Conference on Smart Cloud (SmartCloud) (IEEE, Piscataway, 2016), pp. 20–26
32. M. Villamizar, O. Garces, L. Ochoa, H. Castro, L. Salamanca, M. Verano, R. Casallas, S. Gil, C. Valencia, A. Zambrano, et al., Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures, in 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) (IEEE, Piscataway, 2016), pp. 179–182
33. W. Yu, F. Liang, X. He, W.G. Hatcher, C. Lu, J. Lin, X. Yang, A survey on the edge computing for the internet of things. *IEEE Access* **6**, 6900–6919 (2017)

Chapter 7

ITL: An Isolation-Tree-Based Learning of Features for Anomaly Detection in Networked Systems



Sara Kardani Moghaddam, Rajkumar Buyya, and Kotagiri Ramamohanarao

Contents

7.1	Introduction	186
7.2	Related Work	188
7.3	Model Assumptions and an Overview on Isolation-Based Anomaly Detection	190
7.4	ITL Approach	192
7.4.1	Feature Refinement Process	194
7.5	Performance Evaluation	196
7.5.1	Experimental Settings	197
7.5.2	Experiment Results	198
7.5.3	Strength and Limitations of ITL Approach	204
7.6	Conclusions and Future Work	205
	References	206

With the advances in monitoring techniques and storage capability in the cloud, a high volume of valuable monitoring data is available. The collected data can be used for profiling applications behavior and detecting anomalous events that identify unexpected problems in the normal functioning of the system. However, the fast-changing environment of the cloud brings a need for fast and efficient analytic solutions to monitor the cloud system for its correct operational behavior. The isolation-based method is an effective approach for detecting anomalies. This method randomly samples the data and builds several isolation-trees (iTrees) data structures to find anomalous records. However, a common challenge of iTrees as well as other anomaly detection algorithms is dealing with high-dimensional data that can impact the accuracy and execution time of the process. This is an important issue for cloud-hosted applications where a variety of problems are constantly changing the normal pattern of features from low-level network data to high-level

S. Kardani-Moghaddam · R. Buyya (✉) · K. Ramamohanarao
The University of Melbourne, Melbourne, VIC, Australia
e-mail: skardani@student.unimelb.edu.au; rbuyya@unimelb.edu.au

application performance. Therefore, refining the feature space for the removal of irrelevant attributes is a critical issue.

In this chapter, we introduce an iterative iTree based Learning (ITL) algorithm to handle high-dimensional data. ITL takes the advantage of iTree structure to learn relevant features for detecting anomalies. Initially, it builds iTrees making use of all features of the data. Then, in the iterative steps, it refines the set of the features to find the most relevant ones by selecting highly ranked anomalies discovered in the previous iteration. Experiments are conducted to validate the performance of our proposed ITL method on several benchmark datasets. The results show that ITL can achieve significant speedups with the appropriate choice of the number of iTrees while achieving or exceeding the area under the curve (AUC) values of other state-of-the-art isolation-based anomaly detection methods.

7.1 Introduction

Anomaly detection is an important field of knowledge discovery with rapid adoption in a variety of applications. The main goal is to find interesting patterns in the data that deviate from the expected behavior of the application. In the context of the cloud environment, this process is utilized for a variety of performance management applications. For example, intrusion detection systems provide frameworks that monitor the performance of the network to find misbehaving users, possible misconfiguration, or serious conditions from an attack on the system [1]. Similarly, SMART (Self-Monitoring, Analysis, and Reporting Technology)-based disk failure prediction applications perform regular monitoring and anomaly detection analysis to increase the reliability of storage systems [2].

With the advances in data collection techniques, storage capabilities, and high-performance computing, a huge volume of monitoring data is collected from continuous monitoring of the system attributes. Despite the appealing benefits of access to larger amounts of data for better diagnostics of anomalous events, the great challenge is how to deal with a high volume of information that should be processed effectively in real time. The increase in the volume of data is due to (1) recording of fine-grained measurements for long periods which increases the number of records to be processed and (2) high-dimensional data with many features that describe various aspects of the target system. The curse of dimensionality or having many features can make the problem of anomaly detection in high-dimensional data more complex in terms of the runtime efficiency and accuracy [3]. This is also becoming a critical issue in cloud systems which are exposed to several performance problems at different layers of computing. As a result, the collected performance data is heterogeneous and includes a variety of attributes from low-level operation logging data to hardware-specific features, application performance data, or network-related information. On the other hand, these performance data are exposed to a variety of problems such as different types of attacks and intrusion patterns in network-related performance data. Particularity, the general anomaly detection techniques cannot

perform well for high-dimensional network data with a variety of data types and embedded meaningful subspaces from different sources [4]. Moreover, the collected data is dynamic and rapidly changing. All of these, together, highlight the need for highly adaptable and fast analytic solutions. Therefore, researchers are investigating more efficient techniques with the goal of better explorations of collected data and improving the quality of the extracted knowledge.

Traditional anomaly detection algorithms usually work based on the assumptions that highly deviated objects in terms of the common metrics such as distance or density measures have a higher probability of being anomalous (outliers in statistical methods). While these assumptions are applicable in general, their accuracy can be affected when the base assumptions do not hold, such as in the high-dimensional data [5, 6]. Moreover, in the traditional methods, anomalies are detected as a by-product of other goals such as classification and clustering. More recent approaches, such as isolation-based methods, directly target the problem of anomaly detection with the assumption that anomalies are few and different [7, 8]. However, the problem of having a high number of noisy features can also affect these methods. To improve the efficiency of detection algorithms in high-dimensional data, a variety of solutions such as random feature selection or subspace search methods are proposed [9, 10]. However, the proposed approaches are usually considered as the preprocessing steps which are performed as a separate process from the anomaly detection. Although this separation makes them applicable for a variety of algorithms, finding the relevant features in the datasets with many noisy features can be challenging when the mechanism of target detection algorithms in finding the anomalies is ignored. Therefore, a question arises that is there a way that one can improve the efficiency of anomaly detection by extracting knowledge from the assumptions and the process which leads to identifying potential anomalies in the data? Having this question in mind, in this chapter, we address the problem of anomaly detection in high-dimensional data by focusing on the information that can be extracted directly from the isolation-based mechanism for identifying anomalies. The reason for selecting this technique as the base process is that it is designed to directly target the most common characteristics of anomalous events such as rarity compared to other objects. We exploit the knowledge that comes from the detection mechanism to *identify* the features that have higher contribution in the separation of the anomaly instances from normal ones. This approach helps to identify and remove many irrelevant noisy features in high-dimensional data. The proposed method, Isolation-Tree (iTree)-Based Learning (ITL), addresses the problem of anomaly detection in high-dimensional data by refining the set of features to improve the efficiency of the detection algorithm. These are the features that appear in the short branches of iTrees. The refining procedure helps the algorithm to focus more on the subset of features where the chances of finding anomalies are higher while reducing the effect of noisy features. The process helps to obtain more informative anomaly scores and generates a reduced set of features that improves the detection capability with better runtime efficiency in comparison to the original method that uses all the features.

Accordingly, the major contributions of this work are as follows:

- Proposed an iterative mechanism for structural learning of data attributes and refining features to improve the detection efficiency of isolation-based methods.
- Exploited the inherent knowledge from iTree data structure to reduce the effect of noisy and irrelevant features.
- Efficiently found an essential subset of the features that can effectively detect anomalies. The simplified model is extremely fast to train so that the model can be periodically trained when the important features largely remain unchanged.
- Carried out experiments on network intrusion datasets and other high-dimensional benchmark datasets to demonstrate the effectiveness of ITL in improving the anomaly detection results as well as runtime efficiency.

ITL focuses on the data engineering part of data analytics which also helps to speed up the process of anomaly detection. We have compared ITL with the state-of-the-art feature learning-based framework [11] and show that not only ITL improves results as an ensemble learning method with the bagging of scores, but also it can discover a subset of the features that can detect anomalies with reduced complexity. Since anomalies can be different and dependent on the context of the datasets, we have considered the heterogeneity by analyzing different benchmark datasets with a variety of attributes and anomaly patterns.

The remainder of this chapter is structured as follows. Section 7.2 reviews some of the related works in the literature. Section 7.3 overviews the basic idea of isolation-based anomaly detection and the main assumptions in the problem formulation. Section 7.4 presents the ITL framework and details the steps of the algorithm. Section 7.5 presents experiments and results followed by time complexity and runtime analysis, and finally, Sect. 7.6 concludes the chapter with directions for future research.

7.2 Related Work

The general concept of anomaly detection indicates the exploration and analysis of data to find patterns that deviate from the expected behavior. The concept has been widely used and customized in a range of applications such as financial analysis, network analysis and intrusion detection, medical and public health, etc. [1, 12, 13]. The growing need for anomaly related analysis has led researchers to propose new ways of addressing the problem where they can target unique characteristics of anomalous objects in the context of the target applications. For example, distance-based algorithms address the problem of anomaly detection based on the distance of each instance from neighborhood objects. The greater the distance, the more likely it that the instance presents abnormal characteristics in terms of the values of the features [14, 15]. Alternatively, [16, 17] define the local density as a measure for abnormality of the instances. The objects with a low density in their local regions have a higher probability of being detected as an anomaly. Ensemble-based methods

try to combine multiple instances of anomaly detection algorithms to improve the searching capability and robustness of the individual solutions [11, 18].

Performance anomaly detection has also widely been applied in the context of cloud resource management to identify and diagnose performance problems that affect the functionality of the system. These problems can happen at different levels of granularity from code-level bug problems to hardware faults and network intrusions. The fast detection of the problem is a critical issue due to the high rate of changes and volume of information from different sources. A variety of techniques from statistical analysis to machine learning solutions are used to process collected data. For example, Principal Component Analysis (PCA) is used in [19] to identify the most relevant components to various types of faults. Xiao et al. [2] applies random forests on various exported attributes of drive reliability to identify disk failures. Dean et al. [20] exploits the self-organizing map technique to proactively distinguish anomalous events in virtualized systems. Clustering techniques are utilized by [21] to split the network-related log data into distinctive categories. The generated clusters are then analyzed separately by anomaly detection systems to identify intrusion and attack events. Cao et al. [22] uses entropy concept on network and resource consumption data to identify denial of service attacks.

While the abovementioned approaches show promising results for a variety of problems, the exploding volume and speed of the data to be analyzed require complex computations which are not time efficient. A common problem that makes these difficulties even more challenging is the high-dimensional data. For example, the notion of distance among objects loses its usability as a discrimination measure as the dimension of data increases [3, 5]. Methods based on the subspace search or feature space projections are among approaches that are proposed as possible solutions for these problems [23]. The idea of dividing high-dimensional data into groups of smaller dimensions with related features is investigated in [24]. This approach requires a good knowledge of the domain to define meaningful groups. PCA-based methods try to overcome the problem by converting the original feature set to a smaller, uncorrelated set which also keeps as much of the variance information in data as possible [25]. PINN [26] is an outlier detection strategy based on the Local Outlier Factor (LOF) method which leverages random projections to reduce the dimensionality and improve the computational costs of LOF algorithm. Random selection of the features is used in [27] to produce different subspaces of the problem. The randomly generated sub-problems are fed into multiple anomaly detection algorithms for assigning the anomaly scores. While random selection can improve the speed of the feature selection process, as the selection is completely random there is no guarantee of having informative subspaces of data to improve the final scoring. A combination of correlation-based grouping and kernel analysis is applied in [28] for feature selection and anomaly detection in time-series data. Feature reduction is done by selecting representative features of final clusters. Keller et al. [9] and Pang et al. [11] propose two different variations of subspace searching. The former tries to find high contrast subspaces of the problem to improve the anomaly ranking of density-based anomaly detection algorithms. The subspace searching is based on the statistical features of the attributes and is performed as

a preprocessing step separated from target anomaly detection algorithms. The latter, in contrast, integrates the subspace searching as a sequential refinement and learning in anomaly detection procedure where the calculated scores are used as a signal for the selection of the next subset of the features. Our proposed anomaly detection approach is inspired by such models and tries to refine the subset of the selected features at each iteration. However, we try to take advantage of the knowledge from the structure of constructed iTrees instead of building new models for the regression analysis.

7.3 Model Assumptions and an Overview on Isolation-Based Anomaly Detection

The iterative steps in the ITL process are based on the iTree structure for assigning the anomaly scores as well as identifying features. We choose the isolation-based approach and specifically IForest algorithm [7, 29] in this work due to its simplicity and the fact that they target the inherent characteristic of the anomalies as being rare and different without any prior assumption on their distributions. We note that the target types of the anomaly in this work are instances that are anomalous in comparison to the rest of the data and not as a result of being part of the larger groups [1, 30]. This is also consistent with the definition of anomaly in many cloud-related performance problems especially network and resource abnormalities.

The idea of Isolation-based methods is that for an anomaly object we can find a small subset of the features that their values are highly different compared to the normal instances, and therefore it can quickly be isolated in the feature space of the problem. IForest algorithm demonstrates the concept of the isolation and partitioning of the feature space through the structure of trees (iTrees), where each node represents a randomly selected feature with a random value and existing instances create two new child nodes based on their values for the selected feature. It is demonstrated that the anomaly instances usually create short branches of the tree, and therefore, the *length of the branch* is used as a criterion for the ranking of the objects [29]. Consequently, anomaly scores are calculated as a function of the path length of the branches that isolates the instance in the leaf nodes on all generated iTrees. This process can be formulated as follows [7]: let $h_t(x)$ be the path length of instance x on iTree t . Then, the average estimation of path length for a subset of N instances can be defined as Eq. 7.1:

$$C(N) = \begin{cases} 2H(N-1) - 2\frac{(N-1)}{N} & \text{if } N > 2, \\ 1 & \text{if } N = 2, \\ 0 & \text{otherwise} \end{cases} \quad (7.1)$$

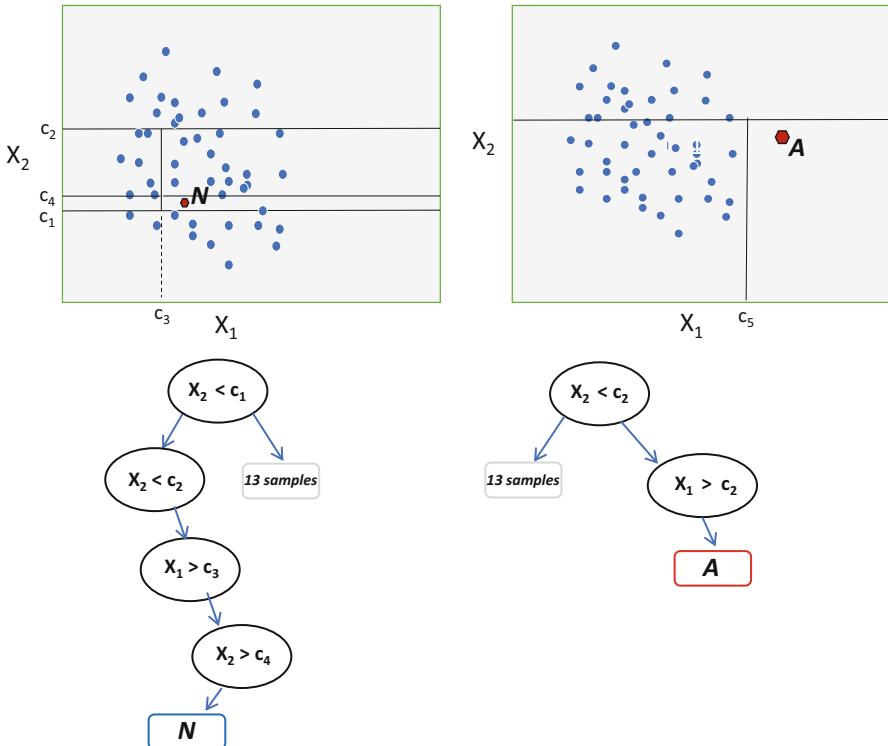


Fig. 7.1 Isolation-based anomaly detection. iTree structures are used to represent the partitioning and isolation process of instances in a dataset with two attributes. The left and right columns show example sequences of partitions to isolate normal and anomaly instances, respectively

where $H(N)$ is the harmonic number and can be calculated as $\ln(N) + \text{Euler_Constant}$. Using $C(N)$ for the normalization of expected $h(x)$ of instance x on all trees, the anomaly scores can be calculated as follows:

$$s(x, N) = 2^{-\frac{E(h(x))}{C(N)}} \quad (7.2)$$

Considering this formula, it is clear that anomaly scores have an inverse relation with the expected path length. Therefore, when the average path length of an instance is close to zero, the anomaly score is close to 1, and vice versa.

Figure 7.1 shows a graphical representation of the isolation technique for a dataset with two attributes X_1 and X_2 . The left and right columns show examples of random partitions on the attribute space and their corresponding tree structures to isolate a normal and anomaly instance, respectively. As it is shown, instance A (anomaly) can be isolated quickly considering the sparsity of values of X_1 around this instance. Though this example is a simple case with just two attributes, the

general idea can be extended to the problems with many features and a variety of distributions.

Considering the above explanations, ITL process is based on the idea that iTrees can also give information on important features for detection purpose. Therefore, ITL analyzes the generated iTree structure to extract information about the features that have more contribution in creating short branches and detected anomalies. In order to better explain the problem, let us assume that the input D is a matrix of N instances, each instance explained with a row of M features such that

$$D = \{(X_i), 1 \leq i \leq N | X_i = (x_{ij}), 1 \leq j \leq M, \\ x_{ij} \in R\}. \quad (7.3)$$

We have excluded nominal data in our assumptions and definition of Eq. 7.3. However, the ITL process is general and can be combined with solutions that convert categorical data to numerical to cover both cases [11]. We formulate the problem as follows: given a matrix D as the input, we try to iteratively remove some irrelevant features from the feature space of D , keeping the more relevant features for the detected anomalies at each step in an unsupervised manner. The goal is to increase the quality of the scores in terms of assigning higher scores to the true anomaly points by reducing the effect of noisy features. The output at each step k is a set of the scores S_k on a set of the reduced features M_k . The idea is that the removal of noisy features makes it easier to focus on the relevant partitions of the data, where the values of the features show higher deviations for the anomalous objects in comparison to the normal ones. As a result, the ranking of the input objects would improve with regard to the true detected anomalies.

7.4 ITL Approach

Figure 7.2 shows the main steps in ITL framework. As we already discussed in Sect. 7.3, the iTree structure forms the base of the ITL learning phase following the assumption that short branches in the structure of iTree are generated by the attributes with higher isolation capability. In another word, a subset of the attributes that are creating the nodes in the short-length branches can form a vertical partition of the data that localize the process on anomaly related subset of the data. As we can see in Fig. 7.2, the process is completely unsupervised with the input matrix as the only input of each iteration (that is we have no information of anomalous instances). There are four main steps in the ITL process and these are:

1. Building iTrees Ensemble: IForest creates a set of the iTrees from input data. This is a completely unsupervised process with a random sampling of the instances/features to create the splitting nodes in each tree.

Algorithm 1: ITL process

input : $D = (X_1, X_2, \dots, X_N)$, $X_i \in R^M$: D is a matrix of N records, each record including M features

Parameter: th : Anomaly score threshold value

output : Reduced Matrix, Scores

- 1 $D' \leftarrow D$
- 2 **while** not (There are unseen features) **do**
- 3 Build $iTrees$ ensemble using iForest on D'
/* Calculate scores for all input instances using Eq. 7.2 */
- 4 $S = (S_k) = (s_{k1}, s_{k2}, \dots, s_{kN}) \leftarrow Scores(iTrees, D')$
/* Filter a small part of the input matrix with higher anomaly scores */
- 5 $D_{\text{subset}} \leftarrow \{x_i \mid x_i \in D' \& s_i \geq th\}$
- 6 initialize $Frequency$ as an array with length equal to the number of features in D_{subset} all equal to zero
- 7 **for** $tree \in iTrees$ **do**
- 8 **for** $x \in D_{\text{subset}}$ **do**
- 9 update $Frequency$ of features by adding the occurrences of each attribute seen while traversing from the root node to the leaf node that isolates x
- 10 **end**
- 11 **end**
- 12 $D' \leftarrow \{x_{ij} \mid x_{ij} \in D' \& \& frequency(j) \geq Average(frequency)\}$
- 13 **end**
- 14 **return**(D' , S)

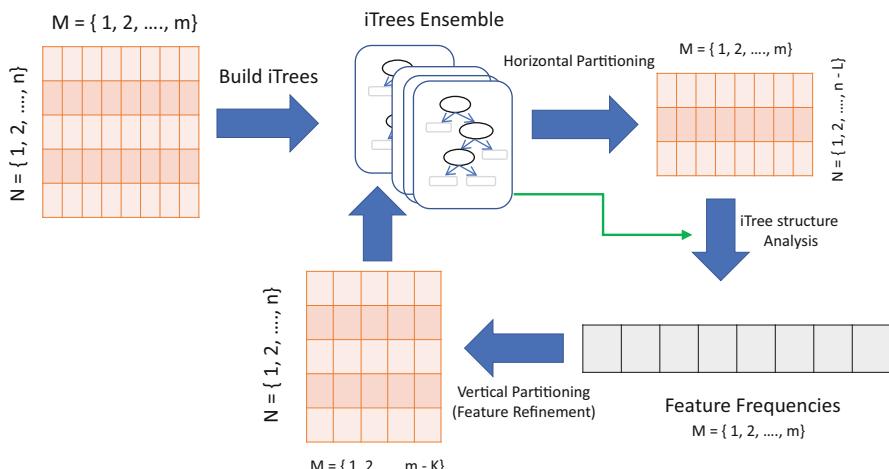


Fig. 7.2 ITL framework. The initial input is a matrix of N instances with M features. An ensemble of iTrees is created. Then, top ranked identified anomalies are filtered. The iTrees are analyzed for filtered instances to create a list of ranked features

2. Horizontal partitioning: The anomaly score for each instance is computed based on the length of the path traversed by the instance on the generated iTrees [7]. The final score shows the degree of outlierness of the instance. Our goal is to discover important features based on their contribution in the isolation of anomaly instances. The low score instances do not affect the determination of the important features for anomaly detection, and therefore, we can remove them to reduce the data size.
3. Extracting Feature Frequencies: We create a frequency profile of occurrences of different features observed during the traversal of short branches of iTrees. These features have a high probability of detecting anomalous instances.
4. Vertical Partitioning: Having a profile of the feature frequencies, a subset of the features with a higher contribution in the abnormality of data are selected and other features are removed. This process creates a vertically partitioned subset of data as the input for the next iteration of the ITL.

This process is repeated multiple times until the termination condition is met. As we continuously refine the features, we expect to see improvement in the anomaly detection process as the detection process becomes more focused on the interesting set of features. Therefore, the set of iTrees built during consecutive steps can be combined to create a sequence of the ensembles. Algorithm 1 shows the pseudo-code of ITL. A more detailed and formal description of the process is presented in the following section.

7.4.1 Feature Refinement Process

We assume that the input D is a matrix of objects labeled as one of the classes of normal or anomaly. These labels are not part of the ITL process as it is an unsupervised mechanism. They are used for evaluating the output results of proposed algorithms and other benchmarks for validation purpose. The goal is to find a ranking of the objects so that the higher values imply higher degrees of abnormality. Considering this objective, the first step of ITL process is to build the initial batch of the iTrees from the input matrix. IForest is used to create t iTrees. To create each iTree, ψ random instances are selected from D , and each node of the tree is created by randomly selecting a feature and a value and splitting the instances based on this selection to form two branches. The output is iTrees ensemble and anomaly scores $S = (s_1, s_2, \dots, s_N)$ computed for all instances based on Eq. 7.2 (Lines 3–4, Algorithm 1).

After creating new iTrees, the next step is to reduce the target instances for the learning procedure (Line 5). A threshold value (th) is defined and all instances with an anomaly score lower than this value are discarded. The idea behind this selection is to focus better on parts of the data which have a higher degree of abnormality based on the iTrees structure as well as reducing the complexity of the problem. As the learning phase is the most time-consuming part of the ITL process, this

reduction dramatically decreases the runtime of the algorithm. The selection can also take advantage of the expert knowledge on the characteristics such as the contamination ratio of the dataset for defining a proper cut-off value of anomaly scores. The output of this step is a subset of the input matrix D (D') with p instances such that $p << |D|$. We emphasize that the process is unsupervised as we do not have the knowledge of anomalous instances. However, based on the assumption that anomalies are few and different, we expect to see many of the anomaly instances in D' . It should also be noted that the generation of each iTree is completely random in terms of the splitting features and value selection. Therefore, one tree may not be informative per se. However, when the random process is repeated to generate many trees, the overall observed patterns confirm the idea of short branch isolation of anomaly instances [7]. This can be observed in Fig. 7.1 as well. Generating iTree structure on high-density regions requires many nodes and splitting conditions to isolate one instance, while for an anomaly instance there is one feature or more that can quickly differentiate that from the rest of the data.

The instances that passed the filtering procedure from the previous step (highly ranked anomalies) are processed by each iTree from the ensemble model to record the frequency of occurrences of features when traversing the trees. The frequency profile of features allows determining important features relevant to detecting target anomalies. According to the formulas in Sect. 7.3 and their interpretation as an iTree structure, we expect to see a subset of more important features for anomalous instances in the short branches of trees. It should be noted again that these are the expected observations from an ensemble of many random trees and are not attributed to any specific iTree. Consequently, we keep the features whose frequencies are higher than the average of the frequency profile (Lines 6–12).

The above steps are repeated multiple times. The output is a set of the anomaly scores for each subset of the data, starting from full data with all features. Therefore, the iteration k of ITL process creates a set S_k of anomaly scores for all instances on the reduced feature set M_k (M_0 is the full set of the features for the first iteration). We note that each iteration would produce a potentially different set of anomalous points and hence a different frequency profile of the features. The termination condition we choose is when the frequency of occurrences for all features is greater than one, meaning that every feature has seen at least one anomalous point in the short branches of iTrees. The idea behind this condition is that as the noisy features are removed during the iterative process, ITL produces better iTrees for detecting the true set of anomalies. Therefore, the observed features become more important in the detection process. When ITL reaches a state that all the features are present in the short branches, it indicates that all current features are contributing to the detection of anomalous instances. Therefore, the termination condition T_k at iteration k is evaluated as follows:

$$T_k = \begin{cases} True & \text{if } Size(M_k) \leq 1 \text{ or} \\ & Frequency_k > 0 \\ False & \text{otherwise,} \end{cases} \quad (7.4)$$

where $Size(M_k)$ evaluates the number of remaining features at the iteration k . $Frequency_k$ is the corresponding frequency profile which is an array of length M initialized by zero (Line 6). The term $Frequency_k > 0$ evaluates the condition that the frequencies of all attributes in M_k are greater than zero. When T_k evaluates to true, ITL process terminates and the final outputs are evaluated as follows:

- *Bagging of the Scores*: Each iterative step of the ITL process produces a score for each data point in D , which represents the degree of anomalousness based on the corresponding set of the reduced features. As we try to improve the detection capability of the ensemble by reducing the noisy features, we expect to get better anomaly scores in terms of the ranking of instances. Therefore, in this approach, the goal is to take advantage of detection results from all iterations by averaging the scores and defining a new score for each instance. Accordingly, the final score of each instance is calculated as follows:

$$S_f(x) = \frac{1}{K} \sum_{k=1}^{k=K} S_k(x), \quad (7.5)$$

where S_f is the final score and S_k is the score at iteration k from K iterations of ITL process.

- *Reduced Level Scores*: ITL produces an ensemble of iTrees on the important features for anomaly detection. The generated iTrees on the reduced features can be used for detecting anomalies in new data. Therefore, the anomaly scores are calculated directly based on the extracted reduced feature set from the process.

7.5 Performance Evaluation

In this section, an empirical evaluation of ITL process on two network intrusion datasets and three other benchmark datasets is presented. The two sets of experiments are designed to demonstrate the behavior of ITL in bagging and reduced modes on the target datasets. First, Sect. 7.5.1 presents the datasets and parameter settings of the experiments. Then, Sect. 7.5.2 shows the comparison results of ITL in the bagging mode with a recently proposed state-of-the-art sequential ensemble learning method and then investigates the improvements made by reduced level features in terms of both AUC and runtime analysis in a set of the cross-validated experiments. Sections 7.5.2.3 and 7.5.3 discuss runtime complexity and weakness/strength points of ITL approach.

Table 7.1 Properties of data used for experiments. N and M are the number of instances and features in each dataset, respectively

	N	M	Anomaly ratio (%)
DOS	69363	37	3
U2R	69363	37	3
AD	3279	1558	13
Seizure	11500	178	20
SECOM	1567	590	6

7.5.1 Experimental Settings

Table 7.1 shows a summary of statistics for the benchmark datasets. All datasets are publicly available in UCI machine learning repository [31].¹ For U2R and DOS datasets which are network intrusion dataset from Kddcup99, a downsampling of attack classes is performed to create the anomaly class. In other datasets, the instances in the minority class are considered as the anomaly.

In order to evaluate the results, we select the Receiver Operating Characteristics (ROCs) technique and present Area Under the Curve (AUC) as a measure of the accuracy of the system. AUC value summarizes the trade-off between true positive and false positive detection rates as shown in Eqs. 7.6 and 7.7 under different threshold values. Higher AUC indicates better performance with regard to the detected anomalous instances.

$$TPR = \frac{TP}{TP + FN} \quad (7.6)$$

$$FPR = \frac{FP}{FP + TN}. \quad (7.7)$$

ITL process is implemented based on the publicly available Python library, scikit-learn [32]. Unless otherwise specified, the values of the parameters for iTree generation step of ITL process are according to the recommended settings as explained in [7]. The value of other parameters is set based on the experimental tunings. The threshold value for the horizontal partitioning (th in Algorithm 1) is determined by assuming a contamination ratio equal to 0.05% for all datasets. This means that the cut-off threshold is identified so that 0.05% of the objects have a score higher than the th , which is good enough considering the number of instances and the contamination ratio in our target datasets. The frequency profiling is done on the branches with a maximum length of 4. This threshold has been selected based on the average length of the trees which is dependent on the sample size and therefore constant in all experiments. To ensure comparativity, the number of trees for the IForest algorithm in all methods is the same and is between 600 and 900 trees.

¹ <http://archive.ics.uci.edu/ml>.

For the comparison, we have selected a recently proposed sequential learning method, CINFO, designed for outlier detection in high-dimensional data [11]. CINFO works based on lasso-based sparse regression modeling to iteratively refine the feature space. As their method is general, we select the IForest-based implementation which considers the scores generated by IForest algorithm as the dependent variable of the regression model. Due to the randomness feature of iTree generation, each experiment is repeated for minimum of 5 times, and the average of results is reported. For CINFO, the number of repeated experiments is based on their recommended values to have stable results [11].

7.5.2 Experiment Results

7.5.2.1 ITL with Bagging of the Scores

Table 7.2 shows the AUC results for the base IForest algorithm as well as both ITL and CINFO learning methods. The best results are highlighted in bold. As the results show, the ITL process improves the performance of IForest by combining the scores from various subsets of the feature space. The best AUC results are achieved for *AD* dataset for which the results of ITL show a dramatic improvement (around 22%) compared to the base method. This is a result of the higher ratio of noisy features in this dataset. In comparison to CINFO method, the same or better performance is observed for 4 of the 5 datasets. The only exception is *Secom* where ITL shows improvements compared to the base, but not as much as the CINFO. This could be attributed to the greedy removal of features in vertical partitioning of ITL as we explained in Sect. 7.4.1. Since the results for *DOS* and *Seizure* are very high, even with the base IForest (higher than 95%), we do not expect to see too much improvement. However, ITL still shows comparable or improved AUC while achieving a reduction of about 8 and 43% in the size of the feature set. In general, ITL shows improved results as well as a reduction of the features between 9 and 97% compared to the original set. These results are especially important when the quality of reduced features is investigated for the detection of unseen anomalies.

Table 7.2 AUC results for the base IForest, ITL, and CINFO. M and M' show the size of the original and reduced features for ITL. The best AUC for each dataset is highlighted in bold

	IForest	CINFO	ITL	ITL Feature Reduction		
				M	M'	Reduction
DOS	0.981	0.971	0.981	37	21	43%
U2R	0.874	0.894	0.901	37	18	51%
SECOM	0.551	0.655	0.594	590	80	86%
AD	0.704	0.850	0.856	1558	54	97%
Seizure	0.989	0.987	0.990	178	163	8%

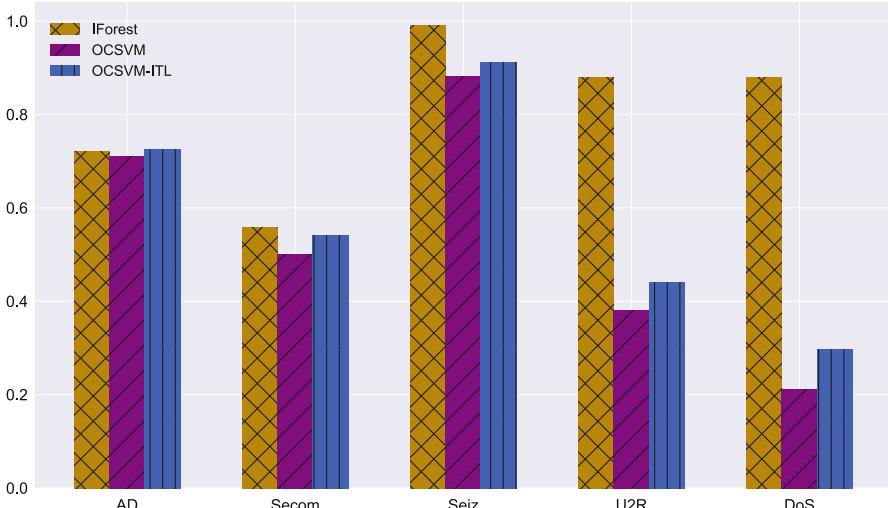


Fig. 7.3 AUC comparison for IForest and OCSVM. OCSVM results are shown when applied on input data with all features and with ITL reduced set of the features. The results are average AUC over cross-validation folds

Therefore, in the following, we further study the effectiveness of the reduced subset of features produced by ITL in anomaly detection results.

7.5.2.2 ITL with Reduced Features

To validate the efficacy of the reduced subset of features on the detection capability of IForest algorithm, a series of experiments are conducted based on the k-fold cross-validation. Figure 7.3 compares the performance of isolation-based technique (IForest) and a kernel-based anomaly detection method (OCSVM). OCSVM is a novelty detection method that can also be used in unsupervised anomaly detection by selecting soft boundaries. The figure also shows the results of OCSVM when applying OCSVM with ITL approach can improve the results of OCSVM, the isolation-based technique (IForest) shows higher performance. Therefore, in the next experiments, the performance of ITL on IForest is studied. The five-fold validation is used to train the IForest model on 4 parts of the data when all features are included in comparison to the data with the reduced features from the ITL process and AUC values of validation parts are reported. Figure 7.4 demonstrates ITL results for different numbers of trees from 1 to 100. As we can see, reduced features can achieve or improve AUC value compared to the full set of the features for a range of number of trees in all datasets. The interesting observation is that the reduction in the number of trees has less impact on the performance, especially for the reduced set as shown in Fig. 7.4. For example, even with 10 trees, the results are very close to the

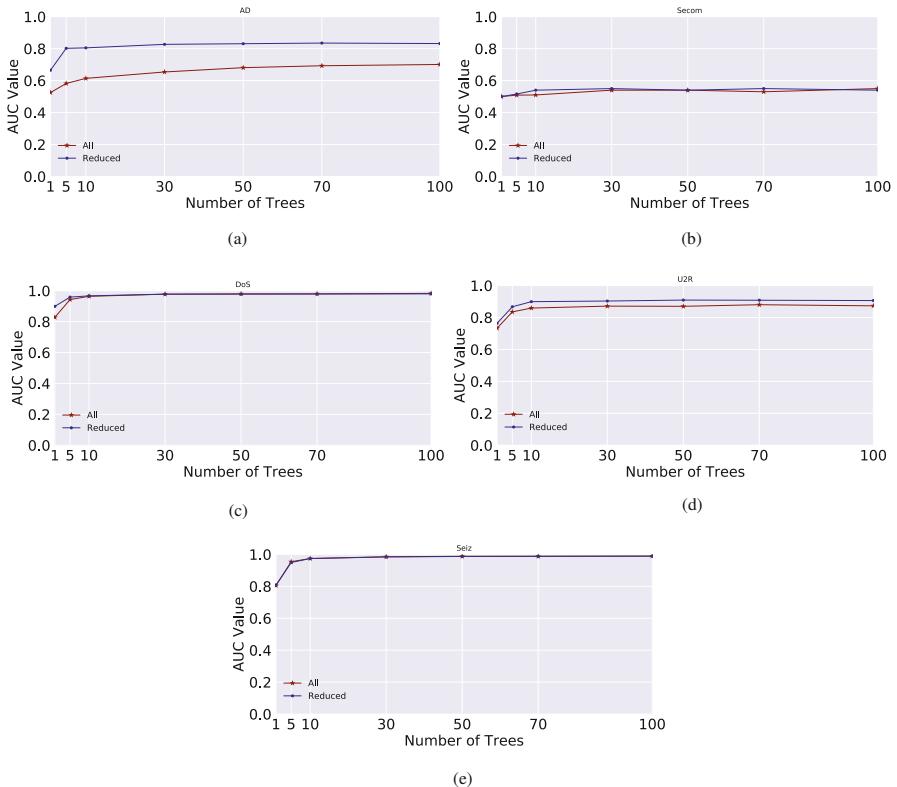


Fig. 7.4 AUC comparison for iForest when applied on input data with all features and with ITL reduced set of the features. The results are average AUC over cross-validation folds. **(a)** AD. **(b)** Secom. **(c)** DoS. **(d)** U2R. **(e)** Seiz

performance of the algorithm with default parameters (100 trees). This improvement can be attributed to having fewer features to be explored during the random selection of the features. In other words, having a subset of the features learned through ITL process, one can achieve improved results with less number of trees. The reduction of features, as well as the number of trees, can help to reduce the complexity in terms of the memory and runtime requirements. Figure 7.5 shows the running time taken for a variety of tree numbers. As we can see, the number of trees can hugely impact the testing time. This is highly important for dynamic environments such as the cloud where the testing should be performed regularly. These results indicate ITL approach as a potential choice to be employed by real-time applications where the new incoming stream of data requires quick online tests for identifying possible problems.

During the ITL learning phase, the number of iTrees in each ensemble is a parameter that should be decided for each iteration. In order to have a better

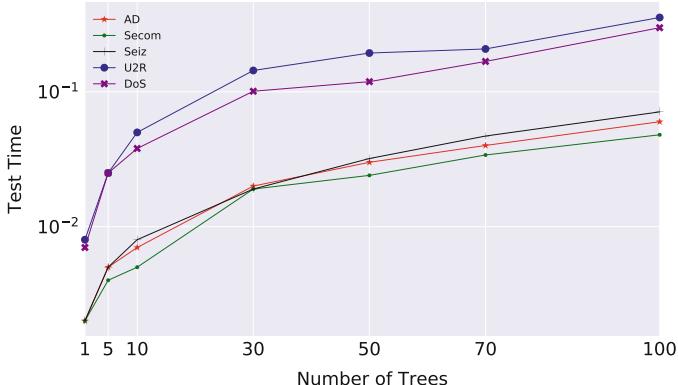


Fig. 7.5 Runtime for the Testing of cross-validated results on the reduced features. Logarithmic scale is used on y axis

understanding of the sensitivity of ITL to this parameter, we run ITL several times for a range of values for the number of trees. Figure 7.6 shows the AUC distribution of each set of the experiments for all datasets. As the results show, ITL is sensitive to this parameter. However, AUC values show improvements with the increased number of trees and are stable for numbers larger than 600. Practically, we found that a value between 600 and 900 trees is sufficient in most cases to have a good trade-off between accuracy and training complexities in terms of memory and runtime.

7.5.2.3 Time Complexity and Runtime Analysis

Algorithm 1 presents the main steps of the ITL process. The main while loop (Line 2) continues until the termination condition of having zero unseen attributes is met. The termination condition is guaranteed to converge as during the vertical partition phase, features with zero-seen or low frequency are removed which reduces the feature space. As a result, the remaining features have more chance to be explored with regard to anomalous instances (and possibly showing in short branches of the tree which increases their potential to be included in high-frequency profile). Since we always have potential anomalies seen in short branches, there is at least one feature with a frequency higher than one which will create the final reduced subspace. Therefore, we finally get to a level where all features are seen at least one time or the reduced subspace has just one feature left. The loop typically converges in less than 5 iterations. Lines 3–11 build IForest models and filter high-rank instances based on the predefined threshold. Considering the IForest trees as the base structure for these steps, it takes $O(t\psi \log(\psi))$ for constructing where ψ is the number of selected subsamples and t is the number of constructed trees. If there are N testing points, it requires $O(Nt \log \psi)$ for determining anomalous points and

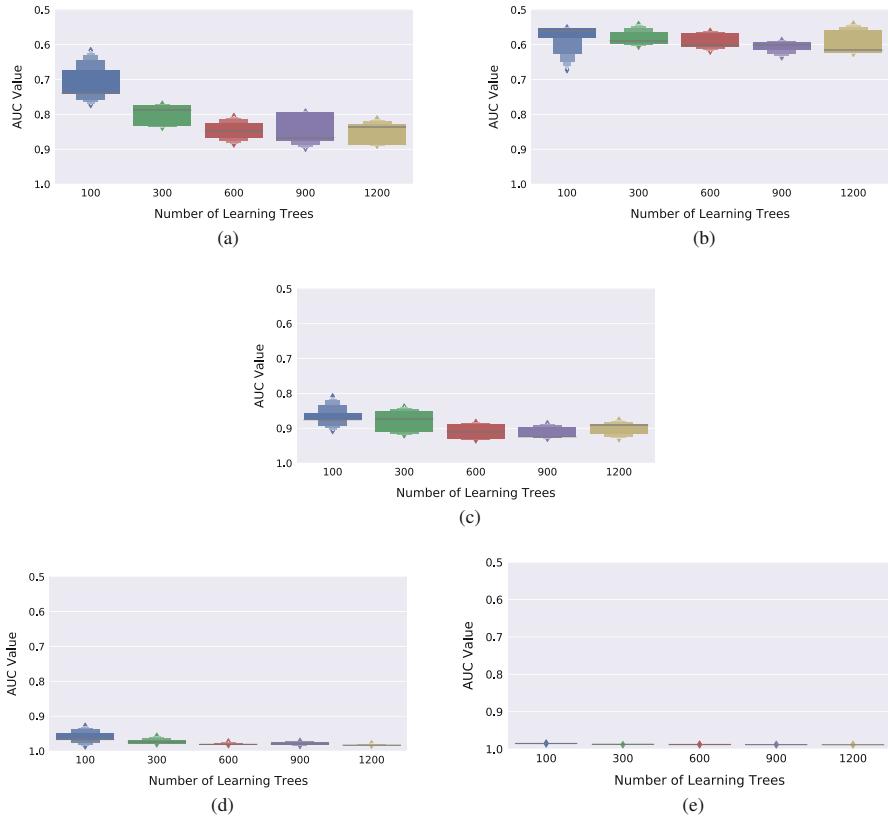


Fig. 7.6 AUC value distribution for ITL reduced features in training. This plot shows the sensitivity of ITL process to different numbers of the learning trees. **(a)** AD. **(b)** Secom. **(c)** U2R. **(d)** DoS. **(e)** Seiz

$O(Kt\log\psi)$ for updating frequency profile of the features, where $K \ll N$ (Line 5) is filtered anomalies (worst-case complexity is order $O(t\psi(\psi + N))$). Therefore, we expect a linear complexity with regard to data size.

IForest is shown to have a very fast and memory-efficient runtime for both modeling and testing purposes. In order to have a clear understanding of the ITL contribution to make this process even faster, a series of execution times with respect to the number of learning trees are presented. Figure 7.7 shows the learning time in ITL, where the main feature refinements are done by constructing iTrees and creating a new subset of features. The diagram shows the learning time for a variety of tree numbers. As it is mentioned before, 600–900 usually is enough to have a sufficient exploration of feature space for target datasets. When the learning phase of ITL is completed, the anomaly detection is done by modeling iTrees with extracted features. To have a better comparison of execution times,

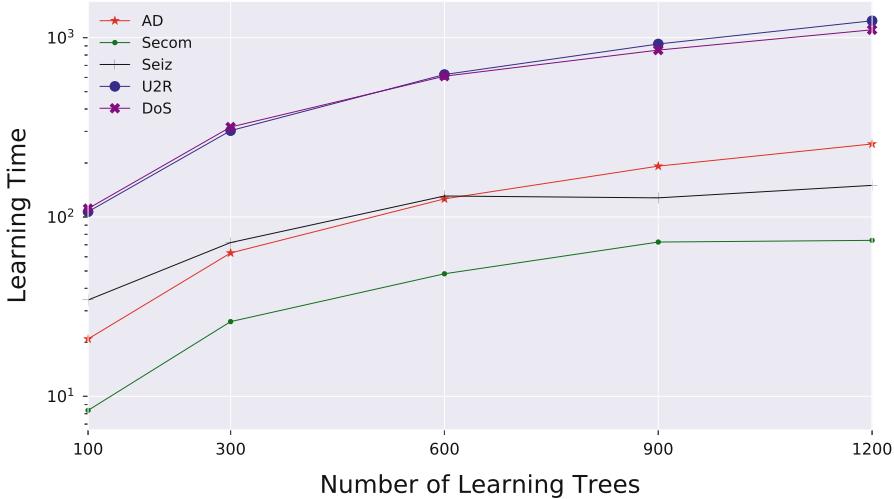


Fig. 7.7 Total runtime of learning phase of ITL. Logarithmic scale is used on y axis

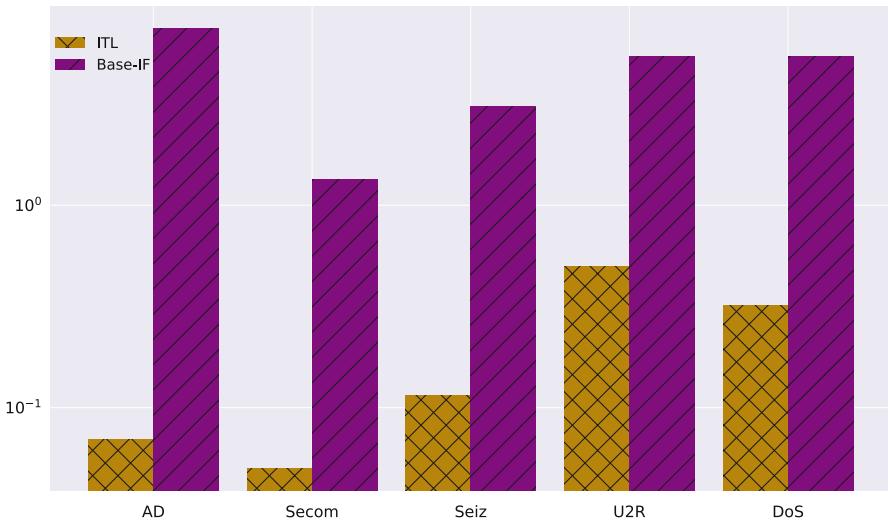


Fig. 7.8 Comparison of modelling times for ITL-produced features with reduced number of iTrees (yellow) and base IForest algorithm (Purple) with default parameters. Logarithmic scale is used on y axis

Fig. 7.8 compares modeling time of ITL-learned features with the reduced number of trees with the base IForest without feature refinements and with the recommended number of trees in the literature. As we can see, ITL process makes a dramatic decrease in modeling times by helping to decrease the number of features/trees which makes the construction of iTrees and training step much faster. It should

be highlighted that this reduction is achieved by keeping or improving the detection accuracy as it is shown in Fig. 7.4. However, the feature refinement process of ITL as shown in Fig. 7.7 is the cost of achieving these results. But the learning phase is a one-time process which is performed off-line, and the final subset is used for subsequent anomaly detection task which is significantly improved in terms of both modeling and testing times as shown in Figs. 7.5 and 7.8, respectively. Considering the context of one application, the learning phase can be done with a low frequency and as a background process. Therefore, systems that require regular updating of their performance models can highly benefit from time/memory reductions of this process.

In conclusion, ITL shows that by targeting the main contributing features which isolate the instances in iTrees, we can reach a refined set of the features that can be used by fewer trees to create a model with better results.

7.5.3 Strength and Limitations of ITL Approach

IForest algorithm, as described in Sect. 7.3, is designed to detect anomalous objects by the ensemble of binary trees from input data. ITL tries to take the advantage of this mechanism to extract information about relevant features that better isolate instances. Since the core of the ITL is iTree data structures from IForest, the same advantages of random-based sampling and feature selection are equally applicable to ITL. Moreover, it can be used as a preprocessing step to learn a reduced set of features for any other anomaly detection algorithm. ITL is a promising method for real-time applications as high detection accuracy can be achieved with small memory and time complexity, and it can perform well without prior knowledge of the specific distribution. We have tested the applicability of our proposed techniques on a variety of datasets with different characteristics and application domains. The benefit of our method is that it can be trained rapidly as trees are a very fast construct. Finding optimal features can be computed in the background without sacrificing response time for anomaly detection. Moreover, ITL is an unsupervised method and does not require training data containing anomaly annotations.

Similarly, ITL inherits the same drawbacks as the base algorithm in detecting local clustered anomalies [8]. This can affect the filtering of instances when the assumption is made that there are a majority of anomaly instances at the top of the ranking list. Adaptive, data-dependent configuration for parameters such as the maximum height of trees or customized split point selection for node constructions may help to reduce this effect but requires more preprocessing and knowledge on statistical characteristics of anomalous data.

7.6 Conclusions and Future Work

In this part, we introduced an iterative learning framework (ITL) for the refinement of features and improvements of the anomaly detection process. Advances in monitoring and storage capabilities provide a high volume of information on the performance of applications and systems to be used for anomaly and fault analysis. This requires real-time analysis of data to quickly identify problems and take appropriate corrective actions. However, high-dimensional data can adversely affect the traditional measures of anomaly detection such as distance between instances in terms of efficacy and time complexity. More recent approaches such as the isolation-based technique try to directly target the main features of anomalies as being different and rare. ITL is designed based on the idea that isolation-based generated trees can give some insights on the importance of the features. Therefore, the learning phase of ITL is based on the knowledge from iTree structures which are binary trees constructed by random selection of the features from domain problems. The assumption is made that the features on the short branches of iTree can be used as a reference to identify relevant features to the detection of anomaly instances. The learning is based on the iterative removal of the noisy and irrelevant features in terms of their importance for isolating anomalies to generate a final subset of the features to be used for anomaly detection. The experiments show that the anomaly scores from IForest algorithm on generated subsets of the data at each iteration can be combined to create a more informative set of the scores in terms of the detection capability of anomaly instances. Moreover, the experiments on five benchmark datasets demonstrate that with the reduced set of the features and choosing a proper number of trees IForest can achieve better results in terms of the detection accuracy while reducing the complexity of the algorithm.

For future work, we plan to enhance ITL framework to identify groups of anomalous metrics that isolate individual faults. The isolation can be achieved for environments that different groups of features are impacted by different types of faults. This helps to distinguish among different anomalies such as various types of attacks. We also would like to extend the ITL idea to more flexible tree structures (for example, trees with more than two branches) to investigate the possibility of further improvements for clustered anomalies. We also highlight that anomaly detection, in general context, considers any significant deviation in the values of the attributes from the past data (training part) as an anomaly which will be reflected in anomaly scores. However, with regard to the required actions after detecting anomalies, some level of knowledge expert may be required. For example, after detecting abnormality in packet-level information that can be a sign of the attacks (as shown by datasets of U2R and DoS in the experiments), application owners may prefer to shut down targeted resources (VM or physical machine) to reduce the cost of wasted resources. The integration of anomaly detection part and resource management modules may bring new challenges in the design of scaling solutions that require further investigations.

References

1. V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey. *ACM Comput. Surv.* **41**(3), 15 (2009)
2. J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, K. Hu, Disk failure prediction in data centers via online learning, in *Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018* (ACM, New York, 2018), pp. 35:1–35:10
3. C.C. Aggarwal, P.S. Yu, Outlier detection for high dimensional data, in *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data* (ACM, New York, 2001), pp. 37–46
4. M.H. Bhuyan, D. Bhattacharyya, J. Kalita, A multi-step outlier-based anomaly detection approach to network-wide traffic. *Inf. Sci.* **348**, 243–271 (2016)
5. A. Zimek, E. Schubert, H.-P. Kriegel, A survey on unsupervised outlier detection in high-dimensional numerical data. *Stat. Anal. Data Min. ASA Data Sci. J.* **5**(5), 363–387 (2012)
6. H. Liu, X. Li, J. Li, S. Zhang, Efficient outlier detection for high-dimensional data. *IEEE Trans. Syst. Man Cybern. Syst.* **48**, 2451–2461 (2017)
7. F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection. *ACM Trans. Knowl. Discover. Data* **6**, 3:1–3:39 (2012)
8. F.-T. Liu, K.-M. Ting, Z.-H. Zhou, On detecting clustered anomalies using sciforest, in *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II* (2010), pp. 274–290
9. F. Keller, E. Müller, K. Bohm, HiCS: high contrast subspaces for density-based outlier ranking, in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, ICDE'12* (IEEE Computer Society, Washington, 2012), pp. 1037–1048
10. H. Shi, H. Li, D. Zhang, C. Cheng, X. Cao, An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification. *Comput. Netw.* **132**, 81–98 (2018)
11. G. Pang, L. Cao, L. Chen, D. Lian, H. Liu, Sparse modeling-based sequential ensemble learning for effective outlier detection in high-dimensional numeric data, in *Thirty-Second AAAI Conference on Artificial Intelligence* (2018), pp. 3892–3899
12. S. Agrawal, J. Agrawal, Survey on anomaly detection using data mining techniques. *Proc. Comput. Sci.* **60**, 708–713 (2015)
13. C. Pascoal, M.R. de Oliveira, R. Valadas, P. Filzmoser, P. Salvador, A. Pacheco, Robust feature selection and robust PCA for internet traffic anomaly detection, in *2012 Proceedings IEEE INFOCOM* (2012), pp. 1755–1763
14. F. Angiulli, C. Pizzuti, Fast outlier detection in high dimensional spaces, in *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery* (Springer, London, 2002), pp. 15–26
15. S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (2000), pp. 427–438
16. M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (AAAI Press, 1996), pp. 226–231
17. Y. Zhu, K.M. Ting, M.J. Carman, Density-ratio based clustering for discovering clusters with varying densities. *Pattern Recogn.* **60**, 983–997 (2016)
18. J. Chen, S. Sathe, C. Aggarwal, D. Turaga, Outlier detection with autoencoder ensembles, in *Proceedings of the 2017 SIAM International Conference on Data Mining* (SIAM, Philadelphia, 2017), pp. 90–98
19. Q. Guan, S. Fu, Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures, in *Proceedings of the 32nd IEEE International Symposium on Reliable Distributed Systems, SRDS'13* (IEEE Computer Society, Braga, 2013), pp. 205–214

20. D.J. Dean, H. Nguyen, X. Gu, UBL: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems, in *Proceedings of the 9th International Conference on Autonomic Computing* (ACM, New York, 2012), pp. 191–200
21. A.B. Ashfaq, S. Rizvi, M. Javed, S.A. Khayam, M.Q. Ali, E. Al-Shaer, Information theoretic feature space slicing for statistical anomaly detection. *J. Netw. Comput. Appl.* **41**, 473–487 (2014)
22. J. Cao, B. Yu, F. Dong, X. Zhu, S. Xu, Entropy-based denial of service attack detection in cloud data center, in *Proceedings of the Second International Conference on Advanced Cloud and Big Data* (2014), pp. 201–207
23. J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: a data perspective. *ACM Comput. Surv.* **50**(6), 94:1–94:45 (2017). <http://doi.acm.org/10.1145/3136625>
24. D. Fesihaye, L. Singaraveluy, C. Chen, X. Huang, A. Banerjee, R. Zhou, R. Somasundaran, Group clustering using inter-group dissimilarities, in *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (2017), pp. 1011–1021
25. P. Filzmoser, R. Maronna, M. Werner, Outlier identification in high dimensions. *Comput. Stat. Data Anal.* **52**(3), 1694–1711 (2008)
26. T. de Vries, S. Chawla, M.E. Houle, Finding local anomalies in very high dimensional space, in *Proceedings of the 2010 IEEE International Conference on Data Mining* (2010), pp. 128–137
27. A. Lazarevic, V. Kumar, Feature bagging for outlier detection, in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (ACM, New York, 2005), pp. 157–166
28. S. Jin, Z. Zhang, K. Chakrabarty, X. Gu, Toward predictive fault tolerance in a core-router system: anomaly detection using correlation-based time-series analysis. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **37**(10), 2111–2124 (2018)
29. F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in *Proceedings of the 8th IEEE International Conference on Data Mining* (IEEE, Piscataway, 2008), pp. 413–422
30. N. Moustafa, J. Hu, J. Slay, A holistic review of network anomaly detection systems: a comprehensive survey. *J. Netw. Comput. Appl.* **128**, 33–55 (2019)
31. D. Dua, C. Graff, UCI machine learning repository (2017). <http://archive.ics.uci.edu/ml>
32. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)

Chapter 8

Digital Twin of a Cloud Data Centre: An OpenStack Cluster Visualisation



Sheridan Gomes, Adel N. Toosi , and Barrett Ens

Contents

8.1	Introduction	209
8.2	Background	211
8.2.1	A Taxonomy of Digital Twin Characteristics	211
8.3	Digital Twin of a Cloud Data Centre Instance System	214
8.3.1	Unity Model.....	214
8.3.2	API Integration.....	218
8.4	Qualitative Study	219
8.5	Evaluation and Results	221
8.6	Discussion and Future Directions	223
8.7	Conclusions	224
	References	224

8.1 Introduction

The beginning of the modern era has seen an exponential rise in Information Technology (IT). The early twenty-first century saw the rise and growth of mobile phones, computers and tablet computers. These devices have now become a staple in households, academia and business enterprises all around the world. As we progress through this age, technology has advanced further and further, giving rise to new Internet-of-Things (IoT) devices ranging from smart fridges and smart TVs to even smart weight scales, to name a few. All these devices, along with IoT and other IT applications, produce a massive amount of data. Data in today's world is one of the most critical and essential resources for companies and people alike. Data centres are built to meet this demand for data storage and analysis, especially cloud data centres where data is stored, accessed and analysed on the cloud. Cloud data centres are computational infrastructures available globally and provide services that allow the storage, access and analysis of data on remote servers. Cloud data centres

S. Gomes · A. N. Toosi () · B. Ens

Faculty of Information Technology, Monash University, Clayton, VIC, Australia

e-mail: adel.n.toosi@monash.edu; barrett.ens@monash.edu

have numerous advantages compared to their conventional counterpart and provide a highly scalable and cost-effective way of accessing computational resources, often in the form of virtual machines (VMs). OpenStack is an open-source cloud computing platform, mostly deployed as infrastructure-as-a-service in both public and private clouds where VMs and other resources are made available to users [5].

While existing management and monitoring tools such as OpenStack dashboard (Horizon) are precious for data centre administrators, data centre management and monitoring present several challenges for the operators [9]. Data centre admins need to have real-time and unified information about their data centre's resources and health status to make quick decisions. For example, in the case of server maintenance where repairs are required on a physical server, it is essential to quickly find a suitable host to accommodate tenants' VMs running on the failing server. In addition, data tables or text table view of data provided by tools such as Horizon often fails to deliver admins' all essential requirements. For example, more detailed data about the exact location of a physical server or its power consumption might be required and tools like OpenStack fail to provide that in their dashboard. Finally, working with the OpenStack dashboard requires a high amount of knowledge and expertise to understand and analyse the dashboard's different aspects. Furthermore, because of how the human brain processes information, visualisation tools can represent large amounts of complex data such as data centre resources much more straightforward than tables or raw data. This research study aims to represent a visualisation tool to help data centre admins better perform their monitoring and management tasks.

In this book chapter, we propose a digital twin of the cloud data centre to tackle the above complexities. A digital twin is a virtual representation of an object. A digital twin can represent different aspects of a system in an easy to understand virtual setting, which allows for higher usability and a better understanding of the whole system. Digital twins have been used to handle many complex systems ranging from aviation to manufacturing [1]. A user-friendly and time-saving monitoring tool for cloud data centres would positively impact the industry by reducing complexity and increasing productivity in resource monitoring and management, which would be essential for a cloud data centre administrator. For testing and research on the proposed hypothesis, a digital twin prototype was developed using Unity3D software. Unity3D was used to develop the prototype as it is easy to use, works well with APIs and has a great support community. The digital twin needed to be implemented to work with OpenStack APIs and some other infrastructure level monitoring APIs to synchronise the digital twin with the cloud data centre in real time. Any changes made to the cloud data centre would be reflected in the virtual environment on the digital twin instance and vice versa.

Once the digital twin prototype is developed using Unity3D to ascertain its usability as a monitoring and management tool, an interview was conducted with multiple participants who are experts in cloud data centres. These participants had to watch a live demonstration of the digital twin in action, illustrating the digital twin's different features. An interview was conducted asking questions to gauge how the digital twin would improve the data centre admin experience compared to

the OpenStack dashboard if various fields such as usability, complexity and time management are considered. The feedback from the experts helped us understand the benefits, differences and future need for improvement.

Below are the highlights of this book chapter:

- A brief taxonomy of digital twin characteristics and elements.
- The design and implementation of a digital twin—a 3D visualisation, monitoring and management tool—of a cloud data centre using Unity3D. To the best of our knowledge, the proposed digital twin of a cloud data centre is one of its first kind in the literature. The digital twin uses data fed from the OpenStack APIs and infrastructure-level devices such as Enclosure Power Distribution Units (ePDUs) to provide an interactive and real-time virtual representation of the different aspects such as VMs, hypervisors and physical servers of the cloud data centre running on OpenStack platform.
- Insights for future directions for building a more advanced digital twin of a data centre based on the expertise and knowledge gained from conducting the research and implementation of our cloud data centre digital twin.

The rest of the book chapter is organised as follows: Sect. 8.2 outlines the related works, background and taxonomy of digital twin characteristics. In Sect. 8.3, the research methodology is explained; this includes the system design of the digital twin, implementation, prototyping and interview process. Section 8.4 describes the findings and discussions of the research and interviews. Lastly, we conclude with the summary of the research conducted and highlight some future works.

8.2 Background

Digital twin technology is described as a virtual representation of a tangible real-world object. The digital twin has two primary components: the physical twin (the real-world physical object) and the virtual twin (a virtual representation of the physical object). The primarily related works of the digital twin are in the aviation and manufacturing industries, with potentials in other sectors being high. A digital twin of a cloud data centre is a new idea, and there is not much research related to that topic [2]. To the best of our knowledge, we are the first to propose a non-commercial digital twin of a cloud data centre with features discussed here.

8.2.1 *A Taxonomy of Digital Twin Characteristics*

Figure 8.1 provides a taxonomy of digital twin characteristics. In the following section, we explain each element of the taxonomy and map our model to that.

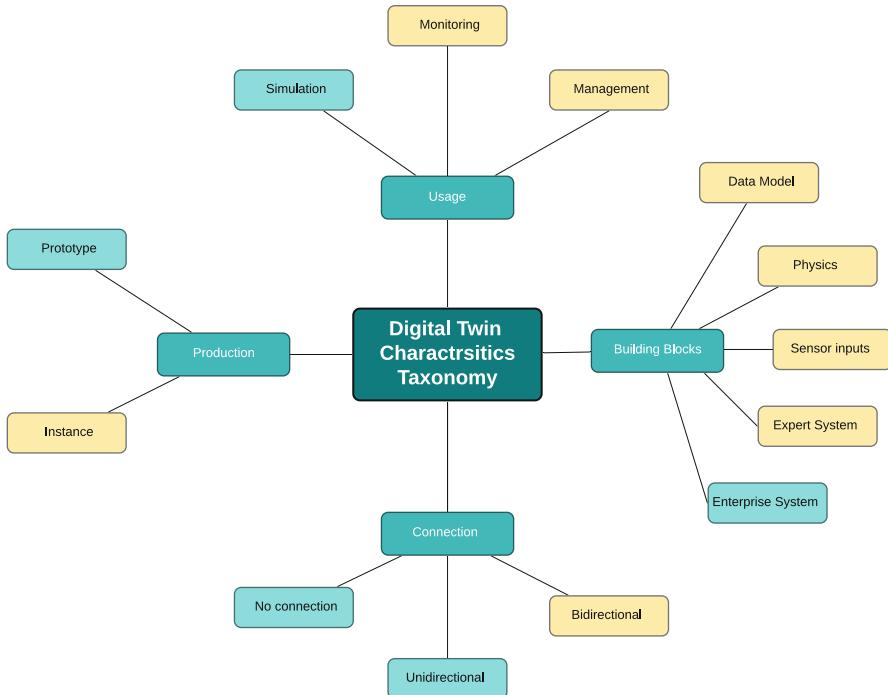


Fig. 8.1 Taxonomy of digital twin characteristics

8.2.1.1 Digital Twin Usage

The usage of the digital twin can be classified into three kinds of project dimensions: *simulating*, *monitoring* and *managing*. Replication/simulation digital twin dimension is where the main goal of digital twin projects is to reproduce the physical object and simulate various aspects to ensure that the physical object is ready to be deployed without any issues. The monitoring dimension of the digital twin is when the digital twin is used to monitor a physical object; this can be done to ensure that the physical item is working correctly and as intended [10]. Lastly, the digital twin's managing dimension is when the digital twin's primary aim is to manage the physical item, which would mean that the physical twin and the digital twin would have a bidirectional connection or two-sided connection. Wherein the data flows from the physical twin to the digital twin and vice versa, allowing the digital twin to manage the physical twin [4]. Instances of digital twin utilisation are a digital twin represented by Mohammadi and Taylor [15], who created a digital twin of a city to visualise the different aspects of the town such as roads; this would help in town planning and management. Another instance of a digital twin used in urban planning was the digital twin represented by Lieberman et al. [11]. They developed a digital

twin used to create an underground city plan. Please note that the digital twin of a cloud data centre we propose in this study is designed to be used for monitoring and managing purposes.

8.2.1.2 Period of Production

The period of production for the digital twin can be classified into two parts. Digital twins developed before the creation of the physical twin, which are called *digital twin prototypes*, and digital twins created along with or after the creation of the physical twin called *digital twin instances* [6]. Ayani et al. [1] represented an example of a digital twin prototype; they used it throughout the configuration phase, which allowed their clients to give feedback and helped avoid costs related to prototyping. An example of a digital twin instance was represented by Luo et al. [12]. They created a digital twin of a milling machine to predict wear and tear and other issues to the milling machine under different conditions. In our proposed model, we develop a digital twin instance of a cloud data centre.

8.2.1.3 Connection Between Physical Twin and Digital Twin

Another characteristic of the digital twin is its connection with its physical counterpart. There are three ways of classifying the connection between a digital twin and a physical twin representing in what way the two parts are linked: firstly, *no connection*, wherein the digital twin and physical twin are not connected and have no association with one another, and secondly, a *unidirectional connection*, wherein the digital twin and the physical twin have a one-sided relationship where data from the physical twin is fed to the digital [3]. An example of this is represented by Luo et al. [12], where they use a unidirectional relationship to predict issues with the device used. Lastly, a *bidirectional connection* between a digital twin and a physical twin, where the data from the digital is fed to physical and vice versa, thereby creating a dual link that data is fed back and forth. For instance, this is represented by Malik and Bilberg [13], where they use a bidirectional connection to control a robot using its digital twin counterpart. Please note that our proposed digital twin of a cloud data centre is designed to have a bidirectional connection with the physical twin.

8.2.1.4 Building Blocks of a Digital Twin

Digital twins can vary in complexity depending upon features and input data. The classification is divided into five levels, with each level increasing in complexity and input data sources. Level one only has one input data source, which is the data model [17]. Level two digital twin is where physics is added to the data model of the level one digital twin creating a more elaborate digital twin. The third level is where a level one digital twin is incorporated with sensor data to increase the productivity

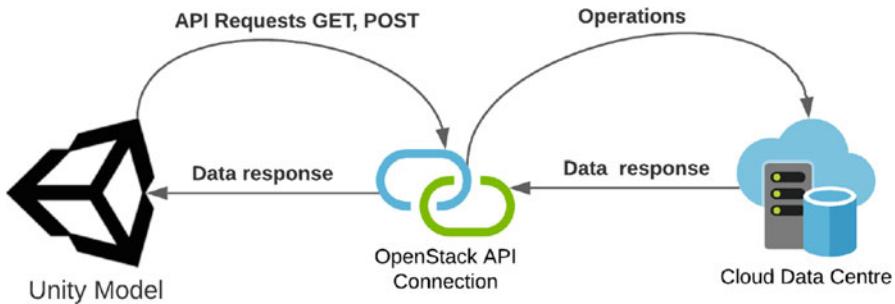


Fig. 8.2 System overview

of the model [14]. Level four has the level three digital twin enhanced with physics and expert systems. Level five is where enterprise systems are added to a level four digital twin [8].

8.3 Digital Twin of a Cloud Data Centre Instance System

To develop the digital twin instance, decisions needed to be made on various design options and other system choices on how the system would look and be developed. The proposed system consists of two main parts: the *unity model* and *API integration* (Fig. 8.2).¹ The two parts of the system need to work in unison to represent the cloud data centre effectively.

8.3.1 Unity Model

One of the primary components of the digital twin is the unity model, which would help visualise the cloud data centre in Unity3D software. The Unity3D software is used for developing full-fledged games for both 2D and 3D. It offers various resources and an abundance of assets primarily used for 3D modelling [18].

8.3.1.1 Main Components

The unity model's design is an integral part of the digital twin system. To ensure that the digital twin represents all the essential elements of the cloud data centre, we analysed the different parts of the cloud data centre and put together a model to

¹ Source code is available at <https://github.com/disnetlab/DatacentreDigitalTwin>.

ensure that the design standards are met. The main components of the cloud data centre that need to be represented in the digital twin are:

- *Hypervisors (physical servers)*: OpenStack supports software that allows the user (data centre admin) to manage physical servers by accessing the underlying hardware component. Hypervisor software gives the user the ability to manage and control the VMs hosted on a particular server [16]. In our proposed unity model, these hypervisors (physical servers) are represented as plates (Fig. 8.3). We leveraged a small-scale OpenStack-managed cluster at [DisNet Laboratory](#) in Monash University to build the digital twin. These servers are classified as hypervisors on the system and are real-world tangible objects. The size of the plate is proportional to the number of VCPUs (virtual CPUs) it supports. Since hypervisors in compute nodes support 32 VCPUs, each plate's dimensions are $x = 8$ and $z = 4$, where x is the width and z the length. As it is depicted in Fig. 8.4, each plate is divided into 32 grid sections, each representing a VCPU.

Power consumption of physical servers is also shown based on the RGB blue to red gradient colour scheme denoted energy consumption from low to high, respectively. We use existing ePDUs in the cluster to read live metering data, including power consumption in each server. We normalise the power consumption of each physical server based on a maximum value and then map the normalised power value to the gradient colour using the following formula:

$$C(R, G, B) = \begin{cases} (\lfloor(\tilde{p}/0.5) * 255\rfloor, 0, 255), & \text{if } \tilde{p} \leq 0.5 \\ (255, 0, \lfloor(1 - (\tilde{p})/0.5) * 255\rfloor), & \text{otherwise,} \end{cases} \quad (8.1)$$

where $C(R, G, B)$ is the colour in RGB scheme and \tilde{p} is the normalised power consumption of the physical server in the range of [0.1].

- *Virtual Machines (VMs)*: VMs are emulating a computer system that is created, stopped or terminated and hosts an operating system (OS) to execute user's applications. In OpenStack, VMs are called *instances* [7], and in the OpenStack APIs they are called *servers*. In our implementation, we represented VMs as boxes similar to the example shown in Fig. 8.3. To be a complete digital twin of

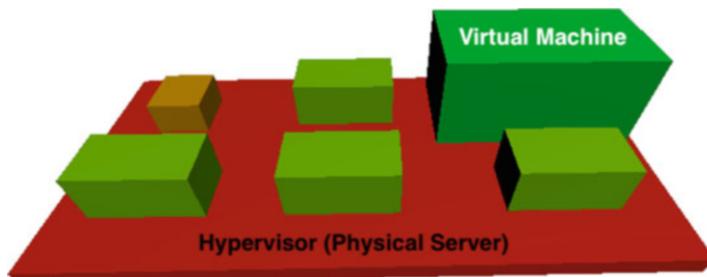


Fig. 8.3 Box (virtual machine) placement on the plate (hypervisor)

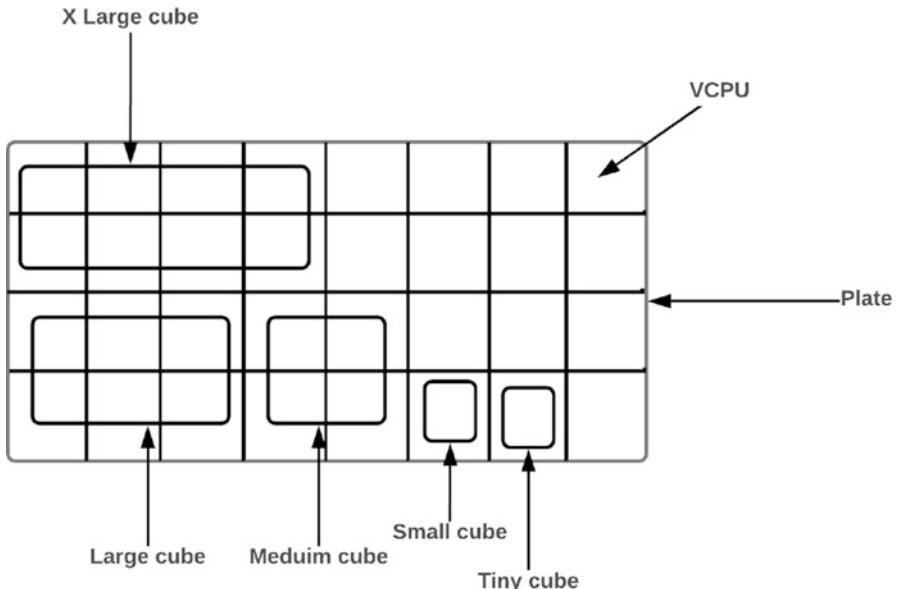


Fig. 8.4 Plate configuration

Fig. 8.5 Power state of a VM shown through transparency



the cloud data centre, the unity model has the boxes (VMs) placed upon the plates (hypervisors), giving a virtual, abstract and simplistic view of how VMs run on a real-world server. The placement of the boxes on the plate is essential for proper visualisation of the cloud data centre. Our proposed placement algorithm places the boxes on the plate until the sum of the boxes' size has reached the limit of the x -axis (right to left of the plate), in our case that is 8. The algorithm moves to the next row and changes the value of the z -axis for each box, and the process then repeats itself until all VMs are placed. Figure 8.3 depicts a sample placement.

Similar to plates, we use transparency to show a VM is suspended or shut off. The system displays boxes in a semi-transparent way to reflect that a VM is suspended or shut off in the digital twin. Figure 8.5 illustrates power on and off VMs as semi-transparent and solid boxes, respectively.

- *Flavours:* The user can configure various aspects of the instance, such as the number of VCPUs, memory and storage [16]. These aspects are represented on the digital twin as the boxes' size. Figure 8.6 shows different flavours represented

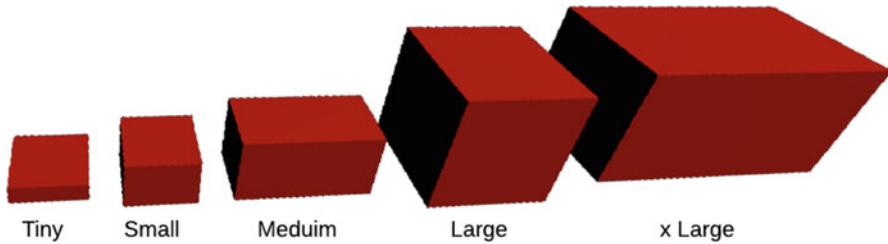


Fig. 8.6 Flavours are represented by various box sizes

Table 8.1 OpenStack default flavours details

Flavour	VCPUs	Disc (in GB)	RAM (in MB)
m1.tiny	1	1	512
m1.small	1	20	2048
m1.medium	2	40	4096
m1.large	4	80	8192
m1.xlarge	8	160	16,384

by various box sizes. The size of the base area of the box is proportional to the number of VCPUs (see Fig. 8.4), and the volume of the box is proportional to the size of the main memory. For simplicity and better visualisation, we do not represent the disk size of the VM in the model. OpenStack, by default, supports five different flavours, as shown in Table 8.1.

- *Projects*: Each instance belongs to a user project or tenant. In the proposed digital twin, the instances that belong to a certain projects are all represented in the same colour and with different colours for each project.

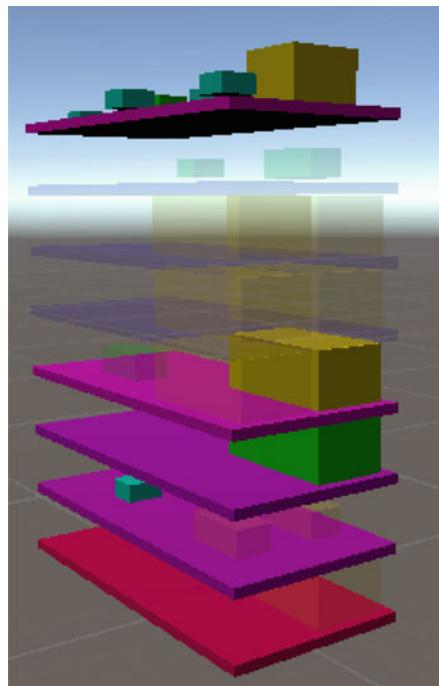
8.3.1.2 Digital Twin Interactive Features

The digital twin also offers a couple of *interactive features* to demonstrate the possibility of cloud data centre management and having the ability to perform interactive operations.

Start and shut off virtual machines and physical servers: The user can shut off or power on a VM or a physical server by long right-clicking on a box or plate, respectively. The box/server will start blinking during the time the switching up or down operation is happening. To know if the VM has successfully turned on or off, the unity model checks with OpenStack every second and compares the old status of the VM to the new status when the status changes, the whole model is instantly refreshed and shows the new changes. A similar process will be done for the physical servers (hypervisors).

Virtual machines migration: To migrate a VM from one hypervisor to another, the user might drag and drop a box from a plate to another. When the migration

Fig. 8.7 Complete model



is happening, the box keeps blinking until the migration is completed, and once migration is over, the box stops blinking, and the model is updated.

Figure 8.7 illustrates a complete unity model and displays how the digital twin represents a rack of servers in a cloud data centre in a 3D visualisation model. One can comprehend a large amount of information quickly by just looking at the model. For example, the bottom physical server/hypervisor is shown to have the highest energy consumption as the red colour represents it. Hypervisors 2, 3, and 4 from the top are switched off as they are transparent. One can also understand every VM's flavour, location, power state and the project they belong to by checking the model. The digital twin model also represents how much resources are used by VMs on different hypervisors.

8.3.2 API Integration

OpenStack provides a full range of different RESTful GET APIs to receive data from the cloud data centre and POST APIs which would allow the Unity model to interact with the cloud data centre, thereby ensuring that the digital twin would have a bidirectional relationship to its counterpart [7].

- GET Requests: The digital twin instance uses GET requests to fetch data from the cloud data centre and for monitoring purposes. The required data includes a list of servers (VMs or instances), a list of hypervisors, a list of flavours and a list of projects.
- Post APIs: The POST requests are needed to implement management and operational features of the digital twin. A POST request is used to manipulate data on the server-side or pass data from the digital twin to the server using API end points. For the interactive elements of the digital twin, we need to call a couple of POST requests. To perform authentication and authorisation, a POST request is sent to OpenStack's keystone service, which returns an authorisation token. This token needs to be used in all request headers for future GET and POST requests.

8.4 Qualitative Study

As part of the research process, an interview was held. The interview session's main aim was to get feedback and evaluate the effectiveness of the digital twin system compared to other existing monitoring and management tools, specifically the OpenStack dashboard. OpenStack dashboard was chosen for the comparison as it is the primary tool used to monitor servers in our cluster.

Study Procedure The study consists of a semi-structured interview process; this interview was conducted on participants selected on specific criteria listed below.

Participants The participants selected for the interview process are experts in the field of cloud data centres and possibly worked with OpenStack. We have selected twelve participants in the interview. The participants were from diverse backgrounds in terms of age, gender and field of work (academia and industry) and are selected according to years of experience, level of study and type of knowledge related to data centres and OpenStack. Table 8.2 shows more detail of each participant involved.

Interview Procedure First, a demonstration of the digital twin was shown to the participants. The software demo is available on youtube.² The demonstration was around 10 minutes and demonstrated all the system's functionalities, displaying and explaining all the visual components of the digital twin and interactive elements. After the demonstration, interview questions are asked. Table 8.3 shows these questions. The first group of questions were asked to gauge the familiarity and knowledge of the participants with the technologies used in creating the digital twin like Unity3D and cloud data centres. Then five questions in the second group

² <https://www.youtube.com/watch?v=pxFmUBuJJS8>.

Table 8.2 Participant details

Participants	Study (degree)	level	Experience with (OpenStack years)	Experience with cloud computing or cloud data centres (years) (type)
P1	PhD		2 years	2 years and work
P2	PhD		3 years	6 years and research projects
P3	Bachelors		8 years	15 years and work and projects
P4	Bachelors		11 years	11 years and work
P5	PhD		5 years	5 years and work and projects
P6	Bachelors		0 years	10 years and work
P7	PhD		5 years	7 years and work and projects
P8	Masters		1 years	2 years and projects
P9	Masters		10 years	15 years and work
P10	Bachelors		0 years	0 years and none
P11	PhD		4 years	4 years and work
P12	PhD		0 years	10 years and manager

Table 8.3 Interview questions*Questions to gauge familiarity and knowledge of each participant*

Q1	How would you rate your familiarity and knowledge of cloud data centres and OpenStack?
Q2	How would you rate your familiarity and knowledge of Unity3D?
Q3	How useful is the digital twin system compared to OpenStack dashboard for the system representation?

Survey style questions

Q4	How familiar is the digital twin system (with instructions)?
Q5	How user friendly is the digital twin system?
Q6	Is the digital twin easy to understand?
Q7	Is the digital twin likely to save time while monitoring?
Q8	How satisfied are you with the digital twin application?

Open-ended questions

Q9	What changes or additional features would you recommend?
Q10	What are the negative aspects of the digital twin?
Q11	What are the positive aspects of the digital twin?
Q12	Any Additional feedback?

of questions in Table 8.3 were asked to evaluate effectiveness of digital twin in management and monitoring of a data centre in a quantitative fashion, where the participants are instructed to answer questions on a Likert scale of 1–5 as follows: (1) very poor, (2) poor, (3) fair, (4) good and (5) very good. Lastly, the participants were asked four open-ended questions shown in Table 8.3.

8.5 Evaluation and Results

As it is shown in Table 8.4, the participants involved rated their familiarity and knowledge of cloud data centre and OpenStack technologies (Q1). Most participants gave a rating of 4 and 5, while there were 1 each for 1, 2 and 3 ratings, respectively. For this question, the mean was 3.82 and the standard deviation was 1.27. The scores indicate that most of the participating interviewees have an insight into the technology used and can provide valuable feedback regarding that. The participants were also asked about their knowledge and familiarity with Unity3D (Q2), with the majority of participants giving a rating of 3 or below, while none gave a rating of 5 with the mean of 2.58 and the standard deviation of 1.08. These scores show that the participants are not very familiar with Unity3D; this adds an element of usability where the user does not need to have experience with unity to understand or use the system. The participants also answered five survey style questions. Their answers highlight the usability, efficiency in terms of time-saving and familiarity with the system. For question Q3, most participants rated the digital twin system's usability relatively high with all participants rating 3 or higher with the mean of 3.92 and the standard deviation of 0.67. The next question Q4 was a question regarding the dimension of familiarity; here, most participants rated the system positively (4 and higher). However, P3 and P11 rated a score of 2 and 3 where they believed that improvements could be made on the side of familiarity by adding more instructions. For question Q5, all the participants gave a considerable high score, which demonstrates that the digital twin is user friendly and can be used even by people who might not possess much knowledge of cloud data centres. For this question, the mean was 3.67 and the standard deviation was 0.65. Question Q6 was also well rated by the participants, most giving a rating of 4 or higher where the

Table 8.4 Survey style questions and answers on a Likert scale of 1–5

Participants	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
P1	4	1	4	5	4	5	5	4
P2	4	3	4	4	3	4	5	4
P3	5	2	3	2	4	4	4	5
P4	4	3	4	4	3	3	3	3
P5	5	2	4	4	3	4	5	4
P6	2	1	3	4	5	5	4	4
P7	5	2	5	4	4	4	5	4
P8	3	2	5	4	3	4	5	3
P9	4	4	4	4	4	4	4	4
P10	1	4	3	4	3	5	4	4
P11	4	4	4	3	4	4	4	4
P12	5	3	4	4	4	4	3	4
Mean	3.83	2.58	3.92	2.83	3.67	4.17	4.25	3.92
Std. Dev.	1.27	1.08	0.67	0.72	0.65	0.58	0.75	0.51

mean of results is 4.17 with a standard deviation of 0.58. Showing that the digital twin performs well in being easy to understand. The next question Q7 dwells on how well the digital twin would help increase efficiency and save time. The majority of the participants gave a rank of 4 or more to this question (mean 4.25), showing that the digital twin system would help in saving time for a cloud data centre administrator. The final question in this section is regarding the satisfaction of the digital twin system. Question Q8 was once again rated highly by all the participants, demonstrating that they were satisfied with the digital twin system.

The final section of the interview consisted of four questions as listed in the bottom section of Table 8.3. These questions were open ended to give deeper insights and feedback into the digital twin system. The first question in this section is question Q9, and the purpose of this question is to gauge if the participants would like any change to be made to the digital twin system or/and add any additional features. Each of the participants had different responses to this question. Most of the participants recommended showing more data about the VMs, hovering over the boxes to show data like the name of the instance, and a search bar to search and highlight a specific VM (box) on the system. They also answered that they would like to see more data represented in the system, data such as which user the VM (box) belongs to. Also, they said that the current plates were homogeneous, and a heterogeneous plate configuration would be better as it would show more data about the hypervisors. Finally, they answered this question by stating they would like to see a VM's history in the digital twin system. Having this history available would help the cloud data centre administrator make better decisions about VMs' placement on different hypervisors during different periods. For example, in a community cloud data centre in a university, a hypervisor might be overloaded every year during the exam period, but during the semester break its load is below average; having this historical data can help the administrator make better decisions and make the system more efficient and save power.

The next question Q10 was regarding the negative aspects of the digital twin system. Some participants said that the system needs to show error messages in case something goes wrong. Others noted that the digital twin maintenance could be problematic, and the digital twin system cannot replicate all the data available on the OpenStack dashboard. One participant stated that the data centre admins will rarely move VMs between hosts in real life. This happens with automated scheduling tools and monitoring systems. Data centre admins interact with the hosts as a full cluster and specify a specific scheduling behaviour rather than specify where the VMs are. Finally, they stated that they would like to see more metrics before moving a VM from one plate to another. Knowing the metrics would help avoid the hypervisors getting crowded and increase efficiency and server utilisation.

The following question Q11 was to find out all the positive aspects of the digital twin. Some participants stated that the digital twin was easy to use, faster than searching for information on the dashboard as it is visualised. It is easy to find an empty hypervisor for migration, and all in all, it provides a good user experience. Others noted that the digital twin simplifies the cloud data centre's monitoring and management, easy for a user to use the whole system. It helps with the simplification

of cloud data centre monitoring and management. One participant added that the 3D view might be more useful as an asset management tool, i.e., servers within the racks, how many free rack units, and how many free network ports are available. Lastly, they answered that the digital twin makes migration and management easy with lots of potential in the future. The migration aspect would help spread the load to different hypervisors by viewing which hypervisor has less load on them or by checking the energy consumption based on colour density.

The final question of the interview was regarding any additional feedback. Feedback given by some was an extension to the project to add the digital twin as a plug-in into OpenStack. Others stated to show more colours, adding more metadata and add more detail to the hypervisor plate. Additionally, they stated to add pop-up windows to display more detail regarding the projects. Lastly, they suggested adding data from a *ceilometer* to add more data regarding the humidity and temperature of the real-world server rack. Another recommendation was to integrate the digital twin with *Ralph3*, which is an asset management system for data centres; it gives a different type of visualisation compared to the digital twin.

8.6 Discussion and Future Directions

The key takeaway of the qualitative study according to the participants' feedback indicates that the digital twin system is user friendly and easy to use. More importantly, the digital twin system helps with increasing efficiency and resource management decisions, such as looking at the plates and knowing which hypervisors have less load on them or overloaded. The digital twin can be extended and improved by adding more details and giving more details to the user, adding more functionality and integration with ceilometer and ralph3. Another key takeaway is that the digital twin cannot replace the OpenStack dashboard as a monitoring tool, not its intended purpose. OpenStack dashboard offers a great deal of information that cannot be all incorporated in a digital twin because of the limitation of the 3D model. However, digital twin provides high-level views of the data centre resources and its status based on OpenStack information which is hard to grasp using the dashboard. It can be used as an extension to the OpenStack dashboard and makes it more user friendly, easier to use and more efficient monitoring and management tool for data centre admins.

The feedback given by the participants of the study helped fuel the future outlook of the project. The following are the directions for future works of the research: firstly, adding a VR (virtual reality) component to the digital twin system. VR would give a different visual sense to the digital twin, using a VR headset to view in detail the digital twin in 3D and move around it to check the different components. One functionality that can be added is when the user clicks on a plate, all the boxes on the plate animate and moves to the space around the server rack where they are spread out in order. Doing this would make viewing the digital twin components easier. Other functionalities are showing error messages, having a search bar to

make it easier to search VMs, hovering a mouse over a box would display more details. For example, these features can be added: displaying the history of VMs and hypervisors in the model, making the digital twin a plug-in on the OpenStack and using a ceilometer to get the real-world server rack's temperature and humidity (e.g., the temperature can be shown as a red to a green hue). Another additional feature would be to show the network traffic and how much traffic is directed to each server. Lastly, this singular digital twin can be multiplied into multiple server racks beside each other, common in data centres.

8.7 Conclusions

This book chapter presented the instance of a digital twin built for OpenStack-managed cluster of servers. The system was designed to help cloud data administrators better monitor and manage different cloud data centre aspects. A 3D model was created to visualise different aspects such as VMs which are boxes in the system and hypervisors, which are plates. The digital twin also allows the user to use a couple of interactive features. To evaluate the effectiveness of the digital twin and gather feedback, a qualitative study was conducted where twelve participants were chosen who were experts in the field of cloud data centres and OpenStack. The participants were asked a series of questions to evaluate how the digital twin system performs compared to the OpenStack dashboard. The feedback was concise and broad, giving new insights into ways that the digital twin outperforms the dashboard to be more user friendly and less time consuming to navigate.

Software Availability

The current version of the software for the digital twin of a cloud data centre has been launched as an open-source project and the source code is available at <https://github.com/disnetlab/DatacentreDigitalTwin>. A software demo is also available at <https://www.youtube.com/watch?v=pxFmUBuJJS8>.

Acknowledgments We would like to thank all the participants who attended the demonstration and filled out survey forms. We also thank the Monash Human Ethics team for approving the ethics application. We thank the anonymous reviewers of the earlier version of this book chapter whose comments/suggestions helped improve this manuscript.

References

1. M. Ayani, M. Ganebäck, A.H. Ng, Digital twin: applying emulation for machine reconditioning. *Proc. CIRP* **72**, 243–248 (2018)
2. S. Boschert, R. Rosen, Digital twin—the simulation aspect, in *Mechatronic Futures* (Springer, Berlin, 2016), pp. 59–74

3. N. Demkovich, E. Yablochnikov, G. Aboev, Multiscale modeling and simulation for industrial cyber-physical systems, in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)* (2018), pp. 291–296. <https://doi.org/10.1109/ICPHYS.2018.8387674>
4. K. Dröder, P. Bobka, T. Germann, F. Gabriel, F. Dietrich, A machine learning-enhanced digital twin approach for human-robot-collaboration. Proc. CIRP **76**, 187–192 (2018)
5. T. Fifield, D. Fleming, A. Gentle, L. Hochstein, J. Proulx, E. Toews, J. Topjian, *OpenStack Operations Guide: Set Up and Manage Your OpenStack Cloud* (O'Reilly Media, Newton, 2014)
6. M. Grieves, Digital twin: manufacturing excellence through virtual factory replication. A White Paper by Dr. Micheal Grieves (2015)
7. R.H. Khan, J. Yilitalo, A.S. Ahmed, OpenID authentication as a service in OpenStack, in *2011 7th International Conference on Information Assurance and Security (IAS)* (IEEE, Piscataway, 2011), pp. 372–377
8. W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, Digital twin in manufacturing: a categorical literature review and classification. IFAC-PapersOnLine **51**, 1016–1022 (2018). <https://doi.org/10.1016/j.ifacol.2018.08.474>
9. R. Kumar, N. Gupta, S. Charu, K. Jain, S.K. Jangir, Open source solution for cloud computing platform using OpenStack. Int. J. Comput. Sci. Mobile Comput. **3**(5), 89–98 (2014)
10. H. Laaki, Y. Miche, K. Tammi, Prototyping a digital twin for real time remote control over mobile networks: application of remote surgery. IEEE Access **7**, 20325–20336 (2019). <https://doi.org/10.1109/ACCESS.2019.2897018>
11. J. Lieberman, A. Leidner, G. Percivall, C. Rönsdorf, Using big data analytics and IoT principles to keep an eye on underground infrastructure, in *2017 IEEE International Conference on Big Data (Big Data)* (IEEE, Piscataway, 2017), pp. 4592–4601
12. W. Luo, T. Hu, W. Zhu, F. Tao, Digital twin modeling method for CNC machine tool, in *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)* (IEEE, Piscataway, 2018), pp. 1–4
13. A.A. Malik, A. Bilberg, Digital twins of human robot collaboration in a production setting. Proc. Manufacturing **17**, 278–285 (2018)
14. R. Minerva, G.M. Lee, N. Crespi, Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models. Proc. IEEE **108**(10), 1785–1824 (2020). <https://doi.org/10.1109/JPROC.2020.2998530>
15. N. Mohammadi, J.E. Taylor, Smart city digital twins, in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (2017), pp. 1–5. <https://doi.org/10.1109/SSCI.2017.8285439>
16. T. Rosado, J. Bernardino, An overview of OpenStack architecture, in *Proceedings of the 18th International Database Engineering & Applications Symposium* (2014), pp. 366–367
17. F. Tao, M. Zhang, Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. IEEE Access **5**, 20418–20427 (2017)
18. J. Xie, Research on key technologies base unity3d game engine, in *2012 7th International Conference on Computer Science & Education (ICCSE)* (IEEE, Piscataway, 2012), pp. 695–699

Part II

Internet of Things

Chapter 9

Industrial IoT Technologies and Protocols



Rahul Devkar, Princy Randhawa, and Mahipal Bukya

Contents

9.1	Introduction	230
9.1.1	What Is IoT?	231
9.1.2	The Paths for Data Transmission Through the Connectivity and Networks	232
9.1.3	Baseline Technologies	232
9.1.4	Connectivity Terminologies	234
9.2	Network Topology	234
9.3	Addressing Manners	237
9.4	Sensors and Transducers	237
9.4.1	Types of Sensors	238
9.5	Components of IoT	238
9.5.1	Communication Protocols	239
9.5.2	IEEE 802.15.4	241
9.5.3	Zigbee	241
9.5.4	Wireless HART	243
9.5.5	NFC Communication Protocol	244
9.5.6	Bluetooth Communication Protocol	245
9.5.7	Bluetooth Piconets	247
9.5.8	Z-Wave	247
9.5.9	ISA 100.11A	248
9.6	Conclusion	250
	References	250

R. Devkar · P. Randhawa (✉)

Department of Mechatronics Engineering, Department of Electrical Engineering, Manipal University Jaipur, Jaipur, Rajasthan, India
e-mail: rahul.189403068@muj.manipal.edu

M. Bukya

Department of Electrical and Electronics Engineering, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, Karnataka, India

9.1 Introduction

In the twenty-first century, technology is leading the domain of data collection and distribution. The “users” of this technology, that is, the Internet users, were/are counted in billions (as shown in Fig. 9.1) and where humans played a major role in generating and receiving data. Thus, we needed a much broader and more complex system which was named the Internet of Things. Even back then, the number of interconnected devices surpassed that of people on earth, according to the latest data collection, we conclude:

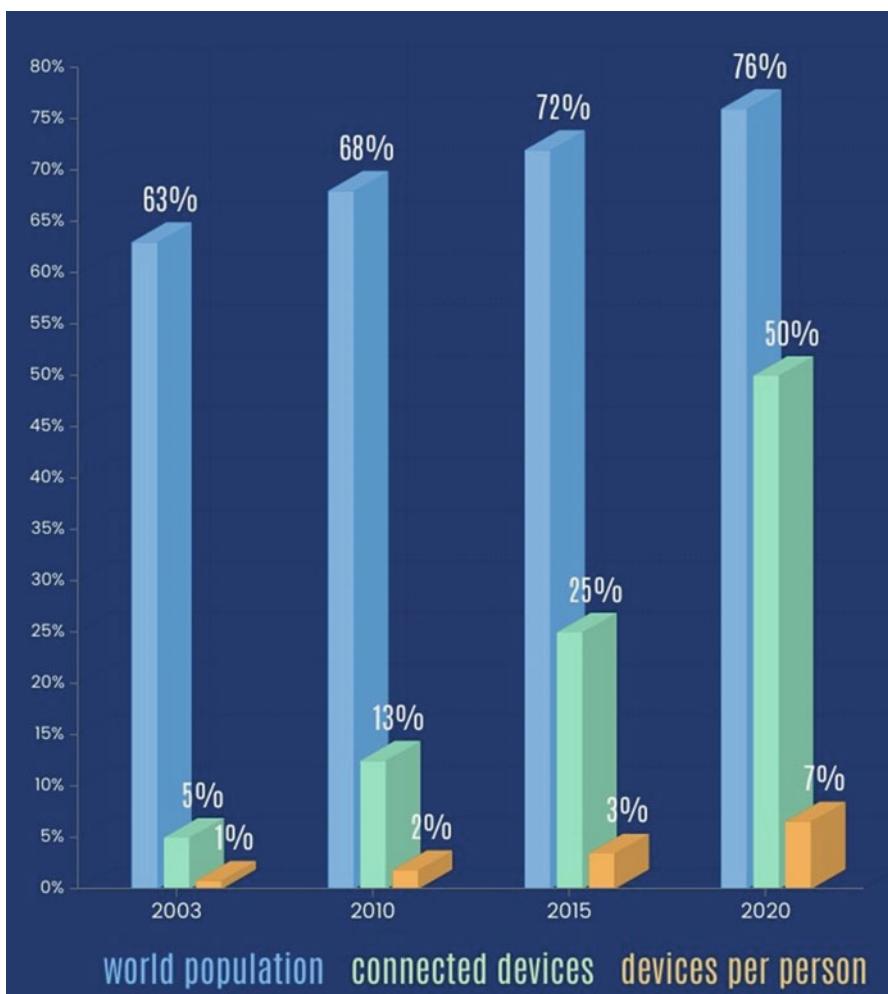


Fig. 9.1 Trends in increasing number of IoT devices

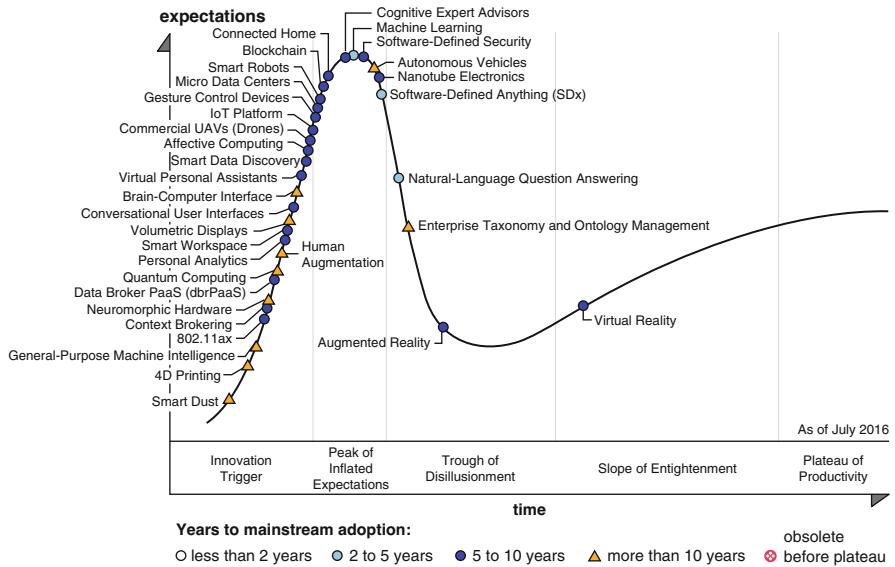


Fig. 9.2 Gartner Hype cycle describing IoT in emerging trends [2]

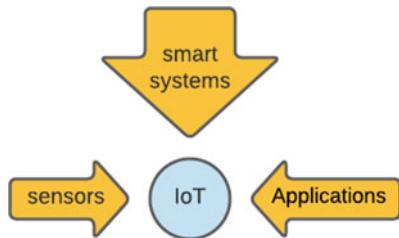
Today, collaborating with every device is the factor that IoT is required in every emerging technology. IoT platforms are still on the top trends in this year's Gartner Hype Cycle because of their use in virtual assistants, homes automation, and super self-driving cars. Figure 9.2 shows that this technology will soon reach its productivity level in 8–10 years [1].

Thus, there is a great need for developing and interconnecting IoT with hardware machines for better performance. We need several different protocols for developing this through which data can be communicated from one place to another through different mediums.

9.1.1 What Is IoT?

The Internet of Things (IoT) is a system of network of electronic objects that contains embedded technology with the Internet to communicate and sense or interact with internally or externally connected objects [1]. With the help of the Internet using wireless technologies, over 11 billion of “Things” (where the word “things” refer to the embedded devices) are connected and communicated. It is estimated that 20 million things would connect to the Internet soon. The emerging technologies in IoT are low-power embedded system cloud computing big data machine learning and networking [2].

Fig. 9.3 Concept of IoT



Modern-day applications of IoT are as follows [14]:

Forest fire detection	Air pollution
Snow level monitoring	Landslide and avalanche prevention
Earthquake early detection	Water leakages
Smart parking	Structural health
Noise urban maps	Smartphone detection
Traffic congestion	And many others

The basic concept of IoT is described in Fig. 9.3.

9.1.2 *The Paths for Data Transmission Through the Connectivity and Networks*

- *Nodes*: They are edge devices or end nodes that form the edge of the IoT ecosystems. It is a combination of hardware and software where sensors are used to capture data and send/pass it to the further networks [3].
- *Gateway*: This is a *medium* through which data (from the sensors) is transmitted/received from the cloud/internet. It is used to bridge two different environments to allow them to exchange real-time data and control of devices [2].
- *Router*: This is a physical embedded device that connects the data coming from the gateway to the Internet wirelessly.
- *Internet*: A virtual world where data can be stored and extracted (Fig. 9.4).

9.1.3 *Baseline Technologies*

There are several technologies that are very closely related to IoT [4]. But some major ones are as follows:

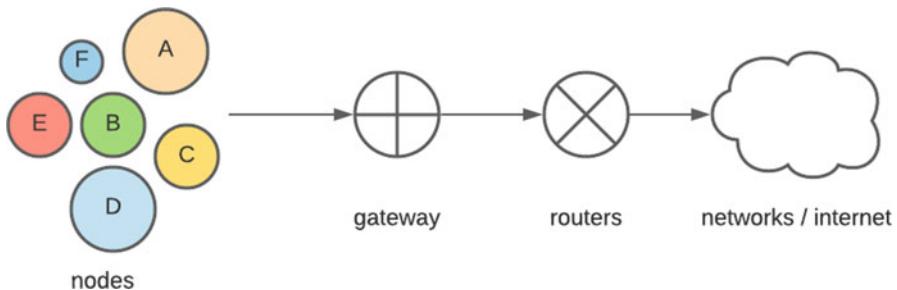


Fig. 9.4 Working of nodes with gateways, routers, and Internet

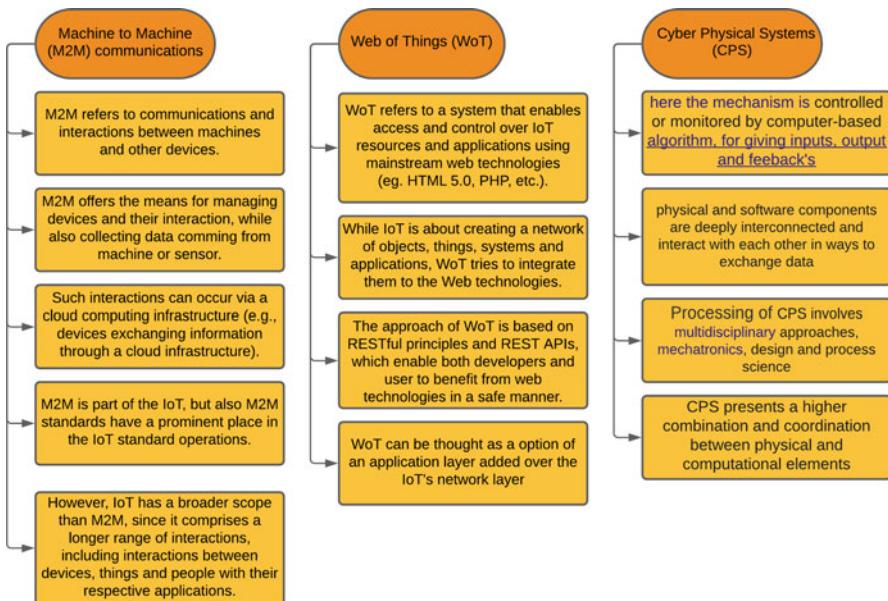


Fig. 9.5 Definitions and differences between M2M, WOT, and CPS

- Machine-to-Machine (M2M) communications [5, 6].
- Cyber-Physical-Systems (CPS).
- Web-of-Things (WoT).

Features and differences of each are mentioned in Fig. 9.5.

9.1.4 Connectivity Terminologies

Connectivity plays a major role in transferring data from one device to another; there are different segments/checkpoints through which the data hops to ensure its correct addressing and security [7].

Different terminologies related to the connectivity of devices are listed below:

- *Local area network (LAN)*: Is a connection of things (which are connected together) in a physical location, where the data is transferred from one device to another without the interferences of Internet like in a building, office, or home. A LAN network can be very small or large, depending upon the number of users like a home network with one to two devices to an office or industrial network with hundreds of users and devices in a limited area [8].
- *Wide area network (WAN)/Metropolitan area network (MAN)*: An integration of two or more remote networks connected by public communication methods. Some WANs connect many LANs together and cover larger geographic areas. WAN can be a collection of LANs or other connectivity networks (like proxy server) that communicate with one another. The Internet can be considered as the world's largest WAN. Today, thus WAN has a variety of uses [4].
- *Proxy servers*: Proxy server is a dedicated hardware or a software system that connects the device/thing to the enterprise network, like sensors, nodes, or other embedded software to the Internet. Many IoT-related problems need an interface that collect data for analysis or provide command and control functions from the cloud, thus proxy servers are used. It is a server, referred to as an “intermediary gateway” because it goes between end-users and the web pages they visit online.
- *Topologies used by LAN networks*: Tree, Cluster-Tree, Mesh, and Star.

Figure 9.6 describes each terminology and Fig. 9.7 shows the connecting topologies.

IoT Gateways with or without proxies responsible mainly for:
Here in the image, L = LAN network, G = GATEWAY.

9.1.4.1 Connectivity Layers

- Service.
- Local connectivity.
- Global connectivity.

9.2 Network Topology

A *network topology* is how a device gets connected to other devices for carrying out the function of data transfer [6]. Understanding what is topology through the sense

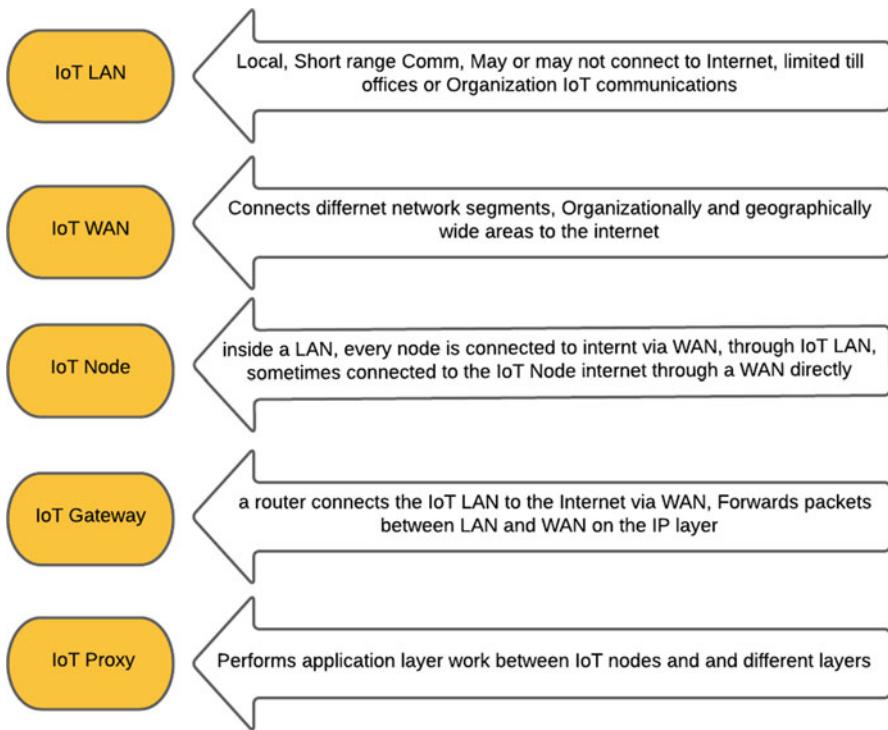
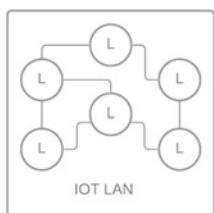


Fig. 9.6 Description of LAN, WAN, NODE, GATEWAY, and PROXY

- Internet connectivity



- IoT LAN intra-connectivity

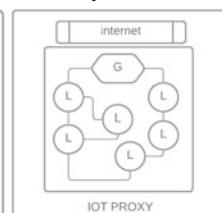
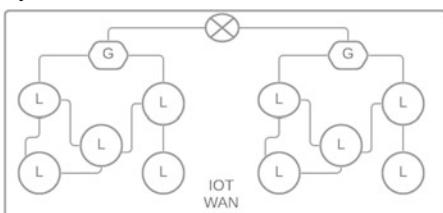


Fig. 9.7 Connecting topologies of LAN, WAN, and PROXY SERVERS

of “Internet of Things;” it means how sensors, actuators, and gateways communicate with one another [9].

There are some topologies that are generally used in a IoT network, which are described with their connectivity in Fig. 9.8:

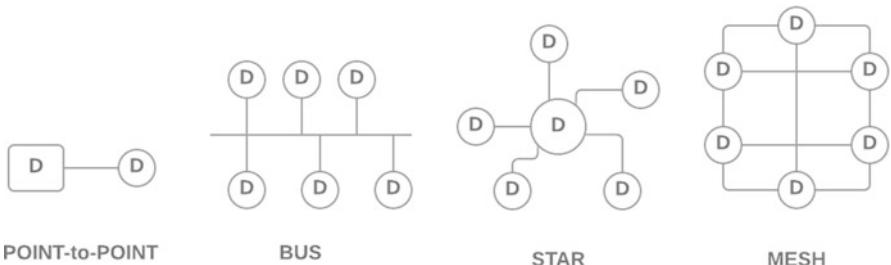


Fig. 9.8 Connectivity of different topologies

- Point-to-point.
- Bus.
- Star.
- Mesh.

Point-to-Point Network Topology

It is a direct/unidirectional connection *between two network nodes*. Here, the data transfer and communication takes place only between two selected nodes or devices, for example, Bluetooth *connectivity* between a mobile phone and a speaker.

Bus Network Topology

Here every node is connected to a *single/parent cable* through the junction connector, where each device has its own address to send and receive data. This parent cable is basically the *backbone* of the network and is called as bus [10].

The data or the signal from the transmitter travels in bidirectional manner in every device connected on the bus cable until it finds the correct address location. If the address of the machine does not match to the address given in the data then the machine ignores that data.

Star Network

Here in Star network, all the other nodes (like sensor nodes) are connected/linked to one central hub (called gateway node). So the central hub acts as the common connecting device for all the other nodes/devices in this protocol. Thus, all the other nodes have to rely on central hub for communicating (i.e., for transmitting and receiving data).

Mesh Network Topology

Here, the nodes are arranged in such a manner that each node, which is within the transmission range, communicate with each other for the data transmission. Here the output of a node can be an input for another node. Here, each data packet passes through many sensors/router nodes to reach the gateway/central hub [11].

Table 9.1 Technical differences in IPv4 and IPv6

Properties	IPv4	IPv6
Developed	IETF 1974	IEF1998
Length (bits)	32	128
Notation	DOTTED DECIMAL	HEXA DECIMAL
No. of addresses	2^{32}	2^{128}
Dynamic allocation of address	DHCP	DHCPv6
IPSec	Optional	Compulsory

9.3 Addressing Manners

It is considered that there are around 4 billion devices that connect to IoT [7], thus sending a specific data to its specific address location becomes challenging. Thus, there should exist a rule for setting address to every device. Thus, we use IPV4 and IPV6 addressing modes.

Internet Protocol Version 4 (IPv4): *IPv4* is used for packet-switched link layer networks (like Ethernet). The amount of address location that it creates is approximately 4.3 billion. The IPv4 emphasizes more on reliable transmission, as it's evident by fields such as type of service, total length, id, offset, TTL, and checksum fields [12].

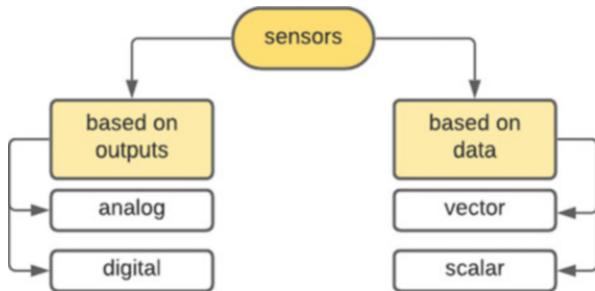
The Internet Protocol Version 6 (IPv6): *IPv6* is much better and advanced as compared to IPv4. The amount of address location that it creates is infinite in number. This is so efficient that, currently, it is replacing ipv4 to accommodate the increasing amount of networks worldwide, thus also helps in solving the problem of addresses getting exhausted. The IPv6 header structure is simpler as it mainly focuses on the addressing part of the source and destination. It works more on addressing than with reliability of data delivery.

Technical differences are described in Table 9.1.

9.4 Sensors and Transducers

Sensor: Sensor is a device that detects the change in the surrounding environment or the state of a device or system and forwards this information in a certain manner for further processing [13].

A sensor is defined for a specific property, thus it only senses the measured property and ignores the other properties so that other properties do not influence the measured property. The resolution of a sensor: This means the smallest change which a sensor detects in the quantity that it is measuring [9].

Fig. 9.9 Types of sensors

Transducers: Transducer is basically a device that converts energy from one form to another, like solar panel that converts solar energy into electrical energy or a loudspeaker that converts electrical signals into sound waves.

Both sensors and transducers can be used to sense a wide range of different energies and mechanically actuated using actuators which can be used to switch voltages and currents.

9.4.1 *Types of Sensors*

There are majorly two different types of sensors, as shown in Fig. 9.9.

- *Analog sensors:* These sensors produce a continuous voltage or signals (at output) which are proportional to the measured quantity.
- *Digital sensors:* These sensors only produce two outputs that is 0 or 1 (digital output) which are the digital describes or representation of the value which is been measured.
- *Scalar sensors:* Here the produced voltage or the signals are proportional only to the magnitude of the quantity which is been measured.
- *Vector sensors:* Here the produced signal or voltage is proportional to the magnitude as well as the direction and the orientation at which the quantity is being measured.

9.5 Components of IoT

In an IoT network, transmitting and receiving data from one node to another needs involvement of many components, which are responsible for the data transmission speed and the security of the network [2]. These components also look after the data allocation and management, backend services, and application management. Thus, these components make the whole IoT system. Those components work in this following manner as shown in Fig. 9.10.

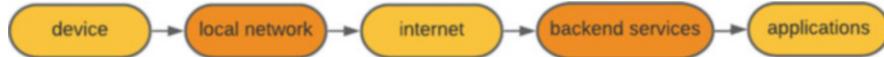


Fig. 9.10 Working of components of IoT

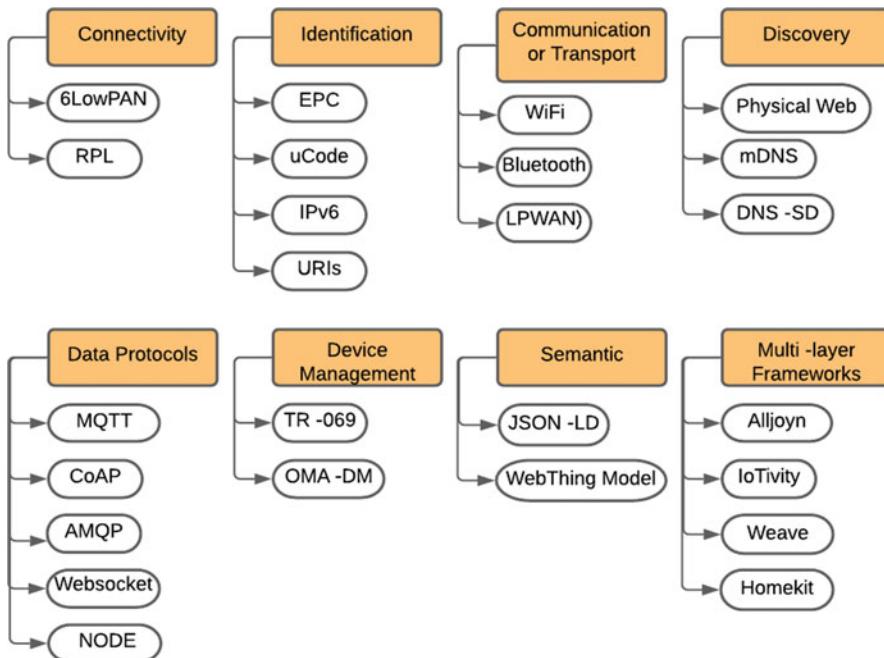


Fig. 9.11 Different protocols w.r.t communication needs

9.5.1 *Communication Protocols*

For an Internet of Things (IoT) system, the basic need is transferring data continuously, therefore the concept of selecting the correct connectivity arises; as IoT is a very wide and diverse system of protocol for connectivity, so we need to find the best suitable communication protocol for a smooth and a secure working of any network [4]. Figure 9.11 describes these communication protocols based on their specific qualities [14].

The abovementioned connectivity protocols have their own strengths and weaknesses depending upon the network criteria; therefore the engineer has to choose them wisely for their networks, for example, making a brain image processing unit involves sensitivity and accuracy [15].

In this chapter, we will be considering some major connectivity protocols (mentioned in Fig. 9.12) which are widely used in big IoT networks and industrial IoT communications [16].

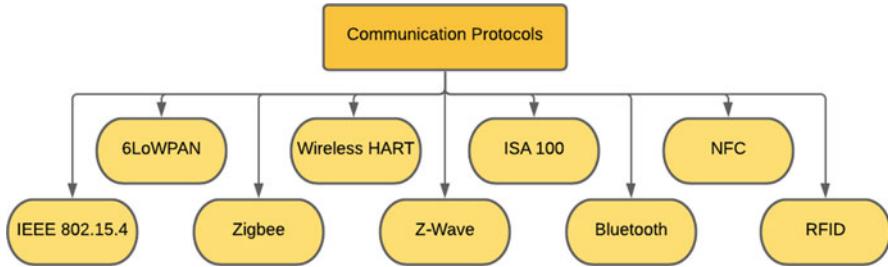


Fig. 9.12 Communication protocols

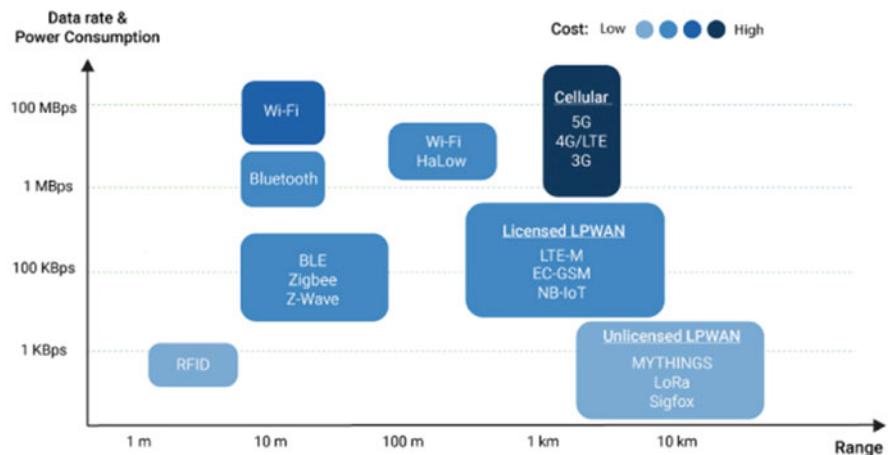


Fig. 9.13 Data rate and power consumption and range of different protocols [18]

All these connectivities have the own level of data transferring rates and power consumption levels as listed in Fig. 9.13.

Data Transferring Rates: The speed at which data is transferred between a peripheral device and the computer; these are generally measured in bytes per second [7].

Power Consumption Level: The electrical energy supplied to operate the hardwares in a given unit time. Power consumption is generally measured in units of watts (W) or per unit time, kilowatts (kW). The energy use is generally more than the energy required because of changes in environmental parameters [7].

9.5.2 IEEE 802.15.4

The IEEE protocol forms the fundamental lower network layers of a wireless personal area type network (LR-WPANs), it basically focuses on low-cost as well as low-speed ubiquitous communication between two devices [2].

Typically, they have many differences when we compare with other protocols such as Wi-Fi which have more bandwidth with less power consumption. Thus, IEEE 802.15.4 basically works on low-power consumption and communication rate. This protocol forms the base of

Zigbee	Wireless HART
ISA100.11a	6LoWPAN

This protocol uses only the first two layers (PHY, MAC) are used in this protocol, with LLC (logical link control) and SSCS (service specific convergence sub-layer) used to connect with all higher levels in the ISM band. It also uses DSSS (direct sequence spread spectrum modulation) [17].

Advantages:

- Resolves highly tolerant of noise and interference and also offers reliable link mechanisms. Uses CSMA-CA (carrier sense multiple access with collision avoidance) for every channel access.
- Interference free access to the same channel that can be given to multiple users or nodes by multiplexing, which allows the user to access at the different times.
- The power consumption can be minimized by in-frequency occurring which happens due to short packet transmission with very low duty cycle.
- Networking topologies used here are Star and Mesh.

Figure 9.14 shows the device connection between FFD and RFD.

9.5.3 Zigbee

Zigbee is the most popular and widely used enhancement of IEEE protocol; this protocol is defined by the 3 layer and above. This standard uses different layers like layers 3 and 4 which define the additional communication development like security, data routing, and node healing [2].

It is the most widely deployed enhancement of IEEE 802.15.4. The Zigbee protocol is defined by layer 3 and above, whereas for the 1 and 2 layer it uses IEEE 802.15. The standard uses layer 3 and 4 to define additional communication development like authentication with valid nodes, security, and a data routing [18].

Networking topologies defined are *Mesh*.

Full Function Device (FFD):

- Can easily communicate with every type of device
- Supports all type of protocols

Reduced Function Device (RFD):

- Can only communicate with FFD present in the topology
- Can work on very low power sources
- Minimal CPU/RAM consumption

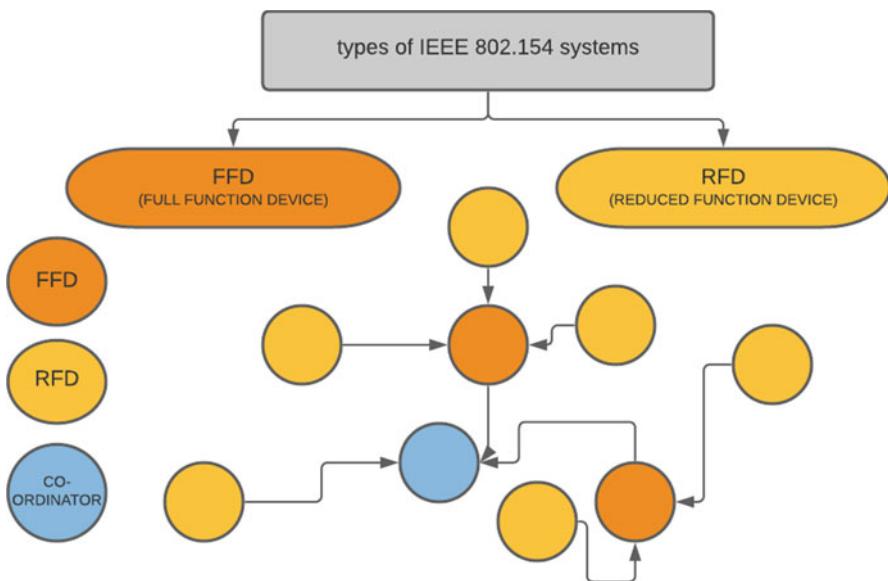


Fig. 9.14 Types of IEEE systems

Networks

The network layer uses AODV (Ad Hoc On-Demand Distance Vector routing). To reach the final location, here AODV makes a routing request to all its immediate nodes. The neighboring nodes relay the same information to the nodes in their peripheral, eventually spreading the request throughout the network.

COMPONENTS OF ZIGBEE: Fig. 9.15 shows different components of ZIGBEE.

Zigbee Types

Zigbee Coordinator (ZC):

- The basic work of a coordinator is to form a network tree and to act as a bridge between two networks. This network has a single Zigbee coordinator which would initially start the network. The major role lies in storing the information of each node connected to the coordinator.
- Does Zigbee coordinator have to work as a repository to secure keys for each node.

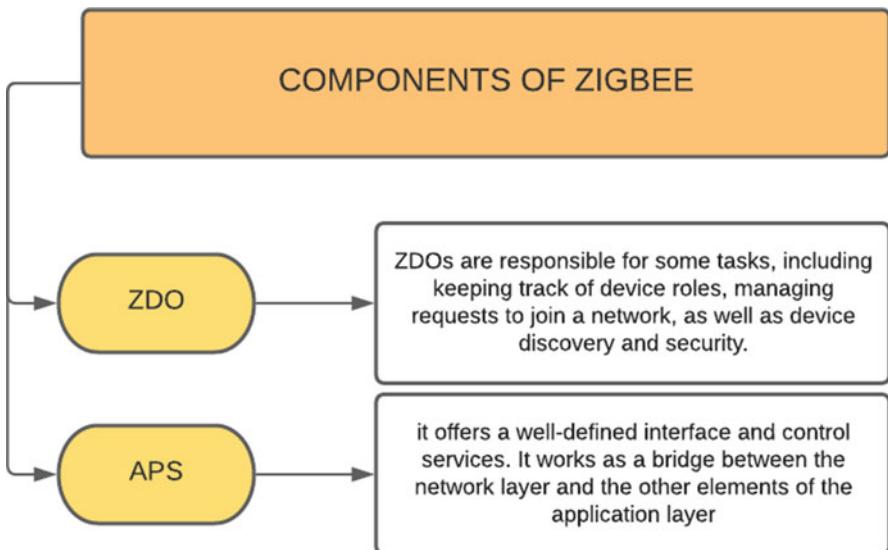


Fig. 9.15 Components of Zigbee

Zigbee Router (ZR):

Here the application layer works, as well as transferring of information from one node to another is done.

Zigbee End Device (ZED):

- ZRD only have the permission to talk to the parent node, thus it cannot transfer information from one node to another node. It also has an inbuilt sleeping mode function which enhances power consumption.

9.5.4 Wireless HART

HART stands for Highway Addressable Remote Transducer Protocol standard and was developed for communication protocol in a network. This is a cheaper and easier protocol among all those available. Devices that work on wireless HART devices can be more convenient in terms of placement, as they can be used for small mechanisms such as correction robots inside a pipe to large implementations such as communication within a large warehouse. The physical and datalink layer distinguishes the wired and invite versions of wireless [4]. Wireless HART have different network layers like:

Congestion Control in HART Protocol

It uses a 2.4 GHz ISM band and also has an algorithm to avoid Interference prone channels by using channel switching post for every transmission. A transmission

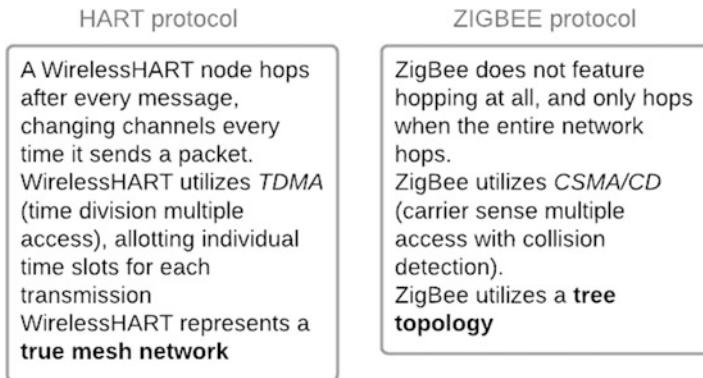


Fig. 9.16 Differences between HART and ZIGBEE

synchronizes every after 10 ms. Wireless HART Network Manager Decides who will send, who will listen, and at what frequency is each timeslot. This also Handles code-based network security and prevents unauthorized nodes from joining the network [16].

Differences between HART and ZIGBEE protocols are shown in Fig. 9.16.

9.5.5 NFC Communication Protocol

NFC or near field communication is similar to radio-frequency identification (RFID). NFCs were 8376888888 designed in such a manner that the protocols get active only when the device is within the range of close proximity of each other; thus, it is power efficient also. All NFC types are similar but communicate in slightly different ways [10].

Types of NFC communication

Active	Passive
1. Active devices can collect and transmit information.	1. Passive devices only transmit the information that they contain, but cannot read it by itself.
2. They have their own power source.	2. It uses the energy from the reader when they come in contact with them.
3. For example, Smartphones are a good example of active devices.	3. For example, NFC tags found in supermarket products are examples of passive NFC.

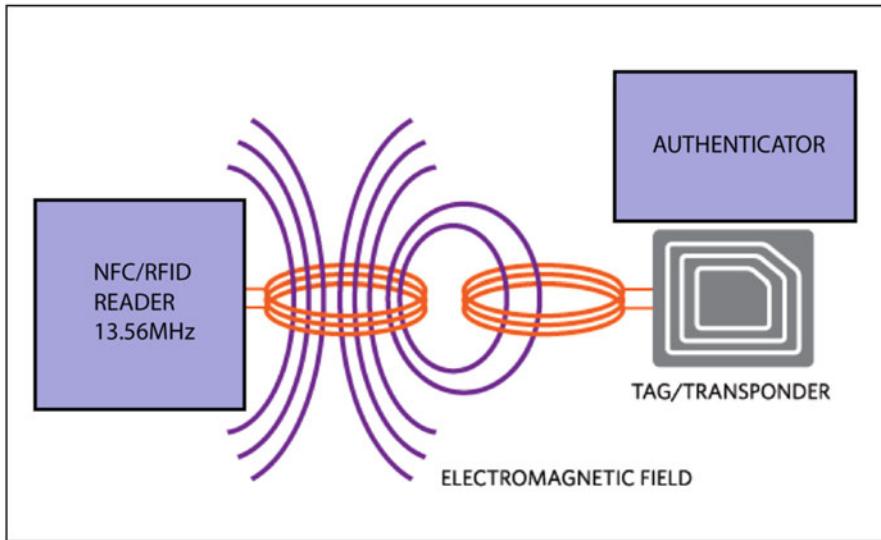


Fig. 9.17 Working of NFCs [21]

Working Principle

NFCs and RFIDs both work on simple and same principle of magnetic induction. A physical space is built, when the reader emits small electrical current that acts like a bridge between two devices. The generated field created by the reader is received by a similar coil present in the slave device where it turns the electrical impulse for further communication, such as identification number on identity cards. Figure 9.17 shows the electromagnetic fields with the schematic for the interaction between tag and reader of an NFC.

Features of NFCs

- It used 13.56 MHz of frequency for data-transmission.
- Data transmitting speed: 106, 212 or 424 Kbps (kilobits per second).
- Data storage capability in every tag is 96–512 bytes.
- Communication range is less than 20 cm.

9.5.6 Bluetooth Communication Protocol

This communication protocol works over a small area as it has a small range, thus created for replacing cables connecting portable units. This communication maintains high levels of security [9].

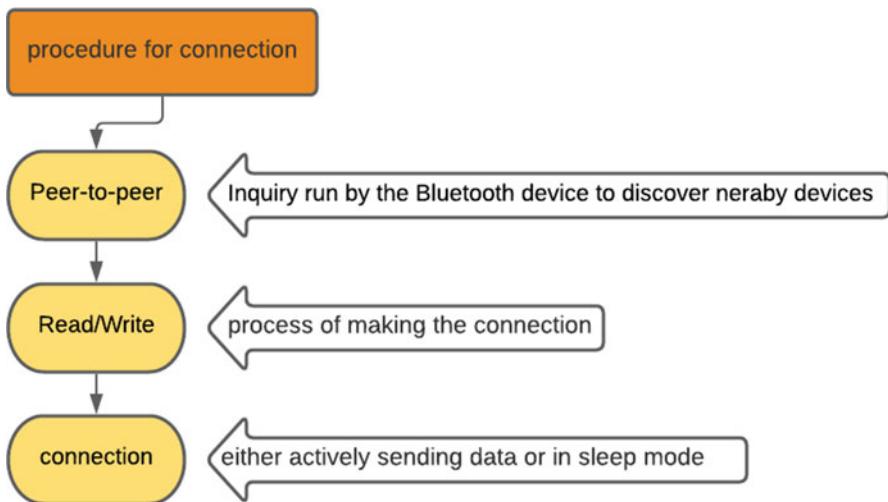


Fig. 9.18 Connection procedure of Bluetooth device

Features of Bluetooth technology

- Bluetooth technology originates from ad-Hoc technology which is also known as ad-hoc Piconets.
- This can be used for various purposes like home automation, industrial automation, and office communications with a frequency range of 2.4–2.485 GHZ.
- This protocol uses spread spectrum hopping principle (full-duplex signal) at a rate of 1600 hops/s.
- Bluetooth supports 1 Mbps data rate to 3 Mbps data rate with combined Error Data Rate for different versions of modules (Fig. 9.18).

Modes of operations	Layers of operation
Active	Software application layer
Sniff	Middleware layer mostly for security purposes
Hold	Data link layer to regain the lost data packets
Park	Physical layer

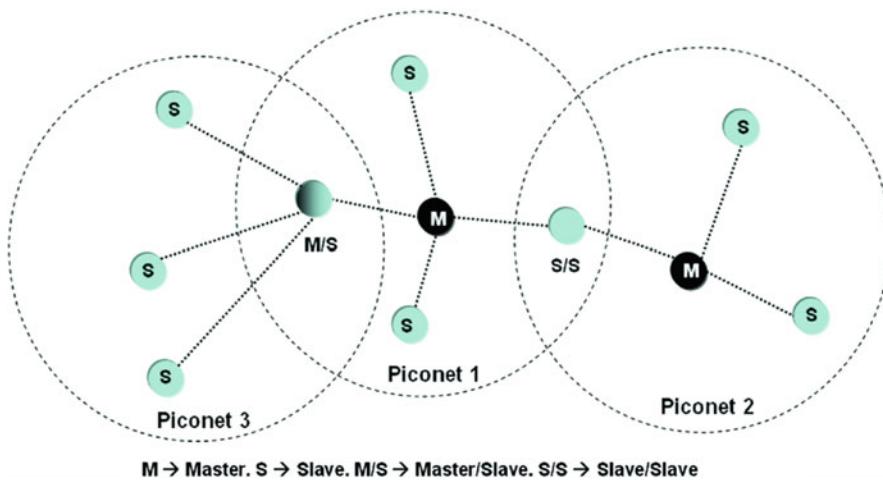


Fig. 9.19 Multiple connectivity of Piconets [22]

9.5.7 Bluetooth Piconets

- When multiple devices communicate with each other, working on Bluetooth communication protocol then the network is said to be Piconets. Piconet enables electronic devices to connect and communicate to other devices wirelessly with the short-range network. It uses topology star where all the slaves connect to only one master.
- Master establishes the Piconet and is also responsible for data transmission control over the other devices.
- A Bluetooth device works on small ad-hoc configurations where they can work either as master or slave.

Features of Piconets

- Within a Piconet, the master plays the major role, as the clock and 48-bit address of master device determines the frequency hopping sequence and the timing of various devices of each node or slave device.
- Each Piconet device supports 7 simultaneous connections to other devices.
- A device can be a part of multiple Piconets (Fig. 9.19).

9.5.8 Z-Wave

Z-wave is one of the best suitable and handy protocol for communication among devices in home automation. It basically uses radio frequencies for signaling, data transfer, and for controlling the node devices in the network [4].

There is different Operating frequency for different countries, like in the USA, it is 908.42 and 868.42 MHz in Europe. It can support 232 nodes in a network.

Networking topologies defined are Mesh.

Technical Working

- Z-Wave works on Manchester channel encoding and GFSK modulation.
- The coordinator (a central network controlling device) manages the whole Z-wave network which consist various nodes.
- Each network working in an area has its own ID (address) and each node has their separate IDs for the devices in the network.
- Nodes of other Z-wave networks cannot communicate with each other. Network ID length = 4 Bytes, Node ID length = 1 Byte.

Here mesh topology is used with the *concept of healing* works, that is, the dead spot is being by-passed and the data is being sent to the targeted node irrespective of any obstruction in the path.

Z-wave has a different working frequency and uses static routers, thus mobile devices networks are excluded, and only stable static devices are considered in the network.

Z-wave is closely related to Zigbee protocol, but still there are differences between them, as follows (Fig. 9.20).

9.5.9 ISA 100.11A

ISA stands for *International Society of Automation*, which is an organization that has designed this protocol mainly for large-scale industrial applications. This protocol supports native and tunneled application layers, with various communication services, like reliability, real-time, and security [4].

- Both transport and Network layers work on IPv6.
- Both, mesh routing and Frequency hopping is supported by the data link layer.
- Physical and MAC layers uses IEEE 802.15.4.
- Networking topologies used here are Star, tree, and Mesh.

Different networks that work under ISA100.11A are as follows:

- Radio frequency link.
- ISA over Ethernet [11, 19].
- Field buses communication [20].

Application layer is responsible for communication services for the user and management processes which makes the systems smooth. For the use of legacy data, tunneling mode is available in the ISA100.11A network (Fig. 9.21).

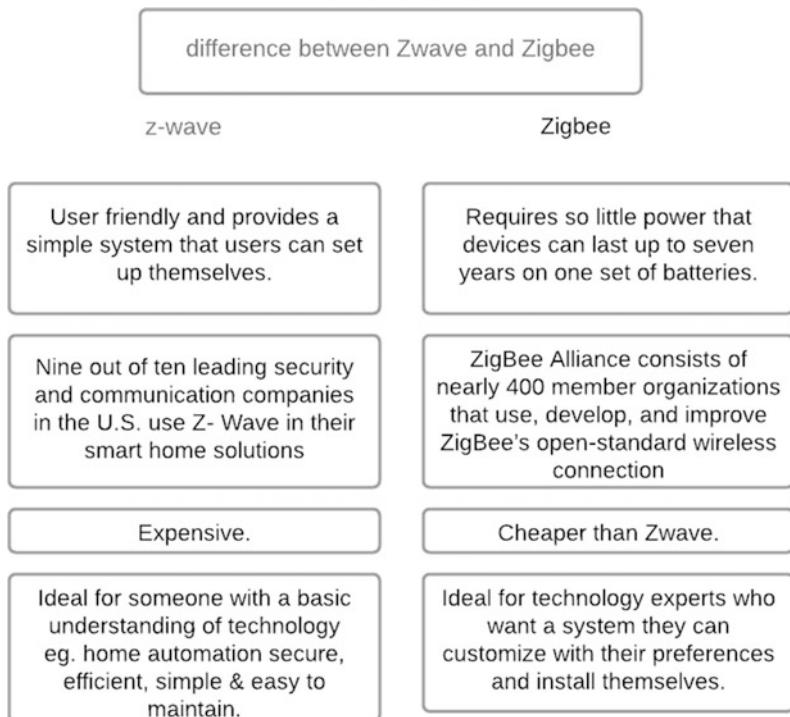


Fig. 9.20 Differences between Z-wave and Zigbee

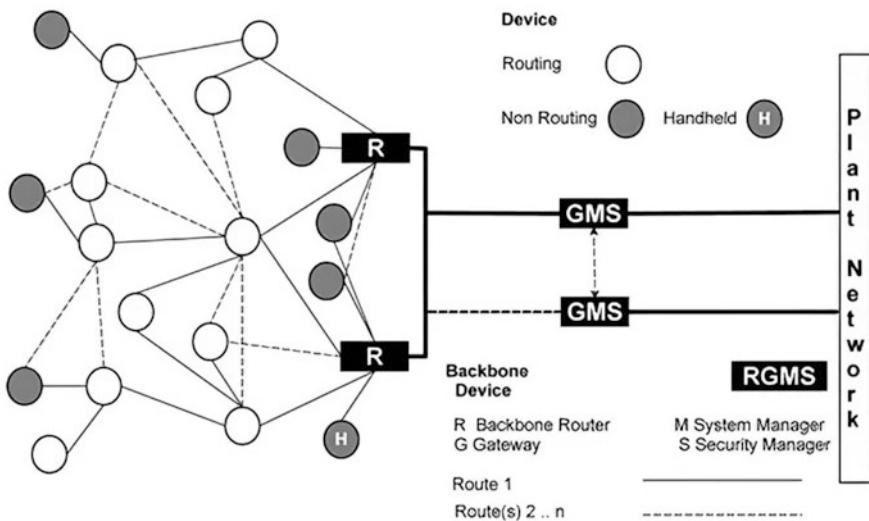


Fig. 9.21 Working of ISA network protocol [24]

Features of ISA 100.11A

- Flexibility in node hopping and working.
- Can be used for multiple protocols.
- Reliability (error detection, security, and channel hopping).
- It uses TDMA, QoS protocol for security.
- Authentication and confidentiality services can be independently made available to the user.
- The security is managed by the network security manager who also distributes the key for different sources [23].

9.6 Conclusion

In this chapter, we discussed what is Internet of Things, how IoT is connected to our daily lives, and what are the applications and uses of it. Thus we also noticed different connectivity/communication protocols for sending data from one location to another. Also these protocols include different layers which are responsible for the reliability and recovery of the data packets, security and real-time working. IoT is an emerging field in which various companies and industries are innovating new microcontrollers to support these protocols which are very useful for these connectivity purposes. In this chapter, we also noted some important protocols which are majorly used in various industries. But with the increase of day-to-day need of users, different protocols would emerge in the future which would be more efficient and beneficial in terms of data consumption data rate, power consumption, storage, strength, etc. Does choosing these different types of communication protocols for implementing them in some services play a major role for using these protocols? So keeping the strengths and weaknesses in mind, these connectivity protocols should be chosen wisely.

References

1. L. Garg, E. Chukwu, N. Nasser, C. Chakraborty, G. Garg, Anonymity preserving IoT-based COVID-19 and other infectious disease contact tracing model. *IEEE Access* **8**, 159402–159414 (2020)
2. P. Lea, *Internet of Things for Architects: Architecting IoT Solutions by Implementing Sensors, Communication Infrastructure, Edge Computing, Analytics, and Security* (Packt Publishing Ltd, California, 2018)
3. G. Fortino et al., Platform-independent development of collaborative wireless body sensor network applications: SPINE2, in *2009 IEEE International Conference on Systems, Man and Cybernetics*, (IEEE, 2009)
4. D.C. Pfister, *Getting Started with the Internet of Things: Connecting Sensors and Microcontrollers to the Cloud*, vol. 1, Institute of Technology in Zurich (ETH Zürich), 1st edn. (Maker Media, Inc, 2011)

5. C. Ma, W. Li, R. Gravina, G. Fortino, Posture detection based on smart cushion for wheelchair users. *Sensors* **17**, 719 (2017)
6. C. Savaglio, P. Pace, G. Aloisio, A. Liotta, G. Fortino, Lightweight reinforcement learning for energy efficient communications in wireless sensor networks. *IEEE Access* **7**, 29355–29364 (2019)
7. S. Cirani, G. Ferrari, M. Picone, L. Veltri, *Internet of Things: Architectures, Protocols and Standards*, vol 3 (John Wiley & Sons, 2018)
8. University of Delft, Communications for IoT – Connectivity & Networking, in *The Internet of Things Magazine (IoTM)* (March 2020), p. 12
9. C. Dow, P. Lea, *Mastering IOT: Build Modern IoT Solutions that Secure and Monitor Your IoT Infrastructure* (Packt Publishing Ltd, 2019)
10. V.H. Princy Randhawa, *International Symposium on Advanced Electrical and Communication Technologies ISAECT '2018* (IEEE, 2018), p. 7
11. S. Al-Sarawi, M. Anbar, K. Alieyan, M. Alzubaidi, Internet of Things (IoT) communication protocols, in *International Conference on Information Technology (ICIT)*, (IEEE, 2017), p. 12
12. Springer Science, *Second International Conference on Computer Networks and Communication Technologies* (Springer Science and Business Media LLC, 2020), p. 10
13. Z. Wang, D. Wu, R. Gravina, et al., Kernel fusion based extreme learning machine for cross-location activity recognition. *Inf. Fusion* **37**, 1–9 (2017)
14. A. Sant, L. Garg, P. Xuereb, C. Chakraborty, A novel green IoT-based pay-as-you-go smart parking system. *Comput. Mater. Contin.* **67**(3), 3523–3544 (2021)
15. A. Gumaei, M.M. Hassan, M.R. Hassan, A. Alelaiwi, et al., A hybrid feature extraction method with regularized extreme learning machine for brain tumor classification. *IEEE Access* **7**, 36266–36273 (2019)
16. S. Albishi, B. Soh, A. Ullah, F. Algarni, Challenges and solutions for applications and technologies in the Internet of Things, in *4th Information Systems International Conference 2017, ISICO 2017* (2017), p. 7
17. D. Dey, J. Dansana, A. Behura, Chapter 52 A survey of datalink layer protocol for IoT, in *Second International Conference on Computer Networks and Communication Technologies*, (Springer Science and Business Media LLC, 2020), p. 20
18. B. Peng, Z.-G. Wei, T. Wu, Corresponding research of android and ZigBee, in *International Conference on Wireless Communication and Sensor Network*, (IEEE, 2014), p. 20
19. S.P. Raja, T. Sampradeepraj, N.N. Prasanth, Performance evaluation on internet of things protocols to identify a suitable one for millions of tiny internet nodes. *Int. J. Intell. Internet Things J. Intell. Internet Things* **1**, 31 (2019)
20. C. Rowland, E. Goodman, M. Charlier, A. Light, A. Lui, et al., *UX For the Consumer Internet of Things*, 1st edn. (Shroff/O'Reilly, London, 2015)
21. D.-I.K. Schwab, *The Fourth Industrial Revolution, Massachusetts Institute of Technology (MIT)*, 1st edn. (Penguin, 2017)
22. M. Kranz, *Building the Internet of Things: Implement New Business Models, Disrupt Competitors, Transform Your Industry* (John Wiley & Sons, 2016)
23. P. Randhawa, V. Shanthagiri, Concept of operations to system design. *Int. J. Instrum. Control Syst.* **5**, 12 (2015)
24. B. Da, R. Li, X. Xu, Oriented Networking (ION) for IoT interoperation, in *2017 Global Internet of Things Summit (GIoTS)*, (IEEE, 2017), p. 12

Chapter 10

IoT for Sustainability



Brian Davison 

Contents

10.1	Global Trends	254
10.1.1	Climate Change	254
10.1.2	Urbanisation	255
10.1.3	Linear Business Models	256
10.1.4	Technology and Automation	257
10.1.5	Summary	257
10.2	Sustainability	258
10.2.1	UN Sustainable Development Goals	259
10.2.2	Perspectives on Sustainability	260
10.2.3	Computational Sustainability	263
10.3	The IoT Fit	265
10.3.1	Architecture	265
10.3.2	Fog Computing	266
10.3.3	IoT and Sustainability	268
10.3.4	Smart Cities	272
10.4	Taking Stock	273
10.4.1	Selective Focus	274
10.4.2	Top-Down Solutions	275
10.4.3	Digital Colonialism	276
10.5	Cellular Architectures	277
10.5.1	Resilience	278
10.5.2	Energy Distribution	278
10.5.3	Local Clouds	279
10.5.4	Smart Villages	280
10.6	Current and Future Challenges	280
10.7	Conclusion	281
	References	282

B. Davison ()

School of Computing, Edinburgh Napier University, Edinburgh, Scotland, UK

e-mail: b.davison@napier.ac.uk

10.1 Global Trends

The COVID-19 pandemic has caused disruption to everyday life for millions of people across the globe. It has forced those in prosperous, developed countries to re-evaluate aspects of their societies that they previously took for granted. Occupations seen as low down the social scale are now reclassified as ‘essential workers’ while entire sectors of the economy have seen redundancies on an unprecedented scale. While some countries were relatively slow to respond to the crisis, the overall speed with which measures were introduced to manage the situation was startling. Virtually overnight, governments introduced financial support schemes for entire populations, transportation came to a standstill and face coverings became commonplace. Within weeks, millions of people accepted the restrictions of lockdown as an entirely reasonable obligation and working from home became the norm for those whose physical presence in the workplace was not essential.

One of the reasons that the world responded so dramatically to the virus was the speed at which it spread. Changes were reported by news channels in real time, and statistics such as the rate of infection and the daily death toll in different parts of the world brought home the immediacy of the threat. The experience has shown that transformational change is possible on a global scale when the situation is serious enough. However, the global pandemic is set against the backdrop of several other pernicious trends that are moving at a much slower pace and which consequently have not yet provoked a comparable response.

10.1.1 *Climate Change*

The first of these trends is global heating brought about by human activity. In the same way that some countries initially responded with complacency or scepticism to the pandemic, there was at first a great deal of resistance to the proposition that humans could have such a significant effect on their environment. With the findings of climate scientists now generally accepted, the International Panel on Climate Change (IPCC) produced a report in 2018 warning of the consequences of allowing the global mean surface temperature (GMST) to exceed pre-industrial levels (1850–1900 mean temperature) by 1.5 °C. These include species loss and extinction, risks to human health, livelihoods, food security and access to water. Economic growth is expected to be affected and extreme weather events will become more common. The surface temperature increase on land is likely to be higher than mean global figure, with the effect even more exaggerated in cities due to the urban heat island effect [31]. A 2020 update by the World Meteorological Organization reported that GMST was already 1.0 °C above pre-industrial levels and likely to reach 1.5° in one or more months between 2020 and 2025 [73].

Given the scale of the potential impacts of climate change, it is sobering to consider the slow pace at which measures to mitigate them are being introduced.

Part of the explanation must lie in the seemingly small quantities involved. A global change of 1.5 °C pales into insignificance for the average person when compared to several thousand deaths per day due to the virus. Likewise, the speed at which COVID-19 can spread is much more alarming than the slow increase in GMST over several decades. As the virus is brought under control, however, the experience could turn out to have beneficial effects. Having been sensitised to global threats and having seen the speed at which transformational change can occur, the general population may be ready to engage in urgent collective action to combat climate change and avoid its worst effects.

10.1.2 Urbanisation

Around 2007, the number of people living in cities exceeded those living in rural areas and the trend has continued since then. In 2018, 55% of the global population were living in cities up from 30% in 1950 [68]. This proportion is set to grow to 68% by 2050 which means that the number of city dwellers will nearly double in that time, assuming an eventual population of 11 billion. From some perspectives, urbanisation is a good thing. The high density of people in a relatively small area offers opportunities for efficiency, security and prosperity, but only if it is accompanied by good governance [68]. With poor management, in contrast, urbanisation can lead to pollution, environmental degradation and the phenomenon of ‘urbanisation without growth’ [6].

Currently, cities account for 3% of the Earth’s land area while consuming 75% of natural resources and accounting for up to 80% of greenhouse gas emissions [49]. The concentration of consumption means that the environmental impact of cities goes far beyond their boundaries. Rural areas face increasing pressure to satisfy the urban need for food with the associated burden of transportation. As they grow, cities are also increasingly vulnerable to natural threats such as water shortages and damage from extreme weather events. The coastal locations of many of the larger conurbations, for example, put them at risk from storm surges, rising sea levels, floods and coastal erosion [64].

A result of the global trend towards greater urbanisation is the emergence of megacities – those with over 10 million inhabitants. In 1990, there were 10 megacities located in relatively developed countries, but by 2018, this number had more than tripled with the new additions located primarily in Asia [68]. By 2030, nine more cities are predicted to join the megacity tier including London where the national per capita GDP is more than \$42 K and Dar es Salaam where it is just over \$1 K [76]. This disparity in wealth will entail contrasting local pressures and priorities for these two very different locations.

10.1.3 Linear Business Models

Human activity places other pressures on the natural world in addition to the consequences of global heating and urbanisation. In the dominant linear model of commercial production in developed countries, raw materials are obtained from the natural environment and turned into products which have a limited lifespan and are then discarded for new improved versions. It is argued that this model, characterised as the ‘take–make–dispose’ pattern, has fundamental limitations which are starting to be felt in the global economy [18]. Quite apart from the increase in business costs that result from dwindling supplies of raw materials, the linear model demands that ever greater areas of land are devoted to the extraction of material and entails ever greater accumulation of waste in the environment. The extractive activity reduces the area available for natural life, and the build-up of waste reduces the quality of habitats that remain.

A particularly salient example concerns the pervasive use of plastics in the global economy. Plastics are typically produced from non-renewable feedstocks derived from petrochemicals and represent around 8–9% of all fossil fuel consumption when energy required for their production is taken into account [47]. Plastics can be consumed either in the form of useful products or as disposable packaging often referred to as ‘single-use plastic’. Plastic waste can be sent to landfill, incinerated or recycled but recycling rates remain low. Globally in 2020, around 14% of plastic waste is recycled and even in the EU which is one of the best-performing regions, the rate is only 30% [47]. Neither of the other disposal strategies are without their issues: The landfill approach requires progressively more sites, while incineration releases carcinogenic chemicals such as dioxins [80]. All too often, plastic waste is released into the environment where it can break down through natural erosion into microscopic particles now known as microplastics which can pervade previously pristine habitats and enter the food chain [80].

To combat the problems associated with linear business models, the World Economic Forum [72] advocates a deliberate and coordinated transition to a circular economy defined as follows:

A circular economy is an industrial system that is restorative or regenerative by intention and design. It replaces the ‘end-of-life’ concept with restoration, shifts towards the use of renewable energy, eliminates the use of toxic chemicals, which impair reuse, and aims for the elimination of waste through the superior design of materials, products, systems, and, within this, business models [18].

The problems caused by linear business models have been growing slowly, and although there is considerable public awareness about them, action has been slow to emerge. In contrast to the COVID-19 pandemic, they are not perceived as urgent. Their effects are typically remote from everyday life and there is a great deal of inertia among commercial interests to move away from existing practices, especially where this would entail a fall in profitability.

10.1.4 Technology and Automation

As in the case of urbanisation, the advance of computing technology into all areas of life has both positive and negative aspects. In many instances, the motivation for greater reliance on technology is that it will improve efficiency. This in turn, the theory goes, will drive down costs for consumers and citizens, increase profits for shareholders and raise the average standard of living in a virtuous cycle of development. Indeed, this chapter goes on to discuss the role of the Internet of Things (IoT) in achieving a more sustainable society. However, it is also important to consider the undesirable consequences of the increased use of automation and machine intelligence. Since the industrial revolution, fears that technology would displace workers have not been borne out; instead, new technologies have created new opportunities and the labour mix has adapted to take advantage of them. A recent World Bank report is optimistic that the same pattern will repeat itself in the case of the revolution currently underway [74]; however, the same report also points out that according to some predictions, 47% of jobs in the United States, for example, could be displaced by automation.

To take a specific example, the imminent introduction of autonomous vehicles will probably mean redundancy within the next 20 years for the majority of those in occupations where driving is the main component. They numbered 3.8 million in the United States in 2015 [3]. On the other hand, fully automated vehicles are expected to enhance safety, make mobility available to all and to increase opportunities for those for whom driving is a complementary adjunct to their main occupation [53]. To avoid the undesirable social and economic effects of such a large disruption, organisations such as the World Bank recommend that governments already start to plan for the changes. Suggestions include placing greater focus on higher-order cognitive and socio-behavioural skills in early years education, enhancing social protections for those whose jobs are at risk and adjusting tax policy to cover the costs of these changes equitably.

10.1.5 Summary

The trends discussed in this section all test the assumptions by which society has developed, especially in Europe and North America, over the last century. They allow us to predict that cities will become larger, denser and more complex, and that technology will continue to pervade everyday life to an ever-increasing extent. They also highlight the pressing need to curb activities that contribute to global heating, and to redesign commercial activity around more equitable values. In order to avoid reaching the limits of a finite planet, transformational shifts in thought, behaviour, corporate governance and social equity are needed. The consequences of not addressing these issues could be dire. COVID-19 has given the entire world a new perspective on its current way of life. Following an examination of the concept

of sustainable development, the question addressed in the remainder of this chapter is what role IoT might have in the response to these burning questions.

10.2 Sustainability

The dictionary definition of *sustainability* is simply the capacity of a system to maintain its present state indefinitely. In the two decades between 1970 and 1990, the popular understanding of sustainability gravitated towards a model built around three interlocking aspects, or *pillars*, as shown in Fig. 10.1. Essentially, the model encapsulates the idea that a comprehensive approach to sustainability must give equal attention to social, environmental and economic factors.

In 1987, the Brundtland report, commissioned by the United Nations in 1983, established the concept of *sustainable development* based on the identification of three interlocking global crises concerning the environment, global economic development and energy use. The report concluded that the risks posed by these crises constituted a long-term threat to human survival and were too large to be managed by individual nations. The report culminated in a call to action advocating a coordinated global effort to address the crises.

Sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs. It contains within it two key concepts [5]:

- The concept of ‘needs’, in particular the essential needs of the world’s poor, to which overriding priority should be given; and
- The idea of limitations imposed by the state of technology and social organisation on the environment’s ability to meet present and future needs.

Fig. 10.1 The three pillars of sustainability



By making explicit reference to human needs, the formulation neatly places human agency in the form of personal, political and corporate responsibility at the heart of sustainability. It also implicitly incorporates the concept of social justice in which the needs of poorer regions of the world are prioritised over those of developed nations. The second key concept directly challenges the false impression that natural resources are infinite and highlights the need to consider the long-term consequences of established ways of life.

10.2.1 UN Sustainable Development Goals

In 2000, the UN launched the Millennium Development Goals (MDGs) which set eight development targets to be achieved within 15 years. By 2015, the proportion of people living in extreme poverty in developing countries fell from 47% to 14%. During the same period, maternal mortality fell by 45% and ozone-depleting chemicals were virtually eliminated from industrial products [67]. To capitalise on the progress made on the MDGs, a new set of 17 Sustainable Development Goals (SDGs) were established in 2016 which have become the current benchmark for sustainability (Fig. 10.2). The declaration defined a series of measurable targets for each goal which UN signatories committed to deliver by 2030 [66].

The need for a balanced approach to achieving the goals may be demonstrated by taking an example. Figure 10.3 shows the renewable energy used by different countries as a percentage of total energy consumption. On this measure alone, some of the poorest countries in the world already achieve an energy mix with up to 80% renewables. However, for many of those countries the renewable fuel in question is



Fig. 10.2 Sustainable Development Goals (["https://www.un.org/sustainabledevelopment/"](https://www.un.org/sustainabledevelopment/)) The content of this publication has not been approved by the United Nations and does not reflect the views of the United Nations or its officials or Member States)

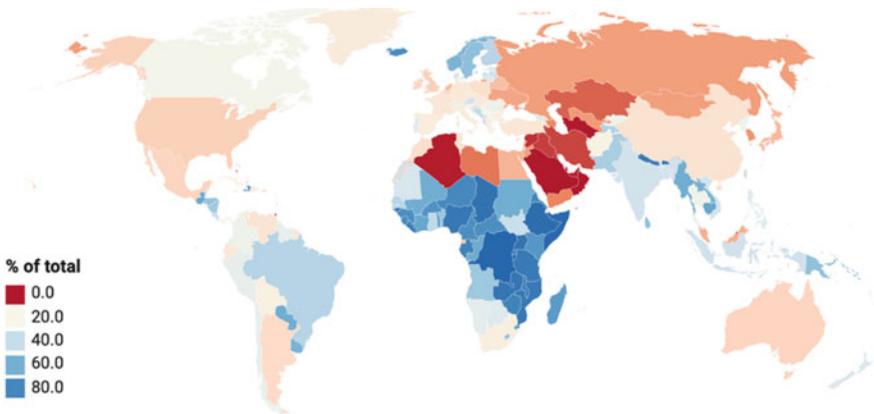


Fig. 10.3 Proportion of renewable energy consumed as a percentage of total energy consumption. (Data: World Bank, 2020 [77])

biomass burned on open fires which contributes to a range of pulmonary diseases [16, 75]. The impact on other indicators such as access to clean energy sources and the mortality rate for children under 5 is thus very negative. It is only by taking a multi-dimensional view that the complete picture can be fully appreciated.

The complexity and scale of the effort required to deliver the SDGs was summarised in the 2019 progress report which advocated a broad portfolio of measures based on technological improvements, lifestyle changes and localised solutions [16]. Coordinated action on consumption, production and conservation policy development would be needed to eliminate hunger (SDG2) while at the same time avoiding biodiversity loss (SDG13, SDG14) and preventing land degradation (SDG12, SDG13).

10.2.2 *Perspectives on Sustainability*

The SGDs reflect the United Nations' status as a supranational organisation with a global perspective. They are intended to capture the web of interconnected pressures faced at the planetary, regional and national levels. On the ground, however, context dictates that some issues take precedence over others. This section outlines some key tensions that cut through generic sustainability concerns.

10.2.2.1 **Developing Versus Developed Countries**

The Human Development Index (HDI) is a statistical indicator that takes values between 0 and 1. The closer to 1, the more 'developed' a country is deemed to

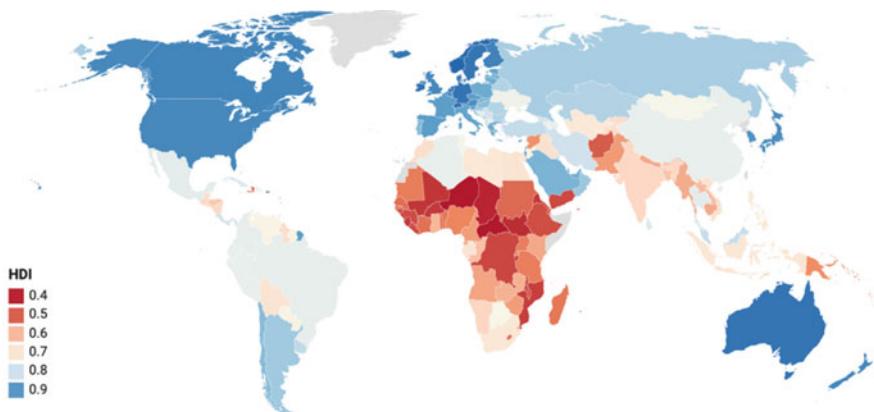


Fig. 10.4 Human development index by country. (Data: UN Development Programme, 2020 [69])

be. The measure is calculated using proxy measures for life expectancy and health, level of education (knowledge) and standard of living [69]. Figure 10.4 shows the data from 2019.

Many of the SDGs are deliberately aimed at raising the quality of life of those at the bottom of the HDI table. For more developed economies, goals such as zero hunger (SDG2) and clean water and sanitation (SDG6) have a much lower priority than others, since those problems have been largely solved. The environmental, energy and climate targets set by the European Commission for 2020, for example focus on the protection of biodiversity, cutting greenhouse gas emissions, sustainable cities and the development of the circular economy [21]. In a recent survey, leaders of developing countries were asked to rank the SDGs by priority. Across the board, poverty was the overriding concern with SDGs 4, 8 and 16 appearing in over 60% of leaders' top six priorities. By contrast, SDG12 and SDG14 were ranked the lowest priorities with figures of 15% and 5.4% respectively [13].

10.2.2.2 Urban Versus Rural

Urban and rural areas have contrasting characteristics that lead to different sustainability concerns. Most food production, for example, occurs in rural areas while urban centres represent concentrations of consumers leading to a net flow of agricultural products from rural areas to urban areas. This creates pressures on land use and biodiversity in rural areas leading to land degradation, deforestation and fragmentation of natural habitats that may be further exacerbated by climate change [42]. In urban areas in contrast, the concerns are much more to do with energy efficiency, the elimination of waste, transportation and pollution control [61]. Rural areas are also more susceptible to extreme poverty and poor access to services and

utilities than urban areas [42]. In contrast, urban priorities revolve around equitable access to education and basic utilities, health and fitness of the population, air quality and noise levels [39].

10.2.2.3 Local Versus Global

The SDGs are primarily concerned with global trends and the statistical indicators used to measure progress against them are measured at the national level [52, 56]. Top-down strategies may be blunt instruments; however, trying to force through a one-size-fits-all approach that may encounter resistance. The 2017 ‘Listening to Leaders’ survey, for example, found that leaders in developing countries prioritised the SDGs differently from ordinary citizens [13]. Leaders were much more likely to focus on economic growth, education and strong institutions, while citizens were more concerned with food security and the health of their cities.

To resolve potential conflicts, the UN acknowledges the need to promote and support localised, bottom-up initiatives. The 2019 progress report on Agenda 2030 stresses the importance of grassroots initiatives and the integration of local knowledge and traditional practices for sustainability [16]. The report notes a multitude of bottom-up efforts working towards the SDGs, but also highlights a lack of research into the realisation of global targets through the aggregation of these bottom-up projects.

One of the mechanisms for individuals to contribute towards the achievement of Agenda 2030 is through participation in citizen science activities. Citizen science typically involves volunteers taking part in data collection but can also include other types of scientific activity. With technology such as smart sensors, mobile computing and broadband Internet, the opportunities for data collection and aggregation, dissemination of results and awareness raising are unprecedented. As well as potentially increasing spatial and temporal data coverage, citizen science has the potential to fill in gaps that exist in traditional sources of data. For example, Fritz et al. [25] argue that the monitoring of food waste (SDG12), climate change (SDG13) and marine pollution (SDG14) are all suitable targets for citizen science in terms of both data collection and the development of monitoring methods.

10.2.2.4 Internal Versus External Focus

General system theory (GST) distinguishes between open and closed systems where an open system exchanges materials with its environment across the system boundary and a closed system does not [4]. GST provides a useful conceptual framework for the design and study of real-world systems such as organisations, countries, cities and ecosystems which are all open in the GST sense. A systems approach typically focusses on the internal workings of the system under study. Organisations, for example, have control over their internal operations and put a lot of effort into refining them to deliver value. Governments create regulatory regimes within

which organisations operate and which define some of the exchanges between an organisation and its environment. Because the organisation has comparatively little external control, it avoids attempts to influence its environment because of the perceived risk of failure and wasted effort. From the point of view of sustainability, this is clearly unsatisfactory. Many problems of sustainability require organisations to attend to the external effects of both inputs and outputs.

In an attempt to embed holistic thinking into organisations and to operationalise the three pillars model, Elkington introduced the concept of the *triple bottom line* for company accounting [17]. Traditionally, the bottom line refers to the amount of profit per share after all liabilities have been accounted for. Elkington proposes that companies should routinely account for value in the environmental and social domains as well. The approach is sometimes called the *People–Planet–Profit* model for that reason, and the literature provides examples of management indicators related to regulatory certification or sanctions, and to quantitative measures of energy used, resources consumed or waste emitted [22]. Elkington situates his discussion in the context of developing trends such as evolving societal values and increasing transparency around company operations as well as against the backdrop of mounting environmental problems. He argues that companies who fail to evolve are likely to overcome by market forces.

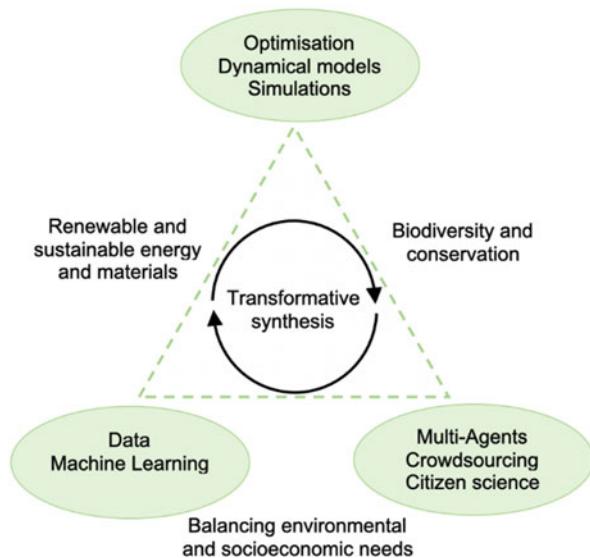
The internal/external distinction can also be applied to any area of activity that consumes energy or resources. For example, automation may be applied in order to reduce the energy associated with a manufacturing process, but if the energy cost associated with the automation equipment is larger than the amount saved elsewhere, the effort is futile. The field of green computing seeks to improve the energy and resource efficiency of computing technology itself thus increasing the benefits of automation.

10.2.3 Computational Sustainability

In 2008, the United States National Science Foundation established the *Expeditions in Computing* programme aimed at funding teams to pursue transformative research agendas over a period of 5 years [65]. One of those programmes, led by Carla Gomes at Cornell University, set out to concentrate the attention and efforts of computer scientists on the challenges around sustainability. The programme established the Institute for Computational Sustainability (ICS) at Cornell as well as coining the term *computational sustainability* itself.

The landscape of computational sustainability research shown in Fig. 10.5 evokes a process of transformative synthesis in which computational techniques are combined in novel ways in pursuit of solutions to sustainability problems [26]. While the techniques themselves may be well-established, it is the intention behind their use and their adaptation to novel contexts which is the hallmark of computational sustainability. In the process, it is not just the application domain that benefits from this inherently interdisciplinary approach; the field of computer sci-

Fig. 10.5 Computational sustainability research landscape. (Adapted from [26])



ence is also enriched through the development of novel combinations of techniques. Publications in computational sustainability fall roughly into three categories as illustrated by these selected examples:

Renewable and Sustainable Energy and Materials

- Finding an optimal mix of renewable energy generation and storage using a dynamic planning approach [35]
- Li-ion battery charging optimisation with multi-arm bandits [29]
- Application of data mining to domestic energy use behaviour [9]

Biodiversity and Conservation

- Modelling bird species distribution using citizen science [63]
- Optimal design of wildlife corridors [14]
- Preserving genetic diversity using an artificial immune system algorithm [58]

Balancing Environmental and Socioeconomic Needs

- Estimating socioeconomic indicators from satellite data [20]
- Urban bike sharing [24]
- Air quality monitoring through citizen science [37]

As the examples demonstrate, the range of topics that fall under the banner of computational sustainability is very wide. It is also an associative field in the sense that work done for other reasons and without explicit reference to computational sustainability could still sit comfortably under its umbrella.

10.3 The IoT Fit

According to the International Telecommunication Union (ITU), the Internet of Things (IoT) is:

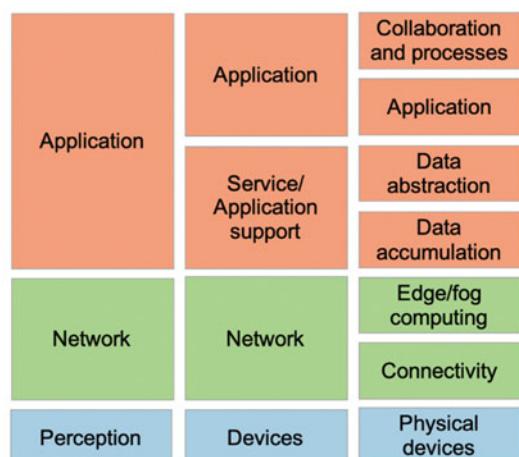
a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving, interoperable information and communication technologies [32].

While other definitions also exist, the ITU formulation captures the essential concept of IoT bridging the divide between the physical and digital domains and offering opportunities to manage complexity through direct communication between things without the need for human intervention.

10.3.1 Architecture

Using a reference architecture to structure a complex system is a mature approach to simplifying the design process, ensuring testability and maintainability, and promoting interoperability and the application of standards. Architectures are often described in terms of layers in which each layer has a specific set of functions and responsibilities and communicates with neighbouring layers in a standardised manner. Currently, there is no single model for IoT and the number of layers in proposed architectures varies between three and seven. Of the three examples shown in Fig. 10.6, the three-layer model is generally agreed to be too simplistic for practical purposes but provides a useful conceptual overview. The ITU proposal makes a distinction between two different categories of service that exist at the top of the hierarchy but does not elaborate on the lower layers. The seven-layer

Fig. 10.6 Example IoT reference architectures: left: simple; centre: ITU model; right: IoTWF model



model, proposed by Cisco for the Internet of Things World Forum (IoTWF), makes allowance for data processing capabilities at lower levels in the architecture, an approach known as *fog computing*. The seven-layer model is widely used, and the concept of fog computing is crucial for some of the arguments presented later in this chapter.

Nodes

The lowest level of each architecture is concerned with the ‘things’ that are responsible for the coupling of the physical and digital domains. They typically include sensors, processing capability and communications, and may also be linked to actuators to provide bi-directional interaction with the physical environment. In many applications, a high-data resolution is critical to delivering the intended benefits. Many nodes are therefore required in the device layer to generate data at an appropriate rate. This places two extrinsic requirements on nodes: their cost must be kept as low as possible, and their power requirements must also be minimised. The result is that nodes are typically simple, resource-constrained devices.

Communications

All models include a communications layer, but the picture is complicated by the inclusion of fog computing. At the detailed level, most architectural models differentiate inter-node communication from the further transmission of data to the cloud using standard internet technologies. The device that bridges between these two networks is usually referred to as a *gateway*, and it is often the gateway device which hosts the resources required for fog computing.

Applications

The data processing in the top layer of the simple model is assumed to take place in the cloud, and that is also the case for the other two models which subdivide the application layer. The resource-constrained nodes do not have the capacity for any significant processing, and cloud service provides a solution that can scale to accommodate an arbitrary demand.

10.3.2 Fog Computing

Architectural models for IoT that rely solely on the cloud for processing and control have several weaknesses that make them unsuitable in certain situations. The main issues are outlined below.

Performance

In 2019, there were 7.6 billion IoT devices in operation and some predictions suggest that by 2030 there could be over 24 billion [44]. The consequent exponential increase in traffic will create network performance and congestion problems [50].

Latency

IoT applications that require a real-time response such as those in safety-critical or health-related contexts are adversely affected when network latency is high. The delay incurred by the transfer of data to and from the cloud in addition to any unavoidable processing time may render such applications unusable.

Security

The data collected and processed by some IoT systems may be sensitive for reasons of safety, privacy or commercial value. The longer the network path traversed by such data, the greater its exposure to potential interception or corruption.

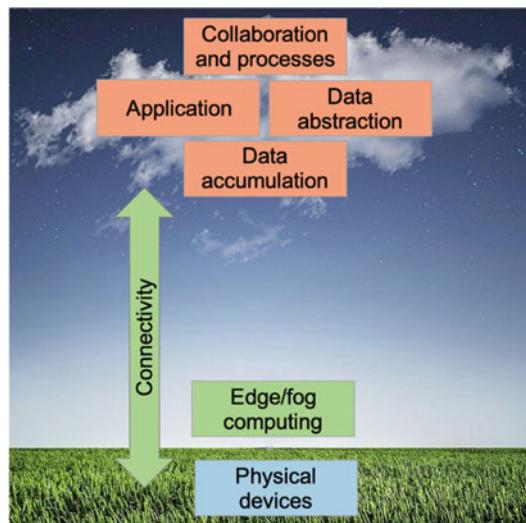
Reliability

Network connections can fail for many different reasons. Relying on a cloud service for processing and control renders an IoT system vulnerable to all such interruptions.

To address these weaknesses while still preserving the ease of use and scalability of the cloud model, fog computing situates the storage and processing of data closer to the nodes, typically by increasing the capabilities of the gateway device. In the same way that natural fog is composed of low-lying water vapour, the term *fog computing* is intended to convey the idea of services similar to those in the cloud, but closer to the devices on the ground as illustrated in Fig. 10.7.

Many technological choices are available for the implementation of a fog architecture. One option would be to use a wireless protocol such as Bluetooth, CoAP or LoRa between the nodes and the gateway, with the backhaul network between the gateway and the cloud based on TCP/IP. In this situation, the gateway could be a small server with appropriate network interfaces. With the advent of

Fig. 10.7 The fog computing metaphor



5G services, an alternative implementation could be based entirely on mobile cellular technology. This particular variation, known as *multi-access edge computing* (MEC), has been specifically designed around a service-based concept to enable distributed computing for IoT [34]. In the MEC model, placing the additional storage and processing capability at the cell base station would ensure optimum performance.

10.3.3 IoT and Sustainability

In the most general terms, sustainability is the husbandry of resources and a major challenge is data visibility: resource stocks need to be measured and monitored, the flow of resources through their lifecycle needs to be overseen and controlled, and the quantities and effects of waste ejected into the environment need to be understood. The required data may be unavailable for a variety of reasons such as those listed below.

Geographical Scale

Measurement is difficult when resources are dispersed over a very wide area which may be difficult to access. Fresh water supplies and fish stocks are good examples. The accumulation of plastic waste in the ocean is only now being recognised as an environmental issue for similar reasons.

Complexity

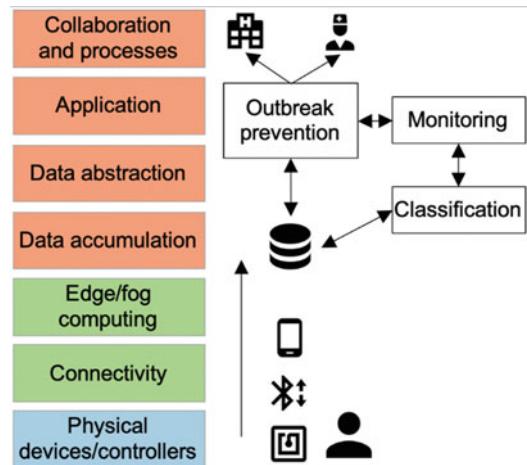
Patterns of resource use may be difficult to monitor because of the complex web of flows that make up the resource lifecycle. The flow of food products in a city falls into this category – identifying specific inefficiencies and waste would require a great deal of effort and coordinated action by many people.

Cost

Monitoring and management activities themselves consume resources. Measurement equipment and staff required to operate it may represent significant expenditure which it might not be feasible for organisations to bear.

The characteristics of IoT make it particularly suitable for addressing problems of sustainability. A recent analysis concluded that 84% of IoT deployments were already addressing or had the potential to address the SDGs [78]. The ultimate goal of IoT is to connect everything regardless of geographical location, giving access to high-resolution data that was previously unavailable. Nodes are designed to be simple, cheap and robust, running for several years without requiring any maintenance. They operate independently of any fixed infrastructure and can be deployed without human intervention using a drone, for example. Over-the-air programming allows them to be updated without physical access, and some are even biodegradable such as those used in health applications [38]. The IoT has also encouraged the development of low-power, low-bandwidth technologies

Fig. 10.8 Outbreak prevention system [57]



such as LoRa¹ and NB-IoT² which vastly reduce the costs associated with data transmission. Thus, the IoT paradigm addresses by design the external and internal perspectives mentioned earlier. The next few sections describe some illustrative examples of IoT and their relationship to the SDGs.

SDG3: Good Health and Wellbeing

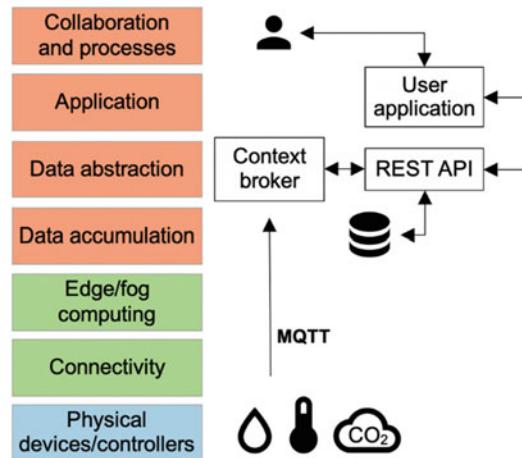
Following outbreaks of Ebola in East Africa in 2014, the U.S. Agency for International Development (USAID) launched a programme entitled *Fighting Ebola: A Grand Challenge for Development* [71]. One of the crowd-sourced innovations to come out of that work was a wearable sensor capable of monitoring 11 vital signs and transmitting them over Bluetooth [62]. Several groups proposed incorporating such sensors into IoT systems, such as the one shown in Fig. 10.8 where it is aligned with the IoTWf seven-layer architectural model.

The sensor transmits data to a mobile device which relays it over the Internet to a cloud datastore. Thereafter, classification and monitoring are performed by cloud applications and alerts are sent to hospitals and medical personnel as required [57]. In the proposed scheme, the mobile device does little more than forward the raw data to the cloud; however, because of the capabilities of modern smartphones, there is the potential for it to act as a fog node and perform local analysis or aggregate the data before it is uploaded to the cloud.

¹ <https://lora-alliance.org/>

² <https://www.gsma.com/iot/narrow-band-internet-of-things-nb-iot/>

Fig. 10.9 Water management via IoT [46]



SDG6: Clean Water and Sanitation, and SDG12: Responsible Consumption and Production

In response to the extremely dry climate and shortage of fresh water in south-east Spain, Muñoz et al. [46] propose an IoT system to monitor and manage the supply of water to agricultural producers. As a result of agricultural production in the area, freshwater reservoirs face depletion and are supplemented by desalination plants. The purpose of the proposed system is to optimise water management by providing feedback to the desalination plants so that they can tailor their production to fit the demand.

The architecture is based on the FIWARE³ open source standard for IoT applications which is driven by the European Union. The system offers standard-based interfaces for interoperability. Devices send data to the cloud using MQTT,⁴ a simple, message-based protocol that allows heterogeneous devices to communicate easily. FIWARE is one of several flexible IoT platforms that aim to make the development of IoT applications as simple as possible. It incorporates a broker process which relays data via MQTT from one element to another. A REST API provides access to a database for the broker as well as for third-party applications. Figure 10.9 summarises the architecture proposed by Muñoz et al. aligned with the IoTWF reference model. The estimated cost saving on desalination as a result of the system was estimated at 87% mainly due to a lower electricity requirement.

SDG15: Life on Land

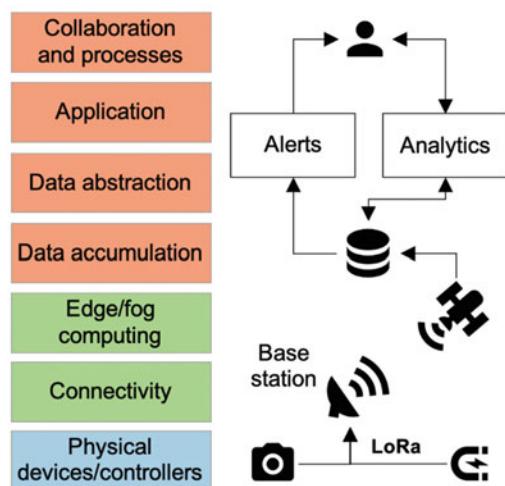
With a long history of wildlife conservation work, the Zoological Society of London are introducing a new version of their remote monitoring system, Instant Detect.⁵

³ <https://www.fiware.org/developers/>

⁴ <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>

⁵ <https://www.zsl.org/conservation/how-we-work/conservation-technology/instant-detect>

Fig. 10.10 Wildlife conservation [40, 59]



The system has already contributed to the protection of endangered species in North America, Africa, Australia and Antarctica, and the upgrade will add new capabilities.

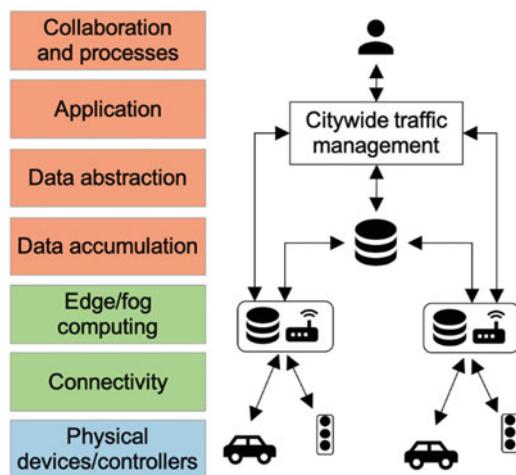
The new architecture relies on LoRa for long-range, low-power communications on the ground, and uplinks to the Iridium satellite network to transmit data to a cloud repository. Sensor nodes equipped with cameras perform local image-recognition processing to determine whether an image needs to be uploaded. Nodes may carry a range of other sensors including metal detectors to identify potential poachers. Figure 10.10 illustrates the architecture of Instant Detect 2.0.

SDG11: Sustainable Cities and Communities

With large and growing numbers of people living in cities, issues related to transport management are becoming more important. Road traffic generates large quantities of airborne pollutants such as carbon monoxide, nitrogen oxides and particulates which are harmful to health. A great number of studies have shown that pollutant levels in busy cities can often exceed recommended levels [1]. Keeping traffic flowing smoothly, avoiding unnecessary stopping and starting and making it easy to find parking spaces can all help to minimise emissions. Many studies have proposed IoT-based solutions for managing traffic flows and for the provision of other services such as collision avoidance and clear routes for emergency vehicles. The example shown in Fig. 10.11 proposed by Mohammed et al. [43] makes heavy use of the fog computing model to provide time-sensitive services such as traffic signal control, and to avoid unnecessary network exchanges between vehicles and the cloud.

In many schemes for urban traffic management, the vehicles themselves are classed as fog nodes because of their onboard computing facilities. It has been suggested that they could be incorporated into an ad-hoc networking relationship

Fig. 10.11 Urban transport management [43]



with other vehicles and with the roadside units, which comprise the fixed fog nodes to provide greater network resilience and flexibility [48].

10.3.4 Smart Cities

Because of the peculiar complexity of the system of systems that makes up the fabric of a modern city, the concept of the *smart city* has become prominent in the IoT literature. Over the last two decades, papers published under the heading of smart cities have covered a very wide range of specific topics. Some take a holistic approach, focussing on city-wide architectures to support multiple homogeneous services, while others focus on specific use cases. One review identified 40 distinct bursts of activity in the academic literature around particular keywords which are shown chronologically in Fig. 10.12 [82].

In an attempt to rationalise the smart city concept, another review identified 84 different definitions spanning the period from 2000 to 2019 which were categorised as *technocentric*, *humanist*, or *collaborative* [15]. The technocentric school focuses on technology as the main driver of smart city development. It emphasises the top-down development of infrastructure from which social benefits follow as a result. Early definitions of the smart city appear to take this optimistic view. The humanist perspective is espoused by the European Union and its main driver is the development of human capital characterised by education, culture and wellbeing. In the humanist view, technology is treated as an enabling tool which does not always match the needs of the citizens if it is introduced in a top-down fashion. The humanist approach also highlights the risk of social polarisation as a result of unequal access to technology. Figure 10.12 shows consistent efforts to define the smart city in humanist terms compared to the bursts of enthusiasm that seem

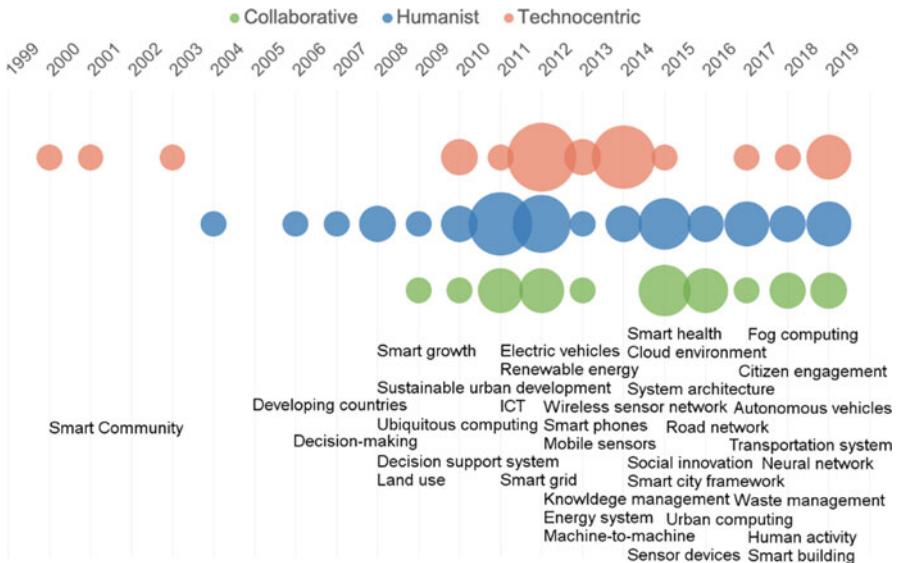


Fig. 10.12 A chronological illustration of the smart city concept [15, 82]. The size of the coloured bubbles represents the number of smart city definitions published in the relevant year

to characterise the technocentric view. Finally, the collaborative approach sees the interaction between stakeholder groups as the defining characteristic of the smart city. Development comes about through a mix of top-down and bottom-up approaches mediated by technology.

Fog computing has a special place in the smart city literature [81]. The weaknesses of the cloud-only model discussed above are intensified by the complex web of socio-technical interactions present in the urban environment and the density of activity. Some authors have asked whether attempts to manage cities through technology will eventually fail as complexity increases, accountability is lost and technology consumes more energy than it saves [10]. However, the fog computing model offers some technological solutions by embedding localisation in the design of smart city architectures. The OpenFog consortium highlights traffic congestion, public safety, energy consumption, sanitation, and public internet connectivity in its smart city scenario [50], and other applications of the fog model are abundant in the current literature [33].

10.4 Taking Stock

IoT is a new technological field which is developing so quickly that new architectures, devices and use cases are being proposed almost daily. Its potential for increasing efficiency in many domains is clear and the correspondence between

the characteristics of IoT and the requirements for sustainability is a natural one. Commercial efforts to develop the technology are intense, and 84% of current IoT projects can be said to be addressing the SDGs at least in part. With such a positive outlook, then, where is the problem? The answer lies in resisting the technocentric approach and addressing the wider implications of IoT deployments, especially in developing countries. Technology has a role to play in sustainable development but is far from being the only consideration. The principles of computational sustainability place a new responsibility on technologists to take this type of measured approach to their work and to take account of the wider context:

Our vision is that computer scientists can and should play a key role in helping address societal and environmental challenges in pursuit of a sustainable future, while also advancing computer science as a discipline [26].

The remainder of this section presents a series of issues for IoT in relation to sustainability which follow from the earlier discussions. By making them explicit, the intention is to highlight the potential for unintended consequences of IoT projects which might otherwise undermine progress towards the SDGs.

10.4.1 Selective Focus

A recent report from a European think tank on IoT for sustainability identified the four most common IoT use cases as smart manufacturing, healthcare, energy and mobility in smart cities [55]. The report also presented statistics to show that 63% of IoT projects in 2018 were concentrated on SDGs 7, 9 and 11 [79] while some SDGs were hardly addressed at all (1, 2, 13, 14, 15). There are three obvious reasons that could be put forward to explain this phenomenon. The first is that the selective approach represents local requirements and should therefore be respected. This response does not stand up under scrutiny, however. Since local concerns are known to differ, they should balance each other out when aggregated into global statistics. The second possibility is that there is some bias in the way the statistics were collected which means that projects with certain characteristics are over-represented and that others are overlooked. This proposition is certainly true at least in part. The statistics come from a dataset of 643 projects collated by market insights company, IoT Analytics.⁶ The company's main business is in compiling insight reports for commercial operators to inform commercial strategy. Only 10 of the projects in the dataset fell into micro category indicating that they had an impact for fewer than 10,000 people. In contrast, 95% of the projects were classified as small or medium corresponding to city and country scale respectively. Additionally, the majority of projects were led by the private sector (70%) and originated in Europe or North America (80%). The third explanation for the uneven

⁶ <https://iot-analytics.com/>

attention is that the popular SDGs are those where commercial interests anticipate the most profit. SDG7 (Affordable and clean energy), SDG9 (Industry, innovation and infrastructure) and SDG11 (Sustainable cities and communities) are precisely those with the scale and financial resources to attract large commercial providers. In contrast, there is little financial incentive to deploy large-scale IoT systems to protect marine life or alleviate poverty.

Whatever the actual reason for the apparent disparities in coverage of the SDGs, a more balanced approach is clearly required. As argued above, concentrating on certain goals at the expense of others can have undesirable consequences.

10.4.2 Top-Down Solutions

The official indicators by which progress against the SDGs is measured are statistics gathered at the national level. To improve their performance against the targets, governments may be tempted to impose solutions in a blanket fashion. The same might be true at the level of the city or region where relevant authorities make decision on the inhabitants' behalf. Country- or city-wide projects are attractive to commercial providers because of the benefits they offer through economies of scale. Installing a large-scale system allows for the selection of a single set of technologies, simplified communications and centralised planning all of which greatly simplify the task. Large-scale installations come with risks, however. The first is that the technology may be quickly superseded which either means that the customer must continue to use obsolete equipment or invest in an upgrade. The larger the deployment, the more an upgrade will cost. Second, once a large amount of money has been spent with a single supplier, the customer is locked into the relationship since the costs of moving to a different supplier are too high. Third, a single solution over a wide area offers an attractive target for cybercriminals. A single weakness could allow access to the whole network of connected devices where a more heterogeneous mix would be more resilient.

As well as technical issues, there may be social consequences of solutions introduced in a top-down manner. Securing buy-in from the local community is more difficult when a solution is imposed from the outside. In addition, it is argued above that a generic solution may not address localised needs. A further risk associated with imposed solutions is the sense of disempowerment and helplessness that may be created among the intended beneficiaries, especially when the imposed solution is perceived as complex [10]. Finally, there is a risk that a top-down solution may suppress small-scale, bottom-up initiatives because of central control over the official system, for example, or because appropriate interfaces for interoperation are not provided. This would have the undesirable effect of excluding potential insights from citizen science and other participatory activities.

10.4.3 Digital Colonialism

The Internet is already dominated by a few large corporations, primarily from the United States. Guided by current commercial wisdom, these entities are still intent on growth despite their current reach, and developing countries represent a major opportunity to extend their user base. Many of them sponsor development programmes in which they offer free products to developing nations, or form investment partnerships with governments for the provision of services at much reduced rates [2]. Microsoft, for example, recently announced that it would be investing \$1.1 billion over 5 years to create a cloud services region for Mexico [41]. The press release claimed that the initiative was about ‘democratizing the access to technology’ and highlighted a number of activities included in the programme such as a project to monitor pelagic sharks off the Pacific coast. While the immediate outcomes of the programme will undoubtedly be positive, the longer-term consequence will be an extension of the corporation’s sphere of economic influence at the expense of any potential local competitors. The result will be a net flow of revenue from Mexico to the United States and to Microsoft in particular.

The phenomenon of digital colonialism has been well documented in the literature. Michael Kwet [36] describes three specific ways in which large corporations seek to dominate markets in poorer regions of the world which are outlined below.

Economics

Digital corporations use their financial resources to enter markets quickly and create a dominant and potentially monopolistic position that no local competitor can challenge. Kwet cites the example of Uber which takes a 25% commission on every trip.

Architectural design

Companies who build technical infrastructure are positioned to ensure their own control over major parts of the technical ecosystem. The significant infrastructure elements for IoT are the communications equipment and the cloud services in the upper layers of the IoT architecture.

Intellectual property

Technology corporations can leverage their existing products to fulfil local needs crowding out competition. Licensing arrangements entail a net flow of revenue from the recipient country to the home country of the provider. One example where a country has resisted this type of domination is China where an internal technology ecosystem has been built by deliberately excluding foreign players.

An example of apparent corporate benevolence that is often held up for criticism is Facebook’s Free Basics⁷ programme which offers a range of Internet services

⁷ <https://developers.facebook.com/docs/internet-org/>

at no cost to users. The free facilities are completely controlled by Facebook, and although users can pay to access services outside the walled garden, many do not have the resources to do so [2]. The offering also allows Facebook to collect data to create profiles of users in developing countries where it later intends to expand its presence [11]. Nor are such initiatives exclusive to the large household names of the technology industry. A deal between Zimbabwe and Chinese company CloudWalk will see the company collect data from facial recognition, and an arrangement for UK company Babyl to provide health care services in Rwanda will allow the company to establish a monopoly position through its ownership of medical data [70]. While addressing some immediate issues, such relationships only serve to entrench existing inequalities between countries.

10.5 Cellular Architectures

To realise the full benefit of IoT for sustainability, the positive aspects need to be exploited while keeping any negative consequences under control. Considering the foregoing discussion, the following set of recommendations can be proposed.

Complexity must be contained

The cloud-centric approach is not viable for certain use cases, and large-scale developments such as smart cities become unmanageable as they become more complex [10]. A fog-first approach is therefore appropriate.

Small-scale, bottom-up developments must be welcomed

To avoid the inertia created by top-down solutions and to exploit the ingenuity of ordinary people and communities; interoperability between devices and systems is vital. Governance therefore needs to be localised so that decision-making is not taken out of the hands of the users.

Heterogeneity is better than homogeneity

Large homogeneous installations are more vulnerable to cyberattacks. Heterogeneity provides greater resilience and is a prerequisite for the aggregation of many small-scale developments [19].

Infrastructure ownership must be equitable

To avoid digital colonialism, the infrastructure should be community-owned where possible. This also safeguards buy-in from the local community, reduces inequality at all levels and ensures that local solutions are tailored to local needs.

Equal attention should be given to all SDGs

A concentration of projects around a small number of SDGs may be detrimental to the overall progress. A balanced approach needs to be actively pursued.

While any attempt at a comprehensive solution to the issues discussed in this chapter must be doomed to failure, it is possible to suggest an approach that addresses the requirements listed here. Against the backdrop of the global trends outlined at the beginning of the chapter and in the face of the COVID-19 pandemic which has underscored the needs from radical, transformational thinking, the next few sections explore at a theoretical level the concept of a cellular architecture for physical environments for which IoT provides the blueprint.

10.5.1 Resilience

COVID-19 has demonstrated the fragility of what used to appear robust. While some scholars distinguish between sustainability and resilience [19], the capacity to withstand shocks is fundamental to a community's continued existence. The trend towards a greater concentration of people in cities means that they are increasingly identified as 'emergency-prone' environments [12]. Not only do infections spread quickly among their dense populations, but many other types of weakness can be identified including the risk of fire, flooding, coastal erosion, chemical concentration, heat stress and failure of utility distribution systems [12, 19, 31, 64]. Several authors have suggested hyper-localisation as a strategy for managing density and improving the urban experience typified by the '20-minute neighbourhood' concept [28]. Primarily aimed at delivering social benefits, the model proposes a planned urban fabric where most needs can be satisfied within a 20-minute walk, cycle or journey by public transport. Extending this idea, each neighbourhood could be designed to be as self-sufficient as possible as a radical response to the threats posed by climate change [45]. This cellular approach would include the local storage of water and energy, local food production using urban farming methods and the ability of each 'cell' to function autonomously at least for short periods. Changing the traditional planning mindset and embedding resilience planning in city governance procedures would be the greatest challenges. IoT technology is a natural fit for enabling cellular structures with its ability to deliver visibility of resource data. Additionally, the fog computing model enables reliable and responsive local processing which does not depend on external facilities.

10.5.2 Energy Distribution

The supply of electricity to city dwellers is just one example of a complex system where a single fault could affect many thousands of people. The typical electricity

grid is designed as a centralised system in which electricity is transmitted from huge generation facilities through three hierarchical levels of network to the eventual consumer [30]. The accelerating integration of small-scale renewable sources into this top-down structure is a destabilising factor which has prompted some radical re-thinking of grid design. One potential solution is to build an inherently distributed architecture around the concept of inter-connected microgrids [27]. As the name suggests, a microgrid is an independent, localised power network that does not depend on external supply or control. Generalising the problem to include other types of energy including heat, biogas and hydrogen leads to the concept of the energy hub which could be used to manage the distribution of energy within and between neighbourhood cells [51]. Local energy storage would also need to be integrated, and the intermittent connection of electric vehicles adds further complication [8]. The smart grid concept, based on dynamic control of supply enabled by smart meters, can be invoked to manage small-scale structures just as well as today's utility-scale services [30]. IoT architectures incorporating fog computing provide an ideal control solution.

10.5.3 *Local Clouds*

The fog computing model is fundamental to the concept of the smart city and to cellular architectures in particular. It satisfies the need for autonomous local data storage and processing without precluding connections to cloud services where appropriate. Although the majority of fog computing examples in the literature envisage a simple fog layer composed of homogeneous devices, the OpenFog reference architecture actually allows for hierarchical arrangements of fog devices in arbitrary configurations [50]. This adaptability allows the paradigm to support neighbourhood cells of any size. Because it is already structured into cells, the mobile communications network is ideally placed to provide the uppermost level of a fog hierarchy. Lower layers could then be sized appropriately for their function. Interoperability is essential in this scenario to ensure that low-cost solutions such as those built on individual or clustered single-board computers are acceptable [23].

The advantages of localised clouds are not restricted to the technological dimension. The placing of infrastructure within the area of use rather than in a remote data centre makes the possibility of local ownership and operation much more realistic. While this might be challenging for mobile network operators, it is an essential element of an equitable deployment. It is a safeguard against digital colonialism in the IoT domain and encourages local engagement, innovation and the development of human capital.

10.5.4 Smart Villages

Given the trend towards larger and denser cities, it is natural to focus on the urban environment; however, the cellular model is equally applicable to the rural case due to its self-sufficient character. Leaders in developing countries have consistently expressed the need for economic development and access to technology [13]. The cellular approach offers a way of structuring small-scale, rural projects so that their benefits are maximised in the short term and their long-term value remains with the local community. This is implicit in the IEEE ‘Smart Village’ programme⁸ in which volunteer engineers deploy renewable energy generation and localised microgrids to enable rural communities. By introducing a data layer based on the fog computing model, the IEEE concept could evolve quite naturally into a comprehensive cellular approach as outlined here.

10.6 Current and Future Challenges

The COVID-19 pandemic has dealt the world a shock from which it is struggling to recover. Costa & Peixoto [12] describe several examples of cities which are mobilising IoT technologies to detect viral outbreaks, trace contacts of infected people and manage the flow of information to health professionals. Other authors are imagining similar IoT-based responses to the crisis [7, 54, 60]. These initiatives demonstrate the power of the IoT concept and the range of problems it can be made to address. While responding to the immediate situation is important, the longer-term threats to human life require more radical thinking. There needs to be a willingness to imagine different ways in which technology – including IoT – can be used to improve the quality of life for all without risking that of future generations.

IoT technology is developing apace. Three clear trends are:

- The shrinking material and energy footprints of devices
- The evolution of communications networks to support low-power, low-bandwidth protocols
- The increasing availability of cheap, robust sensors

It is important that these trends continue, and pressure to further reduce power requirements by such means as ambient energy harvesting needs to be maintained. Greater interoperability between devices is also essential if the maximum benefit from IoT is to be realised. Listed below are several other areas for development which focus less on the technology itself and more on its context of use.

Greater data visibility across all SDGs

While the data needs of some SDGs are well-served, others are not. There is scope for IoT projects to be initiated which are aimed at filling the current data gaps.

⁸ <https://smartvillage.ieee.org/>

Public projects

Market forces are insufficient to address all of the SDGs. Public authorities at local, national and international level need to initiate projects to address those such as SDG14 (Life below water) which are not attractive to commercial interests. The deployment of sensors and the aggregation of data will be crucial to these efforts.

New commercial models

As well moving from linear to circular business models generally, there is also a need to alter the balance of power between large corporations and citizens of developing nations to protect their interests for the long term. A more cooperative approach needs to be developed in which local communities can participate in the ownership and operation of technical infrastructure more readily.

Harmonisation of localised models

The cellular model advocated in the previous section needs considerable elaboration and there are many dimensions to explore. The discussion highlighted several existing examples of localised structures including mobile cellular architectures, fog computing and microgrids. A comprehensive cellular model assumes that the reach of these and other localised facilities will neatly coincide. This assumption needs to be tested through modelling and simulation.

Definition of a reference architecture for cellular infrastructure

An understanding of how the various elements that make up a neighbourhood cell needs to be developed. A reference framework, possibly imagined as a series of layers, would be beneficial.

Protocols for inter-cell communication

A further assumption of the cellular model is that communication between cells and from a cell to the cloud are straightforward. This is also something to be tested through modelling and simulation and extends to material and energy flows as well as data.

10.7 Conclusion

The risks of ignoring the unsustainable aspects of human activity are great, but the slow pace at which they are gathering makes them easy to ignore in the short term. The COVID-19 pandemic has shown that transformational change is possible, however, and provides a salutary lesson on the interdependency of human life and other planetary processes. As the world recovers, attention is focussed not only on methods for avoiding and containing future outbreaks, but also on the processes by which the virus moved so quickly from being a biological curiosity to a global threat. This new period of reflection and international cooperation offers an opportunity to re-evaluate other dimensions of human society and their consequences.

Sustainability is an immensely complex issue with multiple dimensions, some of which conflict with each other. The 17 SDGs provide one means of rationalising the complexity. In the end, though, sustainability must be thought of as an emergent property arising from the interactions of myriad lower-level processes. Achieving the UN goal of sustainable development requires a holistic approach that gives equal attention to all dimensions of the problem simultaneously. Technology has an important role to play in many of the component areas, but it is vital to avoid the optimism of a purely technocentric perspective. Although the IoT is capable of facilitating efforts to increase sustainability through the fine-grained monitoring and control of physical resources, no technology is inherently benign. It is the engineers, developers, policymakers and entrepreneurs who decide how the technology is used. They can choose to employ it to promote sustainable development, or they can further entrench destructive linear business models and exacerbate existing inequalities.

The cellular architecture suggested here is an attempt to bias the design of IoT systems towards greater overall sustainability. It takes advantage of features already present in the fog computing model to imagine a more distributed approach than one based on a cloud-first assumption. It further positions the IoT as the main coordinating technology for the localised management of utilities and data processing services. Hyper-localisation has the potential to enable greater resilience in the face of external shocks, more equitable ownership of infrastructure and a better fit with local requirements. Improved interoperability is the main technical barrier to a more distributed approach, while at the socio-economic level the issue is that the commercial interests of large technology providers remain the dominant driving force behind large-scale IoT projects. An inherently distributed strategy challenges the centralising assumptions built into current business models and cloud-first architectures, but in the wake of the pandemic there is perhaps more willingness to re-evaluate them. Greater cooperation and a more holistic response to shared threats would allow the theoretical benefits of the IoT for sustainability to be fully realised.

References

1. R.A. Abbass, P. Kumar, A. El-Gendy, Car users exposure to particulate matter and gaseous air pollutants in megacity Cairo. *Sustain. Cities Soc.* **56** (2020). <https://doi.org/10.1016/j.scs.2020.102090>
2. P. Arora, *The Next Billion Users: Digital Life Beyond the West* (Harvard University Press, 2019)
3. D.N. Beede, R. Powers, C. Ingram, The employment impact of autonomous vehicles. *SSRN Electron. J.*, 1–33 (2018). <https://doi.org/10.2139/ssrn.3022818>
4. L. Von Bertalanffy, An outline of general system theory. *Br. J. Philos. Sci.* **1**, 134–165 (1950). <https://doi.org/10.1093/bjps/I.2.134>
5. G.H. Brundtland, *Report of the World Commission on Environment and Development: Our Common Future* (Oslo, 1987)

6. D. Castells-Quintana, H. Wenban-Smith, Population dynamics, urbanisation without growth, and the rise of megacities. *J. Dev. Stud.* **56**, 1663–1682 (2020). <https://doi.org/10.1080/00220388.2019.1702160>
7. V. Chamola, V. Hassija, V. Gupta, M. Guizani, A comprehensive review of the COVID-19 pandemic and the role of IoT, Drones, AI, Blockchain, and 5G in managing its impact. *IEEE Access* **8**, 90225–90265 (2020). <https://doi.org/10.1109/ACCESS.2020.2992341>
8. D.A. Chekired, L. Khoukhi, H.T. Mouftah, Fog-computing-based energy storage in smart grid: A cut-off priority queuing model for plug-in electrified vehicle charging. *IEEE Trans. Ind. Inform.* **16**, 3470–3482 (2020). <https://doi.org/10.1109/TII.2019.2940410>
9. C. Chen, D.J. Cook, A.S. Crandall, The user side of sustainability: Modeling behavior and energy usage in the home. *Pervasive Mob. Comput* **9**, 161–175 (2013). <https://doi.org/10.1016/j.pmcj.2012.10.004>
10. J. Colding, M. Colding, S. Barthel, The smart city model: A new panacea for urban sustainability or unmanageable complexity? *Environ. Plan B Urban Anal. City Sci* **47**, 179–187 (2020). <https://doi.org/10.1177/2399808318763164>
11. D. Coleman, Digital colonialism: The 21st century scramble for Africa through the extraction and control of user data and the limitations of data protection laws. *Michigan J. Race Law* **24**, 417 (2019)
12. D. Costa, J.P.J. Peixoto, The COVID-19 pandemic: A review of smart cities initiatives to face new outbreaks. *IET Smart Cities*. (2020). <https://doi.org/10.1049/iet-smc.2020.0044>
13. S. Custer, M. DiLorenzo, T. Masaki, T. Sethi, A. Harutyunyan, Listening to Leaders 2018: Is Development Cooperation Tuned-In or Tone-Deaf? (2018)
14. B. Dilkina, R. Houtman, C.P. Gomes, C.A. Montgomery, K.S. McKelvey, K. Kendall, T.A. Graves, R. Bernstein, M.K. Schwartz, Trade-offs and efficiencies in optimal budget-constrained multispecies corridor networks. *Conserv. Biol.* **31**, 192–202 (2017). <https://doi.org/10.1111/cobi.12814>
15. C. Echebarria, J.M. Barrutia, I. Aguado-Moralejo, The Smart City journey: a systematic review and future research agenda. *Innovations*, 1–43 (2020). <https://doi.org/10.1080/13511610.2020.1785277>
16. P. Ekins, J. Gupta, P. Boileau, GEO-6 Healthy planet , healthy people. United Nations Environment Programme (2019)
17. J. Elkington, Cannibals with Forks. Capstone (1999)
18. Ellen MacArthur Foundation, Towards the circular economy (2013)
19. T. Elmqvist, E. Andersson, N. Frantzeskaki, T. McPhearson, P. Olsson, O. Gaffney, K. Takeuchi, C. Folke, Sustainability and resilience for transformation in the urban century. *Nat. Sustain* **2**, 267–273 (2019). <https://doi.org/10.1038/s41893-019-0250-1>
20. S. Ermon, Y. Xue, R. Toth, B. Dilkina, R. Bernstein, T. Damoulas, P. Clark, S. DeGloria, A. Mude, C. Barrett, C.P. Gomes, Learning large-scale dynamic discrete choice models of spatio-temporal preferences with application to migratory pastoralism in east Africa, in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, (AAAI Press, Austin, 2015), pp. 644–650
21. European Commission, EU policy, strategy and legislation for 2020 environmental, energy and climate targets (2020). https://ec.europa.eu/info/energy-climate-change-environment/overall-targets/2020-targets_en. Accessed 28 Aug 2020
22. A.A. Feil, D. Schreiber, C. Haetinger, V.J. Strasburg, C.L. Barkert, Sustainability indicators for industrial organizations: Systematic review of literature. *Sustain* **11**, 1–15 (2019). <https://doi.org/10.3390/su11030854>
23. D. Fernández-Cerero, J.Y. Fernández-Rodríguez, J.A. Álvarez-García, L.M. Soria-Morillo, A. Fernández-Montes, Single-board-computer clusters for cloudlet computing in internet of things. *Sensors (Switzerland)* **19**, 1–26 (2019). <https://doi.org/10.3390/s19133026>
24. D. Freund, S.G. Henderson, D.B. Shmoys, Bike sharing, in *Sharing Economy: Making Supply Meet Demand*, (Springer, Cham, 2019), pp. 435–459

25. S. Fritz, L. See, T. Carlson, M. Haklay, (Muki), J.L. Oliver, D. Fraisl, R. Mondardini, M. Brocklehurst, L.A. Shanley, S. Schade, U. Wehn, T. Abrate, J. Anstee, S. Arnold, M. Billot, J. Campbell, J. Espey, M. Gold, G. Hager, S. He, L. Hepburn, A. Hsu, D. Long, J. Masó, I. McCallum, M. Muniafu, I. Moorthy, M. Obersteiner, A.J. Parker, M. Weissplug, S. West, Citizen science and the United Nations Sustainable Development Goals. *Nat. Sustain.* **2**, 922–930 (2019). <https://doi.org/10.1038/s41893-019-0390-3>
26. C.P. Gomes, T. Dietterich, B. Dilksina, E. Stefano, F. Fang, A. Farnsworth, A. Fern, X. Fern, D. Fink, D. Fisher, A. Flecker, D. Freund, A. Fuller, J. Gregoire, J. Hopcroft, Z. Kolter, W. Powell, N. Santov, J. Selker, B. Selman, D. Shelcon, D. Shmoys, M. Tambe, C. Wood, W.-K. Wong, X. Wu, S. Kelling, Y. Xue, A. Yadav, A. Yakubu, Z.M. Lou, Computational sustainability: Computing for a better world and a sustainable future. *Commun. ACM* **62**, 56–65 (2019). <https://doi.org/10.1145/3339399>
27. D. Gregoratti, J. Matamoros, Distributed energy trading: The multiple-microgrid case. *IEEE Trans. Ind. Electron.* **62**, 2551–2559 (2015). <https://doi.org/10.1109/TIE.2014.2352592>
28. C. Grodach, L. Kamruzzaman, L. Harper, *20 minute neighbourhood - living local research* (Melbourne, 2019)
29. A. Grover, T. Markov, P. Attia, N. Jin, N. Perkins, B. Cheong, M. Chen, Z. Yang, S. Harris, W. Chueh, S. Ermon, Best arm identification in multi-armed bandits with delayed feedback. *Int. Conf. Artif. Intell. Stat. AISTATS* **84**, 833–842 (2018)
30. S. Hinson, Electricity grids, (2019)
31. IPCC, Global warming of 1.5°C (2018)
32. ITU, Overview of the Internet of things (2012)
33. G. Javadzadeh, A.M. Rahmani, Fog computing applications in smart cities: A systematic survey. *Wirel. Netw.* **26**, 1433–1457 (2020). <https://doi.org/10.1007/s11276-019-02208-y>
34. S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Fryzman, G. Verin, et al., MEC in 5G networks. *ETSI White Pap*, 1–28 (2018)
35. J. Khazaei, W.B. Powell, SMART-Invest: a stochastic, dynamic planning for optimizing investments in wind, solar, and storage in the presence of fossil fuels. The case of the PJM electricity market. *Energy Syst* **9**, 277–303 (2018). <https://doi.org/10.1007/s12667-016-0226-4>
36. M. Kwet, Digital colonialism: US empire and the new imperialism in the Global South. *Race Cl* **60**, 3–26 (2019). <https://doi.org/10.1177/0306396818823172>
37. J.J. Li, B. Faltings, R. Jurca, Incentive schemes for community sensing. *Comput. Sust.*, 1–4 (2012)
38. Y. Li, W. Chen, L. Lu, Wearable and biodegradable sensors for human health monitoring. *ACS Appl. Bio. Mater.* (2020). <https://doi.org/10.1021/acsabm.0c00859>
39. M.M. del Martínez-Bravo, J. Martínez-del-Río, R. Antolín-López, Trade-offs among urban sustainability, pollution and liveability in European cities. *J. Clean. Prod.* **224**, 651–660 (2019). <https://doi.org/10.1016/j.jclepro.2019.03.110>
40. C. McLellan, The Internet of Wild Things: Technology and the battle against biodiversity loss and climate change, in *TechRepublic*, (2020) <https://www.techrepublic.com/article/the-internet-of-wild-things-technology-and-the-battle-against-biodiversity-loss-and-climate-change/>. Accessed 26 Aug 2020
41. Microsoft, Microsoft announces a \$1.1 billion investment plan to drive digital transformation in country including its first cloud datacenter region (2020). <https://news.microsoft.com/es-xl/microsoft-announces-a-1-1-billion-investment-plan-to-drive-digital-transformation-in-country-including-its-first-cloud-datacenter-region/>
42. F.-C. Mihai, C. Iatu, Sustainable rural development under Agenda 2030, in *Sustainability Assessment at the 21st Century*, (IntechOpen, 2020), pp. 9–18
43. T.S. Mohammed, O.F. Khan, A.S. Ibrahim, R. Mamlook, Fog computing-based model for mitigation of traffic congestion. *Int. J. Simul. Syst. Technol.* **19**, 5.1–5.7 (2018). <https://doi.org/10.5013/ISSST.a.19.03.05>

44. J. Morrish, M. Hatton, Global IoT market will grow to 24.1 billion devices in 2030, generating \$1.5 trillion annual revenue, in *Transform. Insights*, (2020) <https://transformainsights.com/news/iot-market-24-billion-usd15-trillion-revenue-2030>
45. R. Muggah, J. Kosslyn, Why the cities of the future are “cellular.”, in *World Econ. Forum*, (2019). <https://www.weforum.org/agenda/2019/11/with-climate-change-accelerating-the-city-of-the-future-is-cellular/>
46. M. Muñoz, J.D. Gil, L. Roca, F. Rodríguez, M. Berenguel, An iot architecture for water resource management in agroindustrial environments: A case study in almería (Spain). *Sensors (Switzerland)* **20** (2020). <https://doi.org/10.3390/s20030596>
47. T.D. Nielsen, J. Hasselbalch, K. Holmberg, J. Stripple, Politics and the plastic crisis: A review throughout the plastic life cycle. *Wiley Interdiscip. Rev. Energy Environ* **9**, 1–18 (2020). <https://doi.org/10.1002/wene.360>
48. Z. Ning, J. Huang, X. Wang, Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wirel. Commun.* **26**, 87–93 (2019). <https://doi.org/10.1109/MWC.2019.1700441>
49. E. O’Dwyer, I. Pan, S. Acha, N. Shah, Smart energy systems for sustainable smart cities: Current developments, trends and future directions. *Appl. Energy* **237**, 581–597 (2019). <https://doi.org/10.1016/j.apenergy.2019.01.024>
50. OpenFog Consortium Architecture Working Group, OpenFog Reference Architecture for Fog Computing (2017)
51. K. Oreounig, R. Evins, V. Dorer, Integration of decentralized energy systems in neighbourhoods using the energy hub approach. *Appl. Energy* **154**, 277–289 (2015). <https://doi.org/10.1016/j.apenergy.2015.04.114>
52. Partnership on Measuring ICT for Development, Joint proposal of ICT indicators for the Sustainable Development Goal (SDG) indicator framework, in *Expert Group Meeting on the Indicator Framework for the Post-2015 Development Agenda*, (New York, 2015), pp. 25–26
53. S. Pettigrew, L. Fritschi, R. Norman, The potential implications of autonomous vehicles in and around the workplace. *Int. J. Environ. Res. Public Health* **15** (2018). <https://doi.org/10.3390/ijerph15091876>
54. M.S. Rahman, N.C. Peeri, N. Shrestha, R. Zaki, U. Haque, S.H.A. Hamid, Defending against the novel Coronavirus (COVID-19) outbreak: How can the Internet of Things (IoT) help to save the world? *Heal Policy Technol* **9**, 136–138 (2020). <https://doi.org/10.1016/j.hplt.2020.04.005>
55. A. Renda, M. Laufer, IoT 4 SDGs - What can the digital transformation and IoT achieve for Agenda 2030? (2020)
56. R. Ritchie, O.-O. Mispy, Measuring progress towards the Sustainable Development Goals, in *SDG Tracker*, (2018) <https://sdg-tracker.org/>. Accessed 28 Aug 2020
57. S. Sareen, S.K. Sood, S.K. Gupta, IoT-based cloud framework to control Ebola virus outbreak. *J. Ambient. Intell. Humaniz. Comput.* **9**, 459–476 (2018). <https://doi.org/10.1007/s12652-016-0427-7>
58. S. Schlottfeldt, M. Emilia, T. Walter, J.A.F. Diniz-filho, M.E.M.T. Walter, Challenges in computational sustainability : A new multi-objective artificial immune system approach to deal with biodiversity conservation, in *Workshop on Evolutionary Multi-Objective Optimization at the IEEE 2015 International Congress on Evolutionary Computation, CEC'2015*, (2015)
59. S. Seccombe, Metal detecting sensors for anti-poaching, in *WildLabs.net*, (2020) <https://www.wildlabs.net/resources/case-studies/metal-detecting-sensors-anti-poaching>. Accessed 26 Aug 2020
60. R.P. Singh, M. Javaid, A. Haleem, R. Suman, Internet of things (IoT) applications to fight against COVID-19 pandemic. *Diabetes Metab. Syndr. Clin. Res. Rev.* **14**, 521–524 (2020). <https://doi.org/10.1016/j.dsx.2020.04.041>
61. A. Sodiq, A.A.B. Baloch, S.A. Khan, N. Sezer, S. Mahmoud, M. Jama, A. Abdelaal, Towards modern sustainable cities: Review of sustainability principles and trends. *J. Clean. Prod.* **227**, 972–1001 (2019). <https://doi.org/10.1016/j.jclepro.2019.04.106>

62. S.R. Steinhubl, D. Feye, A.C. Levine, C. Conkright, S.W. Wegerich, G. Conkright, Validation of a portable, deployable system for continuous vital sign monitoring using a multiparametric wearable sensor and personalised analytics in an Ebola treatment centre. *BMJ Glob. Health* **1** (2016). <https://doi.org/10.1136/bmjgh-2016-000070>
63. B.L. Sullivan, J.L. Aycrigg, J.H. Barry, R.E. Bonney, N. Bruns, C.B. Cooper, T. Damoulas, A.A. Dhondt, T. Dietterich, A. Farnsworth, D. Fink, J.W. Fitzpatrick, T. Fredericks, J. Gerbracht, C. Gomes, W.M. Hochachka, M.J. Iliff, C. Lagoze, F.A. La Sorte, M. Merrifield, W. Morris, T.B. Phillips, M. Reynolds, A.D. Rodewald, K.V. Rosenberg, N.M. Trautmann, A. Wiggins, D.W. Winkler, W.K. Wong, C.L. Wood, J. Yu, S. Kelling, The eBird enterprise: An integrated approach to development and application of citizen science. *Biol. Conserv.* **169**, 31–40 (2014). <https://doi.org/10.1016/j.biocon.2013.11.003>
64. F.N. Tonmoy, S. Hasan, R. Tomlinson, Increasing coastal disaster resilience using smart city frameworks: Current state, challenges, and opportunities. *Front Water* **2**, 1–13 (2020). <https://doi.org/10.3389/frwa.2020.00003>
65. U.S. National Science Foundation, NSF Announces Expeditions in Computing Awards (2008)
66. UN, Transforming our world: The 2030 agenda for sustainable development (2016)
67. UN DESA, *The Millennium Development Goals Report 2015* (New York, 2015)
68. UN DESA, World urbanization prospects (2018)
69. UN Development Programme, Human development reports (2020). <http://hdr.undp.org/en>. Accessed 19 Aug 2020
70. UNCTAD, *Digital Economy Report 2019 : Value Creation and Capture - Implications for Developing Countries* (United Nations, New York, 2019)
71. USAID, Fighting Ebola:A grand challenge for development (2015) . <https://www.usaid.gov/grandchallenges/ebola>. Accessed 26 Aug 2020
72. WEF, *The Future of Nature And Business* (Geneva, 2020)
73. WMO, Global annual to decadal climate update (2020)
74. World Bank, *The Changing Nature of Work 2019* (Washington, D.C, 2018)
75. World Bank, Affordable and clean energy, in *SDG Atlas*, (2018) <http://datatopics.worldbank.org/sdgatlas/SDG-07-affordable-and-clean-energy.html>. Accessed 28 Aug 2020
76. World Bank, Open Data (2020)
77. World Bank, World Development Indicators (2020) . <https://databank.worldbank.org/source/world-development-indicators>. Accessed 28 Aug 2020
78. World Economic Forum, Internet of Things Guidelines for Sustainability (2018)
79. World Economic Forum, IoT for Sustainable Development Project (2019). <http://widgets.weforum.org/iot4d/>. Accessed 15 Jul 2020
80. K.N. Yogalakshmi, S. Singh, Plastic waste: Environmental hazards, its biodegradation, and challenges, in *Bioremediation of Industrial Waste for Environmental Safety: Volume I: Industrial Waste and Its Management*, ed. by G. Saxena, R. N. Bharagava, (Springer Singapore, Singapore, 2020), pp. 99–133
81. H. Zahmatkesh, F. Al-Turjman, Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies - an overview. *Sustain. Cities Soc.* **59**, 102139 (2020). <https://doi.org/10.1016/j.scs.2020.102139>
82. C. Zheng, J. Yuan, L. Zhu, Y. Zhang, Q. Shao, From digital to sustainable: A scientometric review of smart city literature between 1990 and 2019. *J. Clean. Prod.* **258** (2020). <https://doi.org/10.1016/j.jclepro.2020.120689>

Chapter 11

Applications of IoT and Cloud Computing: A COVID-19 Disaster Perspective



Kshitij Dhyani, Thejineaswar Guhan, Prajjwal Gupta, Saransh Bhachawat, Ganapathy Pattukandan Ganapathy, and Kathiravan Srinivasan

Contents

11.1	Introduction	288
11.2	Video Conferencing	288
11.3	E-Gaming and Video Streaming	290
11.4	Future Opportunities	290
11.5	Application of IoT in Transportation	291
11.5.1	Logistics	291
11.5.2	Innovations	292
11.5.3	Applications	293
11.6	Fleet Management [7, 8]	295
11.7	Drone-Based Delivery (DBD) [10]	297
11.8	Autonomous Vehicles [12]	299
11.9	Future Opportunities	301
11.10	Manufacturing Industry During COVID-19	302
11.11	IoT-Enabled Automation for Remote Manufacturing and Businesses	304
11.12	Monitoring Using IoT	305
11.12.1	Vision-Based Control Systems	308
11.12.2	KC 901 Smart Helmet	308
11.12.3	Post-COVID-19 Opportunities	309
11.13	Applications of IoT and Cloud Computing in Healthcare	309
11.13.1	HealthCare	309
11.13.2	New Innovations	311
11.13.3	Applications	312
11.14	Post-COVID-19 Opportunities	316
11.15	Applications of IoT and Cloud Computing in Agriculture	318

K. Dhyani · T. Guhan · K. Srinivasan (✉)

School of Information Technology and Engineering, Vellore Institute of Technology (VIT),
Vellore, Tamil Nadu, India

e-mail: kathiravan.srinivasan@vit.ac.in

P. Gupta · S. Bhachawat

School of Computer Science and Engineering, Vellore Institute of Technology (VIT), Vellore,
Tamil Nadu, India

G. P. Ganapathy

Centre for Disaster Mitigation and Management, Vellore Institute of Technology (VIT), Vellore,
Tamil Nadu, India

11.15.1 Agriculture Monitoring	318
11.16 Summary	319
11.17 Conclusion	320
References	320

11.1 Introduction

Arguably the biggest health crisis of the century, the COVID-19 pandemic has made billions of people change the way they live and work. Even after months of quarantine, the total number of globally identified cases has reached over 80 million; these conditions have made it unsuitable for schools, colleges, entertainment shows, etc., to commence again at the moment. The education and the entertainment industries are, therefore, two of the most affected industries all over the world.

The education industry is arguably one of the most important industries in the world. Millions of students were resigned to their homes without schools for a while as social distancing put a stop to regular classroom lectures; institutes and students turned to virtual classrooms. Online teaching, which was still a novel idea a year ago, has now become the staple practice for educators all over the world. As a result of this, there have been massive developments in the field of cloud computing which has played a crucial role in ensuring a proper learning environment and resources. The entertainment industry has also seen changes similar to the ones in the education industry. Social distancing has halted public events such as watching live sports, going out with friends and going on trips and tours. This has caused a massive increase in Internet traffic with more and more people taking interest in online activities such as vlogging and gaming. This has prompted companies and digital content creators to produce new innovative products and content.

11.2 Video Conferencing

Video conferencing has been around for quite some time now. The main uses were to call far off friends or hold meetings with people separated all around the globe, thus, its usage was not very extensive in our daily lives. However, this has changed during the COVID-19 pandemic period. To continue working while social distancing, video calls were seen as the apt solution and so there was a huge spike in the downloads of video calling apps such as Zoom, Microsoft Teams and Google Hangouts Meet (Fig. 11.1) [1].

This sudden and large-scale shift to video conferencing has resulted in low quality and audio lags in the calls due to the high density of data being transferred simultaneously. This has made working over video calls difficult and often frustrating. To battle this major issue, various companies are working towards enhancing their IT infrastructure further and have boosted their endeavour to shift to the newer concept of edge computing.

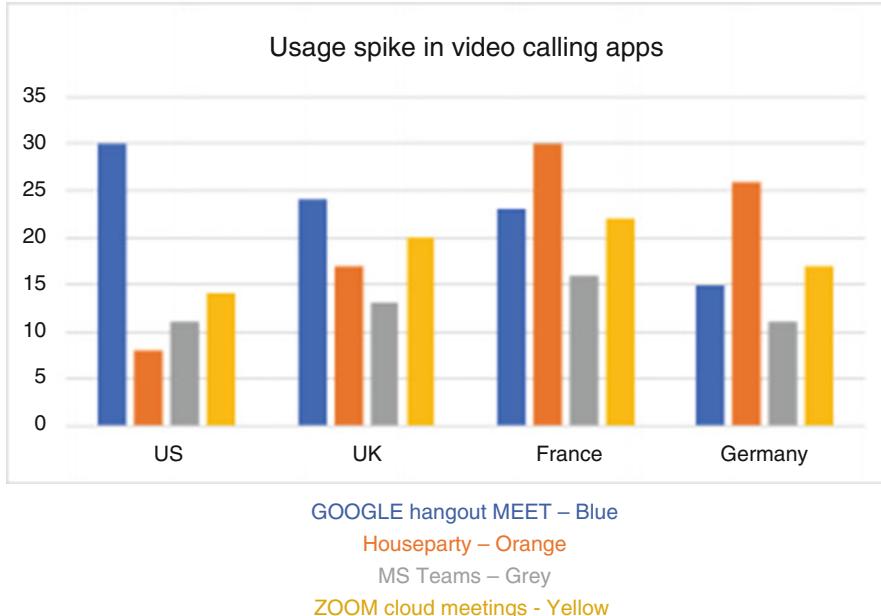


Fig. 11.1 Growth in the total downloads of the mentioned applications

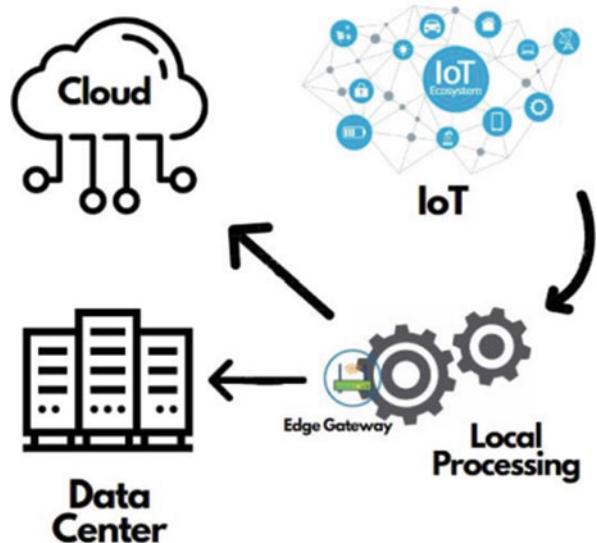
The basic aim of edge computing is to decentralize the current cloud computing system, that is, to bring the data storage and computation to the user. This will help in increasing the data processing speed to a very large extent and, therefore, help applications to reduce latency [2]. Video calls work by compressing analogue data captured by the webcam and the audio device into continuous waves of frequencies and amplitudes representing colour, brightness, audio, etc. The data compression is done using codecs, which compress this data into digital packets. These compressed packets are then transferred over a digital network to the target device. Through edge computing, the application will not have to send the entire data to the cloud, but instead the data will be stored and computed locally and only the relevant data will need to be sent [3]. Below is a diagram [2] summarizing how edge computing works (Fig. 11.2).

Edge computing allows data from Internet of Things devices to be analysed at the edge of the network before being sent to a data centre or cloud.

To decide which data should go to the customer's local database and which should go to the cloud, another network computing concept called Fog Computing is being incorporated. The purpose of fog computing is to give an architecture that can bridge the gap between IoT and Cloud/Edge computing. Although it is a fairly new paradigm, fog computing can be a major factor in enhancing device security and maintainability [4].

Other than the purpose of video calling, edge computing and fog computing have a wide range of applications such as in IoT-based sensors and 5G technology. Edge computing also has a large potential in Augmented Reality devices and applications

Fig. 11.2 Working of edge computing



such as Instagram and Pokemon Go; these devices need to curb latency and edge computing is a highly effective method for that. Edge computing also makes the overall computing process cheaper as most of the data is stored and processed locally.

11.3 E-Gaming and Video Streaming

In the months of lockdown all over the world, a lot of people have taken a liking to E-gaming, vlogging, etc. A Chinese-based game developer company Tencent has seen a surge of more than 30% year-on-year in its revenues for games and video services in the first quarter of 2020 alone [5]. Video streaming has also seen a rise similar to that of E-gaming and video calling. As the lockdown prevented people from going out, more and more people started video streaming, vlogging, etc. This has also given rise to problems similar to video conferencing where players have to complain about high ping rates and latency in the servers. This is another good potential domain where using IoT and edge computing seamless gaming can be established.

11.4 Future Opportunities

The shift from offline learning and entertainment to an online environment has brought with it a large number of limitations and issues. One of the best ways to deal with these is by further developing new computing paradigms such as

Table 11.1 Summary table

Technology	Description	User cases	Benefits
Edge computing and fog computing	Edge computing aims to decentralize data storage and processing by storing data closer to the user and fog computing provides the architecture transferring only the relevant data to the cloud	1. Video conferences 2. E games and video streaming 3. Businesses and corporations	1. Reduced data latency 2. Better user experience and collaboration from far off places 3. Enables social distancing 4. Reduced cost of operations 5. Security and maintainability increase

edge computing and fog computing and integrating them with IoT. This can do wonders in the education and entertainment domains by providing much faster and seamless connectivity between users. Some future applications of this integration are full-fledged distance learning programs and the creation of more opportunities for entertainers such as vloggers and streamers so that their content can reach a greater set of audience and is not ruined by network interruptions and development of larger online games with better graphics.

Overall, there are a lot of opportunities in both education and entertainment domains that can be realized by the integration of cloud and IoT and the circumstances due to COVID-19 have boosted the efforts to achieve this integration to a very large extent; however, we are yet to see these methods applied to any application on a significant scale (Table 11.1).

11.5 Application of IoT in Transportation

11.5.1 Logistics

During this pandemic, the transportation and logistics industry was impacted heavily. Some of the services include multimodal transportation, freight forwarding, warehousing and inventory management. This industry plays a crucial role where components of a final product are built in multiple locations. Hence, for the final product to be built, all the components need to be transported to be under one roof to assemble the final product. The current situation created bottlenecks in the logistics aspect which in turn reduced the rate of production of the final product.

Some of the factors which caused these bottlenecks are the following:

The situation of the pandemic – The disease was spreading at an unprecedented rate as the spread could occur easily. Lockdowns were imposed and most of the activities conducted by people were restricted.

Disrupted supply chains – Due to the restrictions imposed by governments, road transport was scarcely available. As a result, cargos were stuck at multiple ports and warehouses which were essential for production.

Strong demand for e-commerce – Customer preference for the mode of purchase was changing as well. With the circumstances inducing fear in the customers, demand for transportation and logistics was increasing more than ever. So, the industry had to channelize the workforce for e-commerce deliveries as well.

11.5.2 Innovations

Throughout this subchapter, the main focus would be to discuss the applications of IoT in transportation and logistics. The applications were made possible, thanks to innovations in IoT. In this section, RFID, GPS, radar-based systems and cameras are mainly used. Hence, a brief introduction to each of these domains is provided.

RFID: Radio Frequency Identification is a technology where information is encoded into tags or labels. These labels emit waves that contain the information and are passed onto other devices through readers. RFID is part of a group known as Automatic Identification and Data Capture and these types of devices identify and gather information regarding a particular item and add it to the database as an entry without the support of a human.

GPS: Global Positioning System is a technology that helps the user to navigate through places. This is done using a receiver on the vehicle's side which receives transmission regarding the directions. The transmissions are received through satellites. When a request is sent, the timestamp is noted, and the satellite receives the signal and sends a response for the same. The timestamp when the response is received is subtracted from the timestamp of transmission which is then used to calculate the distance. This technology has advanced by suggesting the best possible routes while also suggesting alternate routes based on reviews from other travellers as well, therefore, increasing the accuracy of the system.

Radar-based technology: Radar or *Radio Detection And Ranging* works by emitting radio waves at objects and the reflection helps it determine the nature of the object. Typically, radar is used in aviation but improvements in technology have made vehicles like a car to accommodate such a technology.

11.5.3 Applications

11.5.3.1 Inventory Tracking and Warehousing [6]

The very first impression one may get is why IoT for inventory management. It turns out that IoT along with cloud computing can increase the efficiency of the staff working on inventory management. Traditionally, the person would have to go through individual items and manually input the records sequentially. The job, in general, is very tedious, and along with this brings an array of problems. Some of the problems include the following:

Hard tracking of inventory – When the inventory is stored in a place, an entry is made. If these entries go missing or contain the wrong information, finding the right item becomes extremely painstaking.

Update problems – Similar to the problem above, except if an inventory is moved to another location and a mismatch in entries is caused.

Management of lead time – Consider a situation wherein the components used in an assembly pipeline are sourced from multiple locations. Tracking multiple components consequently becomes extremely time consuming and human errors may just as well further increase the delay to complete the assembly for the inventory.

To solve these problems, IoT devices enabled with the cloud are used. This makes the process seamless and efficient. For this application, RFID tags make a huge difference. Radio Frequency Identification or RFID carries information for a particular item in this case and these items are uniquely identified through an ID. There are quite some options on these tags and some of the tags emit radio waves which are listened to by RFID antennas. The attended waves are sent to the RFID reader, which processes the signal and this signal is transmitted to the cloud where all the other activities can take place. Here, the information could consist of conditions of storage, storage venue and timestamp. One thing to note is that, RFID tags have trouble scanning through metals and liquids. The process would go like this:

A component or inventory either in batches or individual items would be given an RFID tag. Through the reader, the tag is read. This information is processed and passed to the cloud.

The cloud analyses the incoming stream and presents the information in an analytic dashboard like the platform to make it easily interpretable to the user.

Now, this item is done with the production aspect or has passed the outlined pipeline. It is transported to the location of storage through small cranes or robots (can be discussed further).

Just after placing the item in the location, the person who stores the item will scan it and this data is passed through the cloud. Essentially, valuable information such as location, timestamp, conditions, and most importantly the item is being stored.

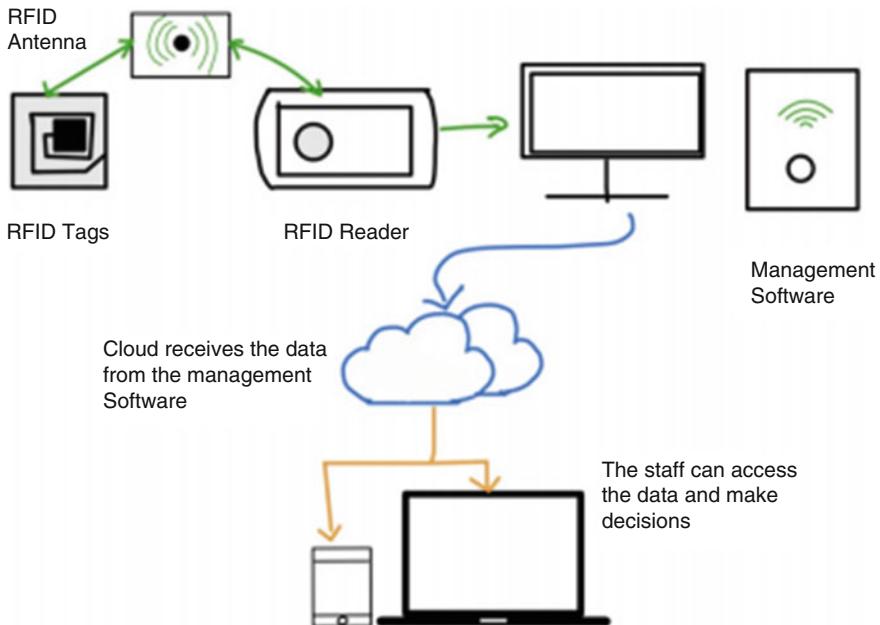


Fig. 11.3 An architecture for the RFID tags with cloud

To showcase this information, the outline of the inventory could be taken and the information of the position could be displayed graphically. Figure 11.3 shows the workflow of the RFID tags along with the cloud.

As seen, the human dependency decreases drastically which is a huge advantage considering the pandemic wherein the human would more likely be a monitor to the process and handle issues when one arises. This process increases the visibility of the flow of inventory. As a result, finding items would no longer create the inconvenience it previously used to. With this implementation, the chances of human error are almost null. Moreover, the tracking of products midway through a pipeline also becomes easier due to the tags. Consider a situation of a missing component crucial for the assembly pipeline. In the traditional approach, finding the root cause of the interruption of the pipeline would take time and finding the missing components can take even more time, therefore, increasing the lead time in the worst-case scenario. In cases of a limited supply of a component, the staff would be notified early on while also giving the staff an approximate of the number of final products which could be assembled. Therefore, the staff could react to the needs as per the demands.

In the end, demand is what drives a business. If a business has demand, it has sales which makes the business run daily. A perk of using cloud computing would be using machine learning-enabled solutions to predict demand for particular items. Some of the items are seasonal and the assembly line would have to make

changes in the production side. It is not just the assembly line but also the storage conditions. For example, some materials may need particular conditions such as humidity to make sure that the material does not deteriorate. As a result, it becomes a necessity that the business adapts to such changes and forecasts from machine learning algorithms can prepare the business for the same.

As a business, it is crucial to know which assets or equipment are used to their full potential. Businesses invest in machinery to get a good return on investment; however, if the equipment purchased has less utilization than expected or not used at all, the business could potentially make losses as the economies of scale on the output decreases. In some cases, businesses lease out equipment and, in such cases, the businesses could avoid heavy transactions by discontinuing such services. IoT helps businesses make such decisions. Based on the tags passed through a machine, the logic in the cloud can determine whether the purchased or leased equipment is living up to the expectations. Most importantly, these decisions would be data-driven, hence giving solid backing to a particular decision.

As stated earlier, efficient usage of machinery will make a huge difference for the business. However, maintenance of machinery is something all businesses have to do to make sure that they run without causing any problem. Typically, machines are maintained in a periodical fashion wherein the employees would check in on the machine at every fixed time interval. This time interval is decided based on the type of system and can last between weeks to perhaps years. However, at times machines are overutilized to meet sudden surges of demand and the machines may not get the cool down time it needs to keep it running for a long period of time. As a result, machines can break down all of a sudden, causing a particular failure which can cost potentially greater than a regular service. So, IoT sensors can be used to check on the status of the machines. By looking at certain parameters of the performance, decisions can be taken for the service of the machine. Also, machine learning-based decisions can be taken as well. Considering the forecast, machine learning-based decisions could also be considered in such cases.

11.6 Fleet Management [7, 8]

As the word suggests, this application focuses on managing a fleet of vehicles, particularly road-based transports, to make informed business decisions. Some of these decisions include committing to the customer with regards to estimated delivery time or date, choice of vehicle for an operation concerning the availability, and most importantly real-time tracking of the fleet in general. The systems earlier used to rely on track based on the status of the delivery. So, the business would only know the status of the vehicle when it would have delivered the cargo.

This small scenario is just a small chunk of the application. The other factor which can be measured is driver performance. In some cases, the cargo transported can be very fragile and expensive. In these situations, a driver would have to take utmost caution to avoid causing damage to the cargo on the back. This is a very

crucial decision to make and IoT makes this happen. Businesses can evaluate the driving of the driver through a metric. The transporter could look at the driving performance of all their drivers and by analysing data collected from the sensor from previous deliveries, can choose the best driver for the job. In this way, the cargo is delivered safely through safe hands. Also, the performance of existing drivers can be improved by giving feedback to make other drivers ready for such situations as well.

This application helps the driver in the current situation for cases of assistance. With the pandemic in place, it would be crucial for the driver to pick safe options for a resort in case of emergencies. Simply, the real-time location feature provided by the sensor could assist the driver to place safe enough. One thing to consider is that unnecessary contact with people may prove to be a safety concern for the driver. Communication with the driver by inspecting the real-time location will reduce the risk of the driver contracting the disease. For example, the driver could be assisted with fuel and by making payment as soon as the vehicle approaches the fuel stations. Additionally, with restrictions imposed all around, the main headquarters could assist the driver with a route fast enough to reach the destination.

With the COVID situation worsening day by day, businesses have to make sure that the equipment used by them is in the best possible conditions. Transports have to avoid unwanted stops caused by mechanical failures which may expose drivers to undesirable conditions. Also, with restrictions such as lockdowns put across by governments at short notices, such delays may be much more expensive than they seem. Potentially, a major technical problem could cause the transport to be stuck during the restrictive period. Fleet management prevents such a problem by monitoring the internals of the cars using sensors to avoid such a problem. If the sensor does detect the problem, the transport can be fixed across before the journey commences, making the process efficient and seamless.

One more application which will be used for a long time is vaccine distribution. The pandemic has caused widespread panic and with vaccines under trials, transporting them would be a challenge and if not handled with caution, could make the vaccine unusable. Promising vaccines tend to need frigid conditions at round -20° to -70°C and most of these vaccines are made of m-RNA. To monitor and maintain such conditions, IoT can be used and such a transportation can be known as cold chain management [9]. For monitoring individual containers of storage, the same RFID can be used but to monitor the condition, thermal sensors can be used and transported. This thermal sensor along with other sensors which monitor other conditions will transmit the data to the cloud which can be centrally accessed. This solution gives a real-time tracking of the location and conditions of the vaccine and enables the transportation systems to be smooth.

The technology used for this application is Radio Frequency Identification or RFID. This is used for tracking the products or cargo within the vehicle and is embedded into microchips commonly known as tags. These tags radiate radio waves which are listened to by a receiver, giving real-time feedback on the cargo carried. To monitor the car's performance, OBD or On-Board Diagnostics is used. OBD is used by the car's system to diagnose itself and this diagnosis can be accessed through

OBD-II, which plays a similar role to a listener. For real-time location tracking, GPS commonly known as Global Positioning System is used. Combined, these devices transmit this data through a cellular network to a cloud, which can be accessed by the headquarters.

11.7 Drone-Based Delivery (DBD) [10]

In the last application, the drivers were the quintessential element to complete the whole process within time. Issues with drivers indirectly meant a delay in delivery of the product. Other than that, it could also be seen that coordination with the driver would also play a crucial role to get the desired outcome. This solution solves most of the driver-related problems - drone-based delivery system. This concept is still taking shape and is a hot topic when it comes to delivery, particularly in the e-commerce sector. To be precise with the issues DBD solves:

Complete control centrally of all drones

The advantage for the business if they have the aim to reduce the carbon dioxide emissions (Most of the drones are battery-based)

It will not be shift-based. When the drone is low on battery, charge it for some time and it is on the go again.

Faster deliveries and the movement is aerial; as a result, the distance covered decreases while also being energy-efficient

The efficiency of inventory and warehouses will have to increase. Faster delivery could mean potential one-day deliveries in a locality.

The drone systems will work on a GPS or global positioning system. The drone would be aided with the best possible route to a location based on decisions made by the GPS. For carrying the package, the drone would have claw-like arms that can hold cargo for delivery. Now, it would be crucial as to where the package would be dropped because at times the cargo can be delicate and easily broken. The drones by Amazon, specifically in the USA, check to drop the package in the compound yard. Also, it has to be cautious about the obstacles it might face on its way. Consider the Amazon drone; it has machine learning and computer vision-based algorithm to detect obstacles such as humans, animals and wires as well. So, the drone can take evasive action to avoid collision with the same. So, there are cameras attached to the drone which captures a live video feed. This feed is processed and the algorithm detects the obstacles on the same. The drone would be in contact with cloud-based servers and these servers provide instructions to the drone with regard to the delivery address and information necessary for the same. Figure 11.4 illustrates the same.

This application is hugely beneficial for the logistics industry during this COVID situation as the problems caused by national restrictions are conquered without causing harm to any human being. Initially, this technology will be expensive to adopt, however, will prove to be valuable considering the cost being human lives. Not only are the lives of people working in the logistics industry saved but also lives

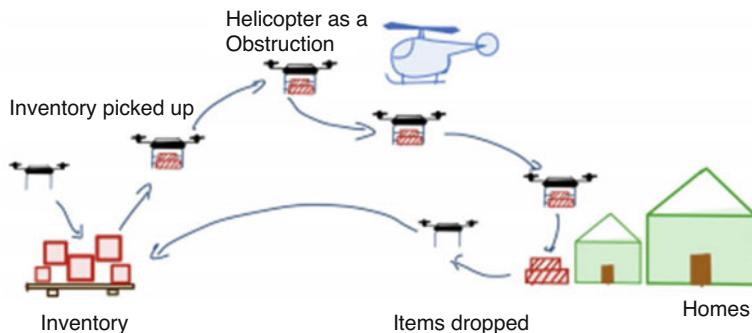


Fig. 11.4 A cycle of drone deliveries

of the people who provide and receive the cargo. This disease is contagious and it is beneficial if unwanted contact or spread is prevented.

Moreover, this application has been used for collecting samples from people who have been suspected to be infected by COVID-19. A use case would be to deliver groceries to people living in COVID hotspots. This, in a way, protects government officials who work closely with such people from contracting the disease themselves.

Drone-based technology is also used for disinfecting public areas to avoid contamination known as disinfection drone [11]. Instead of carrying cargo, the drone can be fitted with a tank and a sprinkler system. The drone will again use a camera-based system to give the operator vision regarding the movement and GPS to navigate to the selected public area. The controller will have access to a sprayer and would have to use both simultaneously to keep the drone moving while it is spraying. This aspect of control is used to fine-tune the actuator of the sprinkler and to turn on/off the sprinkler in the drone. The picture of the drone is displayed on Fig. 11.4. The drone can be automated similar to the technology described above; the GPS would guide the drone and computer vision-based algorithms would help avoid obstacles in the flight path. The drone-based delivery system can carry across 20 to 200 kg of cargo based on the power of the engines used. Such a motive from logistics businesses could also help them fulfil their Corporate Social Responsibility, wherein the business benefits society by disinfecting public places. In addition, drones can be used to detect possible contaminated places. A place can be considered contaminated if a lot of people with the possibility to be COVID positive visit the place. However, predicting this possibility can become effortless by attaching thermal imaging to drones. The data obtained here is the thermal image which looks at the crowd and the temperature based on the thermal image. The drone could pass this data to the cloud which stores and makes predictions of the possibility of using Machine Learning Algorithms. If the possibility is high, a speaker attached to the drone could make announcements to maintain social distancing while summoning the disinfectant drone (Fig. 11.5).



Fig. 11.5 Labelled picture of the dispenser drone [11]

Delivery drones [10] can also be used in applications such as kit delivery wherein drones deliver a kit that contains all the necessary materials to conduct a COVID test by the patient at their residence of stay. The samples can be enclosed in the same casing and sent back to the lab for testing. The security of this mechanism may be questioned; however, it can be solved using OTP or One-time Password where the users would have to possibly recite or type in the OTP on a keypad on the drone. The location of the hospital from which the kits are sent is hardcoded when a drone is purchased. For every test sample collection, the location is sent through the cloud to the drone and it can navigate through GPS. Cloud would play an essential part in this use case as the human dependency, assuming the drones to be self-controlled, would be low, therefore, requiring quick response for movements and this can be done by integrating software which queues workload onto the drone efficiently. Also ingesting data in the form of audio and video required to fine-tune the movement of the drone could be easily done using cloud integration.

11.8 Autonomous Vehicles [12]

Self-driving cars have been the talk for a long time in the transportation industry. These cars, as the name suggests, can be driven by the technology present within the car. However, to gain knowledge of the surroundings, it becomes necessary to implement sensors that provide such data to the processing technology and that is what IoT in this application does. Before moving on, here are some rubrics through which the cars are assessed or classified

Level 0: A vehicle that has a complete dependency on humans. This is a regular car.

Level 1: A vehicle that provides some assistance to humans. Examples of such assistance include cruise control wherein the speed remains constant.

Level 2: A vehicle that has partial automation. The car would have some control, but not the entirety, therefore, human dependency is still present

Level 3: A vehicle has conditional automation. The car in some cases may examine the environment and take actions based on the surroundings. Examples include traffic jam navigation.

Level 4: A vehicle is highly automated, however, is limited to an optimal condition. This can include roads with perfect lane systems.

Level 5: A vehicle is fully automated and there are no conditions concerning self-driving abilities and no human intervention is needed.

Most of the self-driving vehicles, specifically cars, have attained Level 4 in the above rubrics. However, Level 5 is no longer a distant reality as it is expected that these cars will be in mass production in the upcoming years.

The technology used in these differ from manufacturer to manufacturer, however, one common technique is Radar-based technologies along with laser beams and cameras. Inputs from these components are taken and pre-processed, and the system creates a path which the car would follow. To follow the path formed, the systems in the vehicle control the breaking, steering and acceleration aspect of the car. The components detect obstacles and the procedure to avoid them is specified in an algorithm. The vehicle would use GPS-based systems to keep track of the position concerning the final destination and make sure the route taken is the shortest and fastest route possible.

With autonomous trucks also under development, this technology will hugely benefit the logistics industry as the availability of drivers is a huge factor during these tough times. With such technologies, humans would more likely be a reserve as vehicles progress towards self-driven fashion. Also, this would save the lives of many drivers as they will not be exposed to infection while also proving to be a huge advantage for businesses. This method provides benefits such as:

Shift-based system no longer a problem

Could potentially run all the time

Can be controlled and tracked easily without worrying about overpayment to the drivers

One thing to note is that self-driving cars are still under development and it will take time to mass-produce autonomous vehicles. Although the control and dependency of cars by humans have decreased drastically, this comes in because manufacturers are climbing the hierarchy of the autonomous level ladder. Climbing this ladder will more likely impact the technology used to convert vehicles and reduce the actions taken by the driver to navigate the car.

Self-driving vehicles will also have an additional perk of self-maintenance or checkup. As a result, the business can plan for maintenance based on the usage of

cars through the sensors used in cars. This will prevent the cycle service mentality and focus more on the individual needs of the vehicle. (Similar to On-Board Diagnostics)

A feature widely applauded is the car-to-car communication. This typically occurs when a car from a manufacturer is close to another car from the same manufacturer and these vehicles can share information. This could potentially be feedback on an upcoming route as information regarding the itinerary is shared. Currently, this technology is used to share unexpected obstacles directly to the driver in the form of alerts as the vehicles are not fully autonomous yet. This communication is achieved through dedicated short-range communications or also known as DSRC. This system acts similarly to the Wi-Fi used for surfing the internet, except with a range of approximately 300 meters. This step is close to bringing a self-driving culture and maybe a testing ground for potential future applications.

Use cases for the above technology (Car-to-car) [13] can be used to warn cars in the range of places that seem unsafe or contaminated, thereby, helping to reduce the spread. Moreover, such an application could be integrated into a smart city like application. A flow could be defined wherein a car finds an area to be unsafe and the car could notify the drone-based systems to disinfect the area, hence creating a safe environment for the visitors. Also, the authorities could be notified so that the area can be sealed till the time when the disinfection process is complete. Additionally, these cars could be connected to a cloud-based server. These servers can contain a repository of the actions done by previous cars updating the drivers to make decisions regarding the safety of their visit.

Going beyond cars and big trucks, there are plenty of ways where vehicles such as forklifts could be automated as well for warehousing. The advantage of implementing this technology would be synchronization and tracking movement. In some cases, wherein a component gets lost, the history of movements by the forklift could be used to track down those lost components.

11.9 Future Opportunities

Logistics and transportation will have a huge makeover in the post-COVID era. With technologies breaking benchmarks time and again, technology will soon make a huge difference with respect to transportation. In the near future, safety will be given the most priority as accidents are still a common occurrence. Although speed cameras are applied in some areas, people tend to avoid being caught by speed cams. An example of improvement could be to reduce the speed of the vehicle at the first catch with a fine which progressively increases every time the cam catches the car; or perhaps dynamic speed limiters which change the speed of the car based on the limits applied on that particular stretch of road. One thing which remains constant

is the march towards autonomous driving. Regardless of the size of the steps taken, completely self-driving cars which do not even need humans at the wheel may as well be in the horizon in the upcoming years. Table 11.2 summarizes the text.

11.10 Manufacturing Industry During COVID-19

Let us begin by understanding how different industries have been affected by the dawn of Coronavirus and on what levels these effects can be seen.

Workforce: As the dawn of Coronavirus hits the people folk, various industries started lockdown procedures and distancing protocols to prevent the spread of virus. Same was and had to be implemented by the people working in the manufacturing industry on a personal level and on an organizational level as well. Distancing protocols included limiting the number of workers reporting for work, change in shift timings and restrictions made on several social areas. Organizations conducted online meetings wherever possible in order to prevent the spread of contagion. This period has also seen a severe cut down in the workforce in the sectors which have gone cold or are in a danger of going so. Alternatively, this has promoted a worldwide trend of work from home culture for sectors where it is possible.

This has been more significant in IT and software solution-based companies where these organizations have been rigorously hiring engineers via various full time and internship programmes. Online training for skills necessary for work from home arrangements have popped up creating a huge demand in education sector as well. Students, researchers and other people looking to explore different domains have benefitted a lot from this. In this time, a significant demand of cloud services can also be seen as there has been a huge increase in the demand for applications which demand such a service.

Demand and Supply: Speaking of demand and supply which has been fairly skewed in favour of wireless and software industries, manufacturing industries have been facing a lot of shortage in the supply chain domain as well.

Let us look at the food and beverages industry for example. They have seen a significant decrease in the demand as people are either unable to procure such consumables because of commercial places like shops have been closed or are preventing to do so. This is, in turn, the reason for the decline in the supply chain. As many food and beverage manufacturing industries have still not been fully automated and while some of the local factories have seen scarce amount of automation, they have been dealing with the problems of significantly reduced workforce and many have shunned their operations.

The situation has been similar for the paint, aerosol and coating manufacturing industry. China is the key exporter of raw materials like pigments required for the manufacturing of paint and similar products. Thus, the advent of Coronavirus was a heavy blow to the industry. Just like a ripple effect, the industries depending on these industries have seen a massive shortage of supply, for example, the automobile

Table 11.2 The summary of the usage of IoT and cloud computing in logistics

Name	Description	Use cases	Benefits
Automated Inventory Management	RFID tags are used to track inventory where the tags emit radio waves	1. RFID antennas and readers used as checkpoints to track the whereabouts of inventory 2. Cloud integration makes it easier for the access of the data passed as well	1. Reduced Human dependency, therefore containing the COVID spread 2. Tracking of inventory can be done from any location in real time 3. Machinery condition/service monitoring
Automated Fleet Management	Global Positioning System helps user to navigate places and it uses satellite-based technology to track users and guide them	1. Inventory Condition can be assessed on the go 2. Increased efficiency in operations and fuel saving 3. Driver behaviour can be assessed	1. Vehicles' location can be tracked easily by staff 2. With the help of it in the case of fleet management, businesses can pre-book for petroleum and this reduces the spread 3. Cold chain management can be integrated to supply vaccine
Self-Driving Vehicles	Uses Radar to understand the surroundings; such technologies are typically used in aviation.	1. Helps users with disability 2. Rule and algorithm-based driving 3. Driver dependency is still present, however, it is expected to reduce in the coming years	1. With lockdowns in place, such a technology will still make deliveries possible. Reduced dependency on drivers 2. A smart city integration could enable better disinfection of the city (car interacting with another machine) 3. Car-to-car integration can help warn one car about the area the car is about to visit
Drone-Based Technology	As the name suggests, drones which are unmanned aerial vehicles are used to do tasks	1. Aerial Deliveries 2. Disinfection Drones 3. Monitor Drone	1. Can reach places which may seem unsafe or impossible for humans 2. Monitor drones can monitor crowded place, potentially saving the lives of monitoring authorities

industry where paints and coating are required to coat the parts and body of the vehicle on various stages. Many factories like these have been shut down.

The personal care and cosmetics industry also experienced a fall in sales due to closing down of offline shops and outlets.

But unlike these industries, the pharma manufacturing industries have seen a boom in demand and supply is finding it difficult to meet the demands. Many manufacturing nations rely on china directly or indirectly for the supply of raw material and these nations were affected heavily during the start of the pandemic. India halted the export of several pharmaceutical ingredients to ensure the availability within the nation.

11.11 IoT-Enabled Automation for Remote Manufacturing and Businesses

The need and ability to control the various aspects of a manufacturing pipeline remotely and wirelessly has been a domain of interest for many researchers and scholars and the need is increasing day by day. With the introduction of new technologies and the lowering prices of general-purpose sensors and equipment, the scope and practicality have increased by a whopping amount.

During the pandemic, the need to control the manufacturing pipeline in a contactless fashion has been the topic of utmost concern for the process handlers [14]. In such times, companies which saw very less automation in their operations have chosen to invest in IoT-based smart systems, lowering the production costs as well as the manual labour hours.

SCADA [15] stands for Supervisory Control and Data Acquisition. It refers to a system of hardware coupled with software to provide the administrators the total overview of the manufacturing process as well as the ability to control it remotely. In these terms, it may seem that SCADA and IoT are pretty much the same concept. As a matter of fact, they are, except a subtle difference. SCADA refers to a human-machine interaction tool whereas IoT refers to a machine-machine interaction-based system.

Another automation trend to look at is the use of PLC, Programmable Logic Controller. Over the past few decades, SCADA and PLC have been running in parallel and with the introduction of IIoT (Industrial Internet of Things), they are believed to be slightly outdated. In a PLC-based system, the controller receives the information from various sensors in the network, analyses and processes the data and triggers output based on pre-programmed parameters.

But, IoT stands superior to both these technologies as it requires minimum constant effort and smart solutions and error handling. Although these systems might be costlier to install, they reduce production cost significantly, reduce the time required for one cycle of pipeline and return greater profits in the long term.

11.12 Monitoring Using IoT

IoT has provided us with the solution to overlook the ongoing processes and even monitor each and every worker in real time. This monitoring can be done manually and remotely by the concerned authorities or by Artificial Intelligence based systems. Information about various aspects such as temperature, humidity, chemical exposure, radiation and air quality index can be retrieved by using their respective sensors on a general level. This monitoring can be taken up a level by monitoring these aspects for every individual in the workplace.

Such an arrangement is achieved by using sensors in the wristbands and helmets provided to the workers (as seen in Fig. 11.6). This way, the production in-charge can be ensured of the safety and distancing guidelines being followed at all times. For example, consider the OnePlus fitness band. It comes with an SPO₂ sensor. Similar bands can be given to each and every employee and the data from these bands like blood oxygen levels, heart rate and temperature along with location can be sent to the cloud. This data can be retrieved by an AI engine which can continuously monitor the data and predict/classify potential positive employees. These employees can be informed via haptic feedback (like vibrations and sounds) and other protocols like SMS and calling. Along with the employee, such information can be sent to higher officials.

Using the feed from cameras and location trackers in the equipment, one can monitor the number of people present in a particular area. If the number goes beyond the specified amount, the internet can be programmed to alert the workers and take suitable actions.

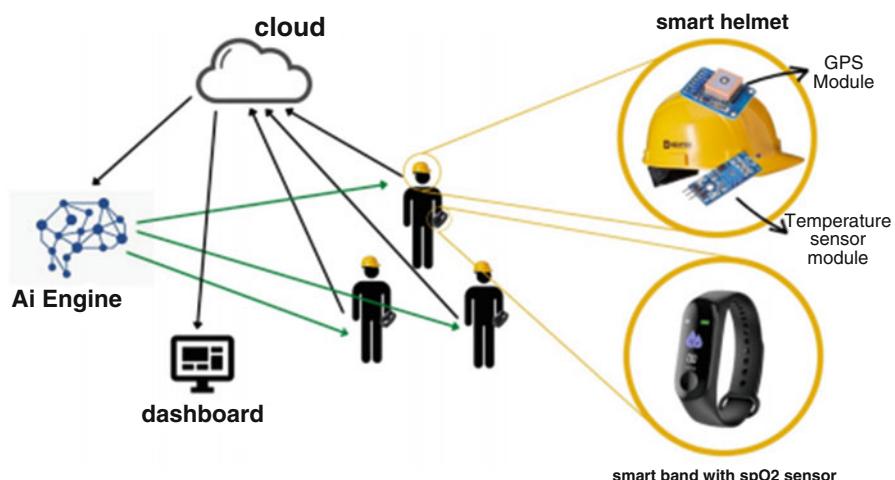


Fig. 11.6 Workforce tracking using IoT

This way, one can ensure that the workplace is healthy and suitable enough for any activities, which is very important in chemical, furnace and radiation-based industries.

This also has an added benefit of contagion tracking [16]. If any personnel found to be COVID positive, his/her activities within the campus can be reviewed and the people who have come in contact can be tracked, warned and quarantined based on his/her location history and that of other employees.

One of the simplest tracking implementations can be seen in the Arogya Setu App by the Government of India. It uses Bluetooth to exchange information between different Bluetooth-enabled devices [17]. The concept is similar to the one discussed earlier; the contacts of an infected person can be traced back till a certain safety threshold. The device owner will be alerted if he/she is near a potentially infected person.

Monitoring can also be done for the industrial equipment [18, 19]. Various aspects depending on the equipment can be monitored in order to predict upcoming faults and failures. The age of the equipment can also be determined by analysing appropriate parameters. Doing this would minimize the downtime by a significant amount as such failures often take time to be resolved. Necessary preparations can be made beforehand for mitigation.

All of the data received from the Internet of Things system employed in the workplace can be uploaded to the cloud in real time to achieve global access to this data. A person sitting in India could monitor the operations in a factory located in Spain, how amazing is that! (Fig. 11.6)

Now let us discuss some of the commonly used and most useful sensors used for industrial monitoring:

Temperature and Humidity Sensors: It is the most commonly used industrial sensor. It is mainly used to predict and prevent condensation on the factory floor. In a furnace-based industry, condensation of humidity on the equipment surfaces and on the factory floor can cause them to become wet and certain parts may get damaged, or certain areas of the factory might become inaccessible depending on the design layout.

Dry Contact Closure Transmitters: These transmitters are used to send an alert or trigger a response whenever a given input changes its state. For example, it is used in a HVAC system for climate control. An HVAC system essentially involves opening and closing of ventilation doors. When the door is left open, the HVAC system runs continuously which might end up being very costly when it is the case for over a 100 loading bays. Thus, the opening and closing of these doors can be automated and monitored by using these dry contact closure transmitters.

Thermocouple Temperature Transmitters: These temperature transmitters are special in a way that they can measure a wide range of temperatures and can handle extreme temperatures easily. They have a very good accuracy and are an essential

part of the ceramic industry. Proper monitoring and timing of certain thermo processes may cut down production hours and hence speed up the manufacturing process.

Current Monitoring Sensor Transmitters: They are mainly used in monitoring current and utility bills. As electricity is an essential part of any manufacturing process, be it powering the equipment or electrochemical processes, it is required in a huge amount. Monitoring can help ensure the best practices for manufacturing optimization along with future failure predictions.

Cycle Counter Transmitters: These are mainly used in scheduling maintenance activities. It keeps track of the number of input switches and schedules outputs based on pre-defined rules and parameters. Its use can be seen in the loading bays where doors are regularly opened and closed. It has a big role in preventing down times and is very effective when used on a large scale.

Pressure Sensor Transmitters: Many industrial procedures have the requirement to read the air pressure or that of different liquids and gases. It is also used to detect leaks or blockages in equipments. Data from these can be sent periodically to the cloud and be monitored by AI or a technician.

Vibration Sensor Transmitters: They are a very common part of predictive maintenance procedures. They can be used to alert for potential failures or issues. Using the data from the normal working and past issues, they can give insights about upcoming issues as the readings go out of specifications. These calculate the RMS vibrations, the minimum and maximum vibration and along different axes X, Y and Z. These are also very useful in areas where direct human monitoring is difficult or impossible.

Air Quality Sensor Transmitters: These sensors are not only used in industrial applications but in offices, societies and even households too! Government guidelines for industries include healthy work environment too and it is among one of the topmost concerns for workers and employers. These sensors are capable of pointing out unnoticeable and unpredictable changes in the environment if monitored carefully.

Activity Sensor Transmitters: This sensor helps to solve the problem of high value-asset relocation. It is attached to high-value assets and it tracks the movement and sends transmissions based on it.

The data from these sensors is uploaded to the cloud from where it is retrieved from a remote software request and the desired data and analytic is displayed on a dashboard (as seen in Fig. 11.7).

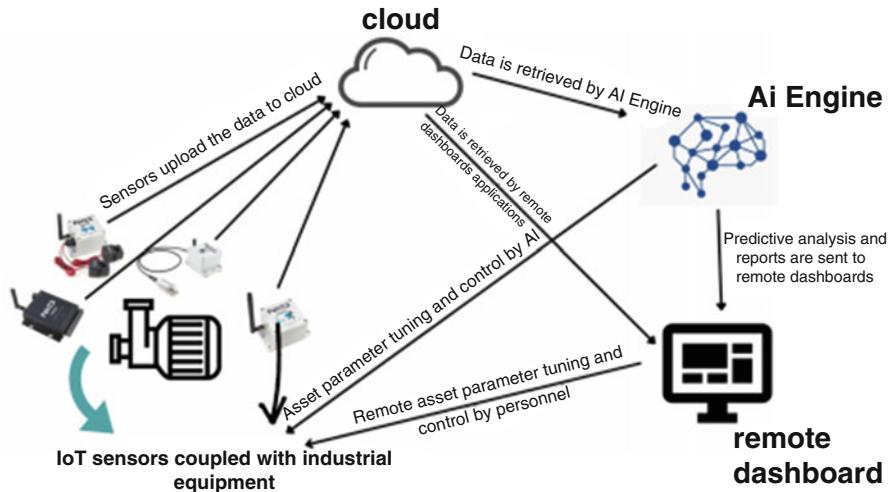


Fig. 11.7 Industrial IoT automation and remote monitoring

11.12.1 Vision-Based Control Systems

Vision-based control systems [20] became quite common during the COVID-19 pandemic. Video feed coupled with infrared sensors are used to enforce social distancing regulations and detecting and tracing potential infected people. These systems have seen wide applications ranging from a wide category of industries from manufacturing to local shops and societies.

Amazon, for example, took body temperatures of employees at the entrance and monitored their activity through cameras and sensors at various areas, feed from which goes to a machine-learning model and determines the distance between them and other safety protocols.

11.12.2 KC 901 Smart Helmet

This smart helmet [21] is constructed with advanced materials and fitted with high precision temperature measurement sensors. It is used in unaware contactless temperature measurement of humans as well as vehicles. It has nine modes namely: single-person temperature measurement mode; large-crowd temperature measurement mode, QR code mode, QR & temperature measurement mode, license plate recognition mode, license plate recognition & temperature measurement mode, thermographic diagnostic imaging mode, night-vision/facility inspection mode, and face recognition mode.

Additionally, Google Location History can be incorporated with the smart helmet to find the places visited by the suspected person after detection.

11.12.3 Post-COVID-19 Opportunities

Because of the pandemic, many industries have chosen to automate their processes and bring manual work out of the equation. This has for sure demanded a good amount of capital investment, thus increasing the production cost for short term. But, in the long term, it proves to be profitable as IoT-enabled smart systems have boosted the production rate and products can be manufactured at a much faster rate. Thus, overall if a company is there for a long run, this might end up being their one of the best investments unless they have very less demand for products in general.

This also means that manufacturers who have invested heavily may need to find some ways to cut costs for some time in order to maintain consistent and future proof business operation. Now, sometimes cutting costs for an already optimized system might end up costing more to the company. So, in such situations, some organizations tend to cut down some of their workforce or transfer them to another domain as the smart systems require minimal manual input. This is one of the reasons why IoT is not very welcomed by the general workforce as it has created an image of them getting replaced by robots. Figure 11.6 represents the workforce tracking using IoT.

As the pandemics open, it is expected for the demands to rise eventually and companies which are among the prime manufacturers need to be prepared beforehand. Thus, this would become a problem for supply chain management. Refer to Table 11.3 for summary of IIoT applications in COVID-19 times.

11.13 Applications of IoT and Cloud Computing in Healthcare

11.13.1 HealthCare

Healthcare has been constantly improving over the past decades. Since the 1900s, the life expectancy has increased more than 2 folds. However, on the advent of COVID-19, the healthcare industry is blasted into pieces and all its weaknesses are exposed brutally. It has resulted in complete mismanagement, utter mishandling and wrecking of the entire healthcare architecture. This is primarily because the health industry had never faced something of this scale and criticality before.

The main factors leading to the havoc of the health industry were as follows:

Table 11.3 IIoT applications summary

Technology	Description	Use case	Benefits
IIoT Asset Optimization & Control	Transmitters placed in appropriate positions on the equipment and connected to the cloud	1. Data from the sensors is retrieved from the cloud to local or remote dashboards for manual monitoring 2. AI can also be deployed to automate monitoring and predicting future faults/maintenance needs	1. Overall 5% increase in equipment effectiveness observed generally 2. Significant increase in ROA (Return On Asset) 3. Predictive analysis to ensure continuous flow of production chain and in time maintenance scheduling
Vision-Based Control Systems	A combination of video feed from cameras coupled with data from infrared sensors placed in strategic locations	High density work areas are equipped with CCTV cameras and infrared sensors are set at shoulder level in numerous places	Number of employees in an area can be determined and respective social distancing rules can be enforced by coupling the system with alarms and alerts
Workforce tracking	Location sensors, Infrared sensors, SPO2 Sensors, etc. fitted in wrist bands and safety helmets, KC901 Helmets and smart bands	Continuous monitoring of temperature and oxygen levels along with their location.	1. Potential positive cases can be identified. 2. In case a person is found infected, contacts can be traced back using his/her location history and the respective personnel can be informed and quarantined.
Remote Employee collaboration	Smart camera powered by speech recognition	Portal through which employees can monitor in factory conditions, hold meetings and manage assets Instruct technicians on procedures	1. Almost all the work being done from home for most employees 2. Minimal requirement of workforce on site. High potential post-COVID scope

1. COVID-19 showed signs of extreme contagiousness, and it could spread mainly through close contact. Hence, curbing the opportunities of coming in contact with people or surfaces is of major consequence especially in a hospital.

2. The failure of the whole healthcare architecture could be blamed on overburdening. Hospitals are often being overcrowded above and beyond capacity.

IoT and cloud computing seem to battle these factors vigorously on the forefront. IoT and cloud computing seem to give the best-combined package to provide comprehensive solutions to the problems thrown by the advent of COVID-19.

The recent advances in IoT and cloud computing not only prove to be gladiators against COVID-19 but also show promise in providing a completely autonomous and thorough healthcare system.

11.13.2 New Innovations

The healthcare industry has been the epicentre in this battle against the Coronavirus; this pandemic has compelled the world to move towards enormous vertical innovation. The healthcare industry has experienced many such novel innovations that proved to be gallantry and richly beneficial to overturn the tables.

The popularization of e-Health has been the polestar that the world has been looking up to. This ideology has given birth to trends such as teleconsultations – remote consultation for the purpose of clinical counsel through electronic means (Fig. 11.8), the mainstreaming of Electronic Health Records (EHR) and much more. Clinical Decision-Making Support (CDMS) systems – tools that provide assistance, recommendations, guidance, advice, directions and alert while dealing with a patient based on the history of their electronic health records are being sought after (Fig. 11.8).

Vital signs monitoring services have started to be spotted in our day to day lives. Tele-homecare-delivering healthcare services and medicine at home and ambulatory e-health healthcare without being admitted to the hospital are being conventionalized. Smart clothing/e-Wear/ e-clothing-clothes are integrated with sensors to

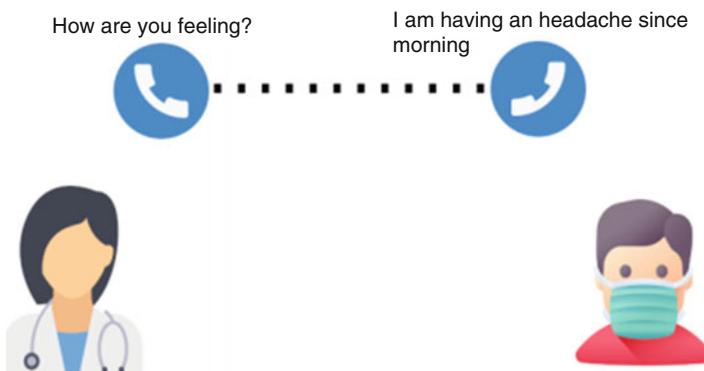


Fig. 11.8 Consultation of a patient through the means of telecommunication

monitor patient's health, are being heavily researched upon and prototypes are being tested in many hospitals. Medical research, education and explorations are being progressed at high speeds using e-learning tools. E-nursing and Robo-nursing are taking over the medical landscape. E-prescribing and e-dissemination are working conjointly by the use of drones and robots. COVID-19 identification and tracking solutions are starting to emerge in the market. Softwares that store, manage and analyse long term laboratory records are being welcomed in the medical research society. The pharmacy industry is being quick to adopt information systems that help them manage their stocks better. At the base root of all these innovations lies the advances in the IoT and cloud computing. The growth in the IoT industry has invited big investors and in turn has promoted research, which has resulted in smaller, cheaper and more compact sensors. Advances in cloud computing have made the data-set readily available and storage of readings from the sensor way easier. All of these, when fused together, inspire innovators to make groundbreaking innovations that benefit the society. The next section discusses a few such innovations which are changing the very perspective of the way in which we look at healthcare.

11.13.3 Applications

11.13.3.1 Remote Healthcare

The healthcare industry can be seen moving from health-care-in-hospital to health-care-from-home. The industry envisions to transform a hospital-centric approach into a complete healthcare experience at the comfort of your home. This concept not only prevents the spread of COVID-19 but also improves customer experience. Many companies have started to sprout and have begun providing extensive health services at home. One such company is CMEDD Health in Bangladesh, which benefitted a population of around two million people during the pandemic by giving healthcare services at the comfort of their homes.

A comprehensive remote monitoring system involves the setup of IoT devices such as a blood pressure monitor, smart glucometer, smart oximeter and smart thermometer [22]. This way, the patient can be monitored without the need for contact and hence is proving to be a vertical which could be explored and exploited extensively. We can further augment it with wearable devices that could track movements, heart rate, calorie exertion, sleep monitoring, periodic cycle monitoring in women, etc. This enables healthcare to become a more personalized and participative sport. It moves the focus from the hospital to the patient. All of this Biomedical Big Data then is being fed to the cloud and autonomous real-time clinical monitoring systems are coming into place [23].

This can, moreover, be coupled with a drone delivery system for medicine where the prescribed medications can be automatically be delivered to the patient's house on a monthly/quarterly/ yearly basis. Many cities have already started to experience medicine supply through drones; this is especially effective in developing countries

Fig. 11.9 Smart clothing with a variety of sensors to monitor patients



where healthcare is not vast spread. Rwanda and Ghana are two African countries that in May 2020 served around 200 medicines remotely through the use of Zipline-enabled drones.

We can also observe the emergence of smart clothing, which implies the weaving of electronics into clothing (Fig. 11.9). This makes the healthcare process invisible and almost effortless; furthermore, it serves as an analysis and inspection step before hospitalization. This conserves a huge tone of effort, money and time for both the hospital as well as the patient. Also, early detection helps recover faster and increases the chances of survival for terminal diseases. COVID-19 noticed an unfolding of a chest-belt which on wearing, provided a lung performance analysis by mapping the topography of the lungs. Leading innovators such Edema Aps are working on a wearable and washable stocking, which detects and tracks the change in the volume/size of the leg for patients suffering from oedema (Fig. 11.9).

All of these devices are essentially reading/detecting a certain measurement in order to estimate an underlying discomfort. This very measurement can be shown in a visually appealing way through mobile applications to the patient. It

is estimated that the number of mobile phones will reach around 8.2 billion by the year 2025; hence, mobile healthcare is a very prominent prospect of the future that can transform healthcare into a DIY – do it yourself activity when coupled with IoT and cloud computing.

There are also prototypes of social robots in the market, which deals with problems related to mental discomfort, anxiety, loneliness, etc. These are completely autonomous and utilize Natural language processing and Machine learning techniques at the base root. They are currently being worked on to provide motivation, courage and motive to the patients who are often disheartened by the discomfort. They could play an important role in remote monitoring/healthcare by providing insights into a patient's mental state and comforting the patient through positive stimulus. Social robots also have the potential to serve as e-nurses and can provide complete assistance in the timely and proper administration of medications. Social robots are being especially useful during the pandemic as governments are imposing strict lockdowns.

All the above when fused together, eliminates the need for hospitals for general healthcare and brings the comfort of the hospitals to your doorstep. It also defeats all the evils of COVID-19 on all fronts; it also seems to cement the loopholes of the current healthcare architecture. Since COVID-19 is followed up by mild symptoms, it can be treated easily by remote healthcare. Hence, by eliminating the need for visiting the hospitals, it not only reduces opportunities for contacting contaminated surfaces and people but also drastically reduces the burden on overcrowded hospitals. This way, hospitals can focus their efforts on treating more severe and critical cases with better management.

With so many sensors and piles of data, security and privacy becomes a major concern. Data on a person's medical records and readings are very welcoming to a lot of ways of exploitation, especially to big agencies looking to maximize their sales. The conventional centralized nature of cloud computing makes the stored data vulnerable. A single break-in results in a reveal all situation; it also gives an obvious location to the hackers for snipping and snooping data packets. However, recent advances in cloud computing have given heat to a decentralized nature. Concepts such as edge computing and fog computing when coupled with health records help make it more secure by pushing it closer to the device. It can further be protected using PKI encryption, cryptographic primitives and cloud-based security to completely resolve the privacy issue. Frameworks such as HealthFog and blockchain methodologies are in talks to protect the privacy of Electronic Health Records (EHR) [24].

The world is moving towards automation, and healthcare is no exception. Long term data stored in the cloud on a person's health, collected by the various IoT devices have a direct implementation in machine learning. Many algorithms and models have been introduced to detect diseases based on anomalies in the reading and measurements. This makes healthcare like an invisible shield, which is watching over your shoulder all the time effortlessly.

11.13.3.2 Robotic Assistantship

Robots are not hijacking the world but they are definitely making us feel their presence in some industries, especially during COVID-19, robots have been at the forefront of the battle [25]. The advances in AI, on the go availability of data sets and very economical pricing of sensors, all connected through IoT has given birth to a new range of robots. Concepts such as telerobots, collaborative robots, autonomous robots and social robots can be seen sprouting at different medical centres. The healthcare industry is taking more of a virtual approach, where concepts such as remote-controlled and remotely-operated telerobots are being given high grounds. These telerobots are equipped with a wide variety of pressure, image, temperature, position, bio, flow, level, motion and radar sensors to carry out remote diagnosis, remote surgeries and treatment with little to no human intervention. Cobots is another burning trend that is being popularly accepted by health organizations across the globe. These collaborative robots (cobots) are being particularly useful in helping patients who are completely isolated to prepare food and to administer the right medication. Doctors around the world have put their lives at risk and are treating patients infected by COVID-19; many doctors have even lost their lives due to coming in contact with contaminated areas [26]. To battle this issue, collaborative robots have been put into place. They alert the medical health expert if they detect a possibility of transmission of the Coronavirus. Another novel implementation of these collaborative robots is to disinfect and sterilize the surfaces that are prone to communicate the Coronavirus. These robots are especially useful in disinfecting areas that are difficult to ingress but are prone to transmit the virus. Robots are being equipped with UV-lights and thermal sensors to effectively destroy the DNA of all viruses in hospital rooms. These extreme disinfection robots (XDBOTs) help speed up the process of vacating the room and getting it ready for the next patient in a very safe and secure manner. Semi-robots are also being heavily deployed in hospitals to ensure the quality of the equipment in their infrastructure. For example, oxygen tanks are being equipped with such semi-robots to ensure that the oxygen tanks never run empty, which is very vital since the Coronavirus causes inflammation in the lungs and hence causes respiratory issues. Robots are extending well beyond the hospital walls; there are instances of robots being deployed on the streets to ensure social distancing. These autonomous robots remind people to practice 6 feet of social distancing in public places which are often hot spots for communication of Coronavirus. All these robots are plugged with the cloud to procure long term data and hence help in identifying vulnerabilities and loopholes in the healthcare architecture. Robo-nursing is another trend on the hot-line. Robots providing medical assistance to patients such that they properly recover from the illness is a very beneficial prospect since proper dosage and timely administration of medicine plays a vital role in recovering from the sickness and developing a strong immunity. Social robots have been the unsung war heroes in this battle against the Coronavirus. As the governments are imposing strict lockdowns and people are being forced into isolation, mental health is being blatantly ignored. People are developing intense anxiety and panic attacks. To battle this problem, social robots

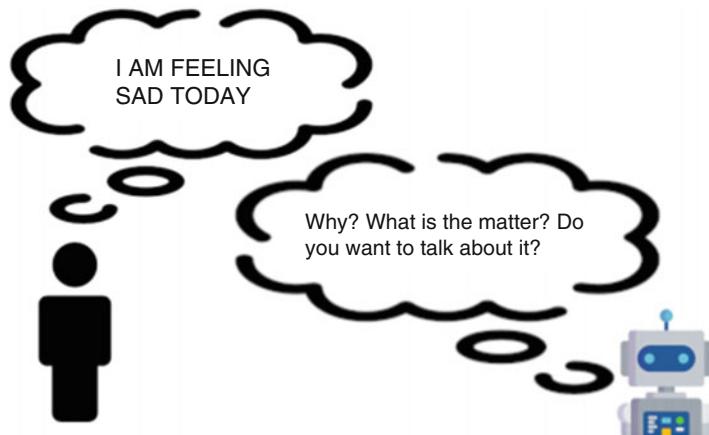


Fig. 11.10 Therapeutic social robots relieving anxiety

have been in high demand. Social robots act as a communication companion and help release stress and anxiety (Fig. 11.10). These social robots aim to cure the patient through continuous and delicate communication [27] (Fig. 11.10).

These communication robots are being augmented with heart rate sensors and infrared temperature to detect stability. The talks with the patients are stored in the cloud and using machine learning algorithms, the social robots ensure that the patient is cured over a period of time.

11.14 Post-COVID-19 Opportunities

COVID-19 when looked upon from an optimistic eye has been a blessing in disguise. It has forced and fasted up innovation in certain sectors and has prepared as well as given us experience for any such pandemic in the future. The lessons that we take away from this pandemic pushes us in the right direction of innovation (Table 11.4).

The post-COVID approach towards innovation should be one with infinite collaborative opportunities, and the general consensus in the innovation society is to jump on multidisciplinary projects. The medical industry should couple with the logistics industry and entertainment industry to produce interdisciplinary and multifaceted projects that reach beyond just treating the disease. We should aim to shift our focus from just treating the disease to a completely seamless experience of effortless healthcare. This integrative and versatile approach towards collaborative innovation opens many doors to novel and interdisciplinary implementations and applications. One such novel application is Neuralink; it is essentially an ultra-tiny and thin neurosurgical robot which has enormous computational powers [28]. It

Table 11.4 Summary of latest innovative applications in healthcare

Technology	Description	Use case	Benefits
Remote Healthcare	Healthcare which is provided without any physical contact	<ol style="list-style-type: none"> 1. Patients can consult through telecommunication. 2. Patients can be monitored through the readings of sensors. 3. Robo-nurses can provide essential medical care. 4. Wearable clothing can be used to monitor and analyse the disease. 	<ol style="list-style-type: none"> 1. Eliminates the chances of spreading through human contact 2. Promotes social distancing 3. Complete and thorough healthcare from the comfort and safety of your home
Robotic Assistantship	Assistance provided by robotic equipments in the healthcare industry	<ol style="list-style-type: none"> 1. Disinfection of contaminated places using collaborative robots 2. Semi-robots to monitor and ensure competency of medical instruments 3. Autonomous robots to alert people about social distancing 4. Social robots to battle anxiety issues during quarantine and isolation 	<ol style="list-style-type: none"> 1. Robots eliminate the possibility of spreading through human contact. 2. Promotes social distancing 3. Robots eliminate the chances of human error and provide more accurate services.

aims to process information from the neurons and claims to fix paralysis. There are claims by the Neuralink Corporation – the company leading the project that the technology is capable of recovering eyesight, hearing and speech. The company also envisions to control computer and mobile devices from your brain and chiefly envisages to build a brain-computer interface. The technology professes to have infinite potential and argues to be very useful in treating a very wide range of neurological disorders. It wishes to redefine the way we interact with objects around us and the world. It also claims to develop a speech system without actual talking using just brain waves and interconnected neural links. There are many such interdisciplinary projects underway across the globe, as the saying goes “If you want something new, you have to stop doing something old”. The world is constantly morphing and the post-COVID era has a very bright future now that we are equipped with the experience of dealing with a global pandemic [29–31].

11.15 Applications of IoT and Cloud Computing in Agriculture

The agriculture industry is an important part of the economy of any country. The COVID-19 pandemic has hit a lot of industries very hard; the agriculture industry is no exception to this. The pandemic has created an uncertainty in supply and demand, and even though agriculture comes under essential services, social distancing and lockdowns have led to labour shortages and personal health issues for farmers across the world. This crisis due to the pandemic has forced farmers to look towards technology; therefore, IoT-based solutions have become popular all across the world.

Some of the existing IoT-based solutions to agriculture are as follows:

11.15.1 Agriculture Monitoring

To increase the chances of successful crop growth, analysing the available environment conditions is a very important step. The agriculture monitoring is done so that counter-measures can be taken against any physical conditions and optimum crop produce can be achieved. The following areas are monitored:

1. *Soil and water monitoring* – The soil pH, moisture, temperature, etc., and the water level, pH, nutrients, conductivity, etc., are measured using IoT-based ground sensors (Fig 11.1) [32, 33]. These sensors measure these parameters and send them to the cloud using wireless networks where this data can be accessed by the end user and compared with the ideal data (Fig. 11.11).
2. *Air and weather monitoring* – The air and weather conditions such as air quality, temperature, humidity, air pressure, and wind speed can often have extremely adverse effects on the crop in non-ideal conditions and, therefore, need to be monitored carefully. For monitoring these factors, we have IoT-based weather stations which give live updates of the air and weather conditions through the cloud and provide a rich dataset that helps in increasing the crop yield to a very large extent.
3. *Illumination monitoring* – Solar conditions such as exposure to photosynthetically active radiation and UV radiations can affect the crop growth to a large extent. These radiations can be measured, studied and modified according to the requirement through radiation sensors, light sensors and actuators.
4. *Crop and Cattle monitoring* – Crop and cattle monitoring is done using specific sensors and cameras put on the crop and the livestock; this helps us determine if the cattle is affecting the crop in any way and also if the crop is being damaged by a bug or disease.
5. *Irrigation control* – Irrigation control works hand to hand with ground and water monitoring. The ground sensors determine the water level in the soil and after

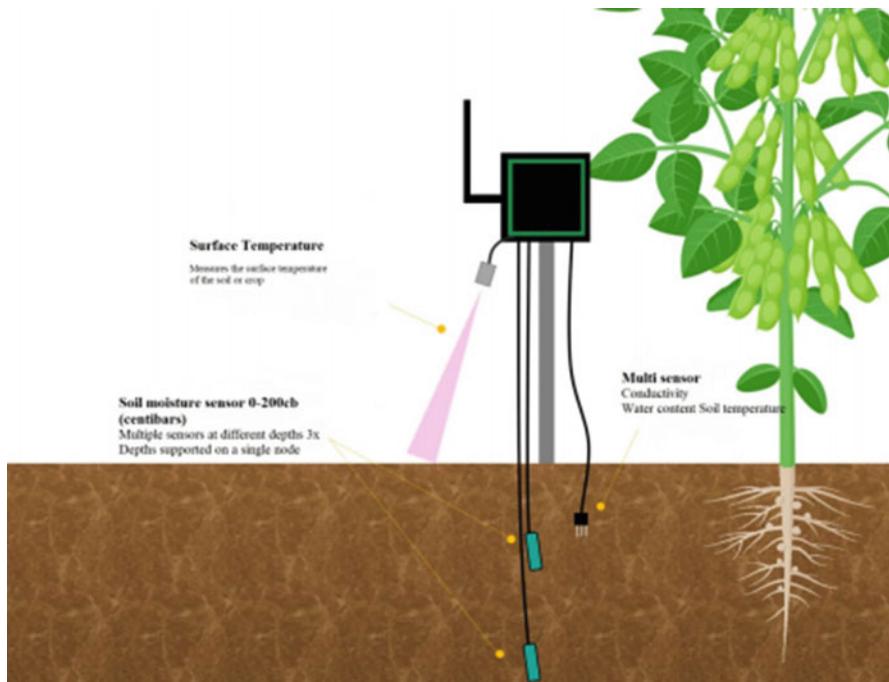


Fig. 11.11 Ground sensor at work

comparing it with required conditions, the automated irrigation system comes to action. The irrigation system also works with the weather stations to avoid irrigating the land immediately before/after rainfall. This prevents water wastage and also helps in ensuring proper water availability for the crop at all times.

11.16 Summary

Therefore, agriculture monitoring provides potential for much higher crop yields and helps in long term cost control due to usage of only required amounts of manure, fertilizers, pesticides and water [34]. Automated systems such as irrigation control systems make it possible to reduce work force and hence help farmers to work with social distancing in effect even in the COVID-19 times and control their costs and make up for losses in the post-COVID era. Moreover, with the recent advancements in edge computing and IoT, the data latency can be shortened even further and the data transfer costs will also reduce significantly.

11.17 Conclusion

This chapter presented a brief insight into the applications of IoT and cloud computing in education, entertainment, transportation, manufacturing, healthcare and agriculture. Moreover, it also discussed the future opportunities for such emerging technologies in these sectors. Nevertheless, there have been enormous developments in the field of cloud computing that had played an essential role in ensuring a proper learning environment and resources. The entertainment industry has also seen changes similar to the ones in the education industry. The transportation and logistics industry got heavily impacted during the COVID-19 pandemic. With technologies breaking benchmarks time and again, technology will soon make a huge difference with respect to transportation. The need and ability to control various aspects of a manufacturing pipeline remotely and wirelessly has been a domain of interest for many researchers and scholars and the need is increasing day by day. The healthcare industry envisions transforming a hospital-centric approach into a complete healthcare experience at the comfort of your home using the latest technologies including IoT and cloud computing. This crisis due to the pandemic has forced farmers to look towards technology; therefore, IoT-based agricultural solutions have become popular all across the world. Finally, this work also discussed the future opportunities provided by these technologies in managing the everyday living all through such epidemic catastrophes.

References

1. L. Sydow, *Video Conferencing Apps Surge from Coronavirus Impact* App Annie Blog. App Annie (3030), <https://www.appannie.com/en/insights/market-data/video-conferencing-apps-surge-coronavirus/>
2. W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges. *IEEE Internet Things J* **3**(5), 637–646 (2016). <https://doi.org/10.1109/JIOT.2016.2579198>
3. *How Video Conferencing Works – VoIP Supply.* (n.d.). Voip Supply. Retrieved January 2, 2021, from <https://www.voipsupply.com/how-video-conferencingworks#:~:text=Once%20digitally%20compressed%2C%20the%20video,video%20images%20and%20audio%20 sounds>
4. K. Tange, M. De Donno, X. Fafoutis, N. Dragoni, A systematic survey of industrial internet of things security: Requirements and fog computing opportunities. *IEEE Commun. Surv. Tutor.* **22**(4), 2489–2520 (2020)
5. A. Brem, E. Viardot, P.A. Nylund, Implications of the coronavirus (COVID-19) outbreak for innovation: Which technologies will improve our lives? *Technol. Forecast. Soc. Chang.* **120451** (2020)
6. R. Bunker, A. Elsherbeni, A modular integrated RFID system for inventory control applications. *Electronics* **6**(1), 9 (2017)
7. Unit-R&D, T. T., *Smart Fleet Management System Using IoT, Computer Vision, Cloud Computing and Machine Learning Technologies.* (2019)
8. P. Killeen, B. Ding, I. Kiringa, T. Yeap, IoT-based predictive maintenance for fleet management. *Procedia Comput. Sci.* **151**, 607–613 (2019)

9. Y.P. Tsang, K.L. Choy, C.H. Wu, G.T.S. Ho, H.Y. Lam, P.S. Koo, An IoT-based cargo monitoring system for enhancing operational effectiveness under a cold chain environment. *Int. J. Eng. Bus. Manag.* **9**, 1847979017749063 (2017)
10. J. Scott, C. Scott, Drone delivery models for healthcare, in *Proceedings of the 50th Hawaii International Conference on System Sciences*, (2017)
11. M.N. Mohammed, H. Syamsudin, N.A. Hazairin, M. Haki, S. Al-Zubaidi, A.K. Sairah, Y. Eddy, Toward a novel design for spray disinfection system to combat coronavirus (Covid-19) using IoT based drone technology. *Rev. Argent. Clín. Psicol.* **29**(5), 240 (2020)
12. P. Bucsky, Autonomous vehicles and freight traffic: Towards better efficiency of road, rail or urban logistics? *Urban Dev. Issues* **58**(1), 41–52 (2018)
13. K. Dhoble, A. Deshmukh, V. Patil, *Car to Car Communication Using IOT*. (2019)
14. S. Kathiravan Srinivasan, S. Kumaran, *Quality Assessment of Friction Welding using Image Super resolution via Deep Convolutional Neural Networks*, *Materials Today: Proceedings*, vol 22, Part 4 (2020), pp. 2266–2273. <https://doi.org/10.1016/j.matpr.2020.03.347>
15. H. Geng, *SCADA Fundamentals and Applications in the IoT*. (2017)
16. S. Fahrni, C. Jansen, T. Kasah, B. Körber, N. Mohr, *Coronavirus: Industrial IoT in Challenging Times* (McKinsey & Company, 2020)
17. M. Cunche, A. Boutet, C. Castelluccia, C. Lauradoux, D. Le Métayer, V. Roca, *On Using Bluetooth-Low-Energy for Contact Tracing* (Doctoral Dissertation, Inria Grenoble Rhône-Alpes; INSA de Lyon, 2020)
18. H.S. Raju, S. Shenoy, Real-time remote monitoring and operation of industrial devices using IoT and cloud, in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, (IEEE, 2016), pp. 324–329
19. K. Momeni, M. Martinsuo, Remote monitoring in industrial services: Need-to-have instead of nice-to-have. *J. Bus. Ind. Mark* (2018)
20. R.B. Abdessalem, S. Nejati, L.C. Briand, T. Stifter, Testing vision-based control systems using learnable evolutionary algorithms, in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, (IEEE, 2018), pp. 1016–1026
21. M. Nasajpour, S. Pouriyeh, R.M. Parizi, M. Dorodchi, M. Valero, H.R. Arabnia, Internet of Things for current COVID-19 and future pandemics: An exploratory study. *J. Healthc. Inform. Res.*, 1–40 (2020)
22. K. Srinivasan, N. Mahendran, D.R. Vincent, C.-Y. Chang, S. Syed-Abdul, Realizing an integrated multistage support vector machine model for augmented recognition of unipolar depression. *Electronics* **9**, 647 (2020). <https://doi.org/10.3390/electronics9040647>
23. G. Sheares, Internet of things-enabled smart devices, biomedical big data, and real-time clinical monitoring in COVID-19 patient health prediction. *Am. J. Med. Res.* **7**(2), 64–70 (2020)
24. S. Tanwar, *Fog Computing for Healthcare 4.0 Environments*. (2020)
25. M. Tavakoli, J. Carriere, A. Torabi, Robotics, smart wearable technologies, and autonomous intelligent systems for healthcare during the COVID-19 pandemic: An analysis of the state of the art and future vision. *Adv. Intell. Syst.* **2000071** (2020)
26. K.P. Iyengar, P. Ish, G.K. Upadhyaya, N. Malhotra, R. Vaishya, V.K. Jain, COVID-19 and mortality in doctors. *Diabetes Metab. Syndr. Clin. Res. Rev.* **14**(6), 1743–1746 (2020)
27. L. Aymerich-Franch, I. Ferrer, The implementation of social robots during the COVID-19 pandemic. *arXiv preprint arXiv:2007.03941*. (2020)
28. E. Musk, Neuralink, An integrated brain-machine interface platform with thousands of channels. *J. Med. Internet Res.* **21**(10), e16194 (2019)
29. A. Saglietto, F. D'Ascenzo, G.B. Zocca, G.M. De Ferrari, COVID-19 in Europe: The Italian lesson. *Lancet* **395**(10230), 1110–1111 (2020)
30. J. Yoosefi Lebni, J. Abbas, F. Moradi, M.R. Salahshoor, F. Chaboksavar, S.F. Irandoost, et al., How the COVID-19 pandemic effected economic, social, political, and cultural factors: A lesson from Iran. *Int. J. Soc. Psychiatry*, 0020764020939984 (2020)
31. Y. Siron, A. Wibowo, B.S. Narmaditya, Factors affecting the adoption of e-learning in Indonesia: Lesson from Covid-19. *J. Technol. Sci. Educ.* **10**(2), 282–295 (2020)

32. A. Vangala, A.K. Das, N. Kumar, M. Alazab, Smart secure sensing for IoT-based agriculture: Blockchain perspective. *IEEE Sens. J.*. <https://doi.org/10.1109/JSEN.2020.3012294>
33. M.S. Mekala, P. Viswanathan, A Survey: Smart agriculture IoT with cloud computing, in *2017 International Conference on Microelectronic Devices, Circuits and Systems (ICMDCS), Vellore*, (2017), pp. 1–7. <https://doi.org/10.1109/ICMDCS.2017.8211551>
34. D. Elavarasan, D.R. Vincent, V. Sharma, A.Y. Zomaya, K. Srinivasan, Forecasting yield by integrating agrarian factors and machine learning models: A survey. *Comput. Electron. Agric.* **155**, 257–282 (2018). <https://doi.org/10.1016/j.compag.2018.10.024>

Chapter 12

Analytics of IoT-Based System for Monitoring Students' Progress in Educational Environment



Moses Kazeem Abiodun , Joseph Bamidele Awotunde , Emmanuel Abidemi Adeniyi , Roseline Oluwaseun Ogundokun , and Sanjay Misra

Contents

12.1	Introduction	323
12.2	Review of Related Work	325
12.2.1	Big Data	327
12.2.2	The Internet of Things	328
12.2.3	Cloud Computing	330
12.2.4	RFID Sensing Technology	331
12.2.5	Applications of RFID Technology	332
12.2.6	Internet of Things and Education	333
12.3	Methodology	334
12.4	Use Case of the Proposed Framework	337
12.5	Conclusion and Future Research Directions	339
	References	339

12.1 Introduction

With the use of a broad variety of advanced technology, from implanted devices and networking technologies to Internet protocols, data analytics, etc., the Internet of Things (IoT) aims to transform conventional objects into intelligent objects [1].

M. K. Abiodun

Department of Computer Science, Landmark University, Omu Aran, Nigeria
e-mail: moses.abiodun@lmu.edu.ng

J. B. Awotunde

Department of Computer Science, University of Ilorin, Ilorin, Nigeria

Department of Electrical and Information Engineering, Covenant University, Ota, Nigeria
e-mail: awotunde.jb@unilorin.edu.ng

E. A. Adeniyi

Department of Physical Sciences, Precious Cornerstone University, Ibadan, Nigeria
e-mail: adeniyi.emanuel@lmu.edu.ng

The potential impression of IoT in driving the financial development of IoT-based services is expected to offer many business opportunities. Based on a study report on the worldwide financial effect of IoT [2], the yearly financial influence of IoT in 2025 will be in the range of \$2:7 to \$6:2 trillion. Around 41% of this sector accounts for healthcare, followed by industry and energy, with 33%percent and 7% of IoT share, respectively. Further industries, such as transport, agriculture, urban infrastructure, defense, and retail, have an IoT market of approximately 15% overall. These estimates show the enormous and sharp progress of IoT products and produced data within a short period. In the years ahead, the estimated market share is expected to improve drastically.

Kevin Ashton proposes the Internet of Things (IoT) for the first time in 1999. Many IoT methods have been created by various researchers such as the Internet of Medical Things (IoMT), the Internet of People, the Internet of All, the Internet of Marks, the Internet of Info, and the Internet of Services [3, 4]. IoT is a worldwide network that combines materials and materials depending on the Internet to connect with the environment [5, 6]. The ubiquitous presence of several objects or things is the essential theory of the IoT [7]. The IoT articles are, therefore, much wider, meaning various physical objects such as individuals, cargo and products determinable by various types of equipment such as sensors, RFID tags, actuators, cell phones, smart cards, and GPS devices [8]. Objects may respond to one another and cooperate with others around to accomplish the goals involved [9].

IoT ties or links object-to-object, object-to-machines, infrastructure-to-machines, person-to-objects, etc. Alternatively, we can say that an IoT can be called a system that links simple/regular corporal substances to recognizable addresses that offer smart facilities and generate enormous amounts of data. At any point in time, these links multiply and form a network of networks [10]. IoT is expected to spread further in the future. In the educational sector, IoT has a wide reach. Learning is about the most important macro accomplishments in human life and it has become a requirement to introduce IoT in the educational field. If there's IoT in developing countries or developed countries, it will help ease human lives. IoT could be described as an intelligent gadget that communicates and interacts with further technologies, structure, substances, and the setting that generates information that is processed for useful action. Smart machines have a user interface that is very effective and efficient, thereby minimizing human interaction. Creativity is the IoT of today's era. It is extensively employed in several fields, including medical care,

R. O. Ogundokun

Department of Multimedia Engineering, Kaunas University of Technology, Kaunas, Lithuania
e-mail: rosogu@ktu.lt

S. Misra

Department of Computer Science and Communication, Østfold University College, Halden,
Norway
e-mail: sanjay.misra:hiof.no

ecological monitoring, protection, agriculture, education, and energy efficiency [11].

The implementation of the IoT in education would result to a full overhaul of educational establishments. The teachers, campus, leadership, education procedure, erudition procedure, etc., are altered. IoT similarly enhances the learning experience by offering different student learning features. Nowadays, conventional learning methods have been discarded by many universities and other educational institutions. Technologies such as tablets and laptops have now replaced them. It allows learners to absorb at the rate at which they are relaxed. It similarly lets scholars at both home and college have a similar learning experience. For schools, IoT not only assists learners but also teachers. With the application of different techniques, the teaching process can be made even simpler. It allows teachers to recognize the speed of learner learning and to concentrate more on those who need additional learning assistance. For the educational industry, this is also a major benefit [10, 11].

IoT Technology will affect every area of our life. Higher learning organizations and universities will work together by correcting and leading the creation of potential IoT innovations, business models, standards, and IoT leadership. University computer science and engineering educators, for example, lead IoT labs for the production of IoT technologies [12]. The way to track the student's time in and time out in different locations around the campus is to define radiofrequency. With the aid of the RFID system and the Internet of Things, the tracking of learners would be easy to access; the system provides reliable information to school administrators and parents through the quality and reliability of the data. IoT defines the universe in which just about everything can be intelligently related and interconnected. In other words, the entire world is becoming one massive knowledge system across the globe with the Internet of Things by using devices that record student time-in and time-out via passive technology for radio frequency identification.

This chapter focuses on enhancing the performance of students in their research by analyzing how students work in a campus setting to assess their readiness to obtain successful results in their tests. The arrangement of the remaining chapter follows this pattern: Sect. 12.2 evaluates the applicable literature, Sect. 12.3 delivers the method established to achieve the set objective, Sect. 12.4 reviews the execution plan, and Sect. 12.5 addresses the conclusion and future work proposed.

12.2 Review of Related Work

In educational settings, the Internet has deeply grounded itself, and now, another technology IoT is gaining ground very fast. It is projected that IoT will make more substantial reforms in the learning industry. Numerous IoT appliances apply to the educational system; interactive displays and digital highlighters are included [13, 14]. Also, by supplying objects with attached sensors as students get close to learning about objects, the smart learner's room allows learners to learn more attractively with real-time data. Authors in [15] address the Internet of Everything (IoE) is a system of data, objects, individuals, and procedures. Not only does the

word IoE concentrate more on IoT, which is the structure of corporal substances, but similarly on proficiencies for millions of related individuals and objects, for instance, respect-awareness and power dispensation.

Santas, [16] IoT helps schools and organizations shape the developing world's curriculum, tasks, assessments, etc., and developments that contradict other theories. IoT adjusts the mode it imparts, visually for greater comprehension, and can support physically disabled individuals encompassed, according to individual to individual. The computerized student monitoring scheme aids to trail distinct accomplishments of persons and also strengthens their co-scholastic abilities. It drives the even tide of data after installation, but formerly it has to be set up correctly and preserved with conversant safety and virus guards. Protecting scholars with the utilization of IoT employing manifold monitoring and geo-fencing signals similarly helps them to monitor their current position and ward status.

According to Bulla et al. [17], for various levels of education, the use of cloud computing can be implemented in many ways. In the instance of primary and secondary colleges, the execution of a wise state server that is popular across the state might have a sub server for each district that would synchronize with the main server. This will inspire broods to acquire, and through animation, the topics can be educated. In the situation of pre-university electronic classes, students can access the knowledge learned at school at any time and ask the teacher for any questions online. Cloud learning aims to redefine the conventional way of teaching and increases students' effectiveness.

According to authors in [18], parents are drawn to the high probable of IoT in education to select this approach to reach the maximum possibility of their school. Digital content is effortlessly shared and stimulated by any student to do so. With fast data flow rapidity, there would be no postponement in obtaining the information and reducing drawbacks. Investigation similarly demonstrated that the student's performance is also relatively higher and more effective in a physical setting provided by IoT. The authors used various parameters to monitored and helped in creating more efficient performance by the student, such as temperature and humidity.

According to Aldowah et al. [19], the processing of data would be streamlined and accessible 24/7, eliminating the physical effort to continuously collect data. When scanned by the RFID reader, each object shows the data to the students. Students can learn and easily exchange content with IoT anywhere. Printing mistakes, typos, and man-made mistakes can be cut down as technology can do this effectively. There will now be simpler personal professional guidance or studying from remote locations. Apt learning for all relies on their skill and speed, which would be documented by expertise automatically. In education, the IoT will perform a very imperative function in observing at how it is intensifying at the moment. Scholars are involved in technology and can explore their true potential if they are surrounded by technology all around them.

According to Abdel-Basset et al. [11], the IoT allows sophisticated attendance monitoring systems to reduce the attendance of proxies at the institute. Innovative safety actions will be strengthened inside the campus to limit unlawful classroom bunking. Adaptive e-books offer better learning for humans. Owing to the vast

capacity of data flowing, numerous security and privacy issues have to be tackled, thus putting scholars' and lecturers' safety at risk. Like money, the implementation of all these gadgets necessitates an enormous asset that presents additional contest. Suitable management of these gadgets entails a great deal of precision and proficiency. Sensors track buses and, if necessary, interrupt school schedules at the same time. Global usual education is accessible unrestricted and regularly around the world.

12.2.1 Big Data

Data is life now, and through various social media tools and on the Internet, a lot of data is generated online. Communications around the world have increased massively, producing vast quantities of data. Recent research revealed that Google accepts over four million requests every minute, user e-mails exceed the cap of 200 million e-mails, users of YouTube upload 72 hours of videos, two million pieces of information are uploaded through Facebook, and 227,000 tweets are generated on Twitter every minute, while users of WhatsApp post 347,222 images and users of Instagram post 216,000 new photos. The present age is the Big Data age, so the data is rising more than ever before on a massive scale. According to Computer World, 70 to 80% of the information is in a formless state in most establishments [20].

Big data technologies have a significant impact on a variety of industries since the last century and continue to do so today, with the potential to become ubiquitous. Unlike most other sectors, the banking industry relied heavily on structured data collection to run its operations. Nonetheless, using big data applications, the information found in different semi-structured and unstructured data sources can be analyzed. Big Data is a paradigm that is changing in almost every sector. The information, which comes from social media, accounts for 80% of global data and 90% of big data. As detailed in the annual digital universe report of the International Data Corporations (IDC) [21], the data is being generated too quickly and this year's estimate will impact the range of 44 zettabytes that would be ten times greater than what we now have [22]. In the quickly emerging areas of Big Data Analytics, big data can be analyzed and accumulated in the financial markets, allowing for the interpretation, rationalization, and utilization of big data for various purposes.

The planet is unified and this helps individuals with the right opportunity and sensors to start producing a large amount of data every minute. According to the study [23], each human being produces data on a large scale in the range of over 6 MB per minute across different sources, a total of 1.7 million bytes of data. The Observance, Authority, and Oversight Committee also reported that for the majority of business organizations, the data size doubles after every one and a half to two years, although 90% of the data has been generated in the last few years [24].

There are a significant number of domains belonging to hundreds of companies that have come to the point that their technology infrastructure needs to be updated to accommodate the amount of data that has now reached Petabytes' scale to stay

in the market. IT companies recommend that clients use the power of a mixture of NoSQL database technologies such as HBase, Paxata, and Hadoop to manage Big Data. However, in terms of the level of complexity and standardization, big data architectures pose an obstacle to implementing the analytical framework [25]. In the field of Big Data Analytics, the lack of qualified manpower (data scientists, engineers, architects) and technological architecture is the next level obstacle. To understand the ultimate potential of Big Data, it is important to concentrate on the requirements: (1) combining several hands and device architectures of expertise and (2) increasing the data manipulation and usability capacity of Big Data technologies.

Big Data is more than increasing sizes of datasets, but also the uncertainty that such growth creates concerning various aspects of data handling, there are a large number of domains belonging to thousands of organizations [26]. Based on the ‘3V’ classic model proposed by Gartner, Big Data can be described as Big Data is voluminous, frequently produced and in different types, requiring cost-effective and specialized tools to work on the data for improved assimilation and good judgment. A combination of unstructured and organized data is Big Data. To turn big data into meaningful information, a set of creative and exceptional methods is needed for its handling, storage, and investigation.

12.2.2 The Internet of Things

Special sensors that link the whole world with the Internet can be represented as the Internet of Things. According to experts, the word IoT denotes the next step in the digital community. The IoT aims to foster the competitiveness of academic and commercial institutions. Industry 4.0, which is based on IoT [27, 28], relates to digital transformation, the latest trend of automation, and data sharing by various suppliers. Many of our cities are now being turned into smart cities by the use of IoT in our generation. Likewise, in an attempt to turn our campus into smart campuses, many of our learning environments have begun to use IoT. Many higher learning institutions are now offering elective IoT-related courses in Computer and Computer Engineering for our undergraduate students [29]. IoT has urban transformation, waste management facilities, parking areas, efficient traffic management, lighting systems, environmental surveillance, efficient irrigation systems, smart home encroachment revealing systems, banks, etc. Besides, social networks have evolved rapidly in order to efficiently enter a vast range of sectors [30–33], and many innovations have been incorporated, with IoT being no exception.

To turn the actual real world into the virtual one, IoT social networks were consolidated with the Internet. The outcomes of the study suggest using the social Internet of Things (SIoT), which has the potential to effectively and efficiently support new IoT applications and networking services [34]. The main objective of the SIoT is to integrate devices into the everyday lifecycle of users using the advantages offered by social networks such as user-friendliness and interconnectivity. To inspire users to link their devices to others to support the SIoT project

and guarantee its sustainability, new enticing services need to be provided [35]. IoT advancements will increase because of the constant improvements in cloud computing, networking, sensors, nano-electronics, intelligent objects, and big data. One of the components of the Internet is the IoT, which helps people interact with each other, connecting people and objects and connecting objects with other things.

Ideally, engineers and technicians would avoid traveling to and visiting warehouses, isolated locations, or busy places because they fear being infected with the virus. Nevertheless, their physical appearance on site is needed because they cannot get away with manually operating the equipment available. By the use of special sensors mounted in machinery and enhanced knowledge, real-world overlays, and remote expert feedback, IoT may help solve their repair problems, conduct machine operation remotely, and AI may also anticipate when machines need repair (predictive maintenance) or when they might be faced with a challenge. In this way, physical visits will be significantly decreased, helping to safeguard workers' health and welfare, thus enhancing efficiencies at the level of service of the facilities. When supermarkets continue imposing limits on sales of goods per customer, future alternatives have also been produced by IoT and AI. Smart shelves, smart fridges, video analytics, and an end-to-end integrated supply chain will help retailers deal with planning challenges and even reduce the extreme behavior of customers due to hysteria.

A couple of years ago, a major US retail chain put IoT trackers in its trolleys to prevent daily theft. Perhaps it is time to start applying this to the store shelves of important homes, sanitizers, and everything else that is already over-selling, in an attempt to properly control supplies and prevent hoarding behavior. Smart watches and fitness trackers will be readily available in the not-too-distant future and people with chronic conditions will be able to record temperature, asthma, and heartbeat without the use of intrusive instruments. For this fact, patients will be able to opt to transmit real-time and past data to public or private hospitals anytime they feel unwell so that medical health IT departments will interact with the wearables and mobile devices of patients. In this way, the treatment of coronavirus or other diseases may be prepared even more effectively and with minimal resources expended optimally. Smart connected medical devices, such as smart home ventilators, together with video and wearables, may support patients monitoring at home, give updates to those in distress, or even alert when paramedics are required to come and move them to the hospital.

Ventilators are important in treating people that have become contaminated. The health services have not been prepared to deal with this magnitude of a pandemic, and a resulting shortage is now widespread in only the most developed hospitals. An IoT-3D printing may be a lifesaver in the face of Coronavirus-induced supply shortages, with an IoT-3D printer that offers critical medical equipment, for example, replacement valves. This is what an Italian company called Isinnova does, taking a 3D printer to a Milan hospital and manufacturing incomplete valves to be shipped to a hospital in Brescia, Northern Italy. Touch screens have been the chosen user interface (UI) until recently: among them are tablets, computers, and even doors. Nonetheless, the fact that Coronavirus is more quickly transmitted from a

contaminated surface than by air has made direct contact sound risky. Many UIs that do not need any physical contact are also usable. Voice has already triumphed over tactile user interfaces, especially through smart speakers and digital helpers. Despite people confining themselves inside the building, there will be increased interest in smart home apps and the voice apps of smart speakers. Another functionality that gains ground in smartphones is beyond speech, biometrics, and their use for eye/face identification such as the use of facial picture identification to open phones or make payments. The largest penetration is in China but the opportunity for the remaining part of the biosphere is enormous. Wearables, for instance, smart payment watches and other use cases (enabled by voice or close contact) can allow us to escape physical surface contact.

12.2.3 Cloud Computing

Cloud computing will play an essential part in absorbing healthcare transformation expenses, optimizing assets, and bringing the new age of technology to life. Emerging policies are targeted at obtaining data at any time, anywhere that can be achieved by transferring health data to the cloud. This contemporary distribution model will make healthcare more productive and operational and lower the price of innovation expenditures [36], but it also presents some obstacles due to issues regarding the security of sensitive health information and compliance with a specific criterion such as HIPAA. Healthcare providers, taking into account these security and privacy concerns, can unquestionably reap the benefits of cloud computing technologies and provide substantial benefits such as assisting to expand the eminence of service for patients and reducing healthcare spending [37].

Cloud computing's critical features are (1) self-service on-demand, (2) wide grid access, (3) resource sharing with other occupants, (4) swift elasticity, and (5) calculated facilities. In complex resources, clouds offer benefits such as processing energy or storage abilities, universal access to resources from anywhere at any time, and high resource versatility and scalability. In several business fields, these advantages have been the purpose of the growing acceptance of cloud computing. This principle has also evidently been adopted in the area of education sectors in recent years. A continuous growth in the number of research papers and books in the of healthcare systems in the literature show that a lot of work has been done in the area of cloud computing in medical application. This is gaining more popularity in recent time in education sectors and in other fields.

The cloud with IoT-based online application works efficiently over the ordinary cloud-based applications. The medical, bank, and military are examples of sectors that have benefitted greatly from emerging models. The cloud-based IoT-based system has helped in providing education sectors with effective services in the areas of monitoring and access to records remotely. IoT-centric application has helped to collect necessary data like frequent changes in health status in real-time. Also, the

captured data from IoT-based devices will be analyzed using artificial intelligence for effective diagnosis of disease at the correct time before the illness reaches advanced stages.

The capacity to share data between different systems would be one of the main advantages of cloud computing. This capacity is something that IT urgently needs for healthcare. Cloud computing, for example, can enable educational stakeholders and professionals to share data such as students record, lecturer references, libraries, insurance data, and research reports stored via various information systems. In the radiological market, where many organizations have switched to the cloud to minimize their computing costs and promote the sharing of pictures, this is already happening [37]. Cloud computing has provided campus monitoring, surveillance, library book tracking to educational institutions. It has also aided insurance providers and other education companies the ability to agree to cooperate and exchange education data to provide improved service quality and minimize costs. Looking at the developments in the industry, it seems that once all the obstacles it presents are resolved, cloud-based schemes will eventually turn out to be the standard in the education sectors.

12.2.4 *RFID Sensing Technology*

Recently, there has been an upsurge in the use of RFID technology for monitoring and recognizing applications. RFID can trace, identify, and control data using easily deployable tags which allow the software to be employed beyond its normal usage in industrial management. It is used in various new fields; RFID tags are useful as standalone sensors or used as a module that incorporates all the circuitry between the antenna and at least one mixing stage of a receiver for other customized commercial product. Making use of RFID technology's sensing abilities will enable the IoT systems to collect and integrate information seamlessly from real-world objects. We have two categories of RFID tag detecting systems:

- (i) Analog RFID sensing: It performs analog handling of the signals physically with the collaboration amongst the reader and the tag without any committed sensing electronics. Without it requiring additional electronics, the reader can get considerably much more information about the object, more than just recognition. The sensing of the RFID analog depends on the knowledge that the hosting object is affected by the workings of an RFID tag, so by probing the variance of the signals backscattered from the tags, recovering data sensing is possible to simply recover sensing data. Delicate covering materials or aggregate mechanisms displaced above the antenna are also used to achieve a more reliable device response [35].
- (ii) Optical RFID sensing: To construct a combined sensor module, the RFID labels are combined with automated mechanisms such as sensory materials, analog-to-digital converters, and microcontrollers [38]. Embedded computers are

Table 12.1 Comparison between RFID and CFRID

Analog RFID	Computation RFID
Have a lower cost	Comes at a higher cost
There is a tradeoff between sensing and communication requirements.	No tradeoff in CFRID
Less accurate	More accurate
Affected by environmental interactions	Immune to environmental interaction
Less reliable result	More reliable result

allowed in computational RFID (CRFID) to use only sifted Radio Frequency (RF) energy to run programs. Computation is the secret to justly pervasive computing applications for the IoT-based and they are battery-free and support “invisible” sensing. The computational radio frequency identification label is used as a communication interface to convey data. To power the circuit, passive radio frequency identification sensors absorb RF energy from RF radiation, accomplish the sensing task, and store the data to be retrieved by RFID readers on the RFID chip. Both types of RFID sensing provide IoT systems with a range of sensing capabilities [39]. Table 12.1 shows a comparison between RFID and CRFID.

12.2.5 Applications of RFID Technology

The use of radio frequency identification high tech for detecting our physical environment has evolved greatly in the last era. New applications that use both analog and digital sensing continue to emerge and will revolutionize IoT applications in a broad range of fields. RFID sensors are making an impact in the medical business in ways that profit both patients and medical professionals [40–42]. Sensors that use wearable and wireless technologies allow reliable and continuous monitoring for medical devices. Wireless sensors do not restrict the individual’s movements, thus enhancing their worth of life. Likewise, medical professionals can assemble data more quickly, simplifying communication in various departments, and delivering emergency treatment. Future agriculture will need, through technology and sensors, effective solutions [43, 44].

Finally, in the field of neuro-engineering, RFID sensing is emerging. Implantable biomedical applications are one of RFID’s hottest technology fields, thanks to the tremendous promise that RFID’s wireless power and data transmission technologies offer to this industry. The RFID physical layer is thus suitable for applications such as neural recording in which embedded sensors need no external energy source other than the RF field.

Table 12.2 shows the main areas where RFID sensing can be used.

Table 12.2 RFID areas of application

Areas of applications	Examples
Health	Patient monitoringPatient mobility and emergency dispatching
Retail	Security, process automation, real-time stock control
Agriculture	Smart irrigation and tractor, key parameter monitoring and sensors innovation
Industry	Real-time monitoring, smart maintenance, and environment-aware objects
Transportation	Smart road networks, environment connected vehicle, people-oriented transportation
Neuro-Engineering	Biomedical device communication, backscatter communication, and low -power communication

12.2.6 Internet of Things and Education

The Internet of Things (IoT) is still very new in higher education systems despite recent implementation prospects (less than 10 years) [23]. IoT typically creates a very good environment that offers an open structure for all online and physical objects. This skill enables the development of different applications to be made possible based on it [24]. Online education is the most accessible IoT-based application as stated in the field of education. Creatively, the most tangible success is online education, which provides a robust, flexible learning platform online that allows instructors and students to work together as educational objects in real-time. This partnership will significantly boost the learning experience, effectiveness, and assessment online that is not delayed.

We look at students' participation in education as a result of their concentration, enthusiasm, curiosity, comprehension, optimism, and passion that students display when they are taught. This can also be applied to their level of excitement for studying and improving their training [37]. Through the various communication tools provided in the IoT-based learning surroundings, students can respond quickly and frequently to questions and/or any kind of inquiries, questionnaires and feedback.

The collaborative tools of IoT-based would contribute to adolescents' positive growth. It will make it possible to create multi-sized young student groups that express their experiences and viewpoints and conclude with social and emotional participation just as behavioral and physical participation. Educators should also describe various teaching methods and tactics available to record the input of the student in real-time.

The variety of the IoT-based encourages ingenuity as part of the smart activity by being able to perceive the world in different and varied means to link physical and virtual objects in such a way that it is possible to change the current territory and turn the current territory into a new one [45]. The ability to interact with a large number of objects allows expectable and unpredictable items or ideas to be generated. The

Internet of Things will promote inspiration in many subjects, incorporating a broad spectrum of educational tools that can turn instructors and learners into creators.

Self-learning, self-didacticism, or self-study is the act of learning in a self-motivated situation about a subject or subjects without formal education, which is regarded as a counterpart to teaching to enable learners to do additional autonomous work [46–48]. The automation features of the IoT-based allow human objects (students, teachers, or hardware components) to interact independently to perform the educational operational job.

Continuous self-directed learning can be integrated into all contexts of human life through IoT, as IoT items are all around us. They build the whole world of interconnectedness, connecting with all, accessible to all and almost anywhere and every time [36, 49]. A learner can use the various educational assets whenever learning, doing homework, learning areas of interest, transferring and accepting any material or feedback from the educators. Educators can transfer and accept any learner materials as well as administrative tasks, online learner assessment, study, and connection from anywhere connected with anything to the abundant research platforms around the world via IoT. For example, a student can connect to the cloud system of the institution that studies from home or anywhere in the city through a registered mobile device, search possible laboratory works (predesigned by a teacher), digitally connect to the laboratory, and experiment with the exercises and obtain the response online.

In an expanded environment in real-time, using IoT in an institution of higher learning will assist learners and teachers share their expertise and experience. The rate of exchange and right of use of generated data by other institutions everywhere and the number of objects available in the learning procedure are growing considerably. This trait contributes to a noteworthy improvement in the output of both learners and educators in terms of learning effectiveness. The basic elements of cognitive learning are primarily considered in certain variables such as self-learning, collaboration, and speed of learning [43]. In the education world, the use of IoT is comparable to ICT-based teaching systems. It can make the teaching and learning procedure more inspiring and educative [30]. As stated, in terms of these parameters, IoT can have direct and indirect greater learning effects on the performance of students.

12.3 Methodology

Figure 12.1 shows the architecture of the proposed framework, which will facilitate the collection of data through RFID IoT-based devices from various students on campus.

The conceptual framework in Fig. 12.1 starts with the students who are expected to register their information on the administrator system that will allow them to be captured by the RFID devices located all around the campus. The administrator

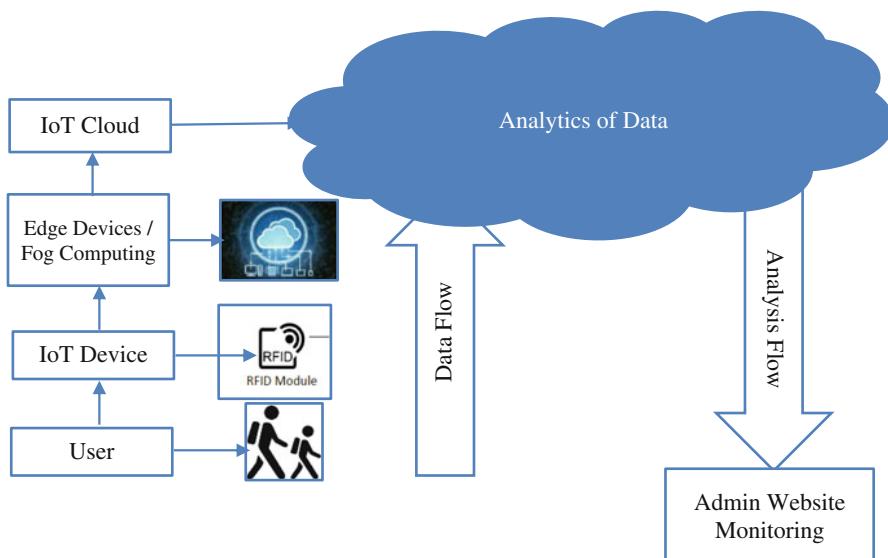


Fig. 12.1 Conceptual framework of IoT Big Data Analytics using RFID

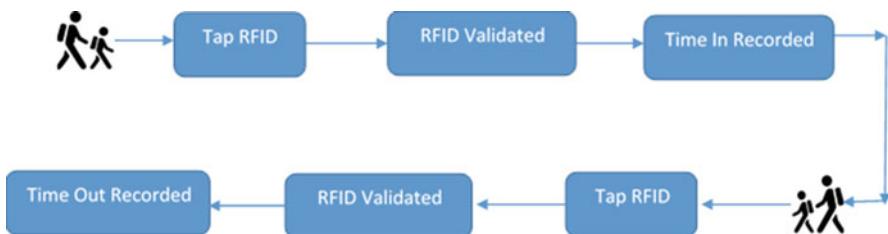


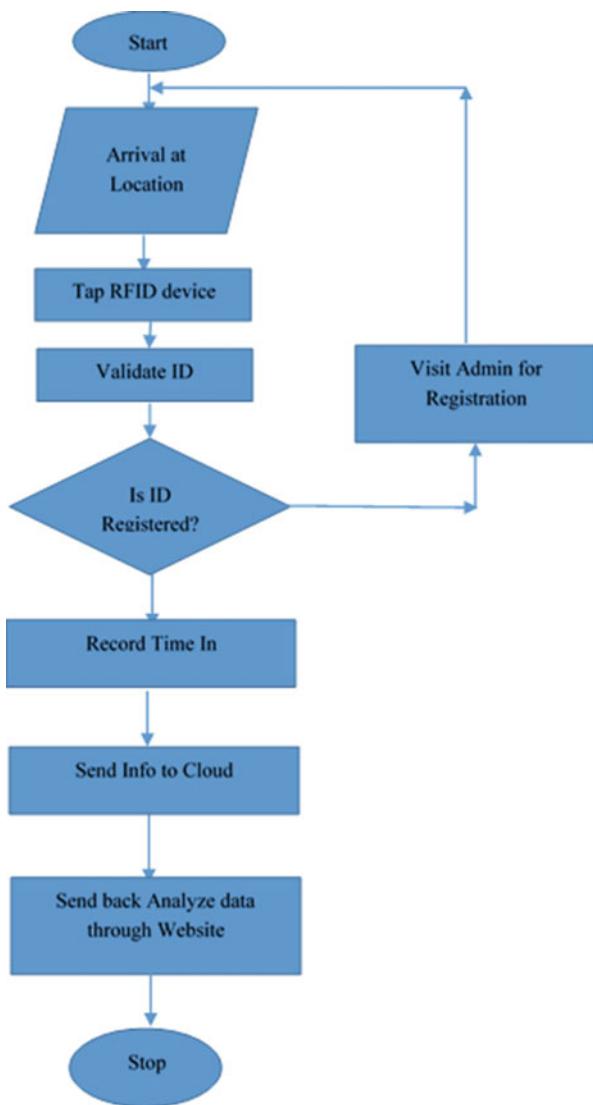
Fig. 12.2 Diagram showing the flow of operation

manages the users' accounts and reports generations. The RFID modules handle the validity of the identification cards.

Figure 12.2 shows what happens when a student taps the RFID device, the device will process and authenticate the tapped ID for confirmation whether the ID is registered or not on the RFID computer. The administrator is charged with registering all the students on the system. Each location on campus is fitted with RFID devices that students can tap in with their ID Card. After the RFID device authenticates the students' ID, the system will record the time-in of the students in the location environment; if the validation is successful, then it shows the information of the ID owner. When the student enters and taps the RFID, the time-in will be recorded to the database. To record the timeout, the student will again tap the RFID; then the system will record the time-out.

The information of the location, time in, and time out of each location are sent to the parent through the website created for this purpose. Parents can log in with

Fig. 12.3 Flowcharts of recording time-in and time-out



their username and password to monitor the activities of their wards on campus with some analysis to show the readiness of their ward toward fulfilling their dreams of success (Fig. 12.3).

The framework starts with a collection of data from students living on campus. The user is referring to students who have the liberty to visit any place on campus. The essence of the study is to monitor the various locations being visited by each student to know how they use their time and from this, useful information can be gathered to be analyzed in a way to provide feedback to parents and the school

administrator. It is structured in a way that no student will be allowed to access any location without tapping the RFID device. The second part of the conceptual framework is the use of the RFID device to track students. The RFID is located at the different centers on campus, especially, the following centers that have been picked to be used: the library, various classrooms for lectures, hostels, cafeteria, relaxation centers, and lecturer's offices.

The RFID devices will be placed at all these locations and the RFID will record the time in and time out of students in various locations. These data are stored in a database system on the cloud to be analyzed and a report is generated. The third component of the layer is the computation of the edge devices/fog. To allow computing directly at the edge of the network, fog computing is suggested, which can deliver faster and enhance the computing capacity. The edge end is close to the origins and users of data, and the cloud servers consist of the core end. In this way, to minimize data processing latencies and the amount of data sent to the cloud, the edge computing model moves computing to the edge of the IoT network. A cloud-based backend can support requests for processing that are less time-sensitive, or for which the source devices themselves do not need results.

In the cloud, each student's time-in and time-out are aggregated and analyzed with various reports available through the website. The website is managed by an administrator, and it is accessible to the parents and staff. The website information will be accessed after successful user login. Each student's record is associated with their parent so that other parents do not have access to other student's records. The reports generate daily, weekly, monthly, and yearly reports of the students' time-in and time-out. The performance of the students will be weighed along with areas where they have visited for a month or in a term to determine the effect that it has on the performance of the students.

12.4 Use Case of the Proposed Framework

Since the framework proposed deals with a large amount of knowledge (termed as big data), the system, therefore, requires an IoT-based cloud database that can effectively process a vast volume of university campus data in real-time. The proposed framework used the Hadoop ecosystem, containing Master nodes, and different data nodes under the Master node to satisfy these requirements. There is HDFS file storage in the Hadoop ecosystem, which divides the data into an equal number of chunks and stores them in different data nodes. Later, parallel processing is done using the MapReduce method on these chunks. Hadoop was essentially used for batch processing, but we used Spark over the Hadoop scheme to use it for real-time analytics. At the Hadoop ecosystem, all the processing calculations results from generation are completed. Finally, based on the results produced by the Hadoop ecosystem, decision making is carried out. Artificial intelligence, analytical thinking, soft computing, and decision models are used for the decision-making technique. For many campus activations, the produced results are used.

The GPS can be used to continuously read the geographic coordinates of the student's current location as well as what they are doing per-time, thus read subsequently by a microcontroller. The capture data in real-time will be uploaded to the cloud database using the MapReduce hosted to be the server through HTTP post request using Wi-Fi adapter placed on the various location where students visited on campus. The student identification number and the timestamp were used to uniquely identify each student within a location in real-time. It is presumed that Wi-Fi networking is given by a Wi-Fi adapter put on the track at each campus site.

As they approach the entrance to any university campus, the RFID reader reads each student's unique ID embedded in the passive tag to locate their movement around the campus where the hardware ensemble is located. This is read by the microcontroller that invokes a PHP script on the server that uses a toggle to adjust the student's status to either inside the university premises, finds the corresponding parent's device id, and pushes a notification message with the position and time from the status database to the respective parent's smartphones via the website application, using the Firebase communication system. On the Android operating system, the front-end mobile framework for the proposed model works. Three groups of users, management, parents and staff can handle the application. Using the Firebase Registration Program, a unique device id is created upon signing up and stored in the database with login credentials. To plot the student's position and path, a Google Map API is incorporated into the application UI.

Using the unique id supplied for each student, a parent can scan the database for their child and choose to follow them. This would allow parents to view their child's location information within the university in real-time such as the library, lecture room, and other locations of the visit, plots on the map, contact the staff or administration, and be notified whenever their child is within the school premises. The Firebase Cloud Messaging service used to push updates using a PHP script cURL request that is triggered any time an RFID tag is read by the hardware ensemble. The Firebase token corresponding to the respective parent is retrieved from the database and pushed to their computer for notification. The application allows student information and parent information to be added or changed by administrators, view information of all locations visited by students on campus, list of onboard students and their data, and contact parents and the guardians. A toggle variable marks students as they are read on or with their RFID tags. Parents are allowed to contact the administrators or emergency services, as well as to access their children's information stored in the database.

The security issues are, last but not least, the most important issue for the people of the smart city. By continuous monitoring of the video of the entire campus, protection is achieved via the proposed framework. However, reviewing all the videos and identifying any mishap with someone via the device in real-time is very difficult. In order to overcome this constraint, the system proposes new scenarios that increase the security of the university-wide system. The device places several emergency buttons with surveillance cameras and microphones at various locations of the premises. If some mishap occurs with anyone like burglary and stolen vehicle, users can just press the emergency button at any nearby location and send the alert

to the nearest university campus security posts. Thus, via surveillance cameras, security forces can begin tracking the surrounding locations and can easily identify the imposter.

12.5 Conclusion and Future Research Directions

The smart campus has the capacity of producing and communicating huge quantities of data produced from IoT-based devices and sensors. The sizes and quantities of networking equipment are more than ever and the volume of data grows even more than ever before. For example, IoT-based devices and sensors can be used to capture and collect campus features and check metrics for student monitoring. The emergence of cloud computing has touched almost all the human life domains; especially the smart educational system has greatly benefitted from the paradigm. Cloud computing has brought a technological revolution needed by IoT-based services like high processing, storage capabilities, heterogeneity, and computation resources among others. Radio Frequency Identification is the way of tracking the student's time-in and time-out. With the help of the system RFID and the Internet of Things, the students' monitoring is easy to access with accurate and reliable data sent to parents and administrators. The IoT provides information through a website, which is available all the time. Parents' and students' communicative process will now be a brilliant fashion of communication. When the RFID students monitoring system is used, the time-in and time-out of the students will be easily sent to parents through the website since the system takes the information of students' time-in and time-out accurately. The system also helps to give a parent peace of mind to know what their ward is doing at school and if there are areas where they need to come in, it can be done easily to curb the excesses of some of the students. In the future, the full implementation will be carried out to examine the effectiveness of the proposed framework and other areas of schools will also be looked into for monitoring, like the kitchen and record processing on campus.

References

1. E.A. Adeniyi, R.O. Ogundokun, J.B. Awotunde, IoMT-based wearable body sensors network healthcare monitoring system. *Stud. Comput. Intell.* **933**, 103–121 (2021)
2. J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, A. Marrs, Disruptive technologies: Advances that will transform life, business, and the global economy. McKinsey Global Institute San Francisco, CA, vol. 180 (2013)
3. G. Fortino, A. Guerrieri, F. Bellifemine, R. Giannantonio, Platform-independent development of collaborative wireless body sensor network applications: SPINE2, in *2009 IEEE International Conference on Systems, Man and Cybernetics*, (IEEE, 2009), pp. 3144–3150
4. C. Ma, W. Li, R. Gravina, G. Fortino, Posture detection based on smart cushion for wheelchair users. *Sensors* **17**(4), 719 (2017)

5. C. Savaglio, P. Pace, G. Aloisio, A. Liotta, G. Fortino, Lightweight reinforcement learning for energy efficient communications in wireless sensor networks. *IEEE Access* **7**, 29355–29364 (2019)
6. A. Gumaei, M.M. Hassan, M.R. Hassan, A. Alelaiwi, G. Fortino, A hybrid feature extraction method with regularized extreme learning machine for brain tumor classification. *IEEE Access* **7**, 36266–36273 (2019)
7. N. Bari et al., Internet of things as a methodological concept, in *2013 Fourth International Conference on Computing for Geospatial Research and Application*, (2013, July), pp. 48–55
8. L. Atzori et al., The Internet of Things: A survey. *Comput. Netw.* **54**(15), 2787–2805 (2010)
9. M. Cata, Smart University, a new concept in the Internet of Things, in *2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER)*, (2015), pp. 195–197
10. M.B. Abbasy, E.V. Quesada, Predictable influence of IoT (Internet of Things) in the higher education. *Int. J. Inf. Educ. Technol.* **7**(12), 914–920 (2017)
11. M. Abdel-Basset, G. Manogaran, M. Mohamed, E. Rushdy, Internet of things in smart education environment: Supportive framework in the decision-making process. *Concurr. Comput. Pract. Exp.* **31**(10), e4515 (2019)
12. H. Aldowah et al., Internet of Things in higher education: A study on future learning. *J. Phys. Conf. Ser.* **892**(1), 012017 (2017)
13. B. D. Ralhan, How IoT is Transforming the Education Sector, Inc42 (2017). Available: <https://inc42.com/resources/io-transforming-education/>
14. J. He et al., Integrating Internet of Things (IoT) into STEM undergraduate education: Case study of a modern technology infused courseware for embedded system course, in *2016 IEEE Frontiers in Education Conference (FIE)*, (2016), pp. 1–9
15. M. Selinger, et al., Education and the Internet of Everything: How ubiquitous connectedness can help transform pedagogy, *Cisco Consulting Services and Cisco EMEA Education Team* (2013)
16. M.T. Saritaş, The emergent technological and theoretical paradigms in education: The interrelations of Cloud Computing (CC), connectivism and Internet of Things (IoT). *Acta Polytechnica Hungarica* **12**(6), 161–179 (2015)
17. C. Bulla, B. Hunshal, S. Mehta, Adoption of cloud computing in education system: A survey. *Int. J. Eng. Sci.* **6**(6), 63–75 (2016)
18. S.P. Mathews, D.R. Gondkar, Solution integration approach using IoT in education system. *Int. J. Comput. Trends Technol.* **45**(1), 45–49 (2017)
19. H. Aldowah, S.U. Rehman, S. Ghazal, I.N. Umar, Internet of Things in higher education: a study on future learning, in *Journal of Physics: Conference Series*, vol. 892, No. 1, (IOP Publishing, 2017, September), p. 012017
20. A. Holzinger, C. Stocker, B. Ofner, G. Prohaska, A. Brabenetz, R. Hofmann-Wellenhof, Combining HCI, natural language processing, and knowledge discovery-potential of IBM content analytics as an assistive technology in the biomedical field, in *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*, (Springer, Berlin, Heidelberg, 2013), pp. 13–24
21. S. Landset, T.M. Khoshgoftaar, A.N. Richter, T. Hasanin, A survey of open source tools for machine learning with big data in the Hadoop ecosystem. *J. Big Data* **2**(1), 24 (2015)
22. L. Garg, E. Chukwu, N. Nasser, C. Chakraborty, G. Garg, Anonymity preserving IoT-based COVID-19 and other infectious disease contact tracing model. *IEEE Access* **8**, 159402–159414 (2020)
23. E. Damiani, C. Ardagna, P. Ceravolo, N. Scarabottolo, Toward model-based big data-as-a-service: The toreador approach, in *European Conference on Advances in Databases and Information Systems*, (Springer, Cham, 2017), pp. 3–9
24. D. Austin, eDiscovery Trends: CGOCs Information Lifecycle Governance Leader Reference Guide. 2012-05-03)[2013-06-10] (2012). <http://www.ediscoverydaily.com/2012/05/ediscovery-trends-cgocs-information-lifecycle-governance-leader-reference-guide.html>

25. J.B. Awotunde, A.E. Adeniyi, K.M. Abiodun, G.J. Ajamu, O.E. Matiluko, Application of cloud and IoT technologies in battling the COVID-19 pandemic, in *Machine Learning for Critical Internet of Medical Things*, (Springer, Cham, 2022), pp. 1–29
26. C.A. Ardagna, P. Ceravolo, E. Damiani, Big data analytics as-a-service: Issues and challenges, in *2016 IEEE International Conference on Big Data (Big Data)*, (IEEE, 2016), pp. 3638–3644
27. L. Atzori, A. Iera, G. Morabito, The Internet of Things: a survey. *Comput. Netw.* **54**(15), 2787–2805 (2010). <https://doi.org/10.1016/j.comnet.2010.05.010>
28. D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges. *Ad Hoc Netw.* **10**(7), 1497–1516 (2012). <https://doi.org/10.1016/j.adhoc.2012.02.016>
29. M.M. Raikar, P. Desai, J.G. Naragund, Active learning explored in open elective course: Internet of Things (IoT), in *Proceedings of IEEE 8th International Conference on Technology for Education, T4E 2016*, (2017). <https://doi.org/10.1109/T4E.2016.012>
30. N. Al-Qaysi, M. Al-Emran, Code-switching usage in socialmedia: a case study from Oman. *Int. J. Inform. Technol. Lang. Stud.* **1**(1), 25–38 (2017)
31. M.K. Abiodun, J.B. Awotunde, R.O. Ogundokun, S. Misra, E.A. Adeniyi, M.O. Arowolo, V. Jaglan, Cloud and Big Data: A mutual benefit for organization development. *J. Phys. Conf. Ser.* **1767**(1), 012020 (2021)
32. Z. Wang, W. Donghui, R. Gravina, G. Fortino, Y. Jiang, K. Tang, Kernel fusion based extreme learning machine for cross-location activity recognition. *Inf. Fusion* **37**, 1–9 (2017)
33. K. Moses, A.J.B. Abiodun, O.R. Oluwaseun, S. Misra, A.A. Emmanuel, Applicability of MMRR load balancing algorithm in cloud computing. *Int. J. Comput. Math. Comput. Syst. Theory* **6**(1), 7–20 (2021)
34. V. Beltran, A.M. Ortiz, D. Hussein, N. Crespi, A semantic service creation platform for Social IoT, in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, (IEEE, 2014), pp. 283–286. <https://doi.org/10.1109/WF-IoT.2014.6803173>
35. M. Al-Emran, S.I. Malik, M.N. Al-Kabi, A survey of Internet of Things (IoT) in education: Opportunities and challenges, in *Toward Social Internet of Things (SIoT): Enabling Technologies, Architectures and Applications*, (Springer, Cham, 2020), pp. 197–209
36. J.B. Awotunde, A.K. Bhoi, P. Barsocchi, Hybrid cloud/Fog environment for healthcare: an exploratory study, opportunities, challenges, and future prospects, in *Hybrid Artificial Intelligence and IoT in Healthcare*, (2021), pp. 1–20
37. O. Ali, A. Shrestha, J. Soar, S.F. Wamba, Cloud computing-enabled healthcare opportunities, issues, and applications: A systematic review. *Int. J. Inf. Manag.* **43**, 146–158 (2018)
38. M.O. Adebiyi, E.E. Adeka, F.O. Oladeji, R.O. Ogundokun, M.O. Arowolo, A.A. Adebiyi, Evaluation of load balancing algorithms on overlapping wireless accesspoints. *Indonesian J. Electr. Eng. Comput. Sci.* **21**(2), 895–902 (2020)
39. EVAL01-Spectre-RM. Available online: <http://www.farsens.com/en/products/eval01-spectre-rm/>. Accessed on 10 Feb 2020
40. S.N.R. Kantareddy, I. Mathews, R. Bhattacharyya, I.M. Peters, T. Buonassisi, S.E. Sarma, Long range battery-less PV-powered RFID tag sensors. *IEEE Internet Things J.* **6**, 6989–6996 (2019)
41. A. Rashee, E. Iranmanesh, W. Li, X. Fen, A.S. Andrenk, K. Wan, Experimental study of human body effect on temperature sensor integrated RFID tag, in *Proceedings of the IEEE International Conference on RFID Technology & Application, Warsaw, Poland, 20–22 September 2017*, pp. 243–247
42. S. Yang, M. Crisp, R.V. Penty, I.H. White, RFID enabled health monitoring system for aircraft landing gear. *IEEE J. Radio Freq. Identif.* **2**, 159–169 (2018)
43. P. Fuhrer, D. Guinard, *Building a Smart Hospital Using RFID Technologies: Use Cases and Implementation* (Department of Informatics, University of Fribourg, Fribourg, Switzerland, 2006)
44. A. Bothe, J. Bauer, N. Aschenbruck, RFID-assisted continuous user authentication for IoT-based smart farming, in *Proceedings of the IEEE International Conference on RFID Technology and Applications, Pisa, Italy, 25–27 September 2019*, pp. 505–510

45. IEEE/EMBS Conference on Neural Engineering (NER), San Francisco, CA, USA, 20–23 March 2019; pp. 791–794. Sensors **2020**, *20*, 2495 16 of 18
46. P. Pande, A.R. Padwalkar, Internet of things — A future of internet: A survey. Int. J. Adv. Res. Comput. Sci. Manag. Stud **2**(2), 354–361 (2014)
47. M. Csikszentmihalyi, Creativity: Flow and the psychology of discovery and invention Harpercollins, ISBN:9780061844034. J. S. Armstrong, “Natural learning in higher education,” *Encyclopedia of the Sciences of Learning* (2011)
48. M. Abdulraheem, J.B. Awotunde, R.G. Jimoh, I.D. Oladipo, An efficient lightweight cryptographic algorithm for IoT security. Commun. Comput. Inf. Sci **1350**, 444–456 (2021)
49. G.A. Aarons, M. Hurlburt, S.M. Horwitz, Advancing a conceptual model of evidence-based practice implementation in public service sectors. Adm. Policy Ment. Health Ment. Health Serv. Res. **38**(1), 4–23 (2011)

Chapter 13

Power System Protection on Smart Grid Monitoring Faults in the Distribution Network via IoT



Owolabi Peter Bango, Sanjay Misra , Oluranti Jonathan, and Ravin Ahuja

Contents

13.1	Introduction	344
13.2	Backgrounds and Literature Survey	345
13.2.1	Internet of Things (IoT): Communication Technology	346
13.2.2	Internet of Things (IoT): Machine: Machine Communication Technology	346
13.2.3	Protection Relay	346
13.2.4	Communication of IoT with Smart Grid	347
13.2.5	Review of the Past Research Work	348
13.3	Proposed Model	349
13.3.1	Components of the Single Line Diagram of Test Distribution Network	350
13.3.2	Design of the GSM Module to Be Integrated into the Tested Distribution Network	350
13.3.3	Simulation of Faults	353
13.4	Result and Discussion	359
13.5	Recommendation	360
13.6	Conclusion	360
	References	361

O. P. Bango () · O. Jonathan

Center of ICT/ICE Research, Covenant University, Ota, Ogun, Nigeria

S. Misra

Department of Computer Science and Communication, Østfold University College, Halden,
Norway

e-mail: sanjay.misra@hf.uio.no

R. Ahuja

Shri Vishwakarma Skill University, Gurgaon, India

e-mail: jonathan.oluranti@covenantuniversity.edu.ng

13.1 Introduction

Reliability and security of electrical energy infrastructure play vital roles today unlike before, as these have become a part of daily human life and activities. The main concern of this research is the protection of the power system against interruptions arising from different types of faults. In this regard, the main component involved in the protection of the power system which is the circuit breaker is considered. The circuit breaker is responsible for opening and closing of the power system in the event of a fault or anomaly in the system in order to protect the equipment.

Traditionally, electrical protection scheme tends to change and become “smart” [1], which improves the fault detection of the system and protect the system remotely in order to monitor every event notification [2]. To become a smart network, home equipment is embedded with the computer system that connects all the household devices together and the Internet. Smart city in an urban setting comprises of computers, sensor chips, cameras and some sensitive equipment that operate in the background. Protection of the power system against faults is, therefore, of paramount importance, although this results in increase in the number of protective devices connected to the Smart world [3].

Traditionally, protection system for residential, office and industrial buildings is designed to use circuit breakers or fuses that isolate the faulty area when fault or overload occurs, which provide limited warning and protection to the system. The world is moving towards automation, therefore, the systems such as power systems, control systems and mechanical systems usually form a combination of one or more systems. Microcontroller-based system can be used to run these complex systems.

Systems that do not have any problem are usually more effective, efficient and flexible [4, 5]. The microcontroller is powered with AC/DC rectifier circuit. The rectifier’s function is to convert the 230 V AC to 5 V DC as the required voltage level for the microcontroller. The emergence of IoT as a technology has brought about what is now known as smart grid. In the smart grid, each device of the power system is considered as an object and identified with a unique IP address which can be used to control the system through the Internet. This effectively results in a connection of various electrical devices that create intelligent and self-sustaining ecosystem [6]. The number of current IoT devices used in the world today is estimated to be about 8.4 billion which is greater than the present human population. According to [7], IoT devices will increase to about 50 billion by the year 2020.

The aim of this chapter is to re-shape the traditional power system by transcending it into the smart grid and micro-grids, which will highly require the flexibility of the distribution management of the power systems and incorporating by the re-design of the protection scheme to match. The National Institute of Standards and Technology (NIST) defined smart grid as the integration of traditional power system with the information and communication technologies [8]. One advantage of smart grid is that it maximizes the energy demand in the distribution system unlike the traditional power system, which increases the efficiency of the system and reduces

the losses [9]. Traditionally, circuit breakers are widely used today for power-system protection; this device is designed to protect the power system against faults in the system. In 2016, Eaton developed a new generation of circuit breakers called Arc Fault Detection Device (AFDD). This circuit breaker protects the power system against electric arcs, leakage current, over current and overvoltage [10].

Papers were reviewed before the modelling of the project. The past work on a particular project [11] was reviewed and modifications carried out. The project was modelled with tested distribution network using Neplan software for load flow analysis and fault analysis of the distribution network. It was concluded that protection of equipments from any types of faults that may occur in the distribution system is crucial to provide safety for the equipments and personnel, and also to provide easy operation for the utility operator monitoring via Internet-of-Things (IoT).

Section 13.1 of this chapter discussed the background and literature review of the project, the purpose of protective device in the distribution network and its roles in the Internet of Things and also the review of the past research work. Section 13.2 is the proposed model of the project and modelling of the tested distribution network before and after the microcontroller is connected to the distribution network using Neplan software for the load flow and short circuit analysis of the project.

13.2 Backgrounds and Literature Survey

This section provides the review of the critical relevant literature to the study. Electrical Substation Communications Standard (IEC-61850) [12] has emerged due to inability of traditional protection systems to provide real-time monitoring and communication features for fast operation of IoT-based integration in smart environments. IEC-61850 is suitable for smart grid power system monitoring and control. However, the IEC-61850 is limited by the fact that they are dedicated for power system utility providers and specialists, and their design does not provide monitoring or notifications service to the end users.

According to [12], Electrical Safety (ELSA) power-system protection device describes the goal of the integrated smart environments like smart homes and cities as protecting the electrical equipment from faults and anomalies that may likely occur in the system. This system supports real-time notifications and protection against electrical failures as well as remote monitoring and control of the coupling or decoupling features of the system. IoT-based intelligent lighting system for public poles lighting integrates circuit breaker which isolates the system in case of a short circuit. ELSA device provides additional protection scheme against a wide range of electrical failure.

13.2.1 Internet of Things (IoT): Communication Technology

Reliable and accurate information of the devices located remotely are required to enable the state of the system to be established. Communications technology that are now available are Wi-Fi, Bluetooth, Zig Bee and GSM/3G/4G; in addition, some new network options are now available such as Thread used for home automation applications as another choice and also whitespace TV technologies are widely used in many countries for IoT-based applications. Common communication technologies used include: Cellular, Z-Wave, Wi-Fi, Bluetooth, Thread, Zigbee, NFC and 6LowPAN among others. The required application for IoT that operates at a long distance can be used for beneficial technologies such as GSM/3G/4G [13].

13.2.2 Internet of Things (IoT): Machine: Machine Communication Technology

This type of communication is referred to as communication between processors, computers, actuators, smart sensors and mobile devices. These types of communication are now available and increasing by the day. The four components used for M2M are sensing, information processing, heterogeneous access and applications and processing.

Actually, M2M has five-part structure such as the following: M2M Device; M2M Area Network (Device Domain); M2M Gateway; M2M Communication Networks (Network Domain) and M2M Applications. The M2M device is capable of responding to the request of data contained within the device, while the M2M Area Network provides the required connectivity between M2M devices and their gateways. The M2M gateways, on the other hand, are responsible for device interworking and interconnectivity of the communication network. The middleware layer is maintained by the M2M Applications, while Communication Networks (Network Domain) ensure communications between the M2M gateway(s) and M2M application [14]. M2M refers to networks typically used for personal network technologies which include Bluetooth, Ultra-wideband or local networks.

13.2.3 Protection Relay

Protection relay is a device that monitors the flow of current in the power system using current transformer (CT) and voltage transformer (VT) as input source. The relay determines the increase or decrease in the current flow in the distribution network based on the comparison between the relay setting and the system parameters provided the fault exist in the network. Overcurrent relay is set to protect a certain part of the feeder, and it is referred to as reach. The reach of the relay is determined by the minimum fault current in order to set the pickup current of the overcurrent relay which is designed to detect the increase in current flow in

the network caused by a fault. The protection scheme used in modern distribution network is based on the short circuit current setting capacity [15, 16].

As soon as the relay detects fault in the network, it will send a tripping signal to the cloud through a microcontroller which performs the processing work and also controls all other devices which are connected to it and isolates a trip command to the circuit breaker to open and isolate the faulty area of the networks. The protective devices mostly used are a combination of digital impedance relays and electromechanical impedance relays.

13.2.4 Communication of IoT with Smart Grid

Smart grid (SG) refers to an intelligent power system with the integration of IoT network [17, 18]. The power system chain begins from the power plant that generates electricity followed by transmission of the electricity to the distribution network and finally to the end users such as smart appliances, factories, houses, electric vehicles and public lighting. The system enables the users of smart meters, smart appliances, sensors and actuators to monitor and control the system. The aim of the smart grid is to balance the energy generation and consumption of the electricity, which allows the monitoring and control of the power system chain. The smart grid of a power system is considered as one of the important infrastructures and has been found to be one among the largest potential for IoT system implementations. It has two-way communication with smart objects such as smart appliances, sensors, smart meters and actuators [9, 19]. For smart grid networks to be set up, it involves the integration of some cordless sensors and other smart devices; such a design will enable all the components to communicate with one another when connected to a network [8].

IoT-based smart grid will enable the utility companies to make use of IoT in order to improve the development of the system and operation of a smart grid which offers great future and enhance efficiency of the smart devices; it also eases congestion, reduces wastage and eliminates human error [20]. The communication of the network is designed to make it possible for smart objects and other components along with transmission and distribution networks, and also the use of smart objects and components at the side of the end user. This will enable to track the actual time of energy demand and consumption of the energy supply when assisting the customers to monitor the usage of their energy and adjust behaviours [21, 22].

Cyber security is one among the challenges faced with IoT devices. The Internet connection with the sensors, devices and other networks are the most targeted online, ransom and theft that cause destruction [23]. Since the IoT-based smart system grid has the potential to connect millions of online nodes that have the capacity to span over a wide range of geographical regions, it is vulnerable to cyber attacks. A cyber attack always results in devastation and financial losses and brings the entire country to a halt. According to [24], it is reported that 54% attacks in the USA discovered targeted the power system infrastructure and the number of attacks

keeps on rising, therefore, security is the most challenging aspect in the deployment and operation of running IoT-based smart grid systems.

According to [25], Power system hacking occurred in December 2015 in Ukraine; the hackers seized the control of Ukraine's power system by hacking the power system supervisory control and data acquisition (SCADA) with Black Energy malware which caused over 700,000 people to be in total blackout for many hours without electricity supply.

According to [26], Electric grid cyber attack also happened in July 2017 in the UK; power system grid supplies electric energy to the UK and Ireland was attacked and left many in blackout. The cyber attack targeted at seizing the power system control that enabled them to isolate a part of the system supply electricity and left many in blackout. The attacker targeted some senior employees of the power utility company using fake emails to destabilize their activities. The emails contained in the technical information of such power system intends to distract their attention pretends to be a genuine information, but actually wanted false information or make the users to click on the links that would trigger malicious software which is known to them as a spear phishing attack.

Cyber attacks can easily destroy the power system utilities physically that causes general blackout to many, render the power network to be isolated and hand over the control of the system to an outsider or jeopardize the activities of the employees and customers' data. According to [27, 28], four major types of attacks affected are data, devices, privacy and network availability attack.

13.2.5 Review of the Past Research Work

According to Shinde et al. [11], in the IoT-Based Feeder Protection And Monitoring System, special attention was discovered using online feeder monitoring via IoT for faults location and isolation of the affected zone in the distribution system; it is stated that online monitoring helps to establish the nature of the occurrence fault and isolate it for future purposes in order to maintain reliability of the power network online with Global System Mobile (GSM), but they failed to discuss about the types of protection system and the load flow analysis being carried out on the distribution system. Failure of protection system in the network may result in a worse situation that leads to system outages which may threaten the safety of the equipment and personnel. Protection issue in the distribution network is associated with the introduction of overcurrent relay to prevent the blinding, false sympathetic tripping, rupturing of protection fuse, failed auto-reclosing and coordination of relays [29]. Faults in the distribution system demand the load flow analysis and short circuit test to be carried out in the system to determine the healthiness of the system and how the system responds to faults before and after faults occur in the system which cannot be achieved without using designed software for the analysis.

Some other works exist in smart IoT environment carried out by authors [30, 31] and [32]. In [30], authors proposed artificial intelligence techniques for electrical

load forecasting in smart and connected communities while in [31], the authors provided clear connection between IoT and smart cities and communities.

13.3 Proposed Model

The major components used in this research work are microcontrollers which required doing the processing of the system and also control all other devices that connect them together. The relay or LED has a predetermined value or a condition that must be fulfilling before sending a signal to the GSM modem. Figure 13.1 represents the block diagram of IoT-based monitoring system.

According to [30], the GSM modem helps to monitor the operation of the power system distribution network from a far off place of the station or a remote area. This will help the operator to monitor the power system easily from the far off location and detect any faults or problems in the distribution network and act accordingly. The microcontroller is powered with a rectifier circuit, which converts 230 V AC to 5 V DC of the desired voltage level required for the microcontroller.

Modelling of Distribution System to Be Integrated with IoT

The system to be modelled comprised the load feeder of 21-mW feeding a number of residential, commercial and industrial units in a simple radial distribution system. But, only the industrial Nitel 11-kV feeder is a model with IoT integration.

NEPLAN software is used to carry out the load flow analysis of the distribution system in order to determine the state of the system before fault is simulated on Nitel 11-kV feeder.

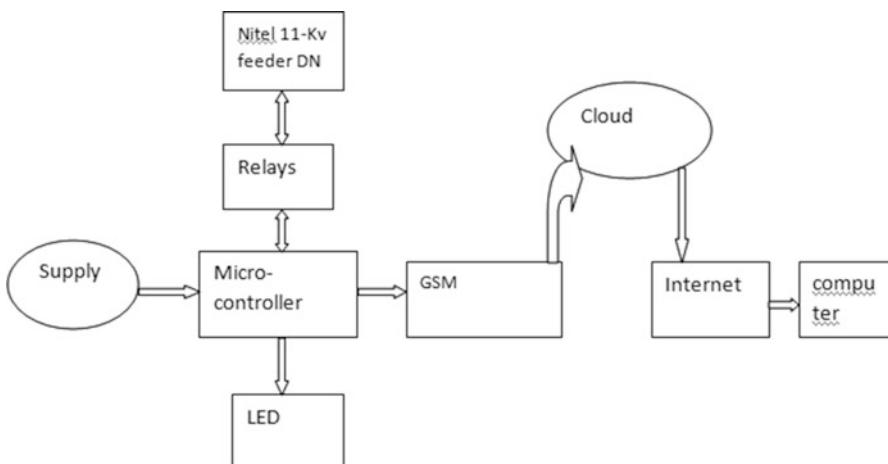


Fig. 13.1 Block diagram of IoT-based monitoring system with distribution network

Also the short circuit test in the distribution system is carried out to determine the fault in current flow and how it affects the voltage profile, power loss and the sensitivity of the protection devices of the distribution system prior to integration of the microcontroller on Nitel 11-kV feeder as well as after the integration.

Table 13.1 shows the results of the load flow analysis of the system under study.

The single line diagram of the distribution network under study incorporated with overcurrent relays to protect the respective feeders is shown in Fig. 13.2; the study is done with simulator using NEPLAN software to perform different types of simulation.

13.3.1 Components of the Single Line Diagram of Test Distribution Network

1. Grid/ Network feeder from Generation to 330/132/33-kV Transmission Station
2. 33-kV bus from Isolo 132/33-kV sub-Transmission station
3. Isolo 33-kV line feeding Isolo 33/11-kV Injection Substation
4. Nitel 33-kV line feeding Nitel 33/11-kV Injection Substation
5. 33-kV bus at Isolo Injection substation
6. 3*15 MVA Power Transformers at Isolo Injection substation
7. 11-kV bus at Isolo Injection substation
8. 12 mW Load feeder at Isoslo Injection substation
9. 33-kV bus at Nitel Injection substation
10. 2*15 MVA Power Transformers at Nitel Injection substation
11. 11-kV bus at Nitel Injection substation
12. 9 mW Load feeder at Nitel Injection substation
13. Protective Relay and Current Transformer (CT)

13.3.2 Design of the GSM Module to Be Integrated into the Tested Distribution Network

Components of the Design

SIM 800L is a quad band GSM GPRS module that works like GSM 850 Mhz frequencies band, EGSM 900 Mhz, DCS 1800 Mhz and PCS 1900 Mhz. SIM 800L supports coding schemes such as CS-1, CS-2, CS-3 and CS-4. SIM 800L is an electronics chip with a total of 88 pins of lga; it provides interface between module and the applications in which it is used. The component is very small in size and can be used in all applications that require small space/area like Smartphone, PDA and other mobile devices.

Table 13.1 Load flow analysis of test distribution network with microcontroller

From Area/Zone	To Area/Zone	P Loss mW	Q Loss MVar	P Imp mW	Q Imp MVar	P Gen mW	Q Gen MVar	P Load mW	Q Load MVar	Iron Losses mW
ID	Node	U	U	Angle U	P Load	Q Load	P Gen	Q Gen	Q Shunt	Area
1077148854	11 kV BUS NITEL INJ S/S	0	0	0	0	0	0	0	0	Zone 1
1077148782	33 kV BUS ISOLO TS	33	100	0	0	18.957	12.088	0	0	Zone 1
1077148788	33/11 kV ISOLO INJ S/S	32.836	99.5	-0.2	0	0	0	0	0	Zone 1
1077148799	33/11 kV NITEL INJ S/S	33.057	100.17	-0.1	0	0	0	0	0	Zone 1
1077148831	11 kV BUS ISOLO INJ S/S	10.75	97.73	-1.7	12	7.437	0	0	0	Zone 1

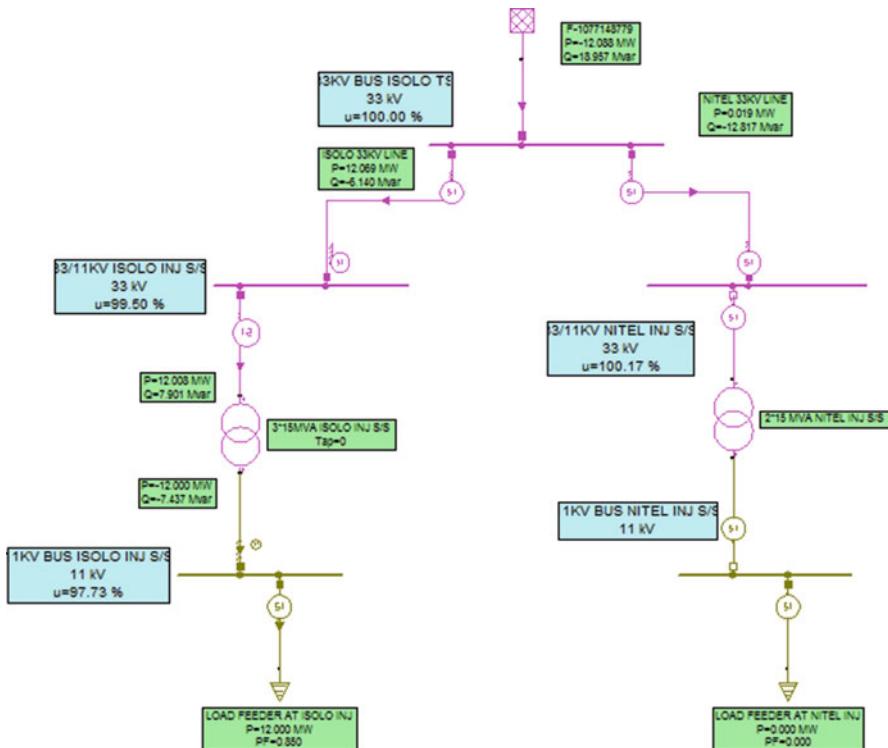


Fig. 13.2 Single line diagram of test distribution system

The power consumption of SIM 800L is very low which saves energy and is powered between the range of 3.4 V and 4.4 V. The temperature is within the normal operation of -40 and 85 °C. Also, it can operate at the wide range temperature that withstands severe conditions. It also has the maximum downlink transfer of 85.6 kbps and uplink of the same value. Integration of TCP/IP protocol that consists of CSD transmission rated 2.4, 4.8, 9.6 and 14.4 kbps. The GSM module has been programmed with the tested distribution network with the system parameters/data to prevent the system from false tripping and blinding tripping. The load flow analysis of the tested distribution system is carried out before and after integration of the design to show that the system can accommodate the design so that the feeders can be controlled automatically.

Load Flow Analysis of the Test Distribution Network Before and After the Microcontroller Is Connected

The load flow analysis result from Table 13.1 reveals that the real power loss (P Loss mW) and reactive power loss (Q Loss MVar) both on the Power Transformer and 33/11-kV lines before and after the integration of the microcontroller are the same which shows that the system is healthy to accommodate the microcontroller design.

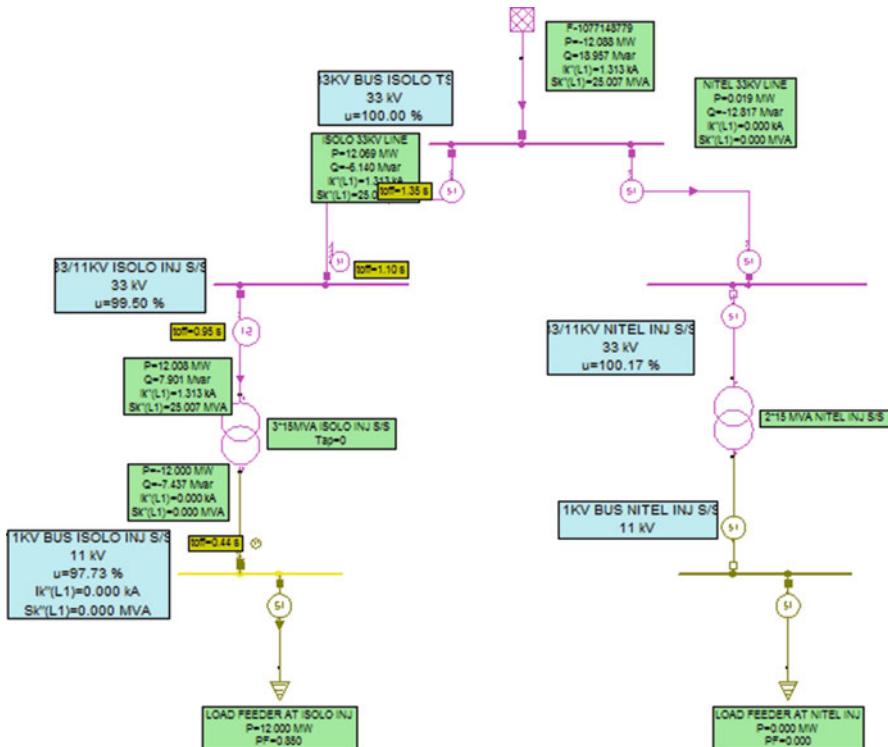


Fig. 13.3 Test distribution system faulted at NTEL Injection substation Isolo 11-kV bUS

13.3.3 Simulation of Faults

Simulation of faults into the distribution network enables the short circuit current flow in the system to be determined and time at which the breaker opens the system.

13.3.3.1 Three-Phase Fault

As shown in the single line test distribution network diagram faulted at NTEL 11-kV Bus in Fig. 13.3, short circuit test is carried out in the system with three-phase fault, and fault current results are shown in Table 13.2.

Table 13.2 shows the node voltages result, fault currents result, and relay tripping time of three-phase fault faulted at Isolo 11-kV bus

The analyses of node voltages result of Table 13.2 from test distribution network when Isolo 11-kV bus is faulted are as follows:

Table 13.2 Node voltages result of Fig. 13.2 test distribution network faulted at Isolo 11-kV bus with microcontroller

	Name	Faulted	Un(kV)	UL-E(kV)	AUL-E (Angle)	UL-L (kV)	AUL-L(Angle)	U(012) (kV)	AU(012) (Angle)	U0 (kV)	AU0 Angle)	Fault type
1E+09	11 kV BUS ISOLO INJ S/S	1	11	6.986	180	12.1	330	6.986	180	0	-90	3 phase fault
1E+09	33 kV BUS ISOLO TS	0	33	16.755	178.6	29.021	330	16.755	178.6	0	-90	3 phase fault
1E+09	33/11 kV ISOLO INJ S/S	0	33	17.426	177.01	30.182	330	17.426	177.01	0	-90	3 phase fault
1E+09	33/11 kV NTEL INJ S/S	0	33	16.755	178.6	29.021	330	16.755	178.6	0	-90	3 phase fault

1. The node voltages of line-earth voltage is 6.986-kV with phase angle of 180.00 degree, but the line-line voltage is 12.01-kV with phase angle of 330.00 degree to the nominal voltage of 11-kV voltage from the table.
2. The node voltages of line-earth voltage is 16.755-kV with phase angle of 178.6 degree, but the line-line voltage 29.021-kV with phase angle of 330.00 degree to the nominal voltage of 33-kV from the table.
3. The node voltages of line-earth voltage is 17.426-kV with phase angle of 177.01 degree, but line-line voltage is 30.182-kV with phase angle of 330.00 degree to the nominal voltage of 33-kV from the table.
4. The node voltages of line-earth voltage is 16.755-kV with phase angle of 178.6 degree, but the line-line voltage is 29.021-kV with phase angle of 330.00 degree to the nominal voltage of 33-kV.

The analysis of node voltages of three-phase fault, the line-earth voltage of the three-phase fault of the node voltages is reduced compare to the line-line voltage of the three-phase fault which is almost twice the line-earth voltage.

13.3.3.2 Fault Currents Result

The analysis of fault currents results from Table 13.3 show the fault currents and corresponding voltages at each of the buses when Isolo 11-kV bus is faulted.

The fault current at faulted Isolo 11-kV bus is 4.615 kA twice the fault current at other buses where backup relay is located at 33/11-kV Isolo Injection substation. At 33-kV bus Injection substation and 33-kV bus Transmission Station, the same fault current of 1.538 kA flow in the circuit. Voltages at each bus from faulted bus to 33-kV transmission station were dropped to the nominal voltage. Voltage at the faulted bus is 6.986-kV with nominal voltage of 11-kV, also at 33-kV bus Isolo Injection substation dropped to 17.42-kV from 33-kV voltage, while at Isolo 33-kV bus Transmission station, the voltage dropped to 16.75-kV from 33-kV.

13.3.3.3 Relay Tripping Time

The analysis of relay tripping time from Table 13.4 shows the time at which the circuit breaker responds to fault current flow in the network and initiates tripping on the faulty feeder.

1. At 0.719s, Isolo 11-kV bus tripped on overcurrent relay when line-earth fault of three-phase fault occurred. The fault current on the bus is 4.615 kA with phase angle 105.3 degree and voltage of 6.986 kV with angle 180.0 degree.
2. At 0.952s 33/11-kV transformer of the injection substation tripped on overcurrent relay when line-earth fault of the three-phase fault occurred. The fault currents at the injection substation is 1.538 kA with phase angle -74.7 degree, and the voltage of 17.426-kV with phase angle 177.01 degree.

Table 13.3 Fault currents result of simulation on three-phase fault

ID	Fault location	NODE	Distance from fault	Elements name	Type	Un kV	UL-E kV	AUL-E	IK'' kA	AIK''	Fault Type
IE+09	11 kV BUS ISOLO INJ S/S	Faulted	0			11	6.986	180	4.615	-74.7	3 phase fault IEC60909
IE+09	11 kV BUS ISOLO INJ S/S	33/11 kV ISOLO INJ S/S		3*15 MVA ISOLO INJ S/S 2 W Transformer					4.615	105.3	
IE+09	33/11 kV ISOLO INJ S/S	1				33	17.426	177.01			
IE+09	33/11 kV ISOLO INJ S/S	33 kV BUS ISOLO TS		ISOLO 33 kV LINE	Line				1.538	105.3	
IE+09	33/11 kV ISOLO INJ S/S	11 kV BUS ISOLO INJ S/S		3*15 MVA ISOLO INJ S/S 2 W Transformer					1.538	105.3	
IE+09	33/11 kV ISOLO INJ S/S	2				33	16.755	178.6		1.538	-74.7
IE+09	33 kV BUS ISOLO TS	33/11 kV ISOLO INJ S/S		ISOLO 33 kV LINE	Line				1.538	-74.7	
IE+09	33 kV BUS ISOLO TS	33 kV BUS ISOLO TS		F-1077148779	Network Feeder				1.538	105.3	
IE+09	33 kV BUS ISOLO TS	33/11 kV NITEL INJ S/S		NITEL 33 kV LINE	Line				0	-90	
IE+09	33/11 kV NITEL INJ S/S	3				33	16.755	178.6			
IE+09	33/11 kV NITEL INJ S/S	33 kV BUS ISOLO TS		NITEL 33 kV LINE	Line				0	-90	

Table 13.4 Relay tripping time result of Fig. 13.2 test distribution network faulted at Isolo 11-kV bus

ID	Name	Node	Elements	Type	Faulted node	Trip time	UL-E (kV)	AUL-E (Angle)	IK''(kA)	AIK'' (Angle)	Description	Zone
1E+09	OC-1077149201	11 kV BUS ISOLO IN/S/S	E-1077148834	Overshoot Relay	11 kV BUS ISOLO IN/S/S	0.719	6.986	180	4.615	105.3	Zone 1	
1E+09	OC-1077149171	33/11 kV ISOLO IN/S/S	E-1077148828	Overshoot Relay	11 kV BUS ISOLO IN/S/S	0.952	17.426	177.01	1.538	-74.7	Zone 1	
1E+09	OC-1077149161	33/11 kV ISOLO IN/S/S	E-1077148796	Overshoot Relay	11 kV BUS ISOLO IN/S/S	1.096	17.426	177.01	1.538	105.3	Zone 1	
1E+09	OC-1077149131	33 kV BUS ISOLO TS	E-1077148795	Overshoot Relay	11 kV BUS ISOLO IN/S/S	1.349	16.755	178.6	1.538	-74.7	Zone 1	

3. At 1.058s 33/11-kV switch-gear breaker of the injection substation tripped on overcurrent protection when line-earth fault of the three-phase fault occurred. The fault currents are 1.538 kA with angle 105.3 degree and 17.426-kV with angle 177.01 degree.
4. At 1.349s 33 kV bus at Isolo Transmission station breaker tripped on overcurrent relay protection when line-earth fault of three-phase fault occurred. The fault currents are 1.538 kA with angle -74.7 degree and voltage of 16.755-kV with angle 178.8 degree.

From the above, the fault currents at the faulted bus are twice of the other buses which confirms that at any point of fault location, the fault currents are always twice the normal current flow in the system. The bus and line at 33 kV level experienced a drop in voltage which confirms that the higher the current flow in a system, the lower the voltage.

The graphical illustration in Fig. 13.4 shows the relay tripping time which indicates that the relay at the point of fault must trip first before any other backup tripping of relays. This shows the coordination of the relay and the zoning of relays.

Curve characteristics of the overcurrent relay of three-phase faults at Isolo 11-kV fault location in the test distribution network with microcontroller is shown in Fig. 13.5.

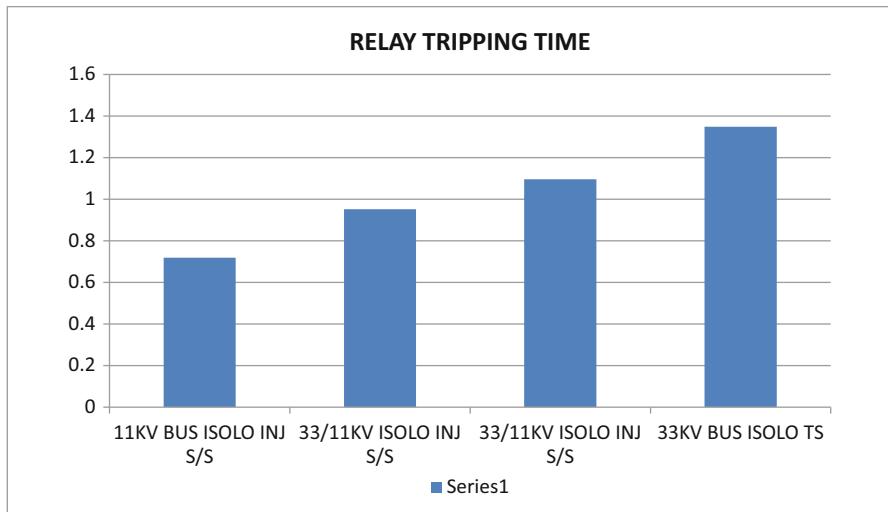


Fig. 13.4 The graphical representation of relay tripping time

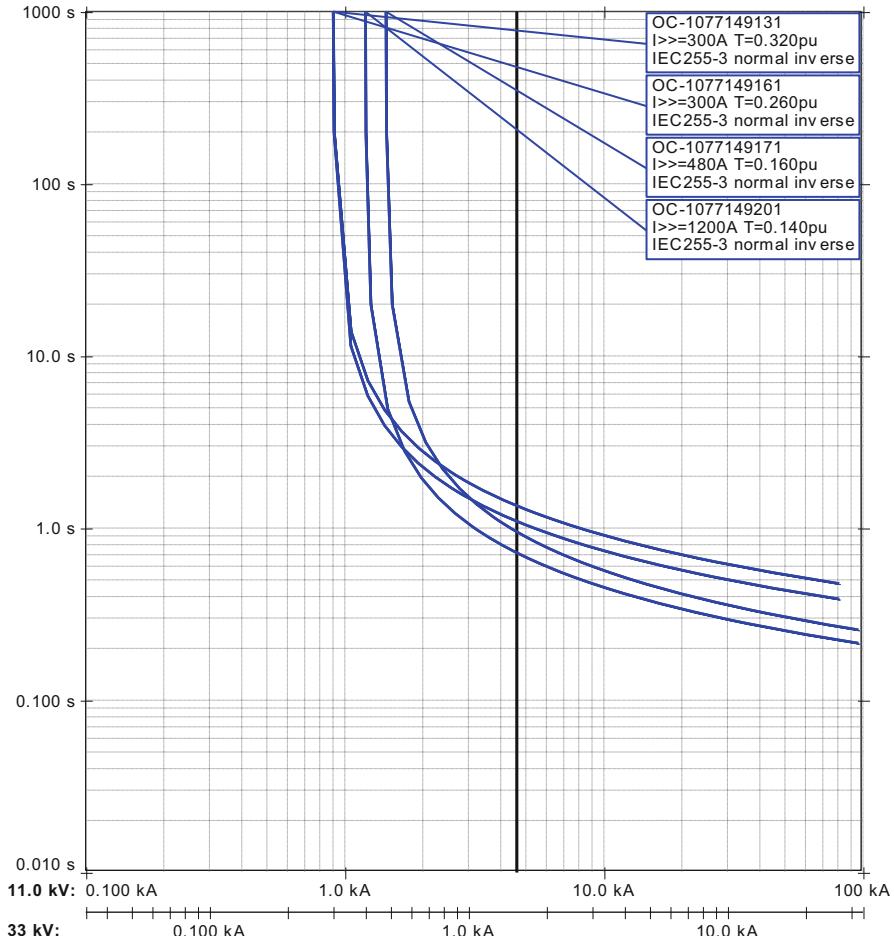


Fig. 13.5 Relay curve characteristics of three-phase fault at Isolo 11-kV bus

13.4 Result and Discussion

Table 13.1 shows the load flow analysis result of the tested distribution system carried out with Neplan software before and after the integration of the design of microcontroller into the power distribution system which confirmed that the system is healthy to accommodate the design via the analysis of the result obtained before simulation of faults into the system. Also, three-phase fault was simulated into the system with a microcontroller as in Fig. 13.3 to confirm that the system will not see the integrated design of the microcontroller as a faulty object to the system via the result analysis interpretation of the result in Tables 13.2 and 13.3 stated above. Table 13.4 shows the setting of the relays to confirm that there is no false or blind

tripping of the relays whenever internal or external faults occur in the system. Figure 13.4 shows the time interval at which each feeder responds to the tripping of faults according to the coordination and zoning of the relays to faults. Figure 13.5 shows the curve characteristics of the relay, the fault current flow in the system, and the corresponding time to trip or isolate the faulty circuit from the system in order to ensure supply is restored to the remaining part of the system.

The study of relay coordination is important for the easy selection of voltage and current transformer; the circuit breaker must be well designed for the distribution network to enable easy isolation of faulty area of the network from the healthy one via proper coordination of the protective relay. The project enables the faults on the distribution network to be monitored easily via the Internet of Things and isolate the faulty area as early as possible.

This work is also related to the affordable and clean energy of SDG of the United Nations.

13.5 Recommendation

The single line diagram of the distribution network under study must be tested by carry out load flow analysis to confirm the healthiness of the network before and after the simulation of the fault into the network with the microcontroller. The incorporation of the protective device with the distribution network feeder and proper coordination of the protective device is very important to protect the system components and provide safety to the personnel. The circuit breaker of the distribution network must be well designed to isolate the faulty part of the network from the healthy one.

The attitude of the protective device of the distribution network here is that the protection coordination should be able to confine the disconnected circuit breaker as the smallest area when a fault takes place. This is to obtain the least interruption of electricity to the end users through monitoring of the system via Internet-of-Things (IoT). To protect system components and for safety purpose, the protection devices must be installed along with the main and lateral feeders with the microcontroller.

13.6 Conclusion

Protection of equipment and feeders of the distribution system becomes more important for the safety of the equipment and personnel. It protects the equipment from any types of faults that may occur in the system and provides easy operation for the utility operator monitoring via Internet-of-Things (IoT). The study of relay coordination is important for the selection of voltage and current transformer, the characteristic of the protective relay, the rating of fuse to be used, and any other relevant information for the optimization of the protective scheme of the system and

selectivity of the relay coordination. Relay coordination activates the devices of the power distribution system by which the fault responds and removes only the affected equipment from the operation. The main objective of the coordination of the relay is to reduce or minimize equipment damage.

Coordination of the protective relay in an-inter-connected of the power distribution system is the main objective to achieve good selectivity of the relay with speed without any issue for the sensitivity of the and time at which the fault cleared. The setting of the relays for protection optimization is the coordination of the relay which is known as the Time Multiplier Setting (TMS) of the relays. The Time Multiplier Setting (TMS) and current setting are the parameters considered in the fault clearance sequence which is considered as optimization of the relay. This chapter shows the importance of the protection system of the test distribution network and the coordination of the overcurrent relay to perform its functions and prevent false tripping and blinding tripping of the network.

Acknowledgement The authors appreciate the Covenant University through its Centre for Research, Innovation, and Discovery for providing research facilities.

References

1. T.M. Fernández-Caramé's, An intelligent power outlet system for the smart home of the Internet of Things. *Int. J. Distrib. Sens. Netw.* **11**(11), 214805 (2015). <https://doi.org/10.1155/2015/214805>
2. B. Mrazovac, M.Z. Bjelica, N. Teslic, I. Papp, Towards ubiquitous smart outlets for safety and energetic efficiency of home electric appliances, in *Consumer Electronics-Berlin (ICCE-Berlin), 2011 IEEE International Conference on*. IEEE, (2011), pp. 322–326
3. R. Piyare, Internet of things: Ubiquitous home control and monitoring system using android based smart phone. *Int. J. Internet Things* **2**(1), 5–11 (2013)
4. M. Frankowiak, R. Grosvenor, P. Prickett, A review of the evolution of microcontroller-based machine and process monitoring. *Int. J. Mach. Tools Manuf.* **45**, 573–582 (2005)
5. J.M. Alonso, P.J. Villegas, J. Diaz, C. Blanco, M. Rico, A microcontroller-based emergency ballast for fluorescent lamps. *IEEE Trans. Ind. Electron.* **44**, 207–216 (1997)
6. S. Sankaranarayanan, A.T. Wan, ABASH—Android based smart home monitoring using wireless sensors, in *Clean Energy and Technology (CEAT), 2013 IEEE Conference on IEEE*, (2013), pp. 494–499
7. D. Evans, *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*. (CISCO System (USA) Pte Ltd, Singapore, 2011). <http://www.CISCO.com/go/ibsg>
8. W. Wang, Z. Lu, *Cyber Security in the Smart Grid: Survey and Challenges* (Elsevier, 2012). <https://doi.org/10.1016/j.comnet.2012.12.017>
9. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions. *Futur. Gener. Comput. Syst.* **29**(7), 1645–1660 (2013). <https://doi.org/10.1016/j.future.2013.01.010>
10. O.M. Machidon, C. Stanca, P. Ogrutan, C. Gerigan, L. Aciu, Power-system protection device with IoT-based support for integration in smart environments. *PLoS One.* **13**(12), e0208168 (2018). <https://doi.org/10.1371/journal.pone.0208168>
11. S. Shinde, S. Jacob, P. Kadam, S. Kachare, IOT based feeder protection and monitoring system. *Int. Res. J. Eng. Technol.* **5**(4), (2018). www.irjet.net e-ISSN: 2395-0056 p-ISSN: 2395-0072

12. O.M. Machido, C. Stanca, P. Ogrutan, C. Gerigan, L. Aciu, S. Lanceros-Mendez, Power-system protection device with IoT based support for integration in smart environments (2018, December 5). <https://doi.org/10.1371/journal.pone.0208168>
13. S. Shinde, S. Jacob, P. Kadam, S. Kachare, IOT based feeder protection and monitoring system. *Int. Res. J. Eng. Technol.* **5**(4), 2073–2076 (2018)
14. S. Liu, B. Chen, T. Zourntos, D. Kundur, K. Butler-Purry, A coordinated multi-switch attack for cascading failures in smart grid. *IEEE Trans. Smart Grid* **5**(3), 1183–1195 (2014). <https://doi.org/10.1109/TSG.2014.2302476>
15. C.F. Wagner, R.D. Evans, *Symmetrical Components* (McGraw-Hill Company Inc., Krieger, 1933). ISBN 10:089874556X/ISBN 13:9780898745566
16. A. Girgis, S. Brahma, Effects of distributed generation on protective device coordination in distribution system (2001, July). ISBN: 0-7803-7107-0. <https://doi.org/10.1109/LESCPE.2001.941636>
17. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **17**(4), 2347–2376 (2015). <https://doi.org/10.1109/COMST.2015.2444095>
18. V.C. Gungor, D. Sahin, T. Kocak, S. Ergüt, C. Buccella, C. Cecati, G.P. Hancke, Smart grid technologies: Communication technologies and standards. *IEEE Trans. Industr. Inform.* **7**(4), 529–539 (2011). <https://doi.org/10.1109/TII.2011.2166794>
19. S. Jain, N. Vinoth Kumar, A. Paventhan, V. Kumar Chinnaiyan, M. Pradish, Survey on smart grid technologies- smart metering IoT and EMS, in *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, (Bhopal, 2014). <https://doi.org/10.1109/SCECS.2014.6804465>
20. D. Miorandi, S. Sicari, F.D. Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges. *Ad Hoc Netw.* **10**(7), 1497–1516 (2012). <https://doi.org/10.1016/j.adhoc.2012.02.016>
21. M. Centenaro, L. Vangelista, A. Zanella, M. Zorzi, Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios. *IEEE Wirel. Commun.* **23**(5), 60–67 (2016). <https://doi.org/10.1109/MWC.2016.7721743>
22. S. Mahalanabish, A. Pradhan, A. Mukherjee, *Smart Grid and the Internet of Things*. (Tata Consultancy Services). www.tcs.com/smart-grid-and-internet-of-thinge
23. K. Billewicz, Possibility of Internet of things technology implementation in smart power grids. *Energetyka* **5**, 264–270 (2016)
24. K. Thakur, M. L. Ali, N. Jiang, M. Qiu, Impact of cyber-attacks on critical infrastructure, in *2016 IEEE 2nd International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing, IEEE International Conference on Intelligent Data and Security*, (New York, 2016). <https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.22>.
25. R.M. Lee, M.J. Assante, T. Conway, *Analysis of the Cyber Attack on the Ukrainian Power Grid* (Electricity Information Sharing and Analysis Center (E – ISAC), 2016)
26. K. Kimani, V. Odul, K. Langat, Cyber security challenges for IoT-based smart grid networks. *Int. J. Crit. Infrastruct. Prot.* (2019). <https://doi.org/10.1016/j.ijcip.2019.01.001>
27. M.A. Ferrag, A. Ahmim, Security solutions and applied cryptography in smart grid communications. *IGI Glob.*, 464 (2016). <https://doi.org/10.4018/978-1-5225-1829-7>
28. X. Li, X. Liang, R. Lu, X. Shen, X. Lin, H. Zhu, Securing smart grid: Cyber attacks, countermeasures, and challenges. *IEEE Commun. Mag.* **50**(8), 38–45 (2012). <https://doi.org/10.1109/MCOM.2012.6257525>
29. A.G. Sukumar Brahma, *Effect of Distributed Generation on Protective Device Coordination in Distribution System* (Clemson University, USA, 2001)
30. V. Alagbe, S.I. Popoola, A.A. Atayero, B. Adebisi, R.O. Abolade, S. Misra, Artificial intelligence techniques for electrical load forecasting in smart and connected communities. *Lect. Notes Comput. Sci.*, 219–230 (2019). https://doi.org/10.1007/978-3-030-24308-1_18

31. M. Olowu, C. Yinka-Banjo, S. Misra, J. Oluranti, R. Ahuja, Internet of things: Demystifying smart cities and communities. *Lect. Notes Netw. Syst.*, 363–371 (2020). https://doi.org/10.1007/978-981-15-3338-9_41
32. L.S. Souza, S. Misra, M.S. Soares, SmartCitySysML: A SysML profile for smart cities applications. *Lect. Notes Comput. Sci.*, 383–397 (2020). https://doi.org/10.1007/978-3-030-58817-5_29

Chapter 14

Medical Data Analysis for IoT-Based Datasets in the Cloud Using Naïve Bayes Classifier for Prediction of Heart Disease



Babatunde Gbadamosi , Roseline Oluwaseun Ogundokun , Emmanuel Abidemi Adeniyi , Sanjay Misra , and Nkiruka Francisca Stephens

Contents

14.1	Introduction	366
14.2	Background and Literature Review	367
14.3	Materials and Methods	371
14.3.1	Data Gathering	372
14.3.2	Data Sample	372
14.3.3	Data Preprocessing	374
14.3.4	Proposed System	374
14.4	Findings and Discussion	375
14.4.1	Weka Software Environment	375
14.4.2	Heart Disease Dataset Analysis	376
14.4.3	Attribute Value Distribution for All Attributes	376
14.4.4	$K = 10$ -Fold Cross Validation Evaluation Technique	377
14.4.5	Detailed Result of Final Analysis (11th Run)	379
14.4.6	Performance Evaluation	382
14.4.7	Discussion	382
14.5	Conclusion and Future Works	383
	References	384

B. Gbadamosi

Department of Mathematics and Statistics, First Technical University, Ibadan, Nigeria

R. O. Ogundokun

Department of Multimedia Engineering, Kaunas University of Technology, Kaunas, Lithuania

e-mail: rosogu@ktu.lt

E. A. Adeniyi

Department of Physical Sciences, Precious Cornerstone University, Ibadan, Nigeria

S. Misra

Department of Computer Science and Communication, Østfold University College, Halden, Norway

e-mail: sanjay.misra@hf.uio.no

N. F. Stephens

Department of Computer Science, Landmark University, Omu Aran, Nigeria

e-mail: Stephens.nkiruka@lmu.edu.ng

14.1 Introduction

Heart disease is an illness that disturbs the heart [1]. Diagnosis of heart disease needs an extremely trained and qualified medical doctor [2]. Data mining, an aggregation of various fields of study, is planned to derive intelligent information from large volumes of data [3]. Data mining effectively extracts healthcare information for medical decisions and the generation of assumptions from comprehensive IoT-based medical datasets in the cloud system [4]. Classification is a widespread concern that is utilized by several applications and to locate an unidentified instance. Investigators concentrate on the design of effective classification techniques for large amounts of data [1]. 17.5 million individuals die every year from cardiovascular disease [5]. These days, heart syndrome is a serious disease that has forced several men to death and has also explained people's limited life span. Existence is dependent on the functioning of the heart, and the heart is still a copious part of the body wherein existence is unlikely. Cardiovascular ailment affects the heart's operation [6] and may lead to death or irritation to an individual before death. Predicting an individual's likelihood of losing adequate blood supply to the heart is one of the main heart failure complications. The forecasting approach can be built by incorporating a variety of autonomous-regression analysis longitudinal studies [7]. Healthcare institutions tend to retain vast volumes of data in their databases by technical progress, and interpreting data is a very difficult process. Data processing methods, image blending methods and machine learning techniques address various difficulties in handling complex data in major hospitals and interpreting data. In this, the image integrates the created image into a simple picture.

Machine learning is also a study of mathematical modeling techniques. And, these types of techniques can be used by computers to execute a certain operation without using overt disruptions. It is a subset of Artificial Intelligence. Deep learning methods operate on a mathematical equation that relies on training samples, making forecasts or making verdicts without being directly trained to do a task. Machine learning has several procedures, such as Random Forest, Decision Tree, and Naive Bayes, to accomplish the task effectively.

Deep learning is related to statistical data, which enables computer-based predictions to be used. Predictive analytics is a data exploration where artificial intelligence focuses on the collection and understanding of data using unsupervised learning. Mathematical simulation is given in the fields of machine learning, theory, and application. ML algorithms and methods can operate on many datasets and allow for conclusions and predictions [8]. Several researchers have implemented machine learning [9, 10], data mining [3] and deep learning algorithms [11] for classification and segmentation [12, 13] leading to prediction [14, 15] and detection [16–18] of diseases.

As far back as the nineteenth century, predictive modeling was first discovered when the government started using early computers, although it had existed for decades. In making predictions on habits and trends, predictive analytics is used. It uses many tools to interpret data and make forecasts, ranging from analytics, data processing, deep learning, artificial intelligence, and so on. In both the

business world and the medical world, it can be used to forecast possible risks and opportunities for individuals around the world [8]. The propagation of ailments has been promoted by the increasing success of global corporations. Most of the time, these viruses have the potential to cross the limits of nations and spread rapidly. It has become a community medical task of immense significance to avoid and diagnose these ailments.

Therefore, this chapter aims to develop a medical data analysis for IoT-based datasets in the cloud using Naïve Bayes Classifier for prediction of heart disease. The Naïve Bayes classifier model was used as the classification model, and it was implemented in WEKA 3.8.

The remaining part of this chapter is organized as follows: Section 14.2 discusses the literature review on cloud computing and related works in the area of ML and AI. Section 14.3 discusses the materials and methods. The ML procedure used for the execution of the system in the study was conferred in the unit as well. Section 14.4 discusses the execution and testing of the system. The findings discovered are also conferred in this section. Section 14.5 concludes this chapter.

14.2 Background and Literature Review

In the IoT model, information and connectivity schemes are invisibly rooted everywhere in the world. This will lead to the creation of an enormous volume of knowledge that needs to be developed, analyzed, and distributed in a smooth, effective, and effortlessly understandable manner. According to Gubbi et al. [19], cloud computing, providing high reliability, scalability, and freedom, is the newest idea to grow. In fact, it offers ubiquitous entry, exploration of resource provisioning and composability obligatory for potential IoT applications. This portal performs as a data transmitter for pervasive radars, a data analysis and simulation system and an Internet visualization comprehension provider. The cloud not only minimizes the cost of installing pervasive applications, but is also extremely scalable. Providers of sensing information can use a storage cloud to view and distribute their findings to the network; computer technology engineers can provide their applications for creation; machine intelligence specialists can deliver their data analysis and ML procedures; and computer animation originators can finally provide a range of simulation techniques. Cloud computing as in-structures, networks, or applications where the maximum power of human creativity can be accessed, can deliver these tools to the IoT. The data created, the equipment used and the method of making complicated visualizations [20] are concealed in this context. Cloud computing is a programming model that delivers on-demand software to end users from a range of computing facilities, including database services and computer infrastructure [21]. The key tools offered by cloud computing are Infrastructure as a Service (IAAS), Application as a Service (PAAS), and Software as a Service (SAAS) Mell and Grance [21]. Many of these suppliers offer computing facilities such as storage and data processing on request [22]. In addition to supplying the servers

mentioned above, cloud computing similarly concentrates on the active optimization of distributed resources by manifold consumers. Western customers, for example, are given a cloud storage service (for instance, email) based on their time zone. The same space is distributed by cloud storage to Asian customers, mostly depending on their time zone.

As a flexible and preordained innovation, cloud computing has recently emerged and is developing rapidly. According to customer needs, the cloud computing platform offers highly versatile, open, and programmable virtual servers, space, computer power, and digital communication. A solution package for the electronic transition of information will also be provided if it is designed for IoTs and paired with current data storage, distribution and preservation technologies. In addition, as long as the computing space allows [23], the user generates knowledge very quickly. The customer, who needs to be available easily, can still use much of the information. One of the key facets of cloud infrastructure is media management, as the cloud facilitates the storage, handling and sharing of large quantities of electronic information. For digital media connected to IoTs, this capability can display a key function. In the future, many more multimedia services will be possible for individuals on the move, for instance, smartphones, tablets and laptop operators, ad hoc vehicle systems, separate ambulances, and rescue operations. Cloud infrastructure plans to display an actual strong responsibility in the organization of programs and in-structure for those resources that are relevant. The server would be used more frequently, particularly with Fog Computing [24], also known as Edge Computing or Micro-Datacentre, an extended network (MDC). A handy approach for handling content in distributed environments is cloud storage. Without the responsibility of handling large storage and processing instruments, it provides streamlined access to information. Another feature cloud providers offer is to share a huge number of online media. In addition to social networks, traditional cloud computing incorporates additional sharing and content management functionality. Similarly, it is not possible to import individual files one after the other if material is to be shared. This issue is addressed by cloud computing so all information can be accessed at once by all entities for whom the data is transmitted. Also, cloud storage can provide more context-aware services, as IoT and sensor networks are not adequately thick in assets to conduct such operations. To generate further tailored and functional software, data deposited in the cloud may even be fully processed [25].

Machine learning is one of the most promising new artificial intelligence developments. The name of an Artificial Intelligence (AI) software is Intelligent Agent. AI has been used for the prediction and detection of diseases [3, 9, 11, 26–28]. Smart towns, computer vision, medical treatment, self-propelled, and machine learning algorithms are used for different topics, for example [9, 10, 13, 29–32]. In these fields, there are various instances of how machine learning can be deployed on edge gadgets [33–35]. The Smart Agent will collaborate with the universe to organize. Via its sensors, the agent can recognize the status of an area and then, via its actuators, it can impact the state. The main AI aspect is the system's control strategy, which means how the signals received from the sensors are sent to the

actuators, that is, how the sensors are related to the actuators, which is done inside the operator by the feature. The overall goal of AI is to create computer intelligence that is human-like. Nevertheless, this goal can be achieved by learning procedures that aim to replicate the way human brain thinks. Hence, AI programs performs the most stimulating things, which include site search, photo tagging, and e-mail anti-spam. Thus, ML was established as an innovative competence for computers, and these days it covers countless business and basic science fields. Autonomous robotics is there, and computer biology is there. Around 90% of the world's data has been developed on its own over the preceding 2 years, and the presence of the ML library identified as Mahout into the Hadoop ecosystem has made it possible to face the problems of big data, particularly unstructured data. Much of the emphasis is on the selection or construction of an algorithm and research in the area of machine learning science depending on the algorithm. This fundamentally skewed viewpoint inhibits the influence of real-world technologies.

Related Works

Classification is among the most critical, essential and influential decision-making methods in medical research. A lot of new research has been developed to forecast coronary heart disease reliably and properly. The following are brief past works done in the area of machine learning and IoT-based medical datasets in the cloud system.

Guan et al. [36] suggested a device that was solely focused on a Support Vector Machine (SVM) capable of forecasting cardiovascular disease effectively. The accuracy rating of the implemented method was 76.5%. The framework contrasted various models such as Regular SVM, Recursive features Exclusion and estimated L0-norm SVM techniques.

Shilaskar and Ghatol [37] employed innumerable grouping and filtering approaches to envisage cardiovascular ailment. They utilized the classification model to improve and filter forward functionality and to pick back-propagation features. Their analysis shows that it simplified the number of parameters and improved the precision of the results. The computer registered an average accuracy of around 85%.

Shao et al. [38] developed a scheme for determining the correctness of the cardiovascular ailment predictive model using machine learning approaches such as regression models and for selecting the top and most significant attributes. The scheme utilized irregular set techniques and multivariate adaptive regression connectors to shrink the size of the descriptive attributes for heart illness identification. The developed scheme has obtained correctness of 82.14%.

Kumar and Inbarani [39] have designed a program that uses Particle Swarm Optimization (PSO) classification strategies to diagnose cardiovascular disease. The framework utilized quick enhancement and relative optimization to determine suitable attributes. The results were used to respond to ML procedures for instance, Multilayer Perception, Backpropagation, SVM and K Nearest Neighbor (KNN) techniques to classify the data set and achieve correctness of 81.73%, 82.30%, 75.37%, and 91.94% correspondingly.

Rajathi and Radhamani [40] suggested a technique using KNN along with Ant Colony Optimization (ACO) approaches to forecast cardiovascular ailment. They matched their performance (70.26%) to four machine learning techniques.

Bashir et al. [41] suggested utilizing SVM, Naïve Bayes, and DT-GI to forecast cardiovascular diseases. The missing parameter in the dataset is extracted in the preprocessing stage. Then, using plurality voting methods, they measured the accuracy of the diagnosis of coronary heart disease. They observed 82% consistency in their test outcome.

Amin et al. [42] suggested a cross approach in which main risk characteristics are used to identify heart disease. They employed two common methods for their method called Genetic techniques and Neural Network (NN) algorithm. Through a genetic algorithm and a broad optimization method, apiece neurons' mass was initialized in the neural networks. The result revealed that their system was easy relative to diverse ones and had correctness of 89%.

Khatibi and Montazer [43] also proposed a fuzzy method built on fuzzy collections and the Dempster-Shafer principle. Two phases accompany the suggested procedure. Second, the inputs are represented by fuzzy components, and the fuzzy collections are operated out by a fuzzy inference method. Second, the model inference engine created confidence and merged the different knowledge using mixture laws. A precision of 91.58% was achieved.

Temurtas and Tanrikulu [44] suggested a model utilizing deep networking methods to characterize the dataset for cardiovascular diseases. They splinted the dataset to threefold cross-validation and then used neural network techniques to ascertain the outcome. In the trial, the precision of their model was 96.30%.

Yumusak and Temurtas [45] employed multiple-layer neurons to model coronary heart illness. They utilized multiple hidden-layers amid the input and output layers and observed an overall precision of the model to be 91.60%. Liu et al. [46] utilized regression and localized linear embedding (LLE) methods to identify coronary heart illness and achieved a precision of around 80%.

Nashif et al. [47] also proposed using various machine learning techniques to track cardiovascular diseases. A cloud-based framework has been adopted in this research, where clinicians can store medical data to verify their cardiac state of health. The framework uses 86.40%, 97.53%, 95.76%, 95.05% and 77.39% naive fruit, SVM, random tree, basic logistic, and ANN.

Singh et al. [48] developed a system for forecasting heart disease, emphasizing the Structural Equation Modeling (SEM) and the Fuzzy Conceptual Diagram. (FCM). They reviewed the compilation of data from the Canadian Public Health Evaluation, which contains 20 important qualities. The SEM model is defined as a correlation between the characteristics of CCC 121 and 20; here, CCC 121 is a function that decides whether a person has cardiovascular disease.

Ghadge et al. [49] also proposed an interactive method for forecasting heart attacks using data analytics. The main effect of this research was to identify the basis for an intelligent cardiac arrest prediction approach that uses massive data and data mining simulation methods.

Motivation

Innumerable research has been accomplished to estimate heart ailment by employing UCI machine learning IoT-based datasets. Diverse percentage of accuracy was realized by employing innumerable ML techniques which were described above. Therefore, the key concept behind this proposed method after evaluating the above previous papers was to build a heart disease prediction system based on the datasets inputted into the system. The motivation of this study is to figure out if Naïve Baye classifier technique is more effective for prediction when used.

14.3 Materials and Methods

This section discussed the materials used for this study in terms of datasets used for the implementation. The machine learning algorithm adopted for this study was also discussed in this section. To predict heart diseases, this chapter follows the following steps: Data gathering, dataset sample, data preprocessing, and proposed system. The following subsections explain the steps in detail.

Naïve Bayes (NB) has remained commonly applied in clinical data processing; it has been found to have well fitted with the distinctive characteristics of patient data [49]. Categorization of the NB is a technique for detecting the occurrence of the critical features of a class that are not related to the occurrence of any other variable. NB defines every segment with a probability description and calls the utmost probable class. It is identified that the NB group functions pretty good in certain areas, not well in others. The output of NB is affected in areas that include redundant clustered and unrelated functions. The naive classifier of Bayes can be described as below. Block capitals like X_i represent parameters, and their meanings are represented by lowercase letters like x_i , and boldface characters like X represent parameter collections.

Let $X = \{X_1, \dots, X_n\}$ be a finite set of random variables identified, labeled attributes in which each function takes values from its D_i field. The collection of all the set attributes is represented by name $= D_1 \times \dots \times D_n$. Let C , such as $c \in \{0, \dots, u - 1\}$, be an unknowable random variable indicating the class of a set of features.

The hypothesis h that attributes a class to any particular set of variables $\{0, \dots, u - 1\}$ is known as a classifier. A discriminant function $f_c(x)$, $c = 0, \dots, u - 1$ is allocated to each class c . The classifier picks a session with the highest discriminant purpose on a given set of variables, written as $h(x) = \arg \max_{c \in \{0, \dots, u - 1\}} f_c(x)$.

The Bayes classifier $h_*(x)$ uses the posterior probabilities given a set of variables as the discriminant function, that is, $f_*(x) = P(C = c | X = x)$. Applying Bayes' theorem from Eq. 14.1 to this function gives

$$(C = c | X = x) = \frac{P(X = x | C = c) P(C = c)}{P(X = x)} \quad (14.1)$$

Since $P(X = x)$ is the same for all classes, it can be ignored. Hence, the Bayes' discriminant function can be written as

$$f^*(x) = P(X = x|C = c) P(C = c) \quad (14.2)$$

where $P(\mathbf{X} = \mathbf{x} | C = c) P(C = c)$ is called the class-conditional probability distribution (CPD). Thus, the Bayes' classifier, written in Eq. 14.3, finds the maximum posterior probability hypothesis given x .

$$h^*(x) = \arg \max_c P(X = x|C = c) \cdot P(C = c) \quad (14.3)$$

Applying the assumption that features are independent given the class on Eq. (14.4), we can get the Naïve Bayes classifier.

$$f_c^{\text{NB}}(x) = \prod_{j=1}^n P(X_j = x_j | C = c) P(C = c) \quad (14.4)$$

The zero-frequency problem, which transpires when a feature value does not occur with all class value, can be solved by adding a value 1 to the count of every feature value-class combination [50].

14.3.1 Data Gathering

Recordset with medical attributes vital to the heart disease prediction was available from the Cleveland Heart Ailment library in the UCI machine learning warehouse (UCI Machine Learning Archive: Heart Ailment Dataset). Out of 13 features, 12 features were taken as input, and one feature was used as output. The input characteristics considered are age, sex, type of chest pain (typical angina, atypical angina, and non-angina), resting blood pressure, serum cholesterol (in mg/dl), fasting blood sugar (>120 mg/dl), maximum heart rate reached, angina caused by exercise, ST-segment peak slope of exercise (slope up exercise, slope down exercise). The output attribute considered is a disease (has heart disease, no heart disease). It is presented in a Microsoft Excel file (.csv) format.

14.3.2 Data Sample

The patient database contains a dataset with medical attributes vital to heart disease, raw data from the Cleveland cardiovascular disease sample accessible in the UCI registry. Out of 13 attributes, 12 features were used as input, and one feature was used as feedback. Input characteristics considered: age, gender, degree of trouble breathing (typical angina, atypical angina, and non-angina), resting blood pressure,

serum cholesterol (in mg/dl), fasting blood sugar (>120 mg/dl), maximal heart rate reached, angina caused by exercise, ST-segment peak slope of exercise (slope up exercise, slope down exercise). The output attribute considered is a disease (has heart disease, no heart disease).

1. Age in years.
2. Sex (Value 1 = male, 0 = female).
3. Typical angina (Value 1 = yes, 0 = no).
4. Atypical angina (Value 1 = yes, 0 = no).
5. Non-angina (Value 1 = yes, 0 = no).
6. Resting blood pressure (Value \leq 145 (min), 145-above (max)).
7. Serum cholesterol (Value \leq 240 (min level), 240-above (max level)).
8. Fasting blood sugar (Value 1 = yes, 0 = no).
9. Maximum heart rate (Value \leq 70 (min), 70-above (max)).
10. Exercise-induced angina (Value 1 = yes, 0 = no).
11. Slope up (Value 1 = yes, 0 = no).
12. Slope down (Value 1 = yes, 0 = no).
13. Disease (Value yes = has heart disease, no = no heart disease)

Figure 14.1 shows the sample of the dataset used for the proposed system implementation and testing.

No	Disease	Age	Sex	ind_tvp_angina	ind_atyp_angina	ind_non_ang_pain	resting_BP	Serum_cholest	blood_sugar_excl20	Max_heart_rate
	Nominal	Nominal	Nominal	Numerico	Numerico	Numerico	Numerico	Numerico	Numerico	Numerico
1	No	63	1	1.0	0.0	0.0	145.0	233.0	1.0	150.0
2	Yes	67	1	0.0	0.0	0.0	160.0	286.0	0.0	108.0
3	Yes	67	1	0.0	0.0	0.0	120.0	229.0	0.0	129.0
4	No	37	1	0.0	0.0	1.0	130.0	250.0	0.0	187.0
5	No	41	0	0.0	1.0	0.0	130.0	204.0	0.0	172.0
6	No	56	1	0.0	1.0	0.0	120.0	236.0	0.0	178.0
7	Yes	62	0	0.0	0.0	0.0	140.0	268.0	0.0	160.0
8	No	57	0	0.0	0.0	0.0	120.0	264.0	0.0	163.0
9	Yes	63	1	0.0	0.0	0.0	130.0	254.0	0.0	147.0
10	Yes	53	1	0.0	0.0	0.0	140.0	203.0	1.0	155.0
11	No	57	1	0.0	0.0	0.0	140.0	192.0	0.0	148.0
12	No	56	0	0.0	1.0	0.0	140.0	294.0	0.0	153.0
13	Yes	56	1	0.0	0.0	1.0	130.0	256.0	1.0	142.0
14	No	44	1	0.0	1.0	0.0	120.0	263.0	0.0	173.0
15	No	52	1	0.0	0.0	1.0	172.0	199.0	1.0	162.0
16	No	57	1	0.0	0.0	1.0	150.0	168.0	0.0	174.0
17	Yes	48	1	0.0	1.0	0.0	110.0	229.0	0.0	168.0
18	No	54	1	0.0	0.0	0.0	140.0	239.0	0.0	160.0
19	No	48	0	0.0	0.0	1.0	130.0	275.0	0.0	139.0
20	No	49	1	0.0	1.0	0.0	130.0	266.0	0.0	171.0
21	No	64	1	1.0	0.0	0.0	110.0	211.0	0.0	144.0
22	No	58	0	1.0	0.0	0.0	150.0	283.0	1.0	162.0
23	Yes	58	1	0.0	1.0	0.0	120.0	284.0	0.0	160.0
24	Yes	58	1	0.0	0.0	1.0	132.0	224.0	0.0	173.0
25	Yes	60	1	0.0	0.0	0.0	130.0	206.0	0.0	132.0
26	No	50	0	0.0	0.0	1.0	120.0	219.0	0.0	158.0
27	No	58	0	0.0	0.0	1.0	120.0	340.0	0.0	172.0
28	No	66	0	1.0	0.0	0.0	150.0	226.0	0.0	114.0
29	No	43	1	0.0	0.0	0.0	150.0	247.0	0.0	171.0
30	Yes	40	1	0.0	0.0	0.0	110.0	167.0	0.0	114.0
31	No	69	0	1.0	0.0	0.0	140.0	239.0	0.0	151.0
32	Yes	60	1	0.0	0.0	0.0	117.0	230.0	1.0	160.0
33	Yes	64	1	0.0	0.0	1.0	140.0	335.0	0.0	158.0
34	No	59	1	0.0	0.0	0.0	135.0	234.0	0.0	161.0
35	No	44	1	0.0	0.0	1.0	130.0	233.0	0.0	179.0
36	No	42	1	0.0	0.0	0.0	140.0	226.0	0.0	178.0
37	Yes	43	1	0.0	0.0	0.0	120.0	177.0	0.0	120.0
38	Yes	57	1	0.0	0.0	0.0	150.0	276.0	0.0	112.0
39	Yes	55	1	0.0	0.0	0.0	132.0	353.0	0.0	132.0

Fig. 14.1 Dataset sample

14.3.3 Data Preprocessing

This procedure encompasses the retrieval of data from the Cleveland database for cardiovascular disease in a standardized format. This process includes the transformation of data, which requires eliminating missing areas, the standardization of data, and eliminating exceptions. Four rows of incomplete values have been deleted, leaving 299 documents. Six columns have since been excluded, leaving 13 columns. Attributes like sex, chest pain type (typical angina, atypical angina, and non-angina), fasting blood sugar, exercise-induced angina, and slope (slope up exercise, slope down exercise) have been transformed to indicator variables.

14.3.4 Proposed System

The projected system framework is displayed in Fig. 14.2. Patients' details (attributes) regarding heart disease comprise the heart disease dataset. Attributes relevant to heart disease were selected and gathered from the Cleveland heart ailment repository accessible on the UCI repository. Preprocessing on the dataset includes transforming the data by removing missing fields, normalization of data, and removal outliers. The Naïve Bayes classification technique is applied, and the dataset is classified (predicted), then the precision of the Naïve Bayes classifier is measured.

The Naïve Bayes Classification algorithm steps as outlined as follows:

- (i) Select cross validation technique (k -fold cross validation technique).
- (ii) Input data is fragmented into training set and testing set.
- (iii) Utilizing training set, preceding probability, $P(c)$ of each class is calculated.

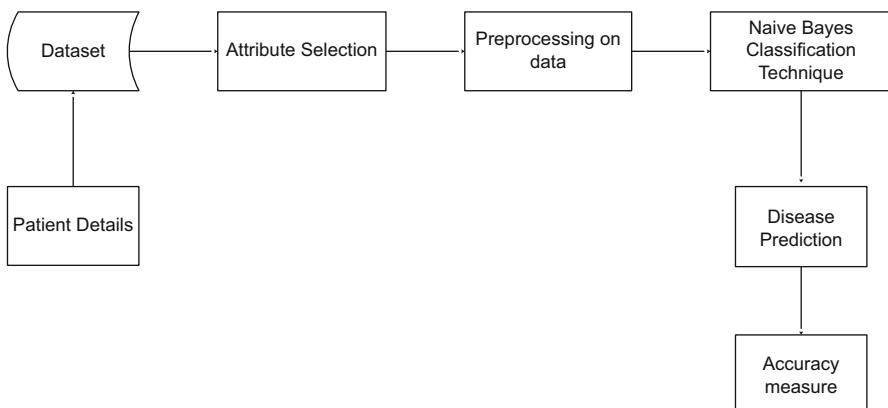


Fig. 14.2 Proposed system framework

- (iv) Utilizing a solitary instance from the test set, conditional probability for each attribute is computed.
- (v) Values obtained are then employed to compute the posterior probability for each class.
- (vi) Class with greatest posterior prospect is allotted as class for that test instance, process is done on each instance in the test set.
- (vii) Accuracy of classifier is computed (percentage of exact predictions divided by the overall number of predictions).

14.4 Findings and Discussion

This section discusses the outcomes gotten from the execution and the testing of the classification using the Naïve Bayes machine learning algorithm. The interfaces of the implementations are also presented in this section.

In this study, the Naïve Bayes classification algorithm was used for implementation and testing. In this section, the interface gotten from the implementation is discussed, and the system was evaluated using the confusion matrix. The study used Weka 3.8 software for the simulation of the Naïve Bayes classifier for the heart disease prediction.

14.4.1 Weka Software Environment

The Waikato Environment for Knowledge Analysis, or WEKA, is an open-source program built in Java and licensed under the GNU General Public License. WEKA's development started at the University of Waikato, New Zealand, in 1992, and was first published for use in 1999. WEKA 3.8. is the current stable update. Weka is fundamentally a suite of ML procedure such as data preprocessing, visualization, classification, regression, and clustering, for data mining activities. Through providing access to SQL databases using JDBC returned by a database query, it can also process the response.

Its graphical user interfaces for ease of access to its functionalities are as follows:

- The explorer interfaces
- The experimenter interfaces
- The knowledge flow interface
- The command line interface

Figure 14.3 shows the WEKA 3.8 software environment used for this study simulation and testing.

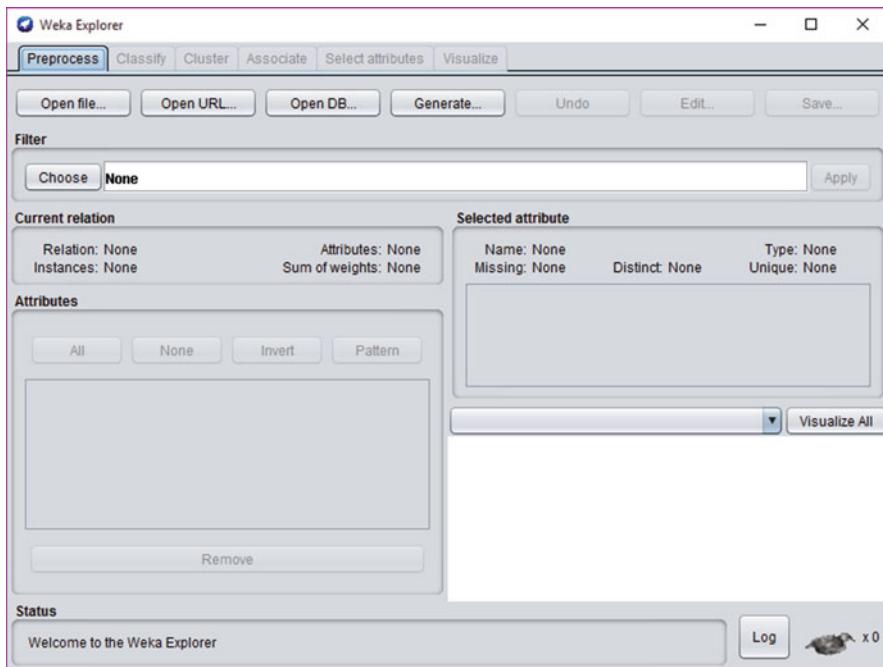


Fig. 14.3 The WEKA explorer interface

14.4.2 Heart Disease Dataset Analysis

The dataset saved in a csv format was uploaded into the explorer workbench by going through the following processes:

- Click on the “open file” tab.
- Locate the data to be uploaded (i.e., HEART_DISEASE_DATASET.csv).
- The result is displayed in Fig. 14.4.

Each attribute was selected so as to see how they were analyzed by the data mining tool. The interface for a selected attribute (Disease) is shown in Fig. 14.5.

14.4.3 Attribute Value Distribution for All Attributes

Figure 14.6 shows the attribute value distribution. The blue-colored regions in the graphs in the figure denote values for “no heart disease,” and the red-colored regions denote values for “has heart disease.” From the graph, it can be observed that patients who have heart disease are fewer than those that do not have heart disease

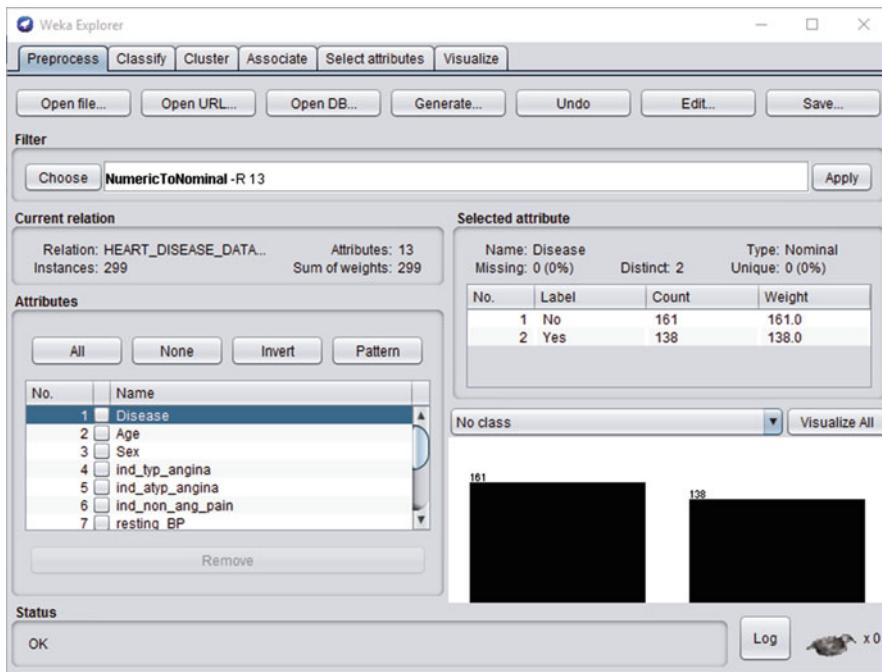


Fig. 14.4 Result display

and are in the age group of 55–63. Most female patients do not have heart disease, while more male patients have heart disease.

Figure 14.7 shows the run test analysis for the proposed system.

14.4.4 $K = 10$ -Fold Cross Validation Evaluation Technique

The classification was checked using a k -fold cross-validation process. The value of k was entered as 10. The precision was determined for each of the ten iterations for ten separate random number seeds. Using all the accuracy received, the overall accuracy as follows (mean) was calculated.

For the heart disease dataset, 90% of the occurrences were used as the training dataset, and 10% of the cases were used as the testing dataset.

- Divide dataset into ten parts (folds).
- Holdout each part in turn.
- Average the results.
- Each data point is used once for testing and nine times for training.

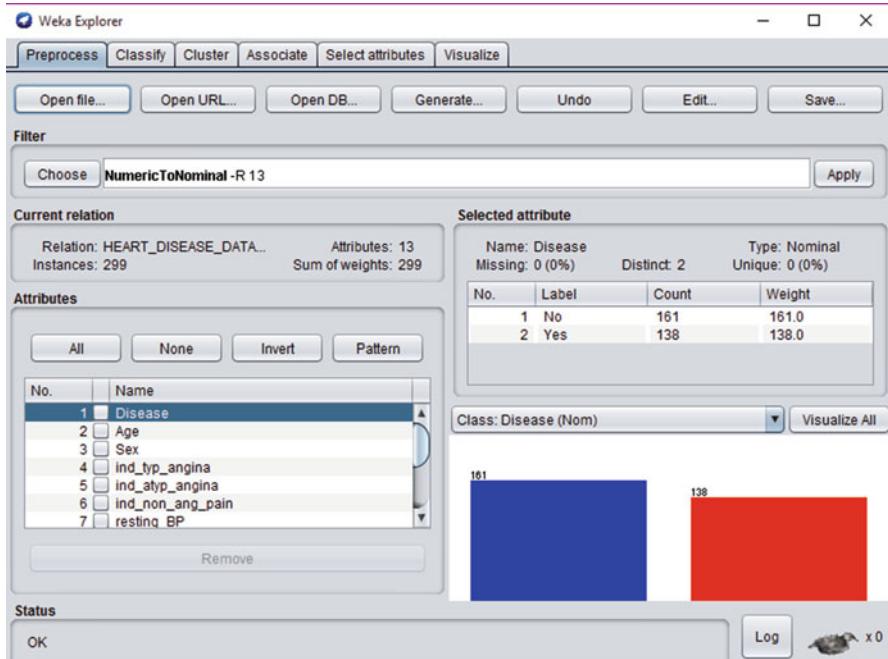


Fig. 14.5 Selected attribute

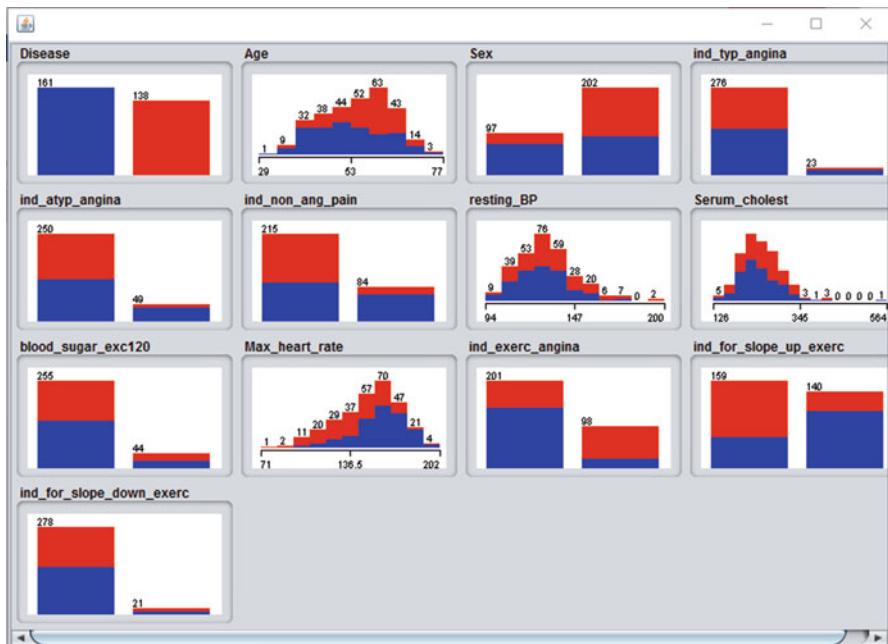


Fig. 14.6 Attribute value distribution

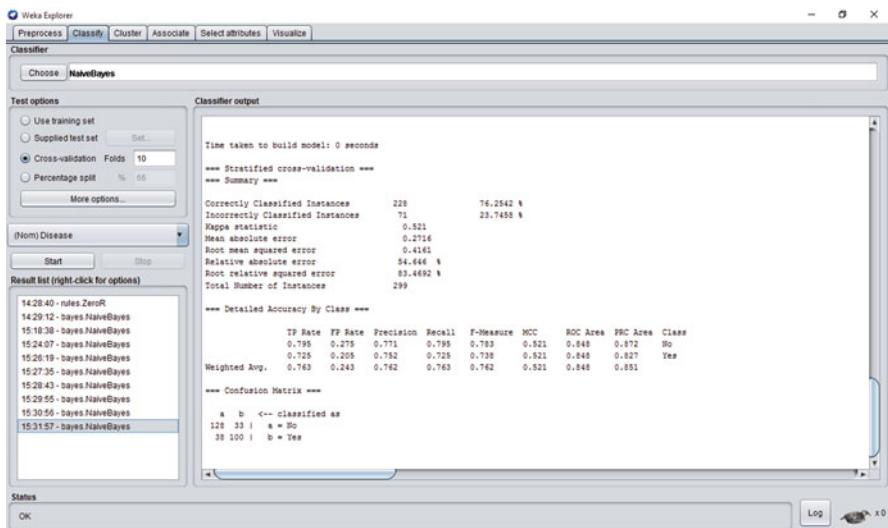


Fig. 14.7 Run test analysis

After cross-validation, WEKA outputs an extra model built on the entire dataset (100% of the instances), the 11th run. This serves as the actual accuracy obtained for this classification. Average Accuracy (Mean)

14.4.5 Detailed Result of Final Analysis (11th Run)

A final analysis is carried out using the entire dataset (100% of the instances) for training the algorithm. The result serves as the actual accuracy obtained for this classification.

```

==== Run information ====
Scheme:      weka.classifiers.bayes.NaiveBayes
Relation:    HEART_DISEASE_DATASET-weka.filters.unsupervised.
             attribute.NumericToNominal-R3-weka.filters.unsupervised.
             attribute.NumericToNominal-R4-weka.filters.unsupervised.
             attribute.NumericToNominal-R5-weka.filters.unsupervised.
             attribute.NumericToNominal-R6-weka.filters.unsupervised.
             attribute.NumericToNominal-R6-weka.filters.unsupervised.
             attribute.NumericToNominal-R9-weka.filters.unsupervised.
             attribute.NumericToNominal-R11-weka.filters.unsupervised.
             attribute.NumericToNominal-R12-weka.filters.unsupervised.
             attribute.NumericToNominal-R13
Instances:   299
Attributes:  13
             Disease
             Age

```

```

Sex
ind_typ_angina
ind_atyp_angina
ind_non_ang_pain
resting_BP
Serum_cholest
blood_sugar_exc120
Max_heart_rate
ind_exerc_angina
ind_for_slope_up_exerc
ind_for_slope_down_exerc
Test mode: evaluate on training data
==== Classifier model (full training set) ====

```

Naive Bayes Classifier

Attribute	Class	
	No (0.54)	Yes (0.46)
<hr/>		
Age		
mean	52.6137	56.7217
std. dev.	9.5078	7.8564
weight sum	161	138
precision	1.2	1.2
0	73.0	26.0
1	90.0	114.0
[total]	163.0	140.0
ind_typ_angina		
0	146.0	132.0
1	17.0	8.0
[total]	163.0	140.0
ind_atyp_angina		
0	122.0	130.0
1	41.0	10.0
[total]	163.0	140.0
ind_non_ang_pain		
0	96.0	121.0
1	67.0	19.0
[total]	163.0	140.0
resting_BP		
mean	129.0435	134.5143
std. dev.	16.3265	18.771
weight sum	161	138
precision	2.1633	2.1633
mean	243.2773	251.5589
std. dev.	53.4476	49.533
weight sum	161	138
precision	2.9007	2.9007
blood_sugar_exc120		
0	139.0	118.0
1	24.0	22.0
[total]	163.0	140.0
Max_heart_rate		
mean	158.2578	139.2376
std. dev.	19.2056	22.5932
weight sum	161	138
precision	1.4556	1.4556
ind_exerc_angina		
0	139.0	64.0
1	24.0	76.0

Table 14.1 Accuracy table using tenfold cross-validation

Runs	Cross validation (ten-fold)
1	76.2542
2	76.9231
3	76.5886
4	75.2508
5	74.9164
6	76.9231
7	75.5853
8	75.9197
9	76.2542
10	75.9197

Average accuracy (mean) = 76.0535
Accuracy = 76%

```
[total]           163.0    140.0ind_for_slope_up_exerc
0                 58.0     103.0
1                105.0     37.0
[total]           163.0    140.0ind_for_slope_down_exerc
0                 153.0    127.0
1                  10.0     13.0
[total]           163.0    140.0
Time taken to build model: 0.01 seconds
==== Evaluation on training set ====
Time taken to test model on training data: 0.02 seconds
==== Summary ====
Correctly classified instances          232      77.592 %
Incorrectly classified instances         67       22.408 %
Kappa statistic                         0.5475
Mean absolute error                     0.2585
Root mean squared error                 0.4014
Relative absolute error                 52.0046 %
Root relative squared error            80.5271 %
Total number of instances               299
==== Detailed Accuracy by Class ====
TP Rate FP Rate Precision Recall   F-Measure MCC
      0.814  0.268   0.780   0.814   0.796   0.548   0.866   0.890   No
      0.732  0.186   0.771   0.732   0.751   0.548   0.866   0.847   Yes
Weighted Avg.  0.776  0.230   0.776   0.776   0.775   0.548   0.866   0.870
==== Confusion Matrix ====
a   b   <-- classified as
131 30   |   a = No
 37 101  |   b = Yes
```

Final Analysis Result Summary

Correctly Classified Instances	232	77.592%
Incorrectly Classified Instances	67	22.408%
Accuracy	= 77%	

Table 14.1 shows the tenfold cross-validation accuracy for the system and it is shown that runs 2 and 6 has the highest cross-validation value.

Table 14.2 Confusion matrix table

	<i>Actual class</i>		<i>Predicted class</i>	
	Total	No	Yes	
N	TN	FP	No	
P	FN	TP	Yes	
N + P	N'	P'	Total	

14.4.6 Performance Evaluation

The study used confusion matrix for the performance evaluation and this is discussed below. The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes. TP and TN tell us when the classifier is getting things right, while FP and FN tell us when the classifier is getting things wrong. Table 14.2 shows the confusion matrix table used for this study.

- True positives (TP): These refer to the positive tuples that were correctly labeled by the classifier. Let TP be the number of true positives [51].
- True negatives (TN): These are the negative tuples that were correctly labeled by the classifier. Let TN be the number of true negatives [51].
- False positives (FP): These are the negative tuples that were incorrectly labeled as positive. Let FP be the number of false positives [51].
- False negatives (FN): These are the positive tuples that were mislabeled as negative. Let FN be the number of false negatives [51].

Confusion Matrix Computed for the Study

a	b	<- classified as
131	30	a = No
37	101	b = Yes

The above is the confusion matrix of the final analysis, which calculates the actual and predicted classification. This means that for:

- Class a = (No), the total number of true negatives = 131 and the total number of false positives = 30
- Class b = (Yes), the total number of false negatives = 37 and the total number of true positives = 101

14.4.7 Discussion

Table 14.3 shows the simulation result evaluation summary table when k-fold (where $k = 10$) cross-validation method by which 90% of the data as the training dataset and 10% of the data as the testing set was used to evaluate the accuracy of the Naïve Bayes classifier, and the result accuracy obtained from this was 76%. A final

Table 14.3 Simulation result evaluation summary table

Cross validation (tenfold) accuracy	Final analysis Accuracy (actual accuracy)	Confusion matrix for final analysis
76%	77%	Correctly classified instances = 232 Incorrectly classified instances = 67 Total number of instances = 299

analysis on the entire dataset, using 100% of the instances as a training set, was carried out and the accuracy obtained was 77%. This serves as the actual accuracy obtained for this classification.

From the confusion matrix of the final analysis, out of 161 negative instances, the Naïve Bayes classifier correctly classified 131 as negative but incorrectly classified 30 as positive. Out of 138 positive instances, the Naïve Bayes classifier correctly classified 101 as positive but incorrectly classified 37 as negative. This also means that out of 299 instances, the Naïve Bayes classifier correctly classified 232 instances but incorrectly classified 67 instances.

14.5 Conclusion and Future Works

This study presents a Naïve Bayes classifier for predicting heart disease. The goal of the research was to develop an analytics model for analyzing medical data. The objectives to realize this was: to review existing data mining techniques for analyzing data, to design the Naïve Bayes classifier model for predicting heart disease and to evaluate the performance of the classifier using cross-validation and confusion matrix evaluation technique. The methodology adopted was, firstly, the Naïve Bayes classifier model was used as the classification model, and secondly, WEKA 3.8 was used as the simulation tool. The results of the study show that when k -fold (where $k = 10$) cross-validation method by which 90% of the data as the training dataset and 10% of the data as the testing set, was used to evaluate the accuracy of the Naïve Bayes classifier, the result accuracy obtained from this was 76%. Using the entire heart disease dataset as a training set, the result accuracy obtained was 77%. This serves as the actual accuracy obtained for this classification. From the confusion matrix of the final analysis, out of 299 instances, the Naïve Bayes classifier correctly classified 232 instances but incorrectly classified 67 instances. Classifying heart disease dataset using a Naïve Bayes classifier model shows that the Naïve Bayes classifier model is efficient and adequate for the classification of data. In future work, additional experiments can be performed with more heart disease attributes. Also, a comparison study of other classification models can be carried out.

References

1. M.A. Jabbar, Heart disease prediction system using associative classification and genetic algorithm. *ICECIT* **1**, 183–192 (2012)
2. M.A. Jabbar, et al., Prediction of heart disease using random forest and feature subset selection. *AISC SPRINGER* **424**, 187–196 (2015)
3. T.O. Oladele, R.O. Ogundokun, A.A. Kayode, A.A. Adegun, M.O. Adebiyi, Application of data mining algorithms for feature selection and prediction of diabetic retinopathy, in *International Conference on Computational Science and Its Applications*, (Springer, Cham, 2019, July), pp. 716–730
4. M.A. Jabbar, Classification of heart disease using artificial neural network and feature subset selection. *Glob. J. Comput. Sci. Technol.* **13**(3), 15–25 (2013)
5. M. Learning, Heart disease diagnosis and prediction using machine learning and data mining techniques: A review. *Adv. Comput. Sci. Technol.* **10**(7), 2137–2159 (2017)
6. V. Krishnaiah, G. Narsimha, N.C. Subhash, Heart disease prediction system using data mining techniques and intelligent fuzzy approach: A review. *Int. J. Comput. Appl.* **136**(2), 43–51 (2016)
7. H. Guizhou, M.M. Root, Building prediction models for coronary heart disease by synthesizing multiple longitudinal research findings. *Eur. J. Cardiovasc. Prev. Rehabil.* **12**(5), 459–464 (2005)
8. H. Almarabeh, E. Amer, A study of data mining techniques accuracy for healthcare. *Int. J. Comput. Appl.* **168**(3), 12–17 (2017)
9. R.O. Ogundokun, J.B. Awotunde, Machine learning prediction for COVID-19 pandemic in India. *medRxiv* (2020)
10. A.A. Adegun, R.O. Ogundokun, M.O. Adebiyi, E.O. Asani, CAD-based machine learning project for reducing human-factor-related errors in medical image analysis, in *Handbook of Research on the Role of Human Factors in IT Project Management*, (IGI Global, 2020), pp. 164–172
11. A.A. Adeyinka, M.O. Adebiyi, N.O. Akande, R.O. Ogundokun, A.A. Kayode, T.O. Oladele, A deep convolutional encoder-decoder architecture for retinal blood vessels segmentation, in *International Conference on Computational Science and Its Applications*, (Springer, Cham, 2019, July), pp. 180–189
12. J.B. Awotunde, R.O. Ogundokun, F.E. Ayo, O.E. Matiluko, Speech segregation in background noise based on deep learning. *IEEE Access* **8**, 169568–169575 (2020). <https://doi.org/10.1109/ACCESS.2020.3024077>
13. A.A. Adegun, N.O. Akande, R.O. Ogundokun, E.O. Asani, Image segmentation and classification of large-scale satellite imagery for land use: A review of the state of the arts. *Int. J. Civ. Eng. Technol.* **9**(11), 1534–1541 (2018)
14. M.O. Adebiyi, R.O. Ogundokun, A.A. Abokhai, Machine learning-Based predictive farmland optimization and crop monitoring system. *Scientifica* **2020**, 1–12 (2020)
15. F.E. Ayo, J.B. Awotunde, R.O. Ogundokun, S.O. Folorunso, A.O. Adekunle, A decision support system for multi-target disease diagnosis: A bioinformatics approach. *Heliyon* **6**(3), e03657 (2020)
16. M.O. Adebiyi, E.B. Adigun, R.O. Ogundokun, A.E. Adeniyi, P. Ayegba, O.O. Oladipupo, Semantics-based clustering approach for similar research area detection. *TELKOMNIKA* **18**(4), 1874–1883 (2020)
17. T.O. Oladele, R.O. Ogundokun, J.B. Awotunde, M.O. Adebiyi, J.K. Adeniyi, Diagmal: A malaria coactive neuro-fuzzy expert system, in *Computational Science and Its Applications–ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020, Proceedings, Part VI 20*, (Springer International Publishing, 2020), pp. 428–441
18. F.E. Ayo, R.O. Ogundokun, J.B. Awotunde, M.O. Adebiyi, A.E. Adeniyi, Severe acne skin disease: A fuzzy-based method for diagnosis, in *International Conference on Computational Science and Its Applications*, (Springer, Cham, 2020, July), pp. 320–334

19. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **29**(7), 24 (2007)
20. M.R. Abdmeziem, D. Tandjaoui, I. Romdhani, Architecting the internet of things: State of the art. *Studies in systems. Decis. Control* **36**, 55–76 (2016)
21. A. Kazeem Moses, A. Joseph Bamidele, O. Roseline Oluwaseun, S. Misra, A. Abidemi Emmanuel, Applicability of MMRR load balancing algorithm in cloud computing. *Int. J. Comput. Math.: Comput. Syst. Theory* **6**(1), 7–20 (2021)
22. S. Hakak, S.A. Latif, G. Amin, A review on mobile cloud computing and issues in it. *Int. J. Comput. Appl. Technol.* **75**(11), 1–4 (2013)
23. M. Aazam, E.-N. Huh, Fog computing and smart gateway based communication for a cloud of things, in *The Proceedings of IEEE Future Internet of Things and Cloud (FiCloud)*, (Barcelona, Spain, 2014), pp. 27–29
24. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **29**(7), 1645–1660 (2013)
25. M. Aazam, E.-N. Huh, M. St-Hilaire, C.-H. Lung, I. Lambadaris, Cloud of things: Integration of IoT with cloud computing. *Architecting the internet of things: State of the art. Stud. Syst. Decis. Control* **36**, 77–94 (2016)
26. I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, M. Imran, Heterogeneity-aware task allocation in mobile ad hoc cloud. *IEEE Access* **5**(2017), 1779–1795 (2017)
27. M. Odusami, O. Abayomi-Alli, S. Misra, O. Shobayo, R. Damasevicius, R. Maskeliūnas, Android malware detection: A survey, in *International Conference on Applied Informatics*, (Springer, Cham, 2018), pp. 255–266
28. S.O. Abdulsalam, A.A. Mohammed, J.F. Ajao, R.S. Babatunde, R.O. Ogundokun, C.T. Nnodim, M.O. Arowolo, Performance evaluation of ANOVA and RFE algorithms for classifying microarray dataset using SVM. *Lect. Notes Bus. Inf. Process.* **402**, 480–492 (2020)
29. R.K. Behera, S. Shukla, S.K. Rath, S. Misra, Software reliability assessment using machine learning technique, in *International Conference on Computational Science and Its Applications*, (Springer, Cham, 2018, May), pp. 403–411
30. A.P. Ikedinachi, S. Misra, P.A. Assibong, E.F. Olu-Owolabi, R. Maskeliūnas, R. Damasevicius, Artificial intelligence, smart classrooms and online education in the 21st century: Implications for human development. *J Cases Inf. Technol.* **21**(3), 66–79 (2019)
31. V. Alagbe, S.I. Popoola, A.A. Atayero, B. Adebisi, R.O. Abolade, S. Misra, Artificial intelligence techniques for electrical load forecasting in smart and connected communities, in *International Conference on Computational Science and Its Applications*, (Springer, Cham, 2019), pp. 219–230
32. E. Okewu, S. Misra, J. Okewu, R. Damaševičius, R. Maskeliūnas, An intelligent advisory system to support managerial decisions for a social safety net. *Adm. Sci.* **9**(3), 55 (2019)
33. H. Xu, *Machine Learning-Based Data Analytics for IoT Devices* (Nanyang Technological University, 2017). <https://doi.org/10.32657/10356/72342>
34. C. Ieracitano, N. Mammone, A. Hussain, F.C. Morabito, A novel multi-modal machine learning-based approach for automatic classification of EEG recordings in dementia. *Neural Netw.* **123**, 176–190 (2020). <https://doi.org/10.1016/j.neunet.2019.12.006>
35. A. Panesar, *Machine Learning Algorithms* (Apress, Berkeley, 2021), pp. 85–144. https://doi.org/10.1007/978-1-4842-6537-6_4
36. W. Guan, A. Gray, S. Leyffer, In mixed-integer support vector machine, in *Mini Symposia & Workshops NIPS*, (2013), pp. 1–6
37. S. Shilaskar, A. Ghatol, Feature selection for medical diagnosis: Evaluation for cardiovascular diseases. *Expert Syst. Appl.* **40**, 4146–4153 (2013)
38. Y.E. Shao, C.D. Hou, C.C. Chiu, Hybrid intelligent modeling schemes for heart disease classification. *Appl. Soft Comput.* **14**, 47–52 (2014)
39. S.U. Kumar, H.H. Inbarani, A novel neighborhood rough set-based classification approach for medical diagnosis. *Procedia Comput. Sci.* **47**, 351–359 (2015)

40. S. Rajathi, G. Radhamani, Prediction and analysis of Rheumatic heart disease using KNN classification with ACO, in *International Conference on Data Mining and Advanced Computing (SAPIENCE)*, (Ernakulam, 2016), pp. 68–73
41. S. Bashir, U. Qamar, M.Y. Javed, An ensemble-based decision support framework for intelligent heart disease diagnosis, in *Information Society International Conference*, (IEEE, 2014), pp. 259–264
42. S.U. Amin, K. Agarwal, R. Beg, Genetic neural network-based data mining in the prediction of heart disease using risk factors, in *Information and Communication Technologies (ICT)*, (IEEE, 2013), pp. 1227–1231
43. V. Khatibi, G.A. Montazer, A fuzzy-evidential hybrid inference engine for coronary heart disease risk assessment. *Expert Syst. Appl.* **37**, 8536–8542 (2010)
44. F. Temurtas, A.C. Tanrikulu, An approach on probabilistic neural network for diagnosis of mesothelioma's disease. *Comput. Electr. Eng.* **38**, 75–81 (2012)
45. N. Yumusak, F. Temurtas, Chest diseases diagnosis using artificial neural networks. *Expert Syst. Appl.* **37**, 7648–7655 (2010)
46. X. Liu, D. Tosun, M.W. Weiner, N. Schuff, Locally linear embedding for MRI based Alzheimer's disease classification. *NeuroImage* **83**, 148–157 (2013)
47. S. Nashif, M.R. Raihan, M.R. Islam, M.H. Imam, Heart disease detection using machine learning algorithms and a real-time cardiovascular health monitoring system. *World J. Eng. Technol.* **6**, 854–873 (2018)
48. M. Singh, L.M. Martins, P. Joanis, V.K. Mago, Building a cardiovascular disease predictive model using structural equation model and fuzzy cognitive map, in *IEEE International Conference on Fuzzy Systems (FUZZ)*, (Vancouver, 2016, July 24–29), pp. 1377–1382
49. P. Ghadge, V. Girme, K. Kokane, P. Deshmukh, Intelligent heart attack prediction system using Big data. *Int. J. Recent Res. Math. Comput. Sci. Inf. Technol.* **2**, 73–77 (2016)
50. X. Liu, H. Zhu, R. Lu, H. Li, Efficient privacy-preserving online medical primary diagnosis scheme on naive Bayesian classification. *Peer-to-Peer Netw. Appl.* **11**(2), 334–347 (2018, March)
51. H. Han, B. Gu, J. Kang, Z.R. Li, Study on a hybrid SVM model for chiller FDD applications. *Appl. Therm. Eng.* **31**(4), 582–592 (2011)

Chapter 15

The Internet of Things in Healthcare: Benefits, Use Cases, and Major Evolutions



Raj Shree, Ashwani Kant Shukla, Ravi Prakash Pandey, Vivek Shukla,
and K. V. Arya

Contents

15.1	Introduction	388
15.2	Related Work	389
15.3	IoT: Transforming Healthcare Industry	392
15.3.1	Delivering Healthcare Solutions	392
15.3.2	Salient Steps for Transforming the Healthcare Industry	394
15.4	Promising Use Cases of IoT in Healthcare	395
15.4.1	Reachability for Remote Patient Care	395
15.4.2	Emergency Care for Patients	396
15.4.3	Healthcare Management	396
15.4.4	Augmenting Surgeries	396
15.4.5	Risk-Based Hardware Management	397
15.4.6	Drugs Management	397
15.4.7	Wearables	397
15.5	State of IoT in Healthcare	398
15.6	Benefits of IoT in Healthcare	401
15.6.1	Cost-Effective	401
15.6.2	Quality-of-Treatment-Service	401
15.6.3	Efficient-Diagnosis-System	401
15.6.4	Action-Based Treatment	402
15.6.5	Medical Resource Management	402
15.6.6	Error Reduction	402

R. Shree · A. K. Shukla · V. Shukla
Babasaheb Bhimrao Ambedkar University, Lucknow, India

R. P. Pandey
Dr. Rammanohar Lohia Avadh University, Ayodhya, India

K. V. Arya (✉)
ABV-Indian Institute of Information Technology & Management, Gwalior, India
e-mail: kvary@iitm.ac.in

15.7	Challengers of Healthcare IoT Deployments	402
15.7.1	Reliable Connectivity	402
15.7.2	Cybersecurity	403
15.7.3	Scalable Platforms	403
15.7.4	Cost	403
15.8	Conclusion	403
	References	404

15.1 Introduction

Internet of Things (IoT) is an emerging technology with objectives that adhere the globe through intelligent devices that have greater capabilities to gather and send information at anytime, anywhere, and compatible with other devices or framework. It provides less redundant data to all adhering devices in the network and ensures IoT consumers to access healthcare facilities securely. Also, the smart healthcare framework helps the consumers to access the data related to healthcare quickly and precisely [1]. Moreover, the use of smart IoT technology has enhanced patient satisfaction and quality of service by informally interacting with doctors as the reachability of the patients to doctor or medical staffs from a remote location becomes quite easy by avoiding the frequent visits for health monitoring. For patients, the financial issues will always be the challenging stuff whenever it comes the scenario of healthcare, but the IoT has made it possible to provide all the major needed facilities in a very efficient manner and at a cheaper cost. In several fields of healthcare, the IoT technology has become a boon through which the reachability of the healthcare-related facilities such as health monitoring and transmission of real-time warnings become possible in remote locations. It is definitely evolutionary for the healthcare industry with the enhancement of smart devices and effective use to facilitate the patients [2].

The emerging and greater advancement in communication and computer technologies, the global market related to the IoT-based environment taking into considerations. IoT itself will not be able to enhance the efficiency of the smart hospitals until and unless the IoT-based technology is being used to determine the success of the model [3]. Many IoT-based technologies such as portable devices, Internet, personal hotspot BYOD, and several other healthcare-detecting techniques have been integrated through which the dream of Internet of Medical Things has become possible. Apart from that, the digital and radio frequency related detection methods greatly enhance the robustness of the healthcare frameworks. Moreover, with the help of the IoT technology, all the resources related to smart healthcare such as patients, medical resources, smart wheelchairs, and other portable devices are made available at a common platform, where the needy people can access the desired facilities ubiquitously. The concept of the smart healthcare industry has developed as patient-centric and healthcare-centric. Both perspectives ensure the patients' safety, quality of service, and services free from medical error [4]. IoT is a still growing field, so the challenges are very common; however, despite all

these hurdles, the intelligent use of the smart devices with precise transmission system, skilled healthcare professionals, and patients are efficiently transmitting the health-related information from one hospital to another. IoT enables health professionals with great opportunities to facilitate the maximum number of patients. In the beginning, the implementation of the IoT-based technology usually needs the industry to pay off the huge volume of assists at different levels. Adversely, the deployment of the smart healthcare industries in developing countries is still a big challenge due to limited availability of the IoT-based technology. Instead of that, there are many other factors to stop the hospitals for getting the financial benefits that enable to adopt latest technological features to facilitate the hospitals. With the effective use of the IoT in the smart hospitals, at least fulfill the necessary facilities to take care with the great solutions to avoid any kinds of failure [5].

15.2 Related Work

Tyagi et al. [6] have developed the models which are known as Framework-as-a-Service (FaaS) and Infrastructure as a Service (IaaS) and implemented a cloud IoT-based healthcare prototype which efficiently helps patients to find cost-effective and easy healthcare solutions. It allows the patients to safely save and transfer their health-related information to the hospitals. Additionally, the proposed framework provides the facilities wherein patients can make their self-assessment and monitor health-related issues and framework guide to find the relevant hospitals as per need. However, the primary use of the cloud-based IoT-driven solutions in healthcare industries is to give precise security to prevent loss of data. In addition, the implementation of this model remains safety to meet the requirements and necessary to test its consequences. An automated supervised decision support system which is called the ICADS, which stands for Intelligent Context-Aware Decision Support has been developed [7]. It has a greater potential to make a robust foundation which assists to reform and prioritize the least necessary facilities which increases the performance of the medical staff, current updates related to the patient's health, proactive plan for emergency assistance, and enhance the QoS. Instead of that, the proposed framework offers promising advantages for the stakeholders of the healthcare organizations. The compatibilities of the systems as per hospital settings requirements have obligations to be managed prior to implementing the proposed framework.

A model which is known as the machine-to-machine (M3) framework is implemented by Gyrard et al. [8]. It empowers to facilitate the reachability of the necessary medical care especially for the patients residing in remote location. Additionally, it encompasses with the wearable devices that collect patient-related records and transfer to smart phones which function as intervening portals and finally transfer to cloud-based MIS to ensure the security concern from source to destination. For managing the patient-related records, the cloud-based MIS assists to provide an efficient management system. Nevertheless, the sensitive health

information of the patients which is generated by the portable devices does not consent by the proposed framework.

A prototype that advances the utilization of IoT-based emerging technology in the urban needy society is developed [9]. The capability of the developed method is to identify the five different sources of advancement of the IoT technology such as cost, health education, nutrition, healthcare, and the mode of employment. It positively affects the urban needies by delivering the access to different facilities such as health-orientation program, healthcare training, and security for food. The main considerations regarding the implemented model facilitate the quality of service to users and experience tangible benefits. These components aid service providers to provide the best services to users and subsequently give maximum user satisfaction. In [10], Jagatheesan et al. proposed a model based on the different healthcare service providers and different user model. It defines the number of interactions to a model which enables the consumer to manage the services facilitated from the proposed model. It encompasses the prospective of the various IoT-driven solutions that is utilized by both healthcare service providers and patients. Nevertheless, it does not satisfy the patients' necessary IoT requirements. Also, due to the data security issues, it restricts from transferring the patients' data in real time and other facilities.

The IoT communication model was developed in [11] as the primary enabler for distributed health applications worldwide. The model has been developed with the primary objective to monitor the patient, physician, and the distributed information database. The major concern is to assist the development of the robust IoT-driven solution in healthcare. Additionally, it features significantly by employing multiple techniques, models, and methods to monitor the level of diabetes. The proposed model is still under the development process; the factors showed the applications in the various phases of the realization also not recommended in real-time integration to collect the information of the patients.

Manashty et al. [12] developed a cloud-based model called as HEAL which stands for Healthcare Event Aggregation Lab. It is featured with the facility to provide healthcare services such as early obtain the refined and observed symptoms patient information. It has integrated with the software to monitor the high level that helps multiple consumers and systems for long-term analysis. Conclusively, it still needed some advancement to increase the efficiency in different composite perspectives. The fusion-based IoT technique with the integration of the composite incidence process and big data analysis used as a reference model is developed in [13]. To make the smart hospital informational-based ecosystem, the proposed model can be worked as a reference with greater potential. Moreover, due to its robustness and greater characteristics, it gives quite efficient results and resolves various issues. Particularly, the complex event processing can assist to transfer and process the patient's information which is obtained from different service providers in real time by the intelligent use of IoT-driven solutions. Pir et al. [14] proposed the IoT-based smart hospital management system. The framework acknowledgment uses as an interface in IoT framework to solve the big data management related

issues. The basic three layers have integrated with the developed model such as a physical, network, and application layer. The function of the first layer, which is also called the observation layer, is to obtain the patients' information and transfer it to the second layer. Finally, the second layer processes the data and transfers the information to the third layer. The framework gets acknowledged to integrate between the second and third layers, that is, function as an interface to analyze the data and send to the application layer only when required. Subsequently, the third layer explains the efficiency of the framework on the basis of obtained information while the patients and other medical staff interact with the system. Chatterjee and Armentano [15] have identified a number of challenges such as the availability of the secure information in real-time assists to implement the smart medical structured model that provides services efficiently. The proposed method has updated the healthcare-related information redundantly and makes it globally available for all dedicated healthcare providers. Also, the most significant feature is to employ the obtained data which assist the implementation of the intelligent clinical decision support model. Gupta et al. [16] implemented the IoT-driven solution as an emergency healthcare monitoring model. It has defined the feasible information collection, fusion, and integration of the IoT-based technique which precisely assists emergency care. Also, it provides patient-centric solutions by obtaining, processing, and transferring the huge volume of information securely in real time and risk free with cost-effectiveness.

Maghdid et al. [17] have implemented the model which relies on using distinct extent of smart phone sensor and facilitate to the doctors or radiologists who have access of smart phones all the time. The primary characteristic of the developed model is that it is platform independent, as it does not need any add-on sensors that give the extreme precision. Moreover, it consists of four different phases which are considered as input sensors extent phases. By using the hybrid method, it assists the sensors configuration layer, computing symptoms disease layer, and predict the disease layer. Furthermore, in the final phase, the ML framework can be further enhanced by employing the transfer learning approach in case of framework functions on the cloud. The framework is more reliable than the modernistic; due to this, the framework relies on multi-readings from multiple sensors mentioning the associated disease symptoms.

An E-quarantine setup for monitoring the Coronavirus infected patients is developed [18] and employed to minimize the infection and to assist in preventing hospital visit and make it available for all instruments, only for high-risk patients. The significant aim of the E-quarantine setup provide the quarantine for patients in their homes to watch patients and categorize the patients as per the disease risk. The proposed model relies on the five fundamental components such as blood pH level, heart rate, blood pressure, body temperature, and respiratory rate. In the case of the Dempster-Shafer technology, it has obtained the greater potential in indexing data with respect to images or videos. The hybrid mechanism is employed to chain data for patients and their respiratory ventilates. Singh et al. [19] developed a wearable band, *namely*, IoT-Q-Band to detect and track the absconding COVID-19 quarantine patients. In the wearable band components, its functions mode and

batteries are chosen such that it must be continuously ON during the quarantine period; it is also validated through current consumption evaluation. Because of the infection potential, the IoT-Q-Band assists in avoiding reuse. Thus, the cost of the proposed prototype is kept low by reusing several smart phones characteristics. As an affordable cure, the IoT-Q-Band can facilitate with low-income especially in the remote locations of the world where it can play a significant role to track people who are under quarantine. In [20], an IoT-based smart healthcare monitoring and management framework is developed. The proposed framework consists of three fundamental phases such as data obtained from battery-powered medical sensors and processing, Hadoop processing, and application phases. Due to the limited capacity of the battery to power the sensor, the work employed an energy-harvesting approach using piezoelectric devices connected to the human body [14].

To minimize the effects of communicable diseases, an IoT-based framework is proposed by Otoom et al. [21]. Moreover, it is employed to provide health records of COVID-19 cases, inform potential COVID-19 cases and to confirm disease and is augmented to develop a machine-learning based predictive model for the disease. The framework also informs these results to healthcare practitioners, who may rapidly react to suspect cases that identify the model in the future, as well as any clinical investigations needed to confirm. It enables confirmation of isolated cases and facilitate appropriate healthcare. Additionally, the proposed model facilitates the real-time framework which potentially minimizes the impact of communicable diseases, as well as reduce the mortality rate through early detection of cases.

15.3 IoT: Transforming Healthcare Industry

15.3.1 *Delivering Healthcare Solutions*

The IoT-driven solutions which efficiently and greatly transform the healthcare industries as follows [22]:

15.3.1.1 IoT for Patients

The medical-oriented IoT-based portable device is incorporated to the smart hospitals that assist precisely and timely regarding the treatment of the patients such as measuring the blood pressure and monitoring of the heart pulse, glucometers, etc. The IoT-driven solutions' reachability takes a place of human minds to make alerts or sense in terms of calorie counts, therapy, doctor appointments, changes in the blood pressure, etc. The most significant roles of the IoT-based smart hospitals make the reachability of medical resources at the remote locations to take care of the patients in real time especially elderly patients possible. Moreover, the IoT has resolved the many challenges in the healthcare industry like those persons or

families who are residing alone at home and getting the best medical facilities and treatment by sitting at home, anytime and from anywhere.

15.3.1.2 IoT for Doctors

The IoT-driven solutions are not only patients-centric but also doctors-centric. They assist in providing efficient and hassle-free services to patients such as online appointments, prescriptions, and health monitoring. Moreover, it is quite effective for the medical staff to provide transferable and transparent services to the doctors and patients. The process of collection of the patient-related information for doctors to see the changes and consequences after giving the treatment becomes too easy by implementing the IoT-based solutions.

15.3.1.3 IoT for Hospitals

The journey of hospitals to become a smart hospital is very much appreciable due to the IoT-driven solutions. In a very short time, the IoT has become very popular in the healthcare industry due to its brilliance and it plays a very significant role as an application in the maximum part of the hospitals as well as to precisely deal with the medical resources such as wheelchairs, defibrillators, nebulizers, oxygen-based devices, and monitoring devices. Additionally, it aids with asset management such as drug stock control and healthcare framework monitoring such as cooling system of the refrigerator and moisture for preventing the vaccine.

15.3.1.4 IoT for Health Insurance Organizations

Due to the greater implementation of the IoT in healthcare, there are many opportunities for the health insurance organizations to provide better transparent services to the patients regarding the easy processing of claims on time. The obtained data facilitate to avoid fraudulent activities regarding the claims. Moreover, with the help of the IoT devices, transparency has become possible between consumers and service providers in financing, valuing, process of handling the claims and risk identification. It assists to complete the entire process. The data obtained from the IoT devices play a very significant role to process the trustworthy claims. As a result, the consumers have enough prospects into the fundamental model after each concerned decision and obtained processes. The four phases which are a constituent part of a typical IoT-driven system are depicted in Fig. 15.1.



Fig. 15.1 The typical four phases of the IoT-driven solutions

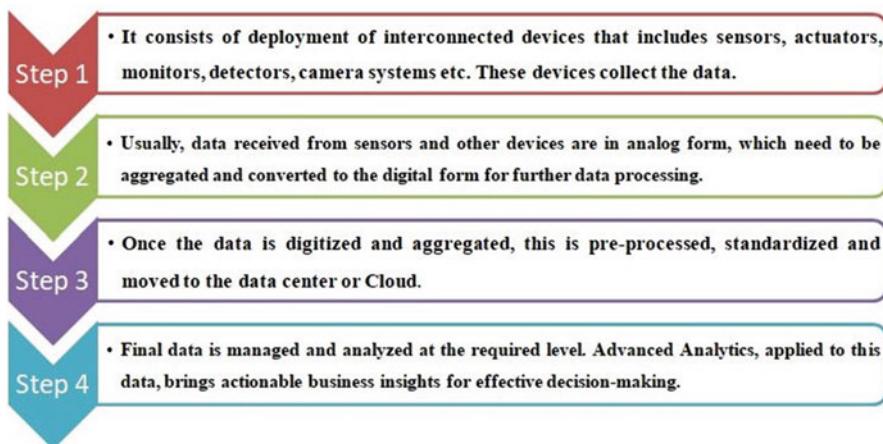


Fig. 15.2 Four-Step architecture of IOT

15.3.2 Salient Steps for Transforming the Healthcare Industry

Due to great intelligence the IoT-driven solutions it gets expanded in healthcare industry and make great opportunities and has the promising ability to transform the big data brought in by connected devices in healthcare industries.

All four steps are connected in such a way that the data driven or processed in every single phase get the value as an input for the next step. All the four stages have been represented in Fig. 15.2.

IoT needs to be defined in such a way that the healthcare industries gain the trust of the consumers and are able to provide the best quality of the services cum treatment at a cheaper cost as of now. Also, ensure continuous enhancement in workflows and efficiency and have a strong belief in IoT-driven solution provided by the healthcare industry [23].

15.4 Promising Use Cases of IoT in Healthcare

The IoT-driven solutions have greatly drawn the attention of the healthcare service providers and patients because their characteristics and applications have distinctly spread out globally [24]. A typical smart hospital equipped with IoT system is shown in Fig. 15.3.

15.4.1 Reachability for Remote Patient Care

Globally, the reachability to the hospitals is a bit hard for patients, especially in case of emergency. Likewise, for the healthcare providers, the major challenge is to provide instant solutions in case of emergency; it has become quite a hard task. But the problem gets efficiently resolved by the IoT-driven solution and makes it reachable for the remotely located patients. The collaborations of the

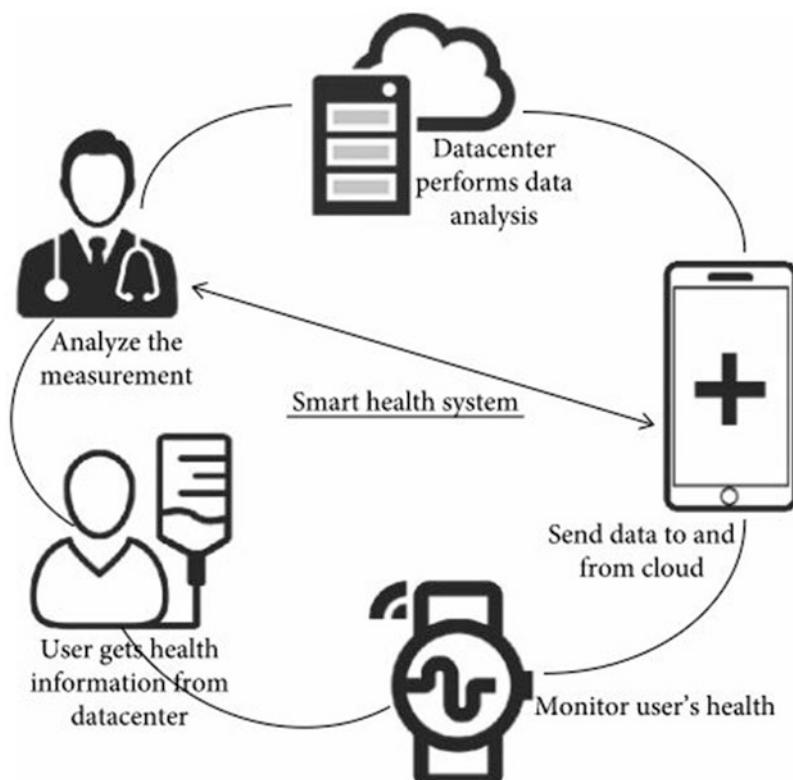


Fig. 15.3 Smart hospital

IoT-driven solutions with the healthcare industry become helpful for the medical staff to take care the patients efficiently and provide the best service on time such as digital appointments, prescriptions, drugs management, and monitoring remote devices, for example, patients are able to utilize the medical resources with the help of the portable device to the cloud-oriented IoT in real-time patient information management.

15.4.2 Emergency Care for Patients

The primary necessity of the healthcare resources is to make it available in emergency cases for preventing the patients from the danger zone. The consequences of the emergency care always rely on the availability of precise admissible information on time. Furthermore, the collecting, saving, processing, and retrieving of data is a time taking process to obtain any results without implementation of IoT. Nowadays, smart hospitals have a greater potential to gather efficient patient-related information for better care and suggest the medical staff regarding immediate care on a priority basis. The obtained information is transmitted in real time to the ER staff for proactive treatment while the patient is supposed to reach the hospital anytime.

15.4.3 Healthcare Management

Currently, the healthcare industry is strongly enhancing the medical facilities and is trying to provide services at a very cheaper cost; this can be possible only by implementing the IoT-driven solutions. The maximum expenses of the hospitals go to manage the hospital-related records, and these expenses get reduced by using IoT-driven solutions which assist in the effective management of the hospital records and make it possible to access the data anytime from anywhere by ensuring data security and being able to easily evaluate the performance of employees. The best and cheapest IoT techniques such as the BLE and RFID effectively manage the inventory and it becomes quite easy to assign medical staff in case of emergency. Simultaneously, the IoT-driven real-time location frameworks assist to facilitate asset monitoring. Moreover, it becomes very precise to manage the medical resources and make it available at a cheaper cost through which more patients get benefitted.

15.4.4 Augmenting Surgeries

As far as healthcare nowadays is concerned, the IoT has also entered the operation theatre and helps in performing various surgeries successfully. These operations are

just a matter of precision which is brought in through supervised-robot surgeons. In the pre and post phases of patient care, the IoT-driven solution works significantly to gather, transfer, and evaluate the patients' records. Also, it assists to determine the difficulty levels of the surgery and make an effective plane to perform the same.

15.4.5 Risk-Based Hardware Management

The IoT-based solution is an integration of the hardware and software which leads to the risk. So, it becomes necessary to pay extra attention to identify risks in the early stages and precisely manage all the risks on time to avoid interruptions and failure of the IoT-based framework. The major challenges faced by healthcare industries earlier include power disruptions, system failures, and cyber attacks, but IoT-driven solutions resolve all these issues and give quite efficient results. If there is a discrepancy in an instrument, the hospital staff get alerts which helps to avoid failure by taking preventive measures.

15.4.6 Drugs Management

The drugs-based business is several million dollars and very complex in operation. There are many stages in the hospital, from drugs storage to management, which are always challenging tasks in healthcare. Presently, all of these issues can be resolved by IoT-driven solutions. Also, it assists to integrate the efficient preventive measure and ensure the best drugs management and patient treatments.

15.4.7 Wearables

The wearables IoT technology assists to care staff that can collect different samples as per the needs of the diagnosis such as sleeping norms, blood pressure, and heart rate. It provides the facilities to observe the information of the patients in real time. In case of an emergency, the portable IoT devices quickly inform the caretakers to handle the patients with efficient care. It is quite efficient for the patients, especially for elderly patients who are residing in a remote location and who need continuous monitoring.

With greater intelligence and characteristics, it has drawn the attention of healthcare service providers as well as patients which assist in determining the organizational restructure capacity and care of the patient. Globally, the IoT-based solution in healthcare is appreciated and renounced.

15.5 State of IoT in Healthcare

In 2019, 87% of healthcare industries have adopted the Internet of Things technology and 76% believed to transform the healthcare industry [25]. The state-of-the-art scenario for use of IoT, benefits of IoT, available IoT technologies, and potential threats from IoT is shown in Figs. 15.4, 15.5, 15.6, and 15.7.

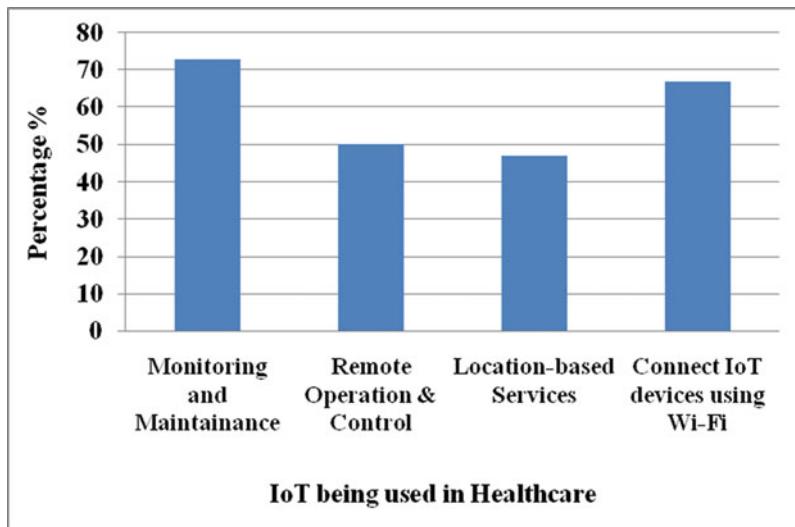


Fig. 15.4 Graphical representation of the IoT being used in healthcare

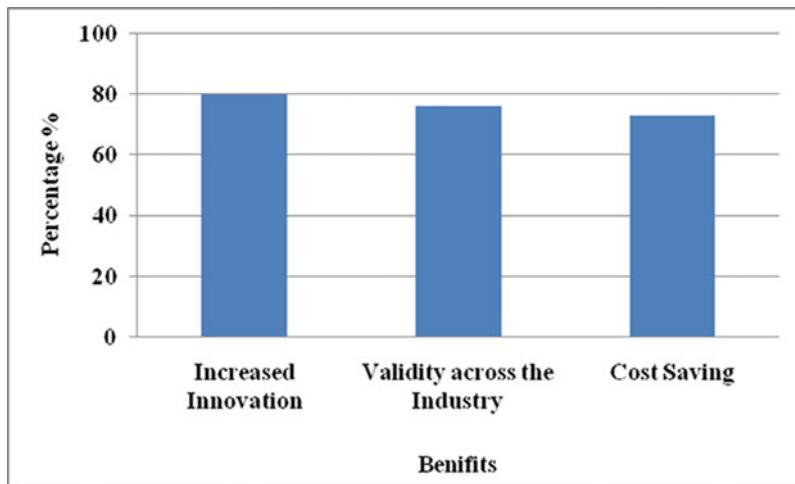


Fig. 15.5 Graphical representation of the benefits from IoT

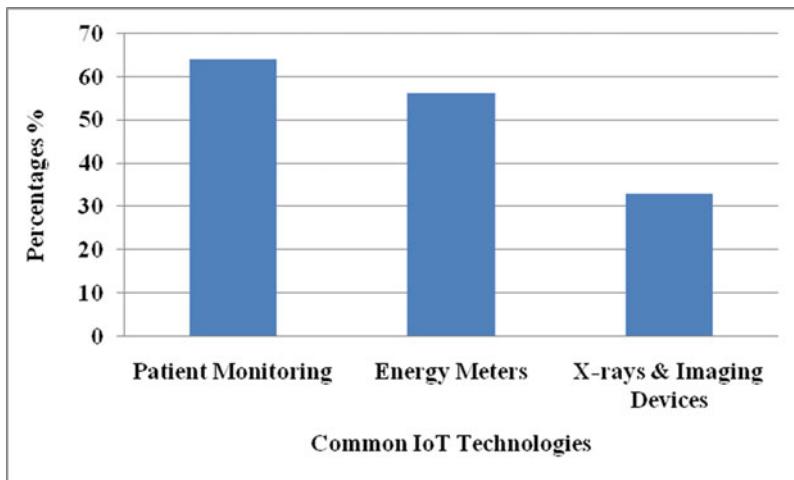


Fig. 15.6 Graphical representation of the most common IoT technologies

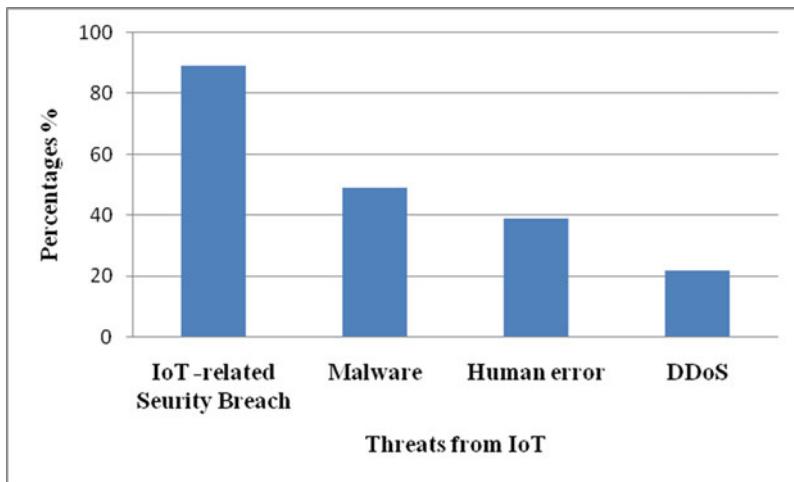


Fig. 15.7 Graphical representation of the threats from IoT

Figure 15.8 graphically represents the number of Internet of Things (IoT) active connections in healthcare in the European Union (EU) in 2016, 2019, 2022, and 2025. The number of IoT healthcare active connections was expected to increase through the years. It was at 0.87 million connections in 2016, and it is expected to reach 10.34 million connections by 2025.

Figure 15.9 is a graphical representation of the IoT-driven solution to transform the landscape of the healthcare industry which incorporates the IoT techniques and patient monitoring framework from remote locations. As of 2018, North America

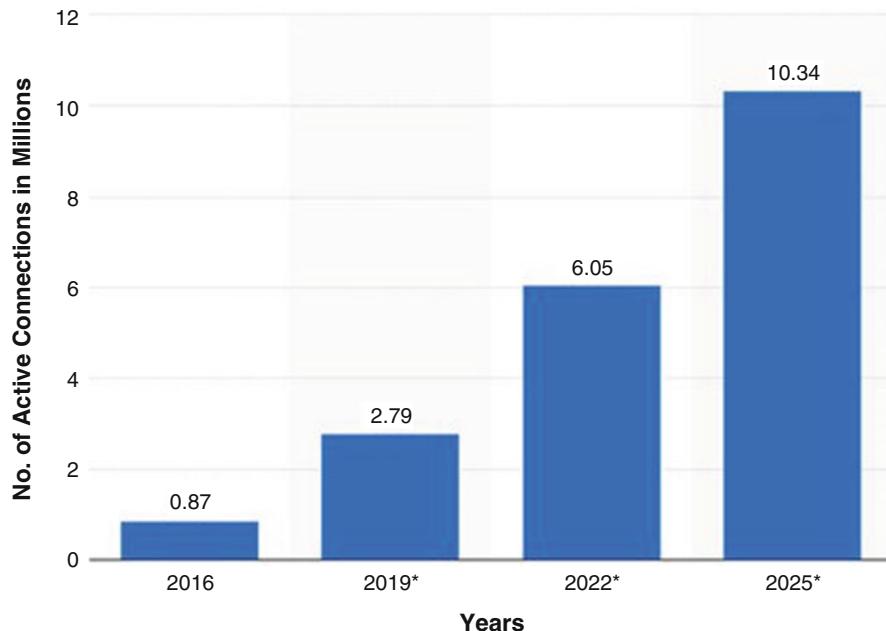


Fig. 15.8 Graphical representation of the year-wise IoT-driven solutions in healthcare in the European Union

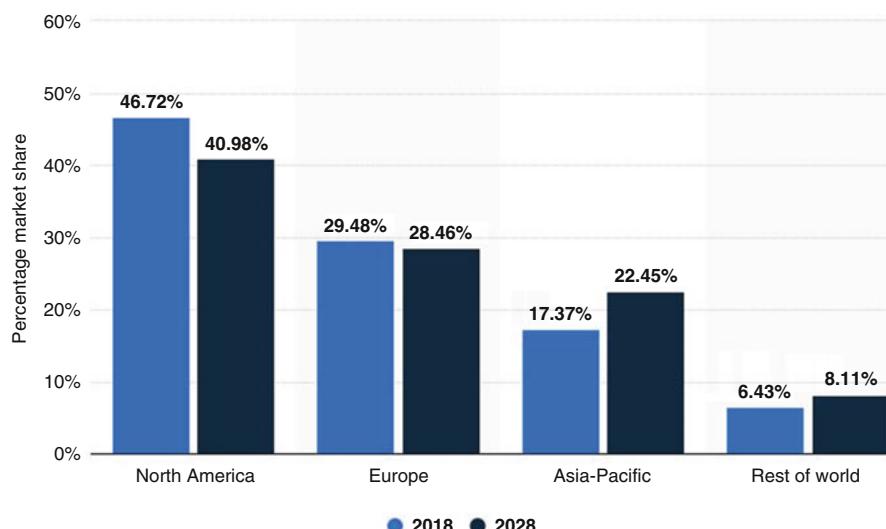


Fig. 15.9 Graphical representation of the global investment in healthcare IoT-driven security in 2018 and 2028



Fig. 15.10 The global smart healthcare market by 2020

held the highest share of the healthcare IoT security market with about 47% of the market. By 2028, the Asia-Pacific region is expected to increase their hold in the market from about 17–22% (Fig. 15.10).

15.6 Benefits of IoT in Healthcare

The benefits of the IoT in healthcare are as follows [26]:

15.6.1 *Cost-Effective*

The IoT-driven solutions make real-time patient monitoring frameworks possible which leads to reduction in unnecessary visits to doctors, hospital stays, and admissions.

15.6.2 *Quality-of-Treatment-Service*

It facilitates practitioners to make evidence-based informed decisions and a complete transparent system.

15.6.3 *Efficient-Diagnosis-System*

Often, the real-time patient monitoring system data assist to diagnose the diseases in the early stages or before symptoms develop based on symptoms.

15.6.4 Action-Based Treatment

The continuous health monitoring system becomes possible by the implementation of the IoT-driven solutions especially in remote locations which leads to action-based treatment.

15.6.5 Medical Resource Management

The medical resource management was quite a challenging task in the healthcare industry but the IoT-driven solutions have greatly enhanced the medical resource management at a cheaper cost.

15.6.6 Error Reduction

The data obtained from IoT-based framework are not only helpful in taking a precise decision, but also provide the best diagnosis and error-free treatment. As it is a growing field, the IoT-based framework faces a few challenges but provides lot of benefits. An IoT-based incorporated system collects great volume of data including sensitive information where data breaches are a very common concern. But, the newly developed IoT-driven solution has a greater potential to meet the security criteria and most significantly to gain the trust of the consumers.

15.7 Challengers of Healthcare IoT Deployments

The healthcare industry is a very concerning and challenging field from the prospective of the IoT-based technology because its pros and cons directly affect humans. Conclusively, due to the high demand in the healthcare sector, a robust framework is still needed to provide the best treatment for the patients. For a successful framework in the healthcare IoT, the following criteria must be fulfilling [27].

15.7.1 Reliable Connectivity

Network failures are not acceptable in connected medical devices that require real-time access or monitoring the patient's data. Moreover, managing the connectivity is a specific issue in mobile devices such as wearables, where the patient travels

anywhere across borders and coverage areas. The cellular-based connection connectivity is often the best solution for IoT deployment covering a huge geographic area. With an open roaming, the IoT devices automatically switch between networks and stay connected to the strongest available signal. The networks that can be accessed rely on the roaming service provider and deployment location of the SIM provider.

15.7.2 Cybersecurity

The landscape of the technology is transforming every day due to its advancement and security issues have become more challenging nowadays especially in the healthcare sector because of sensitive information. So, healthcare technology needs layer-wise security to prevent the patient privacy from cyberattacks.

15.7.3 Scalable Platforms

For making a robust and successful IoT-based healthcare infrastructure, it must be supported and combined with more than one healthcare organization. Patients, doctors, and other authorized professionals must be able to use the equipment, monitor their condition, and troubleshoot it remotely. Conclusively, the IoT-driven solutions must be a scalable, flexible and user-friendly IoT platform that is suited to specific use cases, preferably with strong support staff to help with solution design and to ensure smooth functions.

15.7.4 Cost

The cost plays a very significant role in IoT-driven solutions due to extreme demands in the healthcare sector. IoT-based development can immediately become prohibitive, especially in a competitive environment where various departments have to stake their claims for limited funds.

15.8 Conclusion

Unquestionably, the upcoming trends of the IoT significantly transform the scenario of the healthcare industry with greater capabilities and modernizations. Due to the characteristics of the IoT such as cost-effective, precisions, and robustness, most of the healthcare industry all around the globe has adopted the IoT-driven solutions. Moreover, the renounced healthcare industry as well as small clinics showed a

greater amount of interest by utilizing the benefits of the IoT-based system. The IoT is still a growing field especially in developing countries, but its drastic growth with great potential and robustness has resolved the maximum number of issues to make the reachability of healthcare-related facilities available to the common citizen, especially those residing in remote locations possible. In the future, the landscape of the IoT-driven solution plays a greater role in healthcare with a huge amount of medical facilities available at a very cheaper cost and it makes the efficient reachability to the common citizen possible. The entire healthcare industry is working towards providing the best equipped hassle-free smart hospitals by implementing the IoT-driven framework where doctors, other medical staff, and patients can save time and cost.

References

1. H. Zhang, B.T. Han, Z. Tang, Constructing a nationwide interoperable health information system in China: The case study of Sichuan Province. *Health Policy Technol.* **6**(142151), 142–151 (2017)
2. A.M. Kadhum, M.K. Hasan, Assessing the determinants of cloud computing services for utilizing health information systems: A case study. *Int. J. Adv. Sci. Eng. Inf. Technol.* **7**(2), 503–510 (2017)
3. C. Pagliari, Health information exchange as a complex and adaptive construct: Scoping review. *J. Innov. Health Inform.* **23**(4), 633–683 (2016)
4. Y. Jog, A. Sharma, K. Mhatre, A. Abhishek, Business approach for IoT based health solutions in India with respect to Osterwalder framework. *Int. J. Bio-Sci. Bio-Technol.* **7**(6), 173–188 (2015)
5. W. Sujansky, D. Kunz, A standard based model for the sharing of patient generated health information with electronic health records. *Pers. Ubiquit. Comput.* **19**(1), 9–25 (2015)
6. S. Tyagi, A. Agarwal, P. Maheshwari, A conceptual framework for IoTbased healthcare system using cloud computing, in *2016 6th International Conference—Cloud System and Big Data Engineering (Confluence)*, (2016), pp. 503–507
7. B. Manate, V.I. Munteanu, T.F. Fortis, P.T. Moore, An intelligent context-aware decision-support system oriented towards healthcare support, in *Complex, Intelligent and Software-Intensive Systems (CISIS): 2014 Eighth International Conference on IEEE*, (2014), pp. 386–391
8. A. Gyrard, S.K. Datta, C. Bonnet, K. Boudaoud, Standardizing generic cross-domain applications in internet of things, in *2014 IEEE Globecom Work (GC Workshops)*, (2014), pp. 589–594
9. A. Roy, A.M.S. Zalzala, A. Kumar, Disruption of things: A model to facilitate adoption of IoT-based innovations by the urban poor. *Procedia Eng.* **159**, 199–209 (2016)
10. A. Jagatheesan, S. Maragathavel, M. Sivapurapu, J. Lee, Drops: A multi-producer and multi-consumer data sharing framework with human experience, in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, (2015), pp. 601–602
11. N. Bui, M. Zorzi, Health care applications: a solution based on the internet of things, in *Int. Symp. Appl. Sci. Biomed. Commun. Technol.*, (2011), pp. 0–4
12. A. Manashty, J. Light, U. Yadav, Healthcare event aggregation lab (HEAL), a knowledge sharing platform for anomaly detection and prediction, in *2015 IEEE 17th International Conference on e-Health Networking, Applications and Services (Healthcom)*, (2016), pp. 648–652

13. C.I. Sheriff, T. Naqishbandi, A. Geetha, Healthcare informatics and analytics framework, in *2015 International Conference on Computer Communication and Informatics (ICCCI)*, (2015), pp. 1–6
14. A. Pir, M.U. Akram, M.A. Khan, Internet of things based context awareness architectural framework for HMIS, in *2015 17th International Conference on E-Health Networking, Application & Services (HealthCom)*, (2016), pp. 55–60
15. P. Chatterjee, R.L. Armentano, Internet of things for a smart and ubiquitous eHealth system, in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, (2016), pp. 903–907
16. P. Gupta, D. Agrawal, J. Chhabra, P.K. Dhir, IoT based smart healthcare kit, in *International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT)*, (2016), pp. 237–242
17. H.S. Maghdid, K.Z. Ghafoor, A.S. Sadiq, K. Curran, K. Rabie, A novel AI-enabled framework to diagnose coronavirus COVID-19 using smartphone embedded sensors: Design study. arXiv preprint arXiv:2003.07434 (2020)
18. D.M. El-Din, A.E. Hassanein, E. Hassanien, W.M. Hussein, E-quarantine: A smart health system for monitoring coronavirus patients for remotely quarantine. arXiv preprint arXiv:2005.04187 (2020)
19. V.K. Singh, H. Chandna, A. Kumar, S. Kumar, N. Upadhyay, K. Utkarsh, IoT-Q-Band: A low cost internet of things based wearable band to detect and track absconding COVID-19 quarantine subjects. *EAI Endorsed Trans. Internet Things* **6**(21), 2020 (2020)
20. S. Din, A. Paul, Smart health monitoring and management system: Toward autonomous wearable sensing for Internet of Things using big data analytics. *Future Gener. Comput. Syst.* **91**, 611–619 (2019)
21. M. Otoom, N. Otoum, M.A. Alzubaidi, Y. Etoom, R. Banhani, An IoT-based framework for early identification and monitoring of COVID-19 cases. *Biomed. Signal Process. Control* **62**, 102149 (2020)
22. Y. Qadri, N. Zikria, K. Sung Won, The future of healthcare internet of things: A survey of emerging technologies, in *IEEE Communications Surveys & Tutorials*, (2020), p. 1
23. McKinsey & Company Report, The Internet of Things: How to capture the value of IoT (2018, May)
24. N. Yang, Z. Wang, R. Gravina, G. Fortino, A survey of open body sensor networks: Applications and challenges, in *14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, (Las Vegas, NV, USA, 2017)
25. <https://www.i-scoop.eu/internet-of-things-guide/internet-things-healthcare/>
26. A. Sheldon, *IoT in Healthcare: Benefits, Challenges and Applications, Value Coder* (2019, September)
27. S.B. Baker, W. Xiang, I. Atkinson, Internet of things for smart healthcare: Technologies, challenges, and opportunities. *IEEE Access* **5**, 26521–26544 (2017)

Correction to: Recent Advances in Energy-Efficient Resource Management Techniques in Cloud Computing Environments



Niloofar Gholipour, Ehsan Arianyan, and Rajkumar Buyya

Correction to:

Chapter 2 in: R. Buyya et al. (eds.), *New Frontiers in Cloud Computing and Internet of Things*, Internet of Things,
https://doi.org/10.1007/978-3-031-05528-7_2

This book was inadvertently published with an incorrect affiliation of Chapter 2 author Dr. Niloofar Gholipour. We have now updated the author affiliation as follows:

Department of Software and IT Engineering of Ecole De Technology Superior, University of Quebec, Montreal, Quebec, Canada

The updated original version for this chapter can be found at https://doi.org/10.1007/978-3-031-05528-7_2

Index

A

- Addressing manner, 237
- Agriculture, vii, 318–320, 324, 332, 333
- Anomaly detection, 185–205
- Apache OpenWhisk, vi, 166, 171, 174, 176, 179, 181, 182
- Application, 3, 32, 76, 108, 134, 165, 185, 209, 232, 263, 289, 325, 346, 366, 390
- AWS Greengrass, vi, 166, 167, 171–173, 175–182
- AWS Lambda, 13, 167, 168–173, 175, 177–182
- Azure Functions, 13, 167, 169, 171–182

B

- Big data, 7, 20, 39, 133, 231, 312, 327–329, 335, 337, 369, 390, 394

C

- Characteristics of IoT, 23, 170, 268, 274, 403
- Classification algorithm, 374, 375
- Cloud computing, 3–26, 32–63, 107, 165, 171, 173, 220, 231, 288, 289, 293, 294, 303, 309–316, 318–320, 330–331, 339, 367, 368
- Cloud data center, vi, 33, 35, 45, 50, 62, 69–103, 109, 110, 209–224
- Cloud data center energy-efficiency, 109
- Cloud services, 5, 7–10, 12, 13, 19–21, 32, 33, 62, 107, 266, 276, 279, 302
- Communication protocol, 14–18, 239–240, 243–247, 250

- Components, 4, 6, 10, 14, 37, 39, 41–44, 50, 75, 84, 146, 148, 172, 174, 176, 179, 189, 211, 214–217, 219, 223, 238–250, 257, 282, 291, 293, 294, 300, 301, 329, 334, 337, 346, 347, 349, 350, 360, 370, 390, 391
- Computational sustainability, 263–264
- Computing clouds, 133–161
- Connectivity terminologies, 234
- Container as a Service (CaaS), 5, 10–11, 32, 44, 45, 59, 166, 169
- Containerization, 33, 34, 41, 44, 45, 61, 168
- COVID-19 pandemic, 254–257, 278, 280, 281, 287–320, 391, 392

D

- Data analytics, v, 188, 323, 327, 328, 335, 370
- Data Stream Management Systems (DSMS), 133–161
- Digital twin, vi, 209–224
- D-Storm, vi, 135, 136, 144–148, 151–160
- Dynamic scheduling, 158
- Dynamic VM Consolidation, vi, 69–103

E

- Edge computing, vi, 4, 6–8, 20, 21, 23–25, 166, 169, 181, 268, 288–291, 314, 319, 337, 368
- Education, vii, 257, 261, 262, 272, 288, 291, 302, 312, 320, 325–327, 330, 331, 333–334, 390
- Electricity, 4, 21, 270, 278, 279, 307, 347, 348, 360

Energy efficient solution consolidation, 102
Entertainment, vii, 6, 288, 291, 316, 320

F

Fog computing, 4, 6–8, 20, 21, 23–25,
266–268, 273, 278–280, 282, 289, 291,
314, 337, 368
Function-as-a-Service (FaaS), 5, 7, 12–13, 166,
168–170

G

Genetic algorithm (GA), 38, 108–110, 112,
129, 161, 370

H

Healthcare, vii, 4–7, 21, 22, 274, 277, 309–317,
320, 324, 330, 331, 366, 387–404
Heart disease, vii, 365–383
Heuristic-based algorithms, vi, 136, 141–144,
147, 158
Highly energy proportional servers, 69–103

I

Industrial Internet of Things (IIoT), 304, 309,
310
Infrastructure as a Service (IaaS), 5, 9–10, 32,
33, 37, 44, 45, 139, 367, 389
Internet of Things (IoT), 3, 134, 165, 209, 230,
257, 289, 323, 344, 367, 388
IoT-driven solutions, 389, 390, 392–397,
400–403
IoT in healthcare, 393, 395–403
Isolation-Trees, 185–205

M

Manufacturing, vii, 22, 210, 211, 263,
302–304, 307, 320
MAPE-K, 144–146
Monitoring, v, vii, 7, 9, 19, 22, 44–46, 90,
91, 108, 135, 140, 145, 146, 171, 172,
185, 186, 210–213, 219, 220, 222, 223,
232, 262, 264, 268, 270, 282, 296,
303, 305–312, 314, 318–319, 323–339,
343–361, 388, 391–393, 396, 397, 401,
402

N

Naïve Bayes, 365–383
Network analysis, 188
Network topologies, 234–237

O

OpenFaaS, vi, 13, 166, 168, 171, 172,
174–176, 178–182
OpenStack, 9, 209–224

P

Particle swarm optimization (PSO), 108–110,
114, 369
Performance management, 186
Platform as a Service (PaaS), 5, 11–12, 32, 33,
45, 169, 367
Prediction, 60, 74, 257, 365–383
Protection, 261, 271, 325, 338, 343–361

R

Resource-efficient scheduling, 133–161
Resource management, 8, 22, 24, 25, 32–63,
107–130, 402
Resource usage, vi, 5, 12, 118, 121, 143–145,
156
RFID technology, 331–333

S

SDG-7, 275
Security, v, 5–7, 13, 16–18, 21, 25, 32, 34, 62,
107, 172, 234, 238, 241, 244–246, 248,
250, 254, 255, 262, 267, 289, 291, 299,
314, 327, 330, 333, 338, 339, 344, 347,
389, 390, 396, 400, 401, 403
Serverless, 6, 7, 12, 26, 165–182
Serverless edge computing, 166, 169
Shuffling frog leaping algorithm, vi, 115
Smart grid, 279, 343–361
Software as a Service (SaaS), 5, 13–14, 32, 77,
367
Students location, 335, 337, 338
Sustainability, v, 25, 253–282, 329
Sustainable development, vii, 258–260, 274,
282

T

Transportation, vii, 6, 21, 22, 254, 261,
291–296, 299, 301, 320, 333

U

University campus, 337–339

V

Virtual machine (VM) migration, 34, 37,
47–49, 60, 74–76, 80, 83, 89–91,
96–103, 109, 118, 120
Visualisation, 19, 209–224, 367, 375