Complex & Intelligent Systems
https://doi.org/10.1007/s40747-021-00434-6

ORIGINAL ARTICLE

Implementation analysis of IoT-based offloading frameworks on cloud/edge computing for sensor generated big data

Karan Bajaj[1] · Bhisham Sharma[1] · Raman Singh[2]

Received: 25 March 2021 / Accepted: 9 June 2021

Abstract
The Internet of Things (IoT) applications and services are increasingly becoming a part of daily life; from smart homes to smart cities, industry, agriculture, it is penetrating practically in every domain. Data collected over the IoT applications, mostly through the sensors connected over the devices, and with the increasing demand, it is not possible to process all the data on the devices itself. The data collected by the device sensors are in vast amount and require high-speed computation and processing, which demand advanced resources. Various applications and services that are crucial require meeting multiple performance parameters like time-sensitivity and energy efficiency, computation offloading framework comes into play to meet these performance parameters and extreme computation requirements. Computation or data offloading tasks to nearby devices or the fog or cloud structure can aid in achieving the resource requirements of IoT applications. In this paper, the role of context or situation to perform the offloading is studied and drawn to a conclusion, that to meet the performance requirements of IoT enabled services, context-based offloading can play a crucial role. Some of the existing frameworks EMCO, MobiCOP-IoT, Autonomic Management Framework, CSOS, Fog Computing Framework, based on their novelty and optimum performance are taken for implementation analysis and compared with the MAUI, AnyRun Computing (ARC), AutoScaler, Edge computing and Context-Sensitive Model for Offloading System (CoSMOS) frameworks. Based on the study of drawn results and limitations of the existing frameworks, future directions under offloading scenarios are discussed.

Keywords Context-awareness · Frameworks · Internet of Things · Offloading · IoT applications · Edge/fog computing

Introduction

Internet of Things (IoT) is termed as a connection of networks over the Internet. However, the purpose of this network is not merely the transfer of data or acting as a communication channel; instead, the objective of this network is for enabling the linked devices to communicate and collaborate among themselves to provide some particular service. The aim of IoT is to simplify tasks and enable it to perform smartly by gaining a high degree of intelligence in applications and services with the least human intervention using various sensors, actuators and processors [1]. Internet serves a significant role in IoT services to provide a communication channel and set up a smart interface between people and surrounding objects. Cloud and edge structures act as the critical component of IoT, to provide useful applications, specific services in multiple application domains [2]. IoT brings in automation in all sectors of life referred to as public domain and also makes all physical objects intelligent that can connect, communicate with each other and can make the smart decision by themselves. IoT provides several applications to the various streams of users, and for that, it implements different frameworks. IoT frameworks can be termed as a set of guiding policies, protocols, and principles which simplify the accomplishment of IoT applications [3].

Manyika et al. [4] had predicted the sharp rise of IoT impacting the overall economic sector by \$2.7 trillion to

✉ Bhisham Sharma
bhisham.sharma@chitkarauniversity.edu.in;
Bhisham.pec@gmail.com

Karan Bajaj
karan.bajaj@chitkarauniversity.edu.in

Raman Singh
raman.singh@thapar.edu

[1] Chitkara University School of Engineering and Technology, Chitkara University, Himachal Pradesh, India

[2] Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, Punjab, India

Published online: 19 June 2021

$6.2 trillion per year by 2025. Health services and manufacturing would be the most impacted area in the system. After these sectors, the next most influenced areas from IoT would be farming, energy processing, and security. It is calculated that the sole financial impact of IoT technology in health-related services would be range from $1.1 trillion to $2.5 trillion per year by 2025 [4]. The application of IoT spans in all the domains of society and daily life; it serves in all the fields from environmental information, activity information of living organism to the processing tasks in the industries. In all domains, IoT has no existence without a Wireless Sensor Network (WSN). Acting as a backbone of IoT, sensors collect the data and communicate them. Sensors are connected with the devices having different technologies and vide application areas, which make the incorporation of IoT with WSN challenging [5].

Some of the common issues that arise for the deployment of IoT applications are data management, communication issues, real-time computing and security, privacy and scalability of data [6]. There is also trustworthiness issue related to security and privacy of data in cloud storage scenarios, there may arise a security concern due to continuous connectivity of IoT-based sensors with the edge entities [7, 8], the vulnerabilities can arise specially in sectors like healthcare where a small change in values can be life threatening. Data aggregation is also termed as one big problem for smart grid IoT systems [9] which is somewhere related to data gathering and management issue. Some applications are delay sensitive due to challenges when processing a large amount of data at edge or the cloud level of devices, leading to latency, which is not acceptable in some critical applications like health scenarios, transport management, etc. Some solutions demand high energy, and it's evident that computation-intensive applications require more power and drain the batteries of devices quickly; thus, demanding expensive options or solutions [10].

Figure 1 lists several critical issues faced by IoT, in the paper focus of the study is that using the computation and data offloading in middleware architecture design can aid in dealing with the huge data generation and its processing challenge, also context awareness and device management issues. Architectures serve as a building blocks to fulfil all the essential requirements to solve the fundamental problems faced in IoT [11].

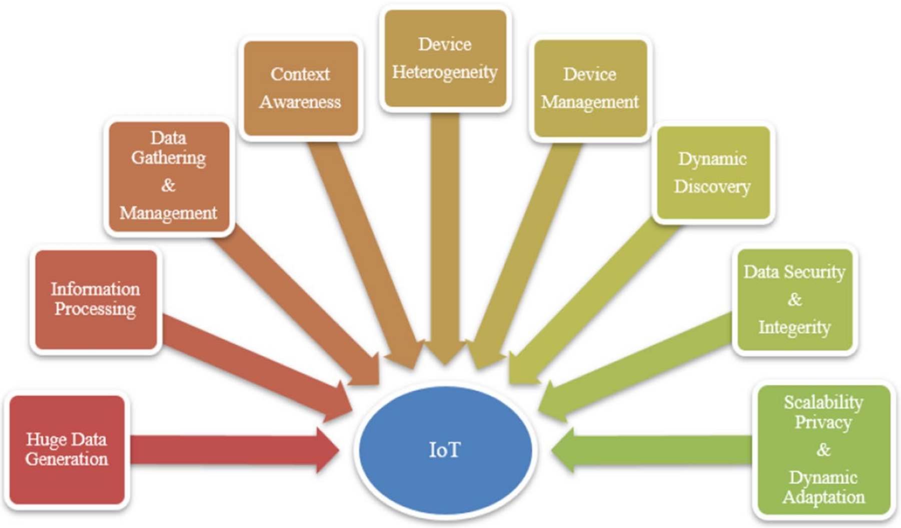
The architecture of internet of things

Limited knowledge and work in the present scenario resist the researchers to get through the scope of the Internet of Things. Middleware plays a vital role in IoT services, and this paper study the role of middleware and its scope to deal some of the critical challenges like time delay, energy consumption, scalability and big data management, etc. using offloading frameworks. Offloading frameworks are part of architectures to improve the overall functionality of IoT applications by developing a better understanding of the associated tool, technologies, and methodology. Its primary purpose is to solve real-life problems using and developing IoT concepts for day-to-day tasks [12].

The architecture serves as the most basic and essential block structure for IoT, and it is vital in terms of design choices for functional and non-functional requirements in IoT environments to serve the increasing scale and complexity of IoT.

Figure 2 shows the basic five-layer general IoT architecture where the bottom layer is perception and sensing layer, this is the physical layer and forms the connection between the real and the digital world. The role of the transport layer is transporting data among different devices

Fig. 1 Key issues faced by IoT



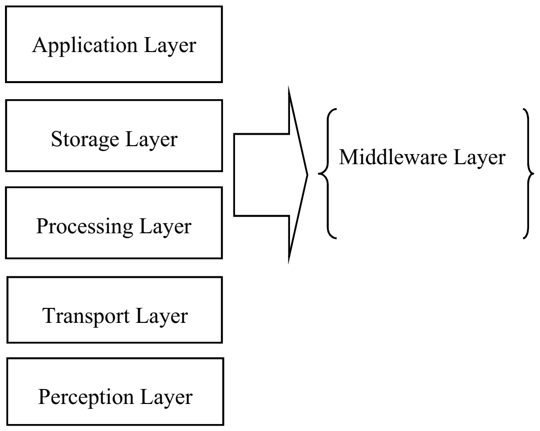


Fig. 2 Basic structural design of IoT

and objects. Enormous sensors lead to a massive amount of data generation necessitating IoT system to be a flexible and high-performance network structure to support different protocols among these devices adequately. Processing layer also called as the middleware layer, analyses and process the data coming from the transport layer. This layer uses large numbers of technologies for analysing and processing work. Massive databases are used for maintaining the data and edge, femto, fog, and cloud computing schemes are used for processing tasks containing big data [1]. Temporary data storage functionalities, data duplication and distribution is provided by a storage layer. The top-most layer of the architecture is the application layer and provides application services of the IoT system to users.

Smart solutions

Edge and fog computing and its integration with cloud computing are one of the promising solutions to address many challenges faced by IoT applications, service-related problems and the limitations of cloud computing [13], Middleware plays a significant role to deal with such challenges and support the delay-sensitive and context-aware services in IoT applications by creating the smart gateway for edge/fog server structures. Local computing and nearby devices can perform a large amount of processing instead of carrying out all storage of data and computing in clouds clusters and thus provide timely and intelligent services.

Computation offloading is a scheme to achieve various performance parameters mainly to reduce the consumption of energy and latency of service among the IoT devices. With the help of offloading, resource-efficient edge/fog computing for IoT applications can be achieved, to provide smart services to the users.

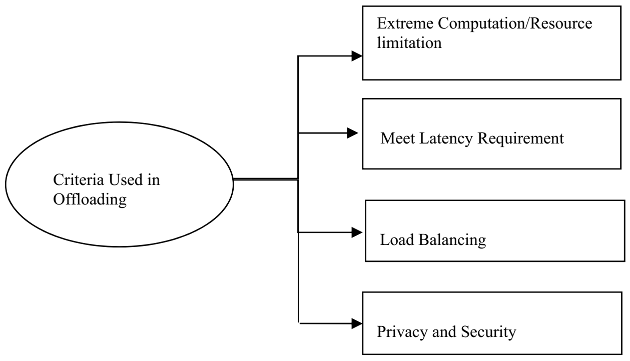

Criteria Used in Offloading
Extreme Computation/Resource limitation
Meet Latency Requirement
Load Balancing
Privacy and Security

Fig. 3 Essential offloading criteria's

Offloading criteria

Understanding the data and its context plays a vital role in offloading. Some criterion is listed in Fig. 3 that acts as a measure to take an offloading decision. Middleware by working as a smart gateway acts as a crucial mediator to monitor the nodes and decide the offloading of applications and services.

During the extreme computation requirement or under constrained resources or when the processing requirement of applications is more than the potential of the native device, devices are not able to fulfil the requirement of processing, and computation. This leads to delay or latency, which is critical to specific delay-sensitive applications. Load balancing is another criterion when the server has reached its maximum limit of processing the tasks, and jobs can be dispersed among other servers using offloading. Offloading is beneficial in the above cases and also this may help in securing the privacy and security of data at edge, femto cloud, or at fog cloud [14–17].

Related work

A systematic review of literature is carried out starting from the basic understanding of context, its role in IoT, how context awareness can help in offloading tasks, decision and role of machine learning and deep learning that can aid in recognition of context and taking an offloading decision based on it. A detailed review and comparative study are carried out of various models and offloading frameworks.

Role of context in offloading

Many researchers define context as understanding the situation of some events, Abowed et al. [18] defined context as the information that can be used to characterise the status of an entity. Context-aware or sensitive applications, look at the details of the data for understanding the behaviour of applications and requirement of services, to identify who's, where's, when's and what's of entities, to utilise the information to decide why the situation is occurring or taking place [19].

Context-aware computing requires both sensing and increasingly learning, as the data coming from the physical devices and sensor sources are large in amount and of continuous nature, making learning and gathering inferences tough. Big data and machine learning, aids by providing similar techniques for processing massive data sets [20]. Several big data techniques, learning algorithms like neural and deep learning, etc. are used to analyse data for IoT applications and services.

To recognise and distinguish the affective context from data, Nalepa et al. [21] showed the integration of context-aware systems with the affective computing prototype. Based on this knowledge, models, which interpret effects, were identified. Activity patterns are important during understanding and learning from data, recognition of patterns inference the knowledge and help in the identification of the context and situation. Researchers are mostly focussing on the implementation of computationally pervasive frameworks to make high-level conceptual models [22].

Under the current scenario, the middleware system architecture suffers by falling short in services and resources [23]. Middleware can provide essential services like collection of data from sensors, its processing and context recognition [22]. As achieving energy efficiency is a major challenge in context-aware applications as there are a continuous extraction and inference learning of data from sensors, there is a need of middleware design that can support context-awareness among application development task.

Offloading for IoT applications

As of now, there are a large number of devices and applications as part of IoT. There is a shift in the requirement of services towards computation and power management. The purpose of computation offloading architectures is to deal with such challenges and process the vast amount of data generated by IoT devices.

Ren et al. [24] proposed a Software Defined Network (SDN) to counter the challenge of generation of Big data across the different geographical locations, an adaptive recovery mechanism using support vector machine is given as solution. In [25], Mobile Edge Computing (MEC) devices are used, and to identify the offloading rate, current battery level of the devices is used. A reinforcement learning-based computation-offloading framework is presented for IoT device. Q-learning model with the combination of deep learning and hot-booting to increase the learning speed is shown.

For many large computation-intensive applications, there is a requirement of other entities to execute the tasks in place of a client device and getting results after processing [26], such mechanism is referred as offloading, where jobs are outsourced. This kind of offloading can be done in between sensors (edge devices), fog or cloud devices, but it turns out to be challenging to perform the real-time or actual processing due to the considerable distance between the cloud and end-user devices. Middleware can address such issues by acting as a smart medium in the middle of end nodes and the cloud. Mobile edge computing, cloudlets are some of the middleware technologies proposed to handle such offloading scenarios. Aazam et al. [27] described the offloading procedure, where the nodes close in proximity of client node that is the receiving node must be involved in the task of offloading for meeting the delay-sensitive requirements of applications. Offloading of tasks needs an intelligent system that can make optimal decisions about whether to offload based on the energy trade-offs and which specific task to be offloaded to the cloud, or a local fog or femto-cloud.

Authors in [28] proposed a partial flooding algorithm, and given an offloading methodology for Internet of Vehicles (IoV) to improve the overall utilization of system. For the reduction of time and energy consumption for mobile devices, a multi-objective optimisation technique is proposed in [29], here Computation Offloading Model (COM) is proposed for IoT-based cloud-edge computing.

With the perspective of context-awareness, many learning solutions and systems have been developed in IoT. These solutions are mainly designed-based, logic-based, and ontology-based, and developed using supervised, unsupervised, and reinforcement algorithms. A hybrid or a mixed approach can be designed to improve them. Deep learning, neural networks can be merged as a novel technique [30].

In the next section, we will be discussing the several IoT offloading frameworks and their major objectives and working parameters.

Offloading frameworks

There are different methodologies vide which the research network has learned different ways to implementing offloading frameworks, the various categories under offloading are: partition of applications, where applications can partition category wise [31], migration of threads, where first threads of applications are created then offloaded [32], migration of the application to the server-side, and distributed offloading [33]. Various internal and external factors like the requirement of applications, condition of network and device computing capabilities, etc. influence the decision of offloading of computation-intensive applications. These internal and external factors affect the offloading decision whether, where and when the offloading should be done.

Edge and fog computing are becoming a promising solution to lessen a load of computation from the cloud. This aids in delay-sensitive and context-aware applications by providing timely services. Instead of using only a cluster of clouds for data storage and performing all the computation processing, edge and fog computing try to utilise the maximum benefit of local computing.

MAUI [34] based on smartphones to make them work longer with code offloading, works on the principle of managed code, which reduces the load on the coder. Program partitioning is used to increase the energy benefits. ThinkAir [35], another model that works on the principle of smartphone virtualisation in the cloud, dynamic allocation of resource and parallel execution, is used for code offloading. Lin et al. [36] implemented a Context-Aware Decision Algorithm (CADA), a decision engine-based approach to decide whether to offload a given method to the cloud servers. CADA algorithm was integrated with ThinkAir.

MobiByte, a context-based model, proposed, is a cloud-based progressive application model for mobile cloud computing. It uses multiple data offloading schemes to increase smartphones devices applications performance, energy efficiency, and execution support [37]. Eom et al. [38] presented a mobile offloading framework called Machine Learning-Based Mobile Offloading Scheduler (MALMOS), having a novel approach of using online machine learning algorithms. It makes the assumption of attributes as independent of each other and also has a drawback of biasing towards earlier observations.

Majeed et al. [39] presented code offloading using Support Vector Machine (SVM), which is an adaptive and dynamic mobile system to take the offloading decision locally or remotely. AnyRun Computing (ARC) [40] uses a dynamic offloading model to choose the most capable nearby local computing infrastructure to support offloading structure. In ARC for offloading, not only the nearby devices are considered, but peer devices can also be taken to perform offloading as code can be run anywhere on any device structure. A round-robin scheduling based offloading approach is used in the Autoscaler [41] to allocate the load among the available servers. It consists of three parts, back-end consisting of servers that act as surrogates, front-end to receive the incoming requests, and load simulator to generate the multiple offloading requests.

An edge-based computing framework was proposed in [42] to help smart city residents by providing situational awareness. The presented framework showed that delivering relevant and essential services to the city residents would be beneficial by processing the IoT data at the edges. This would aid the decision-makers to be situation-aware and deliver services to the people. It is helpful in terms of latency and provides inferential knowledge to city residents. Results showed that using the edge computing services, the requirement of data that needed to be shifted to the remote server decreased significantly. The only limitation of this model is task allocation on edge devices is done based on fixed window size.

Another model named Evidence-Aware Mobile Computational Offloading (EMCO) [43] toolkit and platform is a cloud-based model designed as a new solution to solve the challenges faced during computational offloading. For the categorisation of the contextual parameters and other important factors on the offloading decisions, it makes use of the crowd sensed evidence traces. In this framework, models are constructed in the cloud and are sent to mobile devices; therefore, raw processing of data is not done.

Nakahara et al. [44] proposed a Context-Sensitive Model for Offloading System (CoSMOS). It is a self-adaptive offloading system and works based on the context-aware mechanism for mobile cloud computing (MCC) systems. An Adaptive Job Allocation Scheduler (AJAS) [45] to reduce the job reallocation delay time is proposed, it uses user behaviour pattern to requests the allocation of jobs to other nodes when the user's applications are being executed. Dynamic Energy-Efficient Data Offloading (DEED) framework given by Yan et al. [46] for IoT applications is based on an unstable channel state in the communication model and was proposed for task reliability, energy consumption, and device reliability model. The authors of Energy Efficient Offloading Strategy (EES) [47] developed a new bi-objective model based on firefly technique, which looks for the most advantageous computational device.

MobiCOP-IoT [48] framework uses the concept of surrogates, and it deploys them on both far-end clouds and nearby nodes. These nodes are edge-based and offer several features like automated self-regulating offloading of arbitrary tasks based on the output of an integrated decision-making engine. Having rich features, but still, MobiCOP-IoT needs manual configuring to decide whether the structure should work in cloud or edge mode. A Mobile Edge Computing (MEC) [49], based on an adaptive framework that supports mobile applications with offloading is proposed. It enables the application to dynamically offload among the mobile devices, edges and the cloud. An estimation model and unique design pattern based on DPartner, is proposed to decide the offloading scheme.

A content-based offloading mechanism for Mobile Edge Computing is proposed [50] in which the offloading is based on data transmission rate, built on which the contents are partitioned into separate categories. The transmission rate is considered to identify the priority; users having a low transmission rate are regarded as lesser priority work and offloaded first. It is considered that data having low priority will be having lower utilisation of Small Base Station (SBS). In case the SBS traffic exceeds the threshold, then the resources will be offloaded to the WiFi—app first. A code offloading edge/fog mobile-based Autonomic Management Framework is a computation-offloading model for mobile fog environment [51]. The main components of the framework are fog nodes to support parallelism, code analyser unit identify the basic blocks that are computation hungry and require resources. Resource availability and network status are considered for computing latency and resource demand.

Junior et al. [52] proposed a Context-Sensitive Offloading System (CSOS), which is a machine learning-based framework using J48, JRIP, IBK and Naive Bayes reasoning techniques. The authors selected two techniques—J48 and JRIP for the implementation as these provided the best accuracy to make offloading decisions. The Fog Computing (FC)-based analytical model is proposed for IoT-based healthcare applications [53]. These services are critical in terms of latency, and therefore, require fast processing. The overall latency in communication must be reduced, like computation and network latency for IoT data transmission. An edge-based structure is proposed as a solution for the problem, where edges are used for processing and analysis of data to reduce high latency.

Table 1 lists the authors with the objectives of their models and frameworks, and it also lists the offloading architecture specifying the basis and techniques of offloading used by them. The main aim of a large number of frameworks is to reduce time and energy consumption. The offloading frameworks are classified into two broad categories [54], Virtual Machine (VM) cloning, and Client–Server communication frameworks. In VM clone, the whole application is made on the cloud server by transferring application completely with its operating system; is therefore termed as full image transfer. After finishing the task, the cloned VM state is merged with the client to resume the execution. Client–server frameworks, works using communication protocols to achieve offloading, where only logic imitation termed as replication of logic part is done by pre-installing application part on the server device. Offloading frameworks mainly differ in their techniques and basis of implementation.

Table 2 presents a detailed study of different frameworks, including the data sets or real-time environment for their working, parameters used by them and their offloading and simulation environment. It also details the tools, technologies and the implementation used by the frameworks.

In the present scenario for large-scale IoT applications and systems, the offloading architecture is understudied. Due to a lack of understanding of contexts and situations, offloading is not favoured for large offloading environments. Consequently, there is poor usage and allocation of resources on the cloud, and the study of offloading have mostly shown that instead of benefiting, offloading increases computational effort.

From Tables 1 and 2, we can identify that Client–server model is a better choice over the VM model for the offloading purpose as it saves bandwidth consumption and more near to the real-world situations required for data offloading. Also, most of the frameworks are mobile-based, and therefore, mobile and gaming applications are mostly used for the implementation purpose with cloud-based offloading scenario.

From the literature reviews, it has been identified that very less work has been done on edge structures and edge-based cloud scenarios, and the working models and frameworks based on them were not focussed on the smart requirements or understanding-based offloading approach. Instead, mostly fixed scheduling schemes were used.

Materials and methods

A total of five frameworks having different simulation, working environments, datasets, and working parameters, have been shortlisted for the implementation study. All these models are chosen based on their effective, accurate, and novel approach towards offloading.

EMCO framework

A novel computational offloading approach was given by Flores et al. [43], in which the solution in the form of a toolkit and platform was designed for offloading, for implementation resources are available as open-source on GitHub. Crowdsensed evidence traces were used to categorise the influence of different contextual factors and other parameters on offloading decisions. A simulation environment for the framework is Google Cloud Messaging, LAPSI cloud-based runtime, C4.5 Decision tree classifier with fivefold cross-validation environment based on the Dalvik virtual machine. Implementation is done on lightweight compiler for executing the code. The results obtained from the EMCO framework are in the context of the time taken from a range of 1–30 s to execute on different devices and cloud environment, including pre-caching techniques. The maximum energy consumption in the local scenario was approximately 10,000 J.

In Fig. 4a, the average execution time is calculated for both the applications of chess and backtracking. The time represents both summations of processing time and communication latency. Similarly, Fig. 4b shows the average energy consumption. Only a single local mobile device is taken, which shows that the local device consumes more time and energy for the calculations.

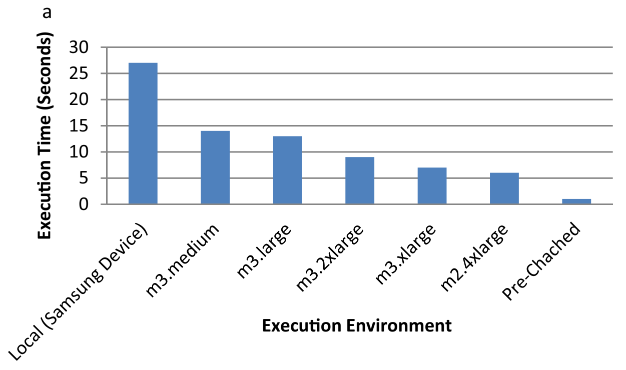
This model had considered only the cloud approach for offloading having smartphone implementation only. Approximate average execution time and energy consumption is 8.33 s and 391 J on surrogate's cloud excluding requirements of the local execution of the devices.

Table 1 Objectives and architecture model of frameworks

Authors	Objectives	Offloading architecture	Techniques	Base/algorithm
Cuervo et al. [34]	Maximizing the energy benefits of offloading code	Virtual Machine (VM) /Client–Server	Fine-Grained code offload	0–1 Integer linear programming
Majeed et al. [39]	Enhance performance and minimize energy consumption	Client–Server	SVM classifier and Scheduler	SVM learning for offloading choice
Ferrari et al. [40]	Foot-print reduction, Higher energy efficiency and using nearby resources for reducing latency	Service Platform for Socially Aware Mobile and Pervasive Computing (SCAMPI) [55, 56]	Fine-grained energy-aware code offloading	Bayesian statistics- Naïve Bayes approach
Flores et al. [41]	Raising offloading capability for large-scale IoT structures	Not-provided	Round Robin scheduler at the front end to allocate load between available servers at the back-end	Operational modes: Concurrent and Inter-arrival rate. N- Concurrent Threads, Minimax algo
Hossain et al. [42]	To reduce the latency and provide the situation awareness	Client–Server (Edge network and cloud middleware)	Fixed window size approach to sample the sensor stream data using edge nodes	Edge computing framework
Flores et al. [43]	Improving responsiveness and battery efficiency	Client (Context profilers, Offload Descriptor, Decision Engine and Evidence Cache)–Server (Evidence Analyzer, Evidence Cache, Code Offload Manager, Push Profiler, AutoScaler& Surrogate Runtime Environment)	Offloading decisions through simple dimensions and for determining optimal dimensions, using analytic process	Crowdsensed evidence, using different contextual parameters and other factors to characterize the influence on offloading decisions
Nakahara et al. [44]	For mobile cloud computing systems, deployment of self-adaptive and context-aware capabilities to enhance cloud support performance	Client–Server (Multiplatform Offloading System MpOS framework) [57]	Data input stage, First validation stage (stimulus awareness linear regression), Second validation stage (interaction awareness) and Offloading decision and conflict solution stage (Capabilities like- meta self-awareness and awareness of goal)	Execution Time and Energy Consumption measures to accurately identify when and which application module or unit should be offloaded (bi-objective optimization-based system)
Benedetto et al. [48]	Reducing running time and energy consumption	Client–Server	Dynamic Profiler-: A network profiler and code profiler (Heuristics-Based Algorithm)	Network environment and evidence of past executions
Alam et al. [51]	Improving performance in terms of latency and energy efficiency	Client–Server	Mobile Fog Nodes: Access Point (AP) and Access Point Controller (APC) units. Inter Fog Communication: Packet Data Gateway presents in Evolved Packet Core	Reinforcement Learning (Markov decision process) and Deep Q-learning [58] based computation offloading
Junior et al. [52]	Based on context reducing consumption of energy and improving runtime	Client (Mobile Device)—Server (Cloud or Cloudlet)	Middleware Integration, Machine-Learning classification algorithms and Robust profiling system	History-based prediction approach. Decision Engine: k-nearest neighbours, decision tree (Dynamic J48 & JRIP)
Shukla et al. [53]	Optimize: Network usage, RAM consumption and Reducing: Delay among healthcare IoT applications	Client (IoT devices, FIS Classification, Patient Health Data (PHD))—Server (Fog Gateway, Virtual Machines)	Q-learning, Markov Decision Process (MDP)	Hybrid ML Algo: Fuzzy Inference System (FIS), Linear Support Vector Classification (SVC) and Reinforcement Learning (RL) technique (Neural Network)

Table 2 Parameters and simulation environment of frameworks

Frameworks	Data set	Parameters	Simulation environment
AnyRun (2016) [40]	Applications: Ant Colony Optimization (ACO), Face Recognition (FR), and Character Recognition (CR)	CPU and RAM usage dimensions, Execution time and Energy consumption	Java Language (Refactoring technique) and Automatizing refactoring technique in Android IDE, Dalvik virtual machine [59]. Android-based devices and Oracle's HotSpot virtual machine [60]
AutoScaler (2017) [41]	Chess Game	Not-Provided	Front-End: AutoScaler, Back-End descriptor JSON format. Back-end: Dalvik machine, Test bed Amazon EC2
Edge computing paradigm (2018)[42]	CityPulse [61] and City of Chicago [62]. Real-time sensor data collection	Humidity, Light (Light, Temperature, Acceleration), Noise (Noise, Acceleration), Parking (Occupancy), SmartPhone (Smart Phone Sensors)	Java Development Kit version 8 & Laravel Framework 5.6
EMCO (2018) [43]	Carat [63] and NetRadar [64]	System profilers: Usage Monitoring-Network and CPU, Device Monitoring- battery state. Sensor data profilers—position or remaining battery	Google Cloud Messaging, LAPSI cloud, C4.5 Decision tree with fivefold cross-validation, environment based on the Dalvik virtual machine
CoSMOS (2018) [44]	Mobile Applications: N-Queen [65] problem and the BenchImage [57], an image processing application	Time consumption by mobile application's, Energy consumption during application execution	Trepn application [66] execution's energy consumption
MobiCOP-IoT (2019) [48]	Benchmark used in our experiments- N Queens	Optimistic mode: On the basis of output of integrated decision offloading the tasks to offsite surrogates. Simultaneous execution of tasks on the client and the server	Android operating system (Nougat 7.1, 0.5.1 developer preview 7 and Marshmallow 6.0), Android execution environment powered by Genymotion Cloud
Autonomic Management Framework (2019) [51]	N-queen problem	Mobile Stations: 30 Stations, Memory- 1 GB/Per node, 900 MHz Processor, 100 Mbps Bandwidth. Fog Nodes: 8, 16 GB/Per node Memory, 1.6 Ghz Processor, 1Gbps Bandwidth. Average Hops—2. Cloud Nodes: Total Number 1 of 8 VMs, 64 Gb Memory/Per VM, 2,44 Ghz Processor, 10 Gbps Bandwidth, Average Hops—13	MATLAB (Mobile Stations, Fog Nodes and Cloud Nodes)
CSOS (2019) [52]	BenchFace [67], BenchImage and CollisionBalls [68]	BenchFace: Detection Algo—Alt tree, Sizes- 3, 6 and 8 in MP. Faces— 77. Number of Tasks- 30. BenchImage: Filter-Cartoonizer, Size-2, 4 and 8 in MP, Image-SkyLine, Tasks- 30. CollisionBalls: Serialization-Java builtin, Size 1938 kb, Balls- 750, Tasks- 10	Weka library classifier, TP Link Communication Between Mobile and Cloudlet—AC1200 (802.11ac, UDP background traffic—Iperf [69] tool, CpuRun [70] and CpuBurn [71] tools having the purpose of utilization of all the available cores of the smartphone's CPU and cloudlets
Fog Computing Analytical Model (2019)[53]	ECG dataset [72] Categorization: low, regular and above normal. Guidelines: Fuzzy functions and fuzzy rules	Age,Chest Pain, restecg, thalrest,trestbps,chol, fbs,num, ca, thalach, sex, exang,oldpeak,slope	iFogSim [73] (Net-Beans) and Spyder (Python), skfuzzy API to model the fuzzy system,Java APIs



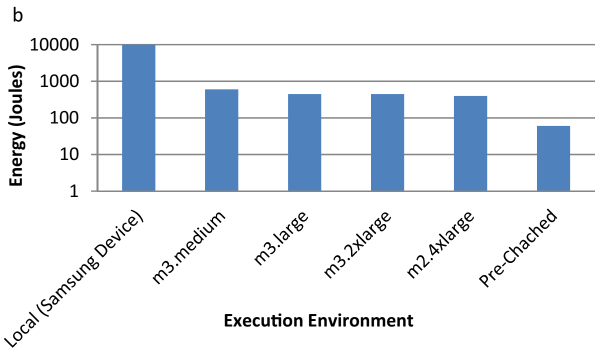


Fig. 4 **a** Execution time of Chess and Backtracking Applications of EMCO Framework. **b** Energy consumption of Chess and Backtracking Applications of EMCO Framework

MobiCOP-IoT

Frameworks given by Benedetto et al. [48] contain surrogates set up as distant clouds and proximate nodes at the edges to enhance the capabilities of resource-constrained devices. Various scenarios were used to examine MobiCOP-IoT; both with the centralised cloud and edge distribution, and performance measurements taken were based on time and energy to complete multiple tasks of the gaming applications. Figure 5a and b show edge computing is having the benefit in terms of performance parameter mentioned as time consumption.

On the similar lines, Fig. 5c and d show benefits of edge computing in total consumption of power. The framework is tested on three applications N-Queens, RenderScript and VideoProcessing.

In Fig. 5, improvement can be noticed in edge scenarios, results show that deployments get better in edge-based systems.

Both scenarios were taken when tasks were fully offloaded to the centralised cloud and to the edge servers, as shown in Fig. 5. These benchmarks are available online as open source. The average results of each task are taken after running five times.

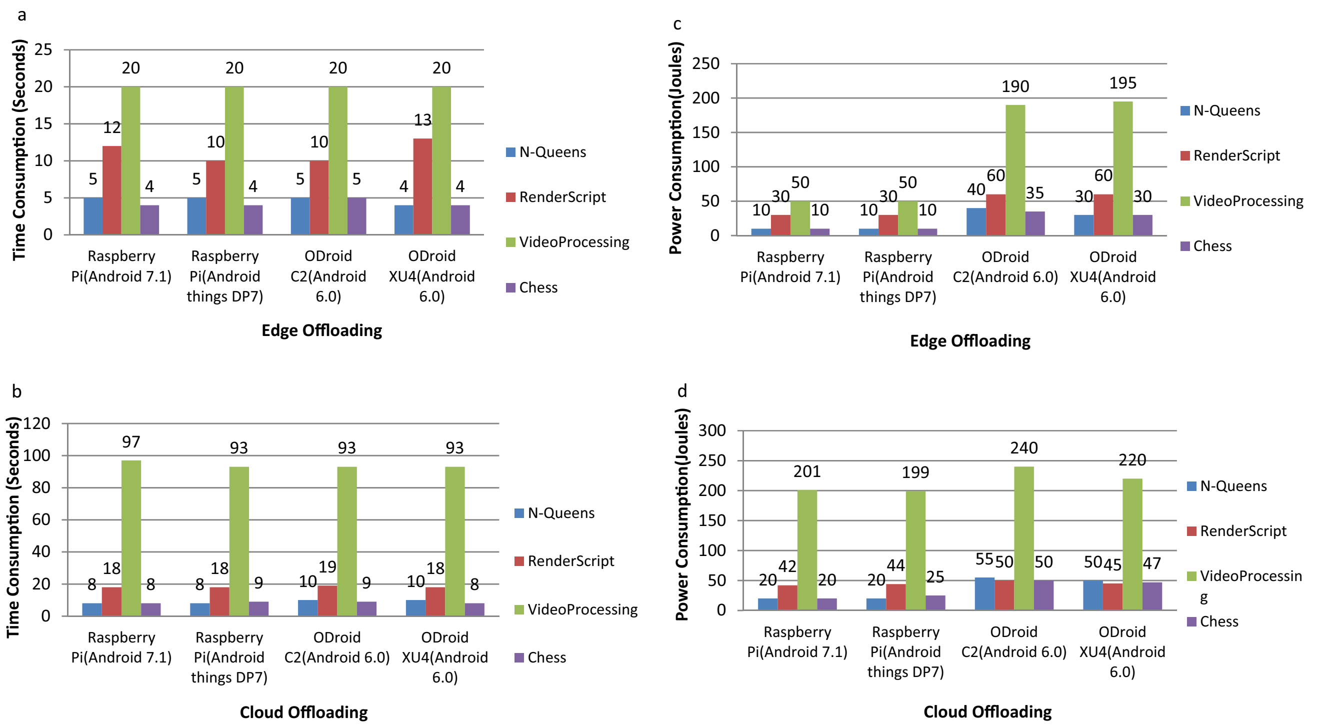


Fig. 5 **a** Performance gain for MobiCOP-IoT in edge scenario. **b** Performance gain for MobiCOP-IoT in cloud scenario. **c** Energy gain for MobiCOP-IoT in edge scenario. **d** Energy gain for MobiCOP-IoT in cloud scenario

Autonomic management framework

Alam et al. [51] proposed an edge/fog-based framework simulated through MATLAB. The N-Queen application is taken to carry out test, Fig. 6a shows the increase in the execution time when the number of queens increased from 1 to 8. Similarly, Fig. 6b shows the raise in energy consumption with the increase in number of queens.

The results from the model tested on dataset with the maximum number of queens taken was 8, variation in time observed was between 25 to 225 s, and energy consumption varied from 10 to 77 J.

Figure 6a shows the approximate execution time mentioned as response time in fog nodes and Fig. 6b shows approximate energy consumption, here eight fog nodes deployed to represent the 8-Queen problem. However, the model was not suitable when the number of queens increased to more than eight.

CSOS framework

Junior et al. [52] proposed a context-sensitive machine learning-based offloading environment (CSOS). Based on the implementation of a classification algorithm, a middleware is designed to work on a robust profiling system. Mobiles are very dynamic in nature; there is a need for an adaptive environment for them. By implementing the classification and profiling, a highly accurate decision engine way designed for mobiles. 10-fold cross-validation was used for training and testing purpose, and it achieved the accuracy of around 95%. The source for the implementation automation program is openly available on the GitHub.

Figure 7a shows the average execution time with an increasing interval of 5 tasks under the contexts data set. The graph shows the execution time as favourable, unfavourable, and unknown contexts of BenchFace & BenchImage application. J48, JRIP & Cloudlet simulations are taken into consideration, and the best time of the model at a particular task is taken into consideration for the calculation.

On similar lines, the average power utilisation is shown in Fig. 7b for BenchFace, BenchImage & CollisionBalls application, and its results show that in the favourable condition, JRIP is more efficient in terms of consumption of energy than the cloudlet and J48 strategy, and it has a smaller number of rules and needs less classification time. In an unfavourable context, J48 and JRIP show promising results by choosing to process locally. Unknown contexts show that for offloading of the computations, JRIP yields better outcome and results in lesser time for execution with lower consumption of energy.

Under the results shown in Fig. 7, the local strategy is not considered as it does not make use of the CSOS framework of offloading, and all processing is done on the smartphone. Further, in this framework, only cloud structure was taken for implementation of offloading structure on a mobile cloud with J48 & JRIP classifier.

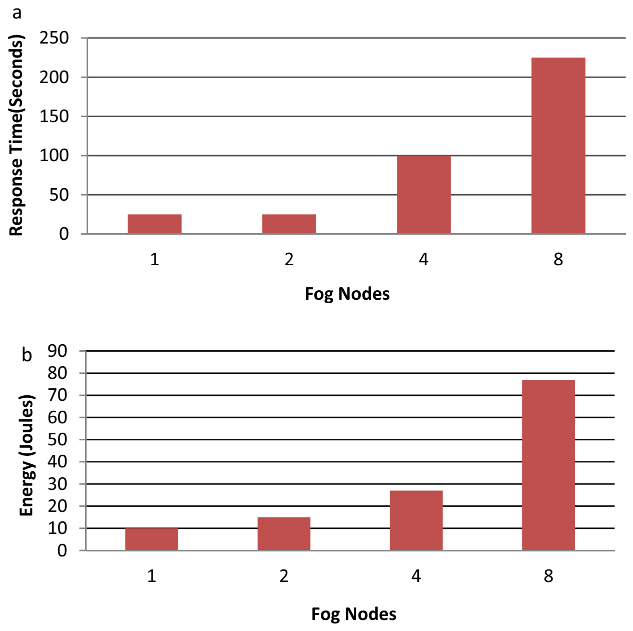


Fig. 6 **a** Execution time of Fog Nodes in Autonomic Framework. **b** Energy consumption of Fog Nodes in Autonomic Framework

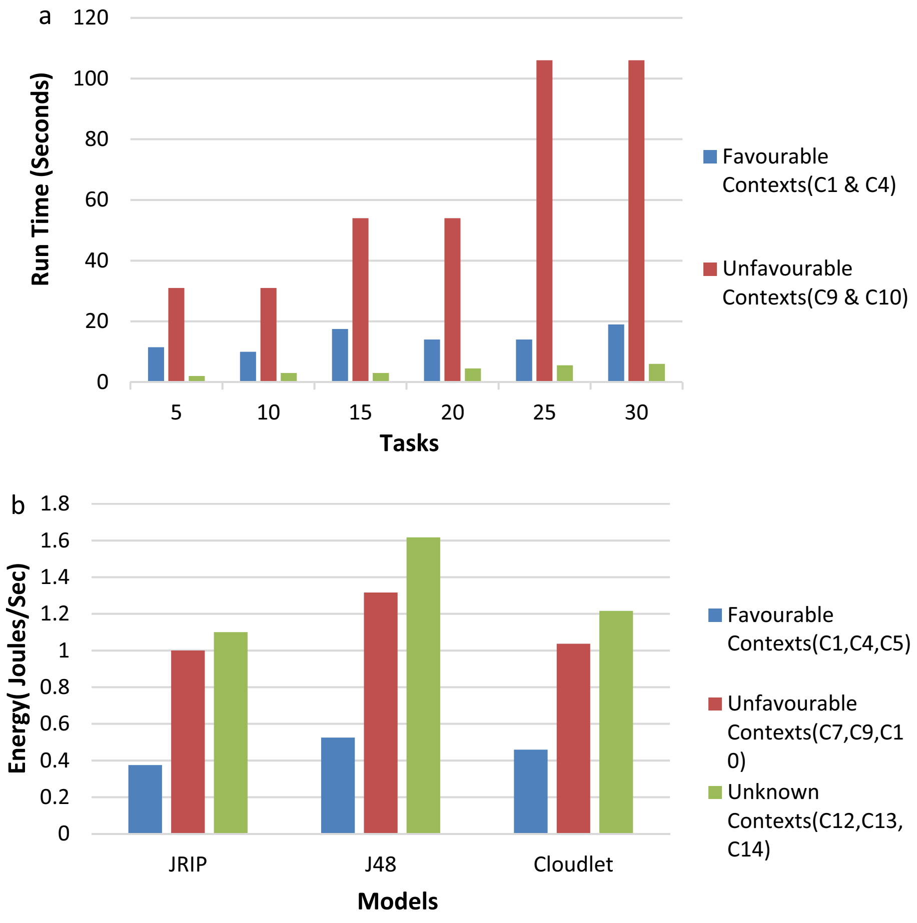
Fog computing framework

A fog-computing framework-based analytical model proposed by Shukla et al. [53], aims at boosting the cloud-based computations by bringing them near the edge devices. A framework was designed for a healthcare system to incorporate the analytical model and hybrid fuzzy-based reinforcement-learning algorithm. iFogSim simulator was used for implementation that includes fog devices, ECG sensors, and cloud servers. Five physical arrangements topologies were used, as shown in Fig. 8.

Simulation setup was implemented on iFogSim (NetBeans), Spyder (Python), skfuzzy API to model the fuzzy system & Java APIs. ECG data set available on the UCI machine learning repository was used, with parameters already specified in the paper mentioned in Table 2.

Results in Fig. 8a and b show physical topology configurations, and this framework showed improved performance and efficiency by minimising latency. Very few parameters of the data set were selected, as in case of health scenarios the accuracy of the results is critical. There might be a rise in overall latency, with the increase in number of parameters or increasing the size of dataset. But this model needs to be tried out with more parameters and larger dataset to

Fig. 7 **a** Execution time of CSOS Mobile-Cloud Environment. **b** Energy Consumption of CSOS Mobile-Cloud Environment



formulate for obtaining desired improvement regarding overall latency, consumption of memory and network.

This section presents that EMCO framework; consider only cloud approach towards offloading with the average execution time of 10 s over six cloud surrogates and average energy consumption of 389 J over five cloud surrogates. MobiCOP-IoT working based on mobile applications showed that edge processing is beneficial over cloud processing. Still, there is a big rise in execution time and power consumption with the increase in the size of an application. For different applications on different devices, the range of average execution time is from 4 to 20 s, and the overall energy consumption is approximately from 10 to 195 J. In the Autonomic framework, the execution time extends beyond 200 s, which is comparatively higher than other frameworks. Here the size of application decreases the performance and energy consumption that goes around 80 J.

Like MobiCOP-IoT, CSOS framework is also a smartphone-based framework. The average variation of execution time is 4 s in case of favourable context, 17 s in unknown context and 103 s in unfavourable context cases. Similarly, the variation in the energy consumption in three different contexts is 0.5, 1 and at max 1.5 J/s, respectively.

Similar to EMCO framework, CSOS is a cloud-based model without considering edge processing. The increase in the number of tasks adversely affects its performance. Fog computing framework based on edge processing working structure, is same as in EMCO and Autonomic framework in terms of non-smartphone-based application and gives an average delay of 0.8 s with more than 85% network and 25% memory consumption. This model gives a good performance in terms of time in general application scenario but required testing when data size increases as other frameworks have large data set over this.

After analysis of the implementation of different frameworks, it can be stated that most of the devices do not have any built-in framework for offloading computation and switches for manual offloading. There is a need for smart middleware gateway design that can work as a solution with the smart-devices operating system to perform the efficient offloading. Designing the middleware which can learn from the sensory data, battery behaviour, context inferences through machine learning and processing of that data, are the major challenges faced, as most of the available solutions are rule-based or logic-based. There is a scope of further improvement using hybrid methods and learning algorithms in future.

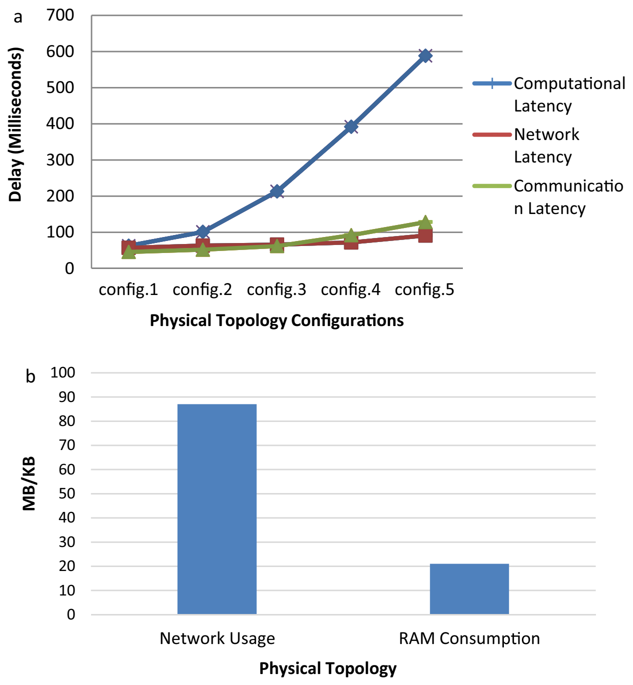


Fig. 8 **a** Delay analysis of Fog Computing (FC) analytical framework. **b** Efficiency analysis of Fog Computing (FC) analytical framework

Comparative analysis and discussion

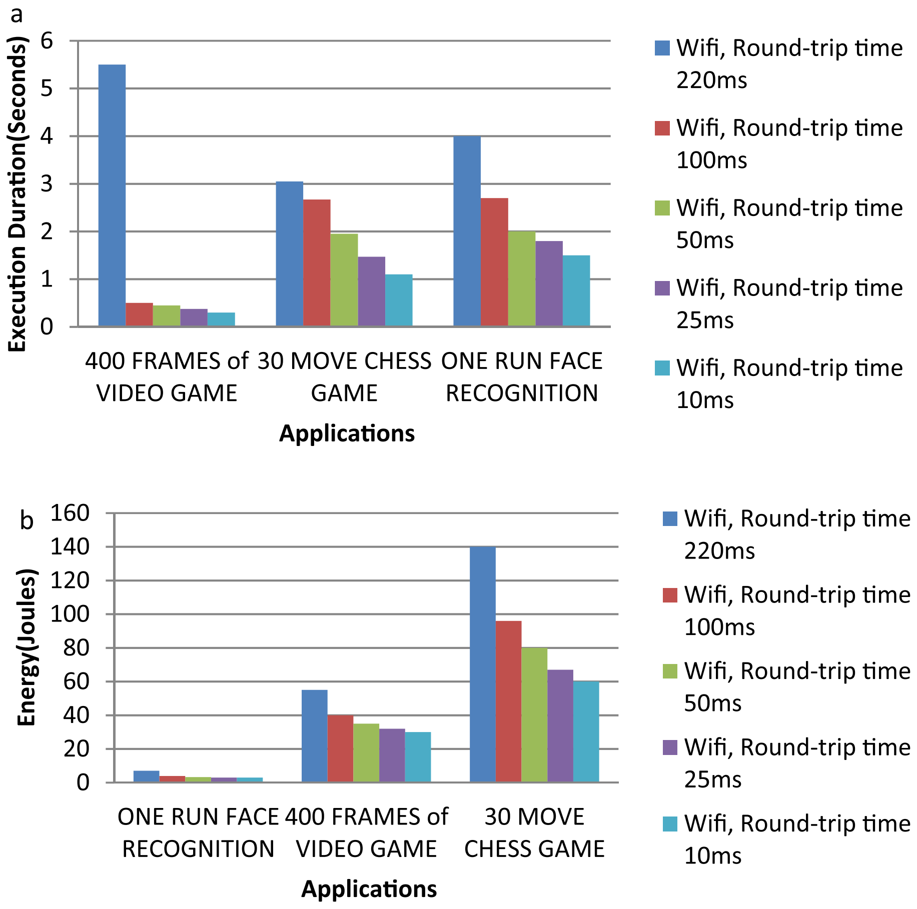
Under this section, some frameworks are taken for comparative study with the frameworks analysed above, based on several performance parameters, models are compared. These frameworks have their working conditions and environment discussed in detail in Sect. "Related work" and Tables 1 and 2.

MAUI is remotely executed on the server with WiFi with Round-trip time (RTT) of 10, 25, 50, 100 and 220 ms as shown in Fig. 9a and b.

In this, three applications are taken for performance and energy consumption parameters running standalone on the smartphone, which are video game for 400 frames and 30 moves for chess game and face recognition application, are taken. MAUI is tested on smartphones with mobile applications and uses program partitioning methodology.

Figure 10a shows the mean time execution and Fig. 10b shows the energy drain for ARC framework, here nexus smartphone is used to run all benchmarks to attain a baseline performance in three modes. The three modes are fixed network topology, full connectivity-based dynamic network topology, and emulated dynamic network topology.

Fig. 9 **a** Execution time of MAUI Model for smartphone code offload. **b** Energy consumption of MAUI Model for smartphone code offload



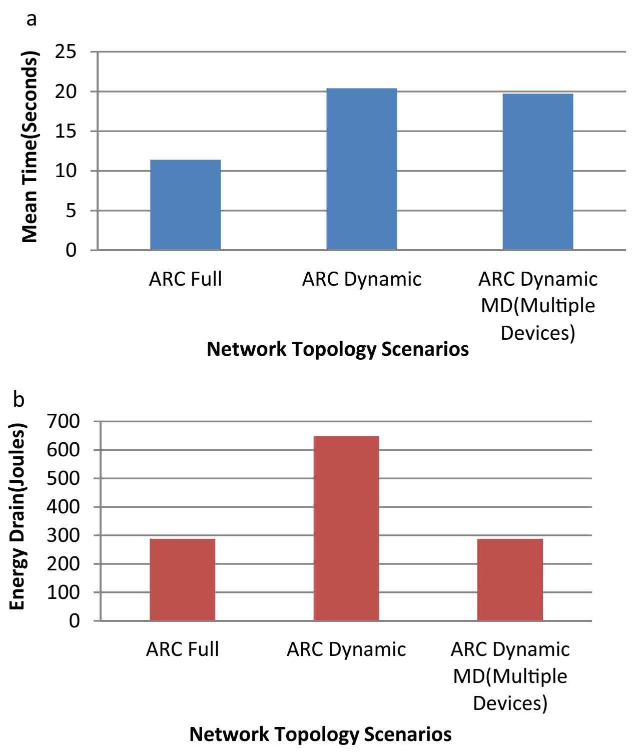


Fig. 10 **a** Mean time execution for AnyRun Computing (ARC). **b** Energy drain for AnyRun Computing (ARC)

In AutoScaler, execution time is based on the minimax algorithm, and varies on different surrogates as that can be seen in the Fig. 11a.

In Fig. 11b, edge computing framework, edges are used for offloading data, storage, processing, and privacy protection. For the partitioning of data for offloading to the edge devices fixed window size approach is used to sample the sensor stream data.

In Fig. 12a and b, the average of the total time taken for execution and their mini-max values are specified for both N-Queens and BenchImage applications.

In Fig. 13a, the average of total energy consumption for N-Queens application is taken and in Fig. 13b, the average for total consumption of energy for BenchImage application considered. Here min–max values are specified for both the applications. The three environment chosen for the analysis are cloudlet WiFi, cloud WiFi and cloud 3G.

From the Figs. 12 and 13, it can be said that, application size and physical distance among the devices and remote servers have a significant influence over the system as the offloading time rises with the transfer time of the dataset.

From Fig. 9, it is observed that the Round-trip time significantly affects the execution time and energy consumption of the applications. A sudden rise represents that this model is highly dependent on the size of the application similar to observed in frameworks like MobiCOP-IoT and Autonomic frameworks.

ARC framework from Fig. 10 represents the mean execution time and energy drain using application datasets described in Table 2. The execution time varies from 11 to 20 s with an exceptionally high-power consumption of around 300 Joules in two modes and about 600 J in ARC dynamic mode. This approach may waste resources during offloading done by the inference engine, which ignores the impact of remote devices for offloading.

AutoScaler can be referred to as a predecessor model of EMCO considering cloud approach, and it takes approximately 16.02 s over 11 devices, which is relatively higher like ARC. If the local devices are not considered, then the average computation time is about 13.6 s, and this, including communication latency and processing time. The main consideration point here is that the cases used are not a realistic practice as a smartphone.

Another framework based on edge computing structure is discussed to process the IoT data [42]. This model resembles the Fog Computing model [53] from the material and method section. This model takes 0.102 s (sec) in offline mode for CityPulse data set, 0.4 s for the Chicago park data set to calculate the results using edge processing. The total processing time in the real-time scenario is approximately 4.35 s, which includes sensor processing time of 0.352 s, edge to cloud network delay of 0.256 s, overall processing time in edge and cloud are 3.5 s and user delay is 0.243 s. Here the only disadvantage is that, fixed window size approach is used during data offloading, which is a rigid approach for all kinds of data whereas Fog Computing model uses the reinforcement learning approach but dataset size used is small and can't be relied for real world applications.

A Context-Sensitive Model for Offloading System (CoSMOS), which is a mobile-based cloud computing offloading decision support model, also termed as self-aware and self-expressive system was proposed [44]. Two mobile applications, N-Queen problem, and BenchImage for image-processing applications used, with the focus on the CPU processing time. Over these applications, N-Queens have a relatively small dataset and BenchImage have a large data set. In the case of N-Queens, the number of queens taken is between 4 and 13. However, when taken number of queens taken are 14 or more, then the processing time and energy consumption increases drastically. For BenchImage application, five different resolutions of three pictures are handled within a range from 0.3 to 0.8 MP.

From the implementation, analysis, and comparison of different frameworks, it is observed that most of the structures are tested on gaming applications and are mobile-based. Their efficiency is affected by the bulkiness of the

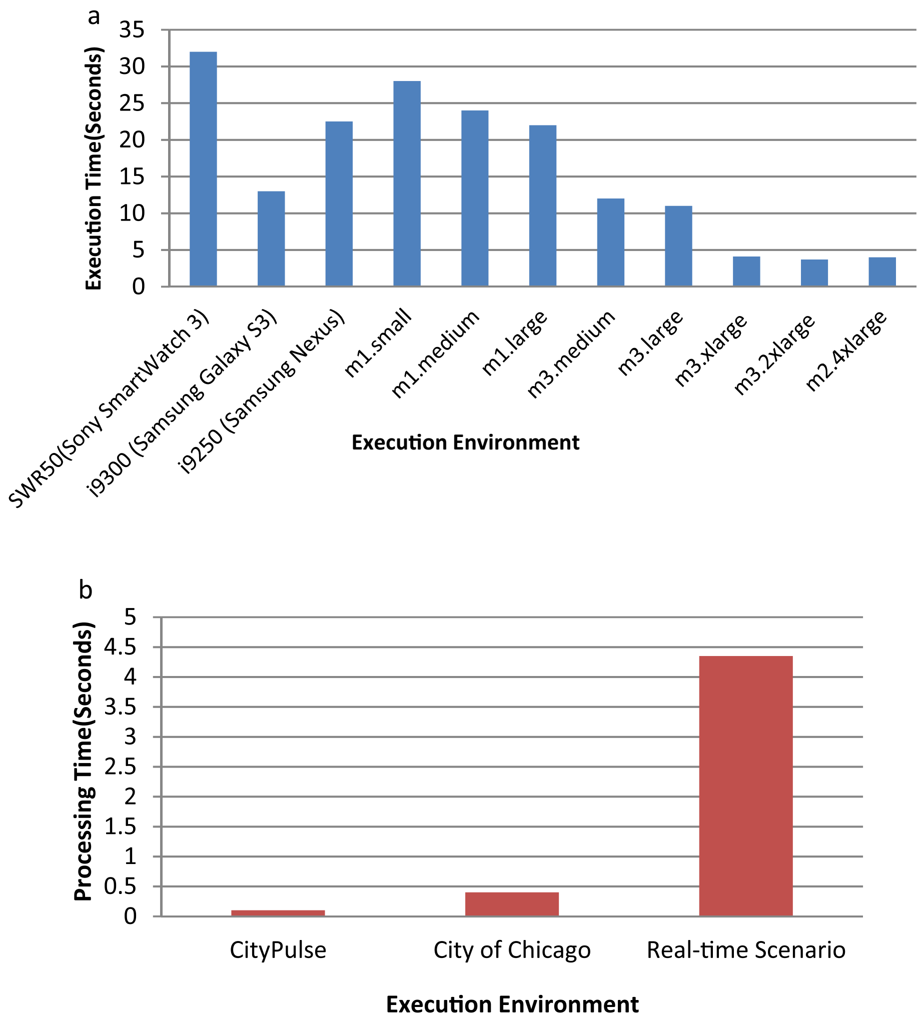


Fig. 11 **a** Execution time of AutoScaler. **b** Processing time of Edge Computing Framework

applications. Edge Computing paradigm and Fog computing frameworks discussed show some useful practical life data set implementations and shows promising results in terms of various parameters like time, energy, network, and system utilisation. However, some of the direct methods used for offloading create a scope for some hybrid, regression, and learning-based approaches for offloading. Scaling-up the requirement of services needs to be handled under offloading architectures, to manage the particular load of the devices. Proper planning for the capacities is required to identify the adequate number of back-end servers.

Future research challenges

There are several challenges in IoT from heterogeneity among the devices, their data to device management, context awareness and processing of information [5]. It has identified that, irrespective of application areas, there are some common key issues that keep on existing, like inherent distribution, data management and making human centric applications [74]. Today cloud services are required to give stable Quality of Service (QoS), as these can be reused. A cost-effective approach is required to provide cloud service composition [75]. Offloading appears as solution for IoT applications and devices but it is not as straightforward as it seems to be, raising further, several challenges from this fusion of technologies. To overcome the problem of computing power limitation, storing capability, and limitations of built-in batteries, the offloading of computations are critically important [45].

- Understanding the context, situation, or status of data and learning from them for computational and data offloading is among the major problems and can have a high impact on making systems and applications as a part of intelligent IoT services [51].
- Most of the designed solutions are based on rules, logic and ontology using supervised and unsupervised learning, along with reinforcement algorithms. There is a

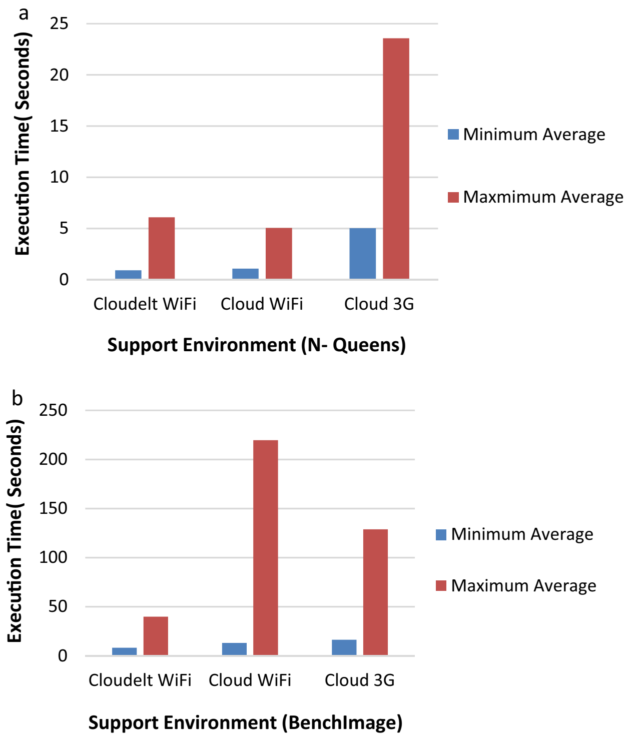


Fig. 12 **a** Average execution time of N Queens Application for CoSMOS. **b** Average execution time of BenchImage Application for CoSMOS

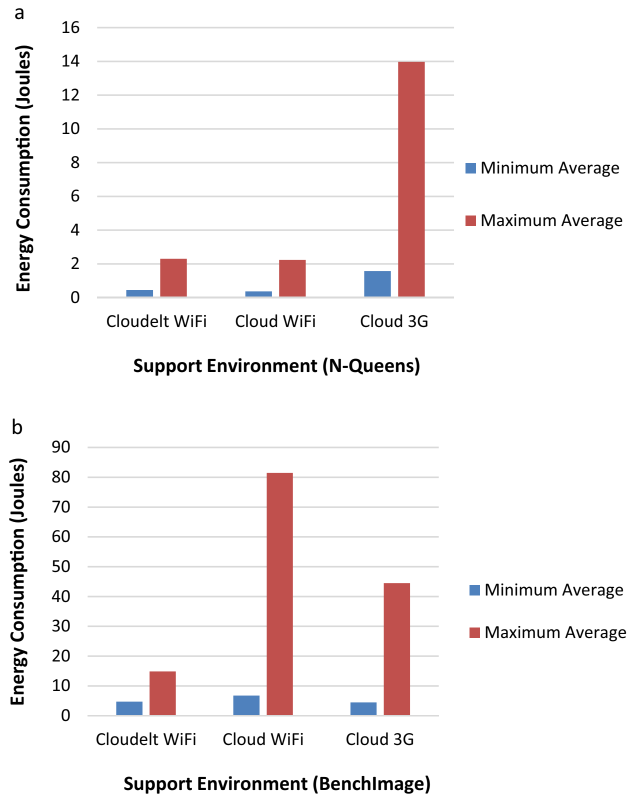


Fig. 13 **a** Average energy consumption of N- Queens Application for CoSMOS. **b** Average energy consumption of BenchImage Application for CoSMOS

scope of improvement using hybrid methods, such as neural methods and deep learning algorithms to achieve better performance [29].

- There is a need for an intelligent system that can make optimal decisions about which specific tasks to be offloaded, to the cloud or Femto-cloud. Designing the middleware which can learn from the sensory data, battery behaviour, and context inferences through machine learning and processing of the data are quite challenging. Middleware devices encounter limitation during providing service due to resource constraints in terms of power, memory, and bandwidth [23, 24].
- Many IoT applications require separate entities to compute and process the tasks on behalf of user devices, like smart home, healthcare, intelligent transport management, Ambient Assisted Living (AAL), Virtual Reality (VR), etc. to produce the results. It becomes challenging to provide real-time computations and delivering fast responses due to significant distance among the cloud servers and end-users [26, 27].
- During transmission and computing for offloading the IoT applications to the cloud, there is the consumption of a large amount of energy because the far-end network experiences a higher latency and network delay. Edge and Fog nodes provide solution and offer the cloud services at near end edges of the network, and this makes IoT applications to run locally with minimum energy utilization and reduce the delay. However, such a structure has a limitation in terms of resource capacity. Resource-intensive IoT applications suffer constrained resources issue under edge/fog nodes implementation [46, 50].
- Current studies in the field of offloading are more focussed on centralisation and coordination of data. Edge computing and fog computing are new areas of research and need establishing frameworks to put these concepts into practice [51].
- Identification of the situation is a crucial issue to take the offloading decision as all of them are not beneficial, and a primary challenge is to identify those situations. A large number of factors influence the efficiency of offloading for making practical and optimal offloading decisions [42].
- The key research issues for offloading computation in fog or edge structures are choosing the approach to offload computation, the module or procedure of applications to offload, and where to offload for minimizing the latency of service computing [47].

- There is need of a smart, intelligent and selective offloading scheme to formulate the decision of whether to offload computation, when to offload and where to offload the tasks, like across local devices at edge level, to the fog cloud, or the cloud structure in proximity.
- Most of the solutions designed are single reasoner, or mobile-based, and therefore, mobile and gaming applications are used for the implementation study and very less work has been done on IoT applications and their implementation on edge-based cloud scenarios, and the working models or frameworks based on offloading with such scenarios have not focussed on the smart requirements or understanding-based offloading approach. Instead, mostly fixed scheduling schemes were used.
- With the results obtained in figures and graphs it can be seen that for most of the applications, their efficiency is affected by the bulkiness of the applications, with the efficient deployment of fog/edge scenarios and smart offloading scheme we can deal with such issues.
- Direct methods used for offloading create a scope for implementation of some hybrid and learning-based approaches for offloading. Such methodologies can take advantage of the reasoning techniques to provide offloading decisions, as higher latency and delay makes the data meaningless and inadequate for end-users.

Conclusion

In this paper, a detailed implementation analysis and comparison of various data offloading frameworks has been carried out, with the aim of understanding and analysing the role of context or situation to perform the data offloading. Some of the existing frameworks based on their novel approach and optimum results are taken for implementation. Under the analysis of implemented frameworks and their comparison with some of the existing frameworks, it has been identified that to meet the performance requirements of IoT enabled services, offloading play a crucial role. From the work carried out it also has been identified the size of the applications play crucial role for achieving adequate performance as increase in time and energy consumption can be seen in the graphs under the implementation, some intelligent approach is required to deal with large data size of applications and then perform the offloading to perform computations.

From the experiments done and results obtained, it has been deduced that offloading is not a straight way approach, rather before offloading some learning of context of data is required which will aid in taking correct decision like where and when to offload. Under the implementation scenarios, it has been seen that some learning methodologies were used in few implementations, but mostly was performed under mobile-based scenarios, there has been scope of improvement by implementing smart middleware design using hybrid learning mechanism to implement the computation offloading.

It also has been identified that there is a future possibility of work in edge structures and edge-based cloud structures for offloading frameworks, as very little work has been done in such scenarios and also mostly fixed scheduling schemes were used which can be improved.

Acknowledgements This work has been carried out in wireless sensor network research facility in Computer Science and Engineering department of Chitkara University, India. This research facility has been funded from the in-house grant of Chitkara University.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Sethi P, Sarangi SR (2017) Internet of things: architectures, protocols, and applications. J Electr Comput Eng 2017:9324035
2. Ray PP (2016) A survey of IoT cloud platforms. Future Comput Inf J 1(1):35–46
3. Ammar M, Russello G, Crispo B (2018) Internet of Things: a survey on the security of IoT frameworks. J Inf Secur Appl 38:8–27
4. Manyika J, Chui M, Bughin J, Dobbs R, Bisson P, Marrs A (2013) Disruptive technologies: advances that will transform life, business, and the global economy. McKinsey Glob Inst San Francisco CA 180:17–21
5. Giri A, Dutta S, Neogy S, Dahal K, Pervez Z (2017) Internet of things (IoT): a survey on architecture, enabling technologies, applications and challenges. In: Proceedings of the 1st International Conference on Internet of Things and Machine Learning, Liverpool, UK, pp 1–12
6. Al-Qaseemi SA, Almulhim HA, Almulhim MF, Chaudhry SR (2016) IoT architecture challenges and issues: lack of standardization. In; Future Technologies Conference (FTC), San Francisco, pp 731–738
7. Khan A, Din S, Jeon G, Piccialli F (2020) Lucy with agents in the sky: trustworthiness of cloud storage for industrial internet of things. IEEE Trans Industr Inf 17(2):953–960
8. Aujla GS, Jindal A (2021) A decoupled blockchain approach for edge-envisioned IoT-based healthcare monitoring. IEEE J Sel Areas Commun 39(2):491–499

9. Saleem A, Khan A, Malik SUR, Pervaiz H, Malik H, Alam M, Jindal A (2019) FESDA: fog-enabled secure data aggregation in smart grid IoT network. IEEE Internet Things J 7(7):6132–6142
10. Li Y, Björck F, Xue H (2016) IoT architecture enabling dynamic security policies. In: Proceedings of the 4th International Conference on Information and Network Security, Kuala Lumpur, Malaysia, pp 50–54
11. Cavalcante E, Alves MP, Batista T, Delicato FC, Pires PF (2015) An analysis of reference architectures for the internet of things. In: Proceedings of the 1st International Workshop on Exploring Component-Based Techniques for Constructing Reference Architectures, Montreal, QC, Canada, pp 13–16
12. Ray PP (2018) A survey on Internet of Things architectures. J King Saud Univ Comput Inf Sci 30(3):291–319
13. Qureshi KN, Jeon G, Piccialli F (2021) Anomaly detection and trust authority in artificial intelligence and cloud computing. Comput Netw 184:107647
14. Elgazar A, Harras K, Aazam M, Mtibaa A (2018) Towards intelligent edge storage management: determining and predicting mobile file popularity. In: 2018 6th IEEE International conference on mobile cloud computing, services, and engineering (MobileCloud), Bamberg, Germany, pp 23–28
15. Masip-Bruin X, Marín-Tordera E, Tashakor G, Jukan A, Ren G (2016) Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. IEEE Wirel Commun 23(5):120–128
16. Huang C, Lu R, Choo KR (2017) Vehicular fog computing: architecture, use case, and security and forensic challenges. IEEE Commun Mag 55(11):105–111
17. Aazam M, Huh E-N, St-Hilaire M (2018) Towards media inter-cloud standardization—evaluating impact of cloud storage heterogeneity. Int J Grid Util Comput 16(3):425–443
18. Abowd GD, Dey AK, Brown PJ, Davies N, Smith M, Steggles P (1999) Towards a better understanding of context and context-awareness. In: Handheld and Ubiquitous Computing. Springer, Berlin, Heidelberg, pp 304–307
19. Sukode S, Gite S, Agrawal H (2015) Context aware framework in IoT: a survey. Aqua Microbial Ecol Int J 4(1):1–9
20. Zaslavsky A, Perera C, Georgakopoulos D (2013) Sensing as a service and big data. In: International conference on advances in cloud computing, Bangalore, India, pp 21–29
21. Nalepa GJ, Kutt K, Bobek S (2019) Mobile platform for affective context-aware systems. Future Gener Comput Syst 92:490–503
22. Yürür Ö, Liu CH, Sheng Z, Leung VCM, Moreno W, Leung KK (2014) Context-awareness for mobile sensing: a survey and future directions. IEEE Commun Surv Tutor 18(1):68–93
23. Wood AD, Stankovic JA, Virone G, Selavo L, He Z, Cao Q, Doan T, Wu Y, Fang L, Stoleru R (2008) Context-aware wireless sensor networks for assisted living and residential monitoring. IEEE Netw 22(4):26–33
24. Ren X, Aujla GS, Jindal A, Batth RS, Zhang P (2021) Adaptive recovery mechanism for SDN controllers in Edge-Cloud supported FinTech applications. IEEE Internet Things J 2021:1–1
25. Shukla RM, Munir A (2017) An efficient computation offloading architecture for the Internet of Things (IoT) devices. In: 2017 14th IEEE annual consumer communications networking conference (CCNC), Las Vegas, pp 728–731
26. Jararweh Y, Doulat A, AlQudah O, Ahmed E, Al-Ayyoub M, Benkhelifa E (2016) The future of mobile cloud computing: Integrating cloudlets and Mobile Edge Computing. In: 2016 23rd international conference on telecommunications (ICT), Thessaloniki, pp 1–5
27. Aazam M, Zeadally S, Harras KA (2018) Offloading in fog computing for IoT: review, enabling technologies, and research opportunities. Futur Gener Comput Syst 87:278–289
28. Li B, Peng Z, Hou P, He M, Anisetti M, Jeon G (2019) Reliability and capability based computation offloading strategy for vehicular ad hoc clouds. J Cloud Comput 8(1):1–14
29. Xu X et al (2020) A computation offloading method over big data for IoT-enabled cloud-edge computing. Futur Gener Comput Syst 95:522–533
30. Sezer OB, Dogdu E, Ozbayoglu AM (2018) Context-aware computing, learning, and big data in internet of things: a survey. IEEE Internet Things J 5(1):1–27
31. Eom H (2014) Extending the capabilities of mobile platforms through remote offloading over social device networks. University of Florida
32. Chun B-G, Ihm S, Maniatis P, Naik M, Patti A (2011) CloneCloud: elastic execution between mobile device and cloud. In: Proceedings of the sixth conference on computer systems, New York, pp 301–314
33. Hassan MA, Chen S (2012) Mobile MapReduce: minimizing response time of computing intensive mobile applications. In: Mobile computing, applications, and services, Los Angeles, pp 41–59
34. Cuervo E, Balasubramanian A, Cho D-K, Wolman A, Saroiu S, Chandra R, Bahl P (2010) MAUI: making smartphones last longer with code offload. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, pp 49–62
35. Kosta S, Aucinas A, Pan H, Mortier R, Xinwen Z (2012) ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: Proceedings IEEE INFOCOM, IEEE, pp 945–953
36. Ting-Yi L, Ting-An L, Cheng-Hsin H, Chung-Ta K (2013) Context-aware decision engine for mobile cloud offloading. In: IEEE wireless communications and networking conference workshops (WCNCW), Shanghai, pp 111–116
37. Khan A, Mur R, Othman M, Khan AN, Abid SA, Madani SA (2015) MobiByte: an application development model for mobile cloud computing. Int J Grid Util Comput 13(4):605–628
38. Eom H, Figueiredo R, Cai H, Zhang Y, Huang G (2015) MALMOS: machine learning-based mobile offloading scheduler with online training. In: 3rd IEEE international conference on mobile cloud computing, services, and engineering, San Francisco, pp 51–60
39. Majeed AA, Khan AUR, Ul Amin R, Muhammad J, Ayub S (2016) Code offloading using support vector machine. Sixth Int Conf Innov Comput Technol (INTECH) 2016:98–103
40. Ferrari A, Giordano S, Puccinelli D (2016) Reducing your local footprint with anyrun computing. Comput Commun 81:1–11
41. Flores H, Xiang S, Kostakos V, Ding AY, Nurmi P, Tarkoma S, Hui P, Li Y (2017) Large-scale offloading in the Internet of Things. In: 2017 IEEE international conference on pervasive computing and communications workshops (PerCom Workshops), (Heidelberg, Germany),pp 479–484
42. Alamgir Hossain SK, Rahman A, Hossain MA (2018) Edge computing framework for enabling situation awareness in IoT based smart city. J Parall Distrib Comput 122:226–237
43. Flores H, Hui P, Nurmi P, Lagerspetz E, Tarkoma S, Manner J, Kostakos V, Li Y, Su X (2018) Evidence-aware mobile computational offloading. IEEE Trans Mob Comput 17(8):1834–1850
44. Nakahara FA, Beder DM (2018) A context-aware and self-adaptive offloading decision support model for mobile cloud computing system. J Ambient Intell Humaniz Comput 9(5):1561–1572
45. Kim H-W, Park JH, Jeong Y-S (2019) Adaptive job allocation scheduler based on usage pattern for computing offloading of IoT. Futur Gener Comput Syst 98:18–24
46. Yan H, Zhang X, Chen H, Zhou Y, Bao W, Yang LT (2020) DEED: dynamic energy-Efficient Data offloading for IoT

applications under unstable channel conditions. Futur Gener Comput Syst 96:425–437

47. Adhikari M, Gianey H (2019) Energy efficient offloading strategy in fog-cloud environment for IoT applications. Internet Things 6:100053
48. Benedetto JI, González LA, Sanabria P, Neyem A, Navón J (2019) Towards a practical framework for code offloading in the Internet of Things. Futur Gener Comput Syst 92:424–437
49. Chen X, Chen S, Ma Y, Liu B, Zhang Y, Huang G (2019) An adaptive offloading framework for Android applications in mobile edge computing. Sci China Inf Sci 62(8):82102
50. Zhao X et al (2019) Deep learning based mobile data offloading in mobile edge computing systems. Futur Gener Comput Syst 99:346–355
51. Alam MGR, Hassan MM, Uddin MZ, Almogren A, Fortino G (2019) Autonomic computation offloading in mobile edge for IoT applications. Futur Gener Comput Syst 90:149–157
52. Junior W, Oliveira E, Santos A, Dias K (2019) A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment. Futur Gener Comput Syst 90:503–520
53. Shukla S, Hassan MF, Khan MK, Jung LT, Awang A (2019) An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment. PLoS ONE 14(11):e0224934
54. Fernando N, Loke SW, Rahayu W (2013) Mobile cloud computing: a survey. Futur generations computer systems 29(1):84–106
55. Conti M, Giordano S, May M, Passarella A (2010) From opportunistic networks to opportunistic computing. IEEE Commun Mag 48(9):126–139
56. Pitkänen M, Kärkkäinen T, Ott J, Conti M, Passarella A, Giordano S, Puccinelli D, Legendre F, Trifunovic S, Hummel K, May M, Hegde N, Spyropoulos T (2012) SCAMPI: service platform for social aware mobile and pervasive computing. SIGCOMM Comput Commun Rev 42(4):503–508
57. Costa PB, Rego PAL, Rocha LS, Trinta FAM, de Souza JN (2015) MpOS: a multiplatform offloading system. In: Proceedings of the 30th annual ACM symposium on applied computing, New York, pp 577–584
58. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
59. Ehringer D (2010) The Dalvik Virtual Machine Architecture. Technical Report. http://show.docjava.com/posterous/file/2012/12/10222640-The_Dalvik_Virtual_Machine.pdf. http://show.docjava.com/posterous/file/2012/12/10222640-The_Dalvik_. Accessed 1 Jul 2015
60. Java SE Hot Spot at a Glance (2014) http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136373.html
61. Puiu D, Barnaghi P, Tönjes R, Kümper D, Ali MI, Mileo A, Xavier Parreira J, Fischer M, Kolozali S, Farajidavar N, Gao F, Iggena T, Pham T, Nechifor C, Puschmann D, Fernandes J (2016) CityPulse: large scale data analytics framework for smart cities. IEEE Access 4:1086–1108
62. C. of Chicago, City of chicago open data (2018) Tech. rep. https://data.cityofchicago.org/. Accessed 26 Jun 2018
63. Oliner AJ, Iyer AP, Stoica I, Lagerspetz E, Tarkoma S (2013) Carat. In: Proceedings of the 11th ACM conference on embedded networked sensor systems—SenSys ’13. https://doi.org/10.1145/2517351.2517354
64. Sonntag S, Manner J, Schulte L (2013) Netradar—Measuring the wireless world. 2013 11th international symposium and workshops on modeling and optimization in mobile. Ad Hoc Wirel Netw 13:29–34
65. Gent IP, Jefferson C, Nightingale P (2017) Complexity of n-Queens Completion. J Artif Intell Res 59:815–848
66. Qualcomm (2015) Trepn power profiler. https://developer.qualcomm.com/software/trepn-power-profiler
67. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In; Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, CVPR, pp 1–1
68. Rego PAL, Costa PB, Coutinho EF, Rocha LS, Trinta FAM, de Souza JN (2017) Performing computation offloading on multiple platforms. Comput Commun 105:1–13
69. IPerf (2017) The ultimate speed test tool for TCP, UDP and SCT. https://iperf.fr/
70. CpuRun (2017) Tool to consume CPU resource by constant usage rate. https://play.google.com/store/apps/details?id=jp.gr.java_conf.toytech.cpurun&hl=pt_BR
71. CpuBurn (2017) The ultimate stability testing tool for overclockers. https://patrickmn.com/projects/cpuburn/
72. Andras Janosi WS, Matthias P, Robert D (2018) UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/heart+Disease. Accessed 25 Feb 2018
73. Gupta H, Dastjerdi AV, Ghosh SK, Buyya R (2017) 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(9):1275–1296
74. Bajaj K, Sharma B, Singh R (2020) Integration of WSN with IoT applications: a vision, architecture, and future challenges. In: Integration of WSN and IoT for Smart Cities, Springer, Cham, pp 79–102
75. Anisetti M, Ardagna CA, Damiani E, Gaudenzi F, Jeon G (2020) Cost-effective deployment of certified cloud composite services. J Parall Distrib Comput 135:203–218