# Multi-Objective Decision-Making for Mobile Cloud Offloading: A Survey

Huaming Wu, *Member, IEEE*

***Abstract*—Running very complex applications on mobile devices is still challenging since they are constrained by limited resources such as memory capacity, network bandwidth, processor speed and battery power. Mobile Cloud Computing (MCC) is a combination of cloud computing and mobile internet, which could effectively alleviate the resource constraints of mobile devices. How to efficiently offload computation-intensive parts of mobile applications from mobile devices to capable cloud servers is one of the keys. In mobile environments, the resource heterogeneity of mobile devices and cloud services, the interruption of heterogeneous wireless networks, the complexity of mobile applications, the characteristic of transferring a large amount of data, are the major bottlenecks that have prevented this technology from being widely used. This paper takes these constraints into account at the same time and explores methods of multi-objective decision making for time- and energy-aware task offloading for MCC. It is designed to ensure the right computational tasks are executed in the right way, at the right time and place.**

***Index Terms*—Mobile cloud computing; mobile edge computing; offloading; decision-making; energy-efficient.**

## I. Introduction

MOBILE devices, such as tablets, smartphones, smartwatches and notebooks, have limited resources in computational capacity, battery lifetime and network connectivity, which prevent them from running very complex applications [1]. There is a rapid growth of power consumption of mobile devices and seriously shorten their battery life as a result when more and more computation-heavy or energy-hungry applications are deployed on them. Responsiveness is another primary constraint for mobile systems. Mobile applications (face recognition, speech and object recognition, natural language processing, mobile augmented reality, etc.) are becoming increasingly intensive and sophisticated that require increasing amounts of computational capabilities [2]. Especially for real-time and user-interactive applications, they have to wait a long time to obtain the results due to the limited processing speed of the mobile systems.

Mobile Cloud Offloading (MCO), which takes advantage of abundant resources hosted by Clouds, is becoming a promising method to solve a number of concerns affecting mobile computing. Its main idea is to release the mobile devices from intensive processing through migrating computation-intensive tasks from mobile devices to remote cloud servers and then receive results from them via heterogeneous wireless networks [3]. MCO can bring many potential benefits, such as improving the performance of mobile applications, reducing the energy consumption of mobile devices and so on.

Mobile network environments usually have a huge impact on the performance of offloading systems since mobile users are easily subject to dynamically changing network conditions due to their mobility [4]. While traditional cloud applications (e.g., iCloud and Siri) have been very successful, on mobile devices they still suffer from a number of shortcomings due to the response time of wireless communication at the network edge. Intermittent connections of wireless networks will cause additional costs in terms of energy consumption and response time, and thus executing applications locally will be more advantageous than offloading them to the remote cloud [5]. Since the extra cost involved in data transfer via a wireless network may be greater than the cost savings from offloading in certain circumstances, an offloading decision of which portion of an application should be offloaded and which not, and where to place the execution (either locally or remotely) should be made based on different decision criteria. Therefore, it will always be difficult to make high-quality offloading decisions at runtime in mobile environments unless with a clear understanding of current and near-future wireless network conditions.

Recently, several surveys on Mobile Cloud Computing (MCC) have been conducted. In [6], the authors reviewed mobile application models in MCC and highlight their advantages and shortcomings. In [7], [8], the authors investigated state-of-the-art mobile augmentation efforts that employ cloud computing infrastructures to enhance computing capabilities of resource-constraint mobile devices. They reviewed existing offloading frameworks by using thematic taxonomy and analyzed the implications and critical aspects of current offloading frameworks within the MCC domain. In [9], the problem of resource heterogeneity in MCCs was tackled. The authors discussed the heterogeneity in convergent computing (i.e., mobile computing and cloud computing) and networking (wired and wireless networks). Further, the impacts of heterogeneity in offloading decision-making were investigated, related opportunities and challenges were identified.

To cope with the stringent requirements of applications on latency (e.g., real-time applications), an emerging concept named Mobile Edge Computing (MEC) / Fog Computing has been investigated [10], [11], [12], which offers connected computing and storage resources at the Internet edge, in close to mobile devices. Due to a wide potential of the MEC, there is a lot of effort both in industry and academia focusing on the offloading decision-making to ensure that the right computational tasks are processed in the right way at the right place and time. In [11], the authors surveyed current

H. Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin, P. R. China, 300072. Email: whming@tju.edu.cn.

research related to the offloading decisions to the conventional centralized clouds and MECs. In addition, it gave a high-level comparison of MCCs and MECs, i.e., MECs can offer significantly lower latency but have limited computational and storage resources with respect to MCCs. In [12], the authors provided a comprehensive survey of the state-of-the-art MEC research and new designs ranging from computation offloading techniques to network architectures.

All above-mentioned papers typically focus on one or two offloading challenges such as resource heterogeneity, data transfer or wireless communications and make offloading decisions based on limited aspects like where to offload, e.g., MCC or MEC. Actually, making good time- and energy-aware offloading decisions have to deal with multiple challenges, such as heterogeneous resources, large amounts of computation and communication, intermittent connectivity and network capacity. In contrast to the above-mentioned surveys, we conduct a comprehensive survey of the research work when taking all these challenges into account simultaneously and analyze the tradeoff between energy consumption and response time based on different offloading decision criteria. If we could find out the optimal solution for offloading decisions, the offloading benefits like performance improvement of applications executing on the resource-constrained mobile devices, could be enhanced. In one word, we need to find what computational tasks and data (*what*), at what point in time (*when*), in what way or through what channel (*how*) and at what place (*where*) to offload.

The rest of this paper is organized as follows. Section II describes characteristics of mobile cloud offloading, major challenges and a list of issues related to offloading decision-making. In each of the following four sections, we focus on the aspects of *when*, *what*, *where*, and *how* to offload for mobile cloud computing, respectively. Finally, the paper is concluded in Section VII.

## II. Offloading Decision Making

In this section, we first provide an elementary material on mobile cloud offloading systems and then describe the major challenges and issues associated with offloading decision-making.

### A. Generic Offloading System

Fig. 1 describes a generic offloading process.

- **Profiling**: On the mobile side, upon receipt of an offloading request, resource information about the device and network characteristics are gathered by the *profiling* module. There are three different kinds of profilers, namely, *program profiler*, *network profiler* and *energy profiler*. Among them, *program profiler* (static or dynamic) collects characteristics of applications, such as the execution time, the memory usage and the size of data; *network profiler* collects information about the network bandwidth and the wireless connection status (connected or unconnected); *energy profiler* is used to collect the energy characteristics of mobile devices through software and hardware monitors [14].
- **Metrics**: Offloading decisions are usually made based on a selected cost criterion. On the one side, energy, monetary cost and storage are cost criteria which are the less the better, and on the other side, performance, robustness and security are benefit criteria which need to be maximized [15]. Among such criteria, energy and performance are the two most important aspects mobile users concern about.
- **Application Partitioning**: On the basis of the collected information, the *offloading decision making* module takes the decision according to the *metrics* module (i.e., minimizing or maximizing some criteria), and then the *partitioning* module is invoked to cut the classes that make up an application into local and remote partitions, where the former is executed locally on the mobile device and the latter will be offloaded to a dedicated cloud server [16]. The application partitioning can be done either statically or dynamically.
- **Offloading Decisions**: The *cloud discovery* module is invoked to find an appropriate cloud service for offloading (*where to offload*). When wireless networks and cloud services are available, or when the offloading for the mobile device is beneficial in terms of energy consumption and/or execution delay (*when to offload*), remote partition classes (*what to offload*) are migrated to the cloud side via a wireless network by the *offloading* module for remote execution (*how to offload*). The offloaded classes can interact with the classes in the local partition [13]. Once completed, the results are sent back to the mobile side.

Most benefits from offloading like time- and energy-saving, can be achieved by optimally deciding *when*, *what*, *where* and *how* to offload. Specifically, *when* is to decide whether to offload or not, according to the knowledge on the amount of computation and communication data, the wireless network conditions and dynamic changes of context, since sometimes offloading is not worthwhile at all; *what* is to decide how much and what should be offloaded, it defines the name of the candidate tasks to be offloaded through application partitioning; *where* describes the type of surrogate and choosing the appropriate offloading target (e.g., local, cloudlet and cloud) in which the application has to be offloaded; *how* introduces offloading plans that enable the device to schedule offloading operations [16].

### B. Major Challenges

Mobile Cloud Offloading (MCO) that migrates heavy computation from mobile devices to remote cloud resources or nearby cloudlets, has been widely used for time- and energy-saving, however, it still faces many challenges. As depicted in Fig. 2, offloading decisions in mobile cloud computing may involve multiple factors, such as the resource heterogeneity of mobile devices and cloud services (*resource*), the complexity of mobile applications (*component*), the interruption of heterogeneous wireless networks (*intermittence*) and the characteristic of transferring a large amount of data (*data*), which may significantly impede the improvement of service quality [17]. Each of them will be illustrated in detail as follows:

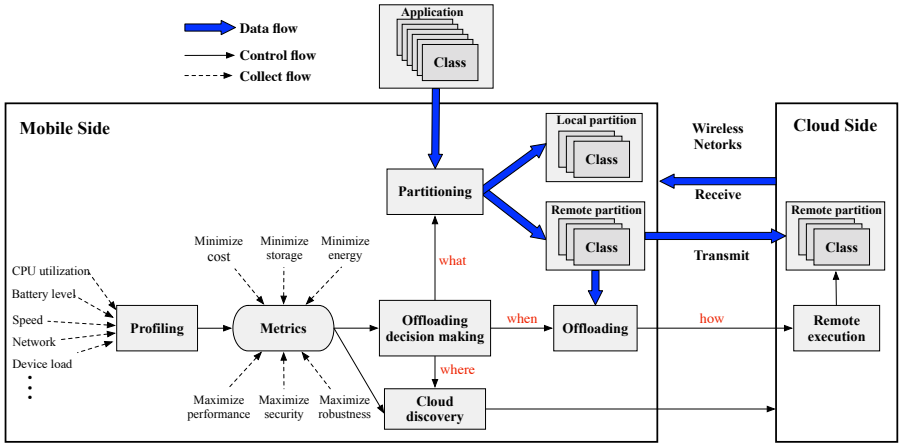


Fig. 1: System architecture of the offloading service [13]

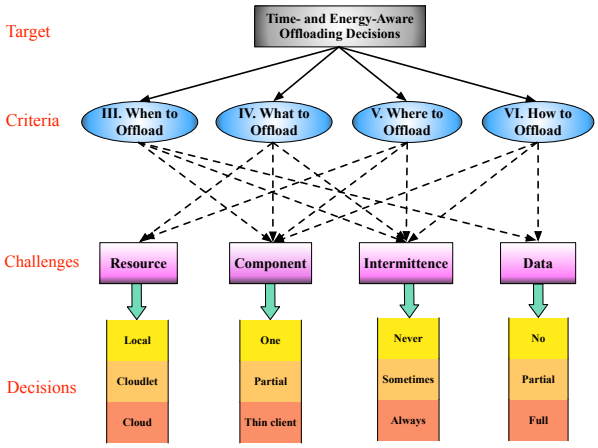


Fig. 2: Challenges and questions for mobile cloud offloading [16]

- **Resource**: The heterogeneous resources include a variability of mobile devices and different cloud vendors providing different services, infrastructures, platforms, and various communication medium and technologies [9]. A mobile device can be thin (without any execution of tasks to be offloaded) or thick (with executions of tasks before being offloaded) [18]. If the resources of mobile devices are not adequate either to execute the application or to achieve the desired performance, mobile cloud offloading will be a good option to short the response time or reduce energy consumption. There are many types of mobile devices, some devices are using android systems while some with iOS systems; some devices are slow while some are very fast; some are equipped with Cellular and WiFi while some can only access WiFi. Besides, a variety of cloud resources can be selected, a mobile device can offload its application either to a remote cloud or a nearby resource-rich middleware like cloudlet [19]. Mobile devices can also discover unknown surrogates nearby by service discovery techniques and then offload tasks to them [18].
- **Component**: A rich mobile application can be very complex, which consists of several components, including offloadable and unoffloadable tasks. Since offloading the whole application to the cloud is not always possible or effective, a decision of which portion of the application should be offloaded and where to place the execution (locally or remotely) should be made based on either the minimum response time or the minimum energy consumption [20], [21]. Further, a diversity of mobile applications such as delay-sensitive and delay-tolerant applications will cause different amounts of computation and communication costs.
- **Intermittence**: The intermittent connectivity of mobile wireless networks has a huge impact on the offloading decisions. Mobile devices use heterogeneous wireless interfaces (with various bandwidths and network latency) to offload tasks to the cloud or access the cloud service. For instance, application tasks that require high responsiveness should not be offloaded using long latency wireless networks; data-intensive tasks should not be offloaded using low-rate networks [18]. Issues such as unstable connectivity and intermittent connectivity of mobile wireless networks may exist due to heterogeneous wireless environments, device mobility and cloud resource availability, which incur high latency and energy consumption [22]. Further, the offloading process will be interrupted when a mobile device moves outside the coverage of wireless networks, executing applications locally seems more advantageous under such circumstances.
- **Data**: There is a huge growth in global mobile data traffic. The amount of data during the offloading process can be divided into three parts: (i) the amount of input data when the task is offloaded to the cloud for remote execution, (ii) the amount of communication data between the mobile device and the remote cloud during remote execution, and (iii) the amount of output data generated from the remote execution of the task [18]. Transferring large amounts of data, such as a large database of video, audio and sensor data, from a mobile device to a remote cloud, will incur additional communication costs, which could be critical for data-intensive tasks that might not benefit from offloading. For example, remote execution may reduce task execution time, however, data transfer during the remote execution may consume more energy than during the local execution. Thus, offloading decisions should

be made: how to effectively transmit data and when to offload data from the mobile devices to the cloud server for time- and energy-savings.

Hence, according to Fig. 2, the optimal decisions between local and remote executions should be carefully made for each application when considering the response time and the energy consumption, as well as the status of mobile wireless networks. To meet the goals of saving energy consumption of mobile devices, improving application performance, or achieving both of them, offloading decisions can be made based on multiple perspectives (e.g., *when*, *what*, *where* and *how*), each describing the attributes a mobile device must meet. In order to minimize the required programmer effort, offloading decisions can be made to determine where to process an application (e.g., local, cloudlet, or cloud); how many subtasks to be offloaded to the cloud (e.g., one, partial or thin client); how much data to be offloaded to the cloud (e.g., no offloading, partial offloading or full offloading); when to do offloading (e.g., never, sometimes or always) [16].

Many research efforts have been devoted to determining the right time, the right component, the right place and the right way to offload. The issues of time and energy saving on mobile devices are becoming increasingly critical. For ease of reference, related works are summarized in Table I.

### C. Main Objectives

Response time and energy consumption are two primary aspects for mobile systems that must be considered when making offloading decisions. Especially for resource-scarce devices, computation offloading is the key to empower these devices and augment their performance (not only energy), i.e., they can run code by means of the cloud that would never run locally. The performance of an offloaded task is judged based on the goals set by the user. Accordingly, we consider three objectives as follows:

- **Shorten Response Time**: From the perspective of a mobile device, response time is defined as the duration between sending the application to the cloud and receiving the results back from the cloud. Reducing the responsiveness is becoming increasingly important, especially for computation-intensive mobile applications. When the amount of computation is very large, it takes such a long time to get results that it fails to meet the user's need, and thus it should be offloaded to the cloud, in order to save time and improve performance. Therefore, we take decisions to offload only if response time will reduce no matter the impact on energy consumption.
- **Reduce Energy Consumption**: The energy spent on the mobile device during the offloading period, is another primary aspect that must be considered. We aim to optimize the energy consumption of a mobile device by estimation and evaluating the tradeoff between the energy consumed by local processing versus offloading the application for remote execution [54]. In this situation, offloading is taken only if energy consumption is expected to reduce no matter the expected impact on response time. Extending battery lifetime is one of the most crucial design objectives of mobile devices because they are usually equipped with limited battery capacity when applications are becoming increasingly complex [55]. Many research efforts have been devoted to minimizing the energy consumption.
- **Achieve combination of the above**: both energy and time saving are crucial design objectives of mobile cloud offloading. We do offloading only if both the response time and energy consumption are expected to improve [32]. It is possible that achieving one offloading goal may affect the realization of the other goal. For example, executing a task on a service node might decrease the response time of the task; however, it might not conserve the mobile device's battery energy. We study the tradeoff between the mean energy consumption and mean response time, which is a non-trivial multi-objective optimization problem. For example, sometimes a short remote execution time for a task is more important though more energy will be spent due to offloading than locally execution, and vice-versa [18].

## III. When to Offload

Remote execution on a cloud server is not an always advantageous strategy because of the need for additional data communications, which may increase the response time and/or energy consumption when the task-related data is transferred [18]. Sometimes the time and energy saved from offloading is not able to cover the extra communication costs between the mobile device and the cloud, in which case, we will opt to execute the application locally instead of offloading it to the cloud. Therefore, mobile cloud offloading is an opportunistic alternative, but not a must. We have to find the right time to offload, e.g., when the wireless network is available, when the amount of communication data is small or when the amount of computation is large [16].

### A. Computation vs. Communication

In Fig. 3 the local computation time is $t_{\mathrm{m}}$, the speedup factor $F$ indicates how powerful a cloud server is in terms of execution speed when compared with that of the mobile device. Mobile cloud offloading has the potential to shorten the execution time and reduce the energy consumption on the mobile device, but the savings from offloading need to exceed the additional communication cost between the mobile device and the cloud [56]. Therefore, offloading makes sense only when on device execution cost is much larger than on server execution cost, which can be decided on time- or energy-saving criterion.

- **Time-Saving**: The time incurred by offloading is the sum of computation time on the cloud server and the predicted costs of transferring the related data and it should be smaller than the execution time on the mobile device in order to save time. Thus, it is worthwhile to offload the computation rather than execute it locally when

$$t_{\mathrm{m}} > \frac{t_{\mathrm{m}}}{F} + \frac{D}{B}, \tag{1}$$

TABLE I: Comparison of Current Offloading Works

| Year | Paper | Main Contribution | Decisions | Partitioning | Time Saving | Energy Saving | Time & Energy Saving |
|---|---|---|---|---|---|---|---|
| 2004 | [1] | Adaptive offloading for pervasive computing | what | √ | √ | | |
| 2007 | [23] | Performance analysis of offloading systems | when/how | | √ | | |
| 2007 | [24] | Context-sensitive energy-efficient offloading | when | | | | √ |
| 2008 | [25] | Use bandwidth to make offloading decision | when | | √ | | |
| 2009 | [26] | Enable mobile phones as interfaces to cloud | when/what | √ | √ | | |
| 2009 | [19] | Cloudlet-based resource-rich mobile computing | where | | √ | | |
| 2010 | [27] | Dynamic partition between mobile devices and clouds | what | √ | | | √ |
| 2010 | [28] | Fine grained energy-aware code offload | what | √ | | √ | |
| 2010 | [29] | Study energy tradeoffs | when | | | √ | |
| 2010 | [30] | Stable and adaptive link selection algorithm | when | | | | √ |
| 2011 | [31] | Elastic execution between mobile devices and clouds | what | √ | | | √ |
| 2012 | [32] | Dynamic resource allocation and parallel execution | where | | | | √ |
| 2012 | [33] | Mobile Augmentation Cloud Services (MACS) | what | √ | √ | | |
| 2012 | [34] | Present a dynamic offloading algorithm | what | √ | | √ | |
| 2012 | [35] | Propose an efficient code partition algorithm | what | √ | √ | | |
| 2013 | [36] | Evaluate computation offloading trade-offs via an application | when/how | | √ | | |
| 2013 | [37] | An Efficient MultiSite Offloading (EMSO) algorithm | where | √ | | √ | |
| 2013 | [38] | Energy-efficient data transmission strategy | how/when | | | | √ |
| 2013 | [39] | An offloading framework, Ternary Decision Maker (TDM) | when | | | | √ |
| 2013 | [5] | Feasibility of mobile cloud systems in a real setting | when/how | | | | √ |
| 2013 | [40] | Measure the energy consumption | what/when/how | √ | | √ | |
| 2013 | [41] | Energy-efficient scheduling policy for offloading | what | √ | | √ | |
| 2014 | [42] | A partitioning scheme taking the bandwidth as a variable | what | √ | | | √ |
| 2014 | [43] | Energy and performance-aware task scheduling | what | √ | | | √ |
| 2014 | [44] | A queueing analytic model for delayed offloading | when/how | √ | | | √ |
| 2014 | [45] | A partitioning based workflow optimization algorithm | what | √ | √ | | |
| 2014 | [46] | On-demand service offloading with a decision support system | when/how | | | | √ |
| 2015 | [47] | A stochastic model for dynamic offloading | when/how | | √ | | |
| 2015 | [22] | A Genetic Algorithm (GA) based offloading method | what/when | √ | | | √ |
| 2015 | [48] | A model for computational offloading in realistic setups | when | | | | √ |
| 2016 | [49] | Performance analysis of offloading in heterogeneous networks | how | | √ | | |
| 2016 | [50] | Cloudlet deployment in local wireless networks | where | | | | √ |
| 2017 | [51] | A Client Aware Certification (CAC) model | when | | √ | | |
| 2017 | [52] | Analytical model of hybrid cloud offloading | where/what | | √ | | |
| 2017 | [53] | A novel statistical cost model for applications to be offloaded | what/when | √ | √ | | |

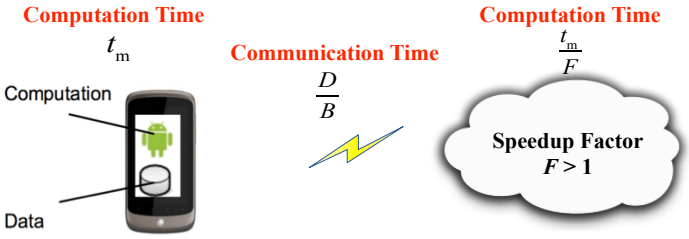


Fig. 3: The offloading process in mobile cloud computing

which holds true under several conditions: large $F$ that the server is much faster than the mobile device, small $D$ that only a small amount of data is exchanged, and large $B$ that the network bandwidth between the mobile device and the server is high [55].

- **Energy-Saving**: A performance seeking offload cannot guarantee energy reducing, as the energy overhead of data transfer may exceed the energy savings from reduced CPU usage [57]. The energy spent due to offloading must be smaller than the energy consumed by the mobile device, which has to satisfy:

$$p_m t_m > p_{idle} \frac{t_m}{F} + p_{tr} \frac{D}{B}, \tag{2}$$

where $p_m$ is the power for computing at the device, $p_{idle}$ is the power when the device is idle and $p_{tr}$ is the power for sending and receiving data. Here for easy of understanding, only gives a simple and intuitive explanation. More complex power models can be found in [58], [59], [60]. MCO could potentially save energy for mobile users, but not all applications were energy-efficient when migrated to the cloud. It depends on whether the computational cost saved due to offloading outperforms the extra communication cost or note.

Offloading decisions depend on whether the mobile device benefits from offloading or not. As sometimes it may be not worth offloading at all, decisions have to be made when encountering with large communication data or low bandwidth. Some works [29], [55], [15] have tried to determine when it is optimal to offload computational tasks to a dedicated server, whereas on the contrary, local execution is more advisable. As shown in Fig. 4, decisions should be made according to the ratio of communication versus computation required

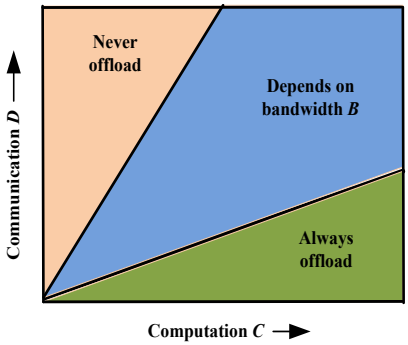


Fig. 4: Tradeoff between computation and communication [29]

by the application, by dividing three intervals, namely, *never offload*, *tradeoff* and *always offload*. There exists a trade-off between computation and communication. When a large amount of computation combined with a very small amount of communication such as face recognition and chess game, it is better always offload for such applications. However, a small amount of computation combined with a large amount of communication such as image searching, it should preferably never offload [36]. In the middle area, it depends on the network bandwidth $B$.

### B. Interrupted vs. Uninterrupted

Due to unstable network connections or cloud conditions, the execution of an offloading task may suffer from failures.

- **Connection Failures**: The access to the cloud is usually influenced by uncontrollable factors, such as the instability and intermittency of wireless networks. The user mobility and quality of wireless connection may lead to connection failures. When a network connection suddenly breaks down, an offloaded computation will suffer severe performance degradation since the request may be dropped due to connection failures.
- **Cloud Failures**: The cloud servers cannot be seen as perfectly reliable systems since the reliability is typically ensured through the SLA negotiated with the provider. The cloud servers sometimes will be unreachable during data center downtime. According to a survey, 93% of enterprises that suffered from a data center downtime for more than 10 days, filed for bankruptcy within a year of the outage[1].

As shown in Fig. 5, once a failure occurs, the offloading action will be interrupted and the offloading task will be re-executed from the scratch. The mobile device has to wait for the recovery, but how long should it wait? To deal with this problem, we set a soft deadline $T_{\text{deadline}}$. There are two options, if we wait some time ($\leq T_{\text{deadline}}$) and the connection is repaired, the task will retry offloading in this case; if the connection is not repaired before the deadline expires, then the task will be executed locally on the mobile device.

This system can be described as an $M/M/1$ modulated queue, where it suffers occasionally a disastrous breakdown occurring in the offloading phase. A failure of the system

[1]https://lifelinedatacenters.com/data-center/data-center-downtime/

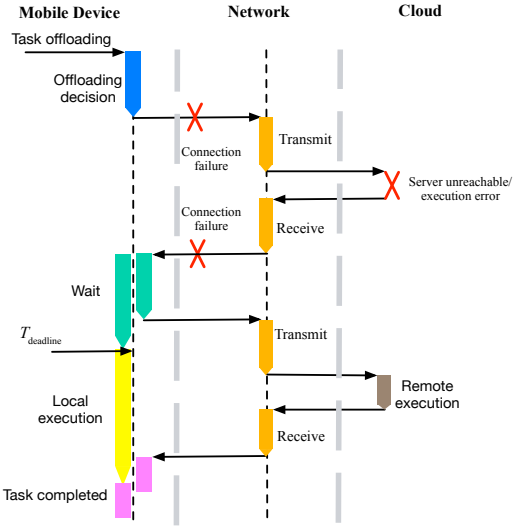


Fig. 5: The offloading model with failures

rejects all jobs present, and a repair process starts immediately. When the connection is down and undergoing a repair process, new arrivals become impatient: each individual job, upon arrival, activates an 'impatience timer', exponentially distributed with an abandonment rate $\xi$ (=$1/T_{\text{deadline}}$). If the connection is repaired before the assigned deadline $T_d$ has expired, it is then offloaded to the cloud. If the system does not change its environment from the repair phase to the offloading phase before $T_{\text{deadline}}$ expires, the job is arranged for immediate local processing rather than offloaded to the cloud. Further, optimality analysis of the energy-performance tradeoff for delayed offloading systems has been analyzed, which captures both energy and performance metrics and also intermittently available wireless networks [47], [49]. They tried to answer the following questions: (i) Given a deadline, how to choose the best offloading model and what parameters do the response time and energy depend on? (ii) How to choose the deadlines to optimize the metric by trading off the response time and energy consumption?

### C. On-the-Spot Offloading vs. Delayed Offloading

From Fig. 6, there is a queue of data to be offloaded from a mobile device to the remote cloud server. By dynamically control the relationship between energy cost, data queue backlog and estimated rate, we can optimally determine when to make transmission decisions to save energy.

- **On-the-Spot Offloading**: When there is any wireless network available, all traffic is immediately offloaded to the remote cloud, regardless of the network quality [61]. Since this kind of offloading strategy does not take the network condition into account and the poor-quality inferior channels may be used, which could be a waste of energy.
- **Delayed Offloading**: When there is currently no high-quality network available, the offloading process can be delayed up to a given deadline, or until a suitable network becomes available [44]. Instead of offloading the task to the cloud directly, we can choose not to offload when the

link is bad or when the amount of data is large. This is an energy-efficient offloading strategy since we take the network conditions into account.

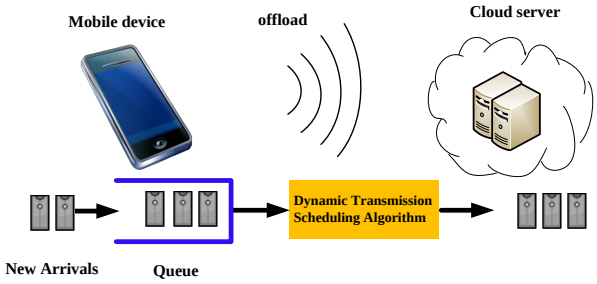


Fig. 6: Framework of delayed offloading [62]

Given a set of available links, such as 3G/EDGE, 4G LTE, WiFi access points and even 5G, energy information and data arrival queues, we can determine whether to use any link or which link to use for data transfer, while keeping the transmission delay bounded. Ultra-dense small cell base stations will be deployed in future wireless networks technologies (such as 5G), so the data has a high probability to be offloaded through the small cell base stations [63], [64]. Dynamic transmission scheduling algorithms based on the Lyapunov optimization have been recently proposed in [30], [62], which determines when and on which network to offload data so that the energy cost is minimized by leveraging delay tolerance. They used the transmission energy cost as a penalty function and dynamically determined when offloading decisions are made in order to minimize the energy cost by accepting a small delay (queue length).

Different types of applications usually give different relative importance to both factors of response time and energy consumption. There exists a fundamental tradeoff between the mean energy consumption and mean response time for different applications [30], [16]. As for *Delay-Tolerant Applications* (e.g., iCloud, Dropbox and participatory sensing) deal with video, audio, sensor data, or access large databases on the Internet, which are less sensitive to network delays. Thus, response time is less critical and optimizing energy usage is more relevant. But as for *Delay-Sensitive Applications* (e.g., language translator, face recognition, video conferencing, vehicular communications), mobile users desire a fast response when using applications. The offloading scheme in which cloud services are available with short network latencies (e.g., WiFi networks) can serve in a better way by providing low response time [16].

Using WiFi to offload large volumes of data from a mobile device to the cloud can be more energy-efficient than cellular radio. Since WiFi connections are not always available, we should decide when to transmit data and across which network interface. Recently, several groups have worked on optimizing the tradeoff between the energy consumption and response time. Rahmati *et al*. [24] suggested seamless offloading operation by switching between several transmission technologies, and examined the tradeoff between energy consumption for WiFi search and transmission efficiency when the WiFi network was intermittently available. Energy-efficient delayed network selection has been used to optimize the tradeoff between energy usage and delay in data transmission by intentionally deferring data transmission until the device meets an energy-efficient network [38]. Researchers have further suggested the use of "delayed offloading": if no WiFi connection is available, (some) traffic can be delayed up to a chosen deadline, or until WiFi becomes available [44].

## IV. What to Offload

Usually mobile applications can be decomposed into a set of fine-grained or coarse-grained tasks which consists of sequential and parallel components. It is not always necessary or effective to offload all computation components to the remote cloud since sometimes high communication delay will be generated or because some tasks must access local features. Therefore, we should find the right component to offload, i.e., judiciously determine what should be deployed on the cloud server and which parts of the application should be left on the mobile device to achieve a particular performance target such as the lowest response time or the least energy consumption [20].

### A. Classification of Application Tasks

Different applications emerge in a mobile device according to some process and each consists of several tasks. Since not all the application tasks are suitable for being offloaded to the cloud server for remote execution, they need to be weighted and distinguished as:

- **Unoffloadable Tasks**: Some should be unconditionally executed locally on the mobile device, either because transferring relevant information would take tremendous time and energy or because these tasks must access local components (e.g., camera, GPS, user interface, accelerometer or other sensors) [28]. Tasks that might cause security issues when executed in a different place should also not be offloaded (e.g., e-commerce). Local processing consumes the battery power of the device, but there are no communication costs or delays.
- **Offloadable Tasks**: Some application components are flexible tasks that can be processed either locally on the processor of the mobile device, or remotely in a cloud infrastructure. Many tasks fall into this category, which are hard to be simply classified as "tasks suitable for offloading" or "tasks suitable for local processing", the offloading decision depends on whether the communication costs outweigh the difference between local and remote costs or not [55].

We do not need to take offloading decisions for unoffloadable components. However, as for offloadable ones, since offloading all tasks of an application to the remote cloud is not necessary or effective under all circumstances, it is worth considering what should be executed locally on the mobile device and what should be offloaded onto the remote cloud for execution based on available networks, response time or energy consumption. The mobile device has to take offloading decisions based on the results of the dynamic optimization problem.

### B. Application Partitioning

Application partitioning is a method to split the execution of the application between the mobile side and cloud side so that the total execution cost is minimized [65]. It plays a critical role in high-performance offloading systemsg. The more accurate and lightweight the profiling information is, the more correct decisions can be made, and the lower overhead is introduced [32].

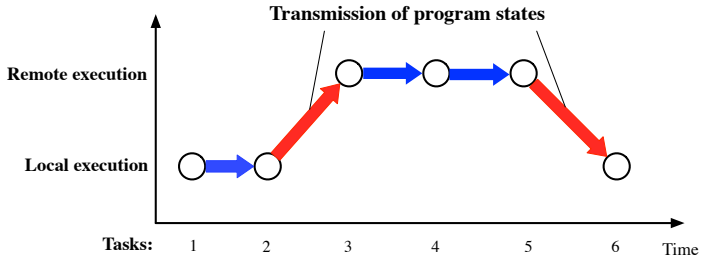


Fig. 7: The execution flow of an application

Application partitioning is used to decide which part of the computations may be advantageously offloaded and which not. As shown in Fig. 7, tasks 1, 2 and 6 are processed locally on the mobile device while tasks 3, 4 and 5 are offloaded to the cloud server for remote execution. Through partitioning, a mobile device can benefit most from offloading. Applications can be partitioned statically during development or dynamically during execution:

- **Static Partitioning**: It is determined beforehand which parts of the application should run locally and which parts should be offloaded, depending on contextual parameters, such as computational intensity of each module, the size of data and state to exchange, battery level, delay constraints and channel state [66]. The optimal partitioning for offloading is calculated based on the estimation of communication cost and computational costs before the program execution. The former depends on the size of transmitted data and the network bandwidth, while the latter is impacted by the computation time [20]. The advantage of static partitioning is that it only requirs a low overhead during execution and thus applies to a fixed number of partitions, however it works well only if the parameters related to the offloading decisions are accurately known in advance or predicted [22].
- **Dynamic Partitioning**: The requirement of resources for a task may change in its input data and the user-defined goals (e.g., response time, battery consumption). Also, the availability of resources may change at the service nodes (available CPU power, memory, file cache, etc.) and at the wireless network (bandwidth, network latency, etc.) [18]. Thus, optimal partitioning decisions must be made dynamically at runtime in order to adapt to different network conditions, server state, delay constraints, and so on. Given the variability of the wireless channel, dynamic partitioning seems more appropriate, but it has an associated higher signaling overhead, which must be taken under control [67].

Among the sets of partitions offered by the partitioning result, a mobile device should judiciously determine what portion of an application is worth offloading to the cloud and what should be executed locally [20]. An offloading strategy selects a subset of tasks to be offloaded, considering the balance between how much the offloading saves and how much extra cost is induced. According to different CPU speeds of the mobile devices, network bandwidths, transmission data size, and the speed of the cloud servers, we will have different partitioning results. Therefore, the partitioning algorithm should be dynamically adapted to changing environment.

### C. Partitioning Algortihms

CloneCloud [31] used a combination of static analysis and dynamic profiling to partition applications automatically at a fine granularity while optimizing execution time and energy use for a target computation and communication environment. However, this approach only considers limited input/environmental conditions in the offline pre-processing and needs to be bootstrapped for every new application built. Dynamic partitioning of applications between weak devices and clouds was presented in [27] and [42], to better support applications running on diverse devices in different environments. They addressed how dynamic partitioning can address these heterogeneity problems by taking the bandwidth as a variable. ThinkAir [32] exploited the concept of smartphone virtualization in the cloud and provided method-level computation offloading and enhanced the power of mobile cloud computing by parallelizing method execution using multiple VM images.

Calculations can naturally be described as graphs in which vertices represent computational costs and edges reflect communication costs [68]. By partitioning the vertices of a graph, the calculation can be divided among processors of local mobile devices and remote cloud servers. Traditional graph partitioning algorithms (e.g., [69], [70], [71]) cannot be applied directly to the mobile offloading systems, because they only consider the weights on the edges of the graph, neglecting the weight of each node.

We can adopt partitioning technologies to identify offloaded parts for energy saving. The energy cost of each application task was profiled. Then a cost graph is constructed according to the profiling results, in which each node represented a task to be performed, and each edge indicated the data to be transmitted between the mobile device and the remote cloud. Finally, the remote parts were executed on remote cloud servers for reducing energy consumption. Some works [72], [20] have explored the methods of how to deploy application tasks in a more optimal way, by dynamically and automatically determining which portions of the application should be offloaded to the cloud, what should be performed on the mobile device.

Estimating the energy consumed on the mobile device for task offloading to the cloud is fundamental to making a correct offloading decision [59]. Some works [29], [40] built energy models to approximate the energy consumption of offloading. The energy models can be used to construct the aforementioned cost graph or make offloading decisions. However,

they did not provide an effective method to obtain optimal offloading decisions. MAUI [28] was a system that enables energy-aware offloading of mobile code to the infrastructure by deciding at runtime which methods should be remotely executed, and achieves the best energy savings possible under the mobile device's current connectivity constraints. Its main aim is to optimize energy consumption of a mobile device, by estimating and trading off the energy consumed by local processing vs. transmission of code and data for remote execution.

Some works [73], [41] considered a response time constraint when partitioning application tasks for execution on mobile devices and servers, which is an important issue for many interactive applications. To achieve energy saving while satisfying a given deadline, some works [34], [74] showed low complexity to solve the problem of offloading decision making (i.e., to determine which software components to execute remotely under mobile network environments). Beraldi *et al*. [5] showed that rather than always offloading the whole application remotely, running partial components locally can be more advantageous. They proposed a novel generic architecture that can be integrated into any mobile application, which aims to automate the offloading decision and improve the application's response time while minimizing the overall energy consumed by the mobile device. The partitioning algorithm introduced in [35] aims at reducing the response time of tasks on mobile devices. It finds the offloading and integrating points in a sequence of calls by depth-first search and a linear time searching scheme, which can achieve low user-perceived latency while greatly reducing the partitioning computation on the cloud. Some application partitioning solutions [26], [1], [33] heavily depend upon programmers and middleware to partition the applications, which limits their uses.

## V. Where to offload

We need to find suitable cloud service to carry out offloading best, i.e., to find the right place to offload. With the development of MCC and MEC, where to offload has become a crucial issue.

### A. Multi-Criteria Decision Making

A variety of Clouds with different characteristics are emerging these days for data storage and processing, e.g., Amazon EC2, Apple iCloud, and Google App Engine. Such systems use proprietary cloud platforms to provide different types of services. For example, cloud data centers designed specifically for healthcare services can provide a platform for big data storage and parallel computing capabilities for data mining [75]. Offloading the same program to different clouds may perform different amounts of computing within the same duration due to the different speeds of cloud servers, and may cost different communication time due to the wireless network and cloud's availability. Therefore, a method for optimal cloud service selection is needed [76].

The goal of cloud service selection is to find an optimal cloud among a certain class of clouds that provide the same service, which best carries out the offloaded tasks [77]. As shown in Fig. 8, three basic steps are required in the process of cloud service selection, namely, *matching*, *ranking* and *selecting*.

- **Matching**: To find a list of available cloud services that are functionally matched with a service request by a mobile user. On the mobile device side, upon receipt of an offloading request, the *service request* module invokes the *cloud discover* module to find an appropriate cloud service according to the task of *SLA management* that keeps track of SLA of customers with cloud providers and their fulfillment history. The candidate cloud services are registered based on the collected information in the *cloud register* module.
- **Ranking**: To evaluate and rank the available cloud services according to QoS values and the results of criteria and sub-criteria calculation. The *criteria calculator* module depends on the tasks of qualitative and quantitative measurements. Qualitative criteria are those that cannot be quantified and are mostly inferred based on previous user's experiences, e.g., *security*. Quantitative criteria are those that be measured by using software and hardware monitoring tools [78], e.g., *bandwidth*, *VM cost* and *speed*.
- **Selecting**: The *decision maker* module is invoked to choose the optimal cloud service according to the ranked list of cloud services. And then the *offloading invoker* module is triggered to partition the application into local and remote partitions, and the latter is then offloaded to the selected cloud.

There are three decision hierarchies listed in Fig. 9. The first level is called target hierarchy, meaning what the object is. Here, it aims at finding the optimal cloud service amongst available cloud services which satisfy the essential requirements of the mobile device. The second level is called criteria hierarchy, and five criteria: *performance*, *security*, *bandwidth*, *availability* and *cost* are considered for cloud service selection. The criteria can be classified into two categories: subjective criteria and objective criteria. The former is defined in linguistic/qualitative terms while the latter has a monetary/quantitative definition. Root criteria can be made up of sub-criteria. The bottom level is named decision hierarchy, in which we can make the final decision in choosing one of the alternative clouds based on the analysis in criteria hierarchy.

The selection process can be a hard task since a variety of data needs to be analyzed and many factors need to be considered. Some works [77], [76] combine the methods of analytic hierarchy process (AHP) and fuzzy technique for order preference by similarity to ideal solution (TOPSIS), which are ideal ways to do multi-criteria decision making. AHP is employed to obtain weights of the criteria for each cloud service and fuzzy TOPSIS is to determine the priorities of the alternative clouds in the decision-making process [79]. Thus, we should make offloading decisions carefully to determine which resource to use and then offload tasks to the most appropriate server according to the energy and computing demand of the task.

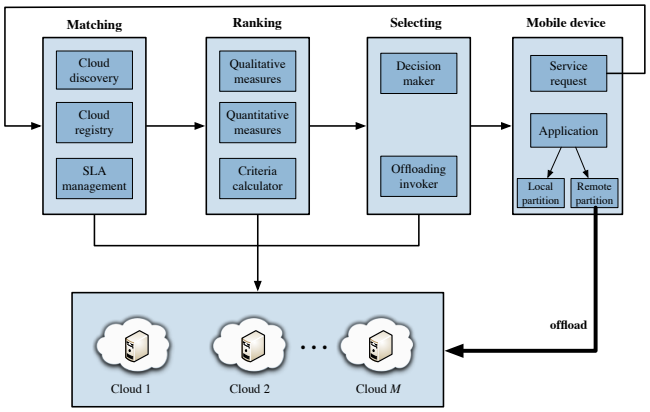


Fig. 8: Steps of selecting an optimal offloading destination [16]

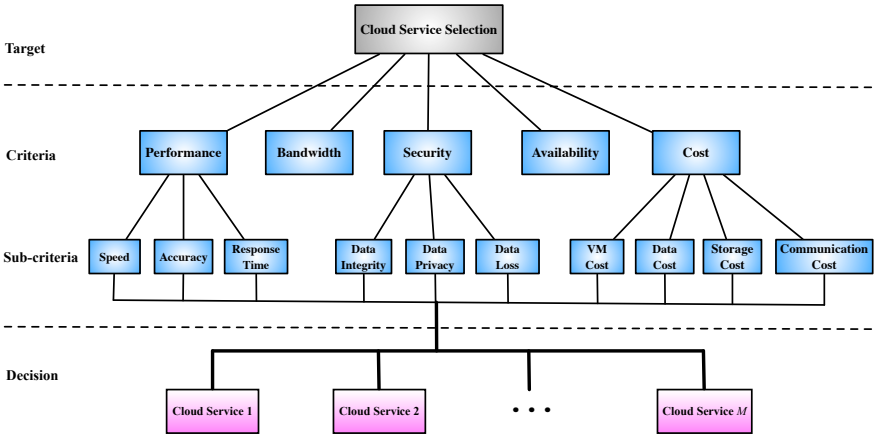


Fig. 9: The decision hierarchy of cloud service selection [77]

### *B. Cloudlet-based Decision Making*

Except for several similar cloud services (from different cloud vendors) that can be offered to a mobile device, nearby cloudlets are also alternative destinations for offloading. Mobile users can also offload applications to nearby mobile resource-rich devices to reduce energy consumption and improve performance. Satyanarayanan *et al.* [19] proposed a VM-based cloudlet in mobile computing, to which a mobile device connects over a WLAN network, with the argument against the use of the cloud due to higher latency and lower available bandwidth when connecting. In essence, Cloudlets make use of mobile devices simply as a thin-client to access local resources, rather than using the mobile devices' capabilities directly and offloading only when required. A Mobile Cloud Middleware (MCM) was also introduced in [80] as an intermediary between the mobile device and the cloud in order to manage the asynchronous delegation of mobile tasks to cloud resources and decrease the time it takes to offload tasks from mobile devices to the cloud.

Fig. 10 illustrates a generic MCO system, organized as a two or three-level hierarchy:

- **Two-Level Offloading**: Rather than running applications locally and directly requesting data from content providers, a mobile device can offload parts of its workload to a cloud server via one or more communication networks, taking advantage of the abundant cloud resources to help gather, store, and process data. This kind of offloading scheme depends critically on a reliable end-to-end communication and the availability of the cloud [38]. In addition, it suffers from high network access latency and low network bandwidth. Access to the cloud is often affected by uncontrollable factors, such as the instability and intermittency of wireless networks.
- **Three-Level Offloading**: Rather than relying on a remote cloud to address the resource poverty of a mobile device, we can use a nearby resource-rich middleware (e.g., cloudlet, MEC or local cloud) via a WLAN hotspot to

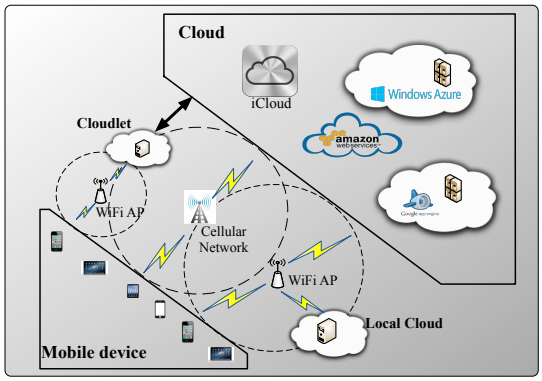


Fig. 10: A general offloading model

decrease latency and lower battery consumption [19]. The application task is first offloaded to the middleware, which is well-connected to the internet and available for use by nearby mobile devices, and then it is migrated onto the remote cloud through a stable internet connection. This architecture reduces latency by using a single-hop network and potentially saves battery by using WiFi or short-range radio instead of broadband wireless which typically consumes more energy [28].

Therefore, an application can deploy their components on multiple application processing nodes such as mobile device, cloudlet and cloud, i.e., there could be multiple offloading destinations and targets [81]. Based on the computational requirements and constraints, offloading decisions should be made on where to offload: (i) tasks are executed locally on the mobile device; (ii) tasks are executed on a nearby cloudlet or edge server [10] with data transferred between the mobile device and the cloudlet, e.g., via Bluetooth; (iii) tasks are executed on a remote cloud server with data transferred between the mobile device and the cloud, e.g., via a cellular network.

### C. Hybrid Offloading Decision-Making

Considering the cloud is far from the mobile users, Mobile Edge Computing (MEC) has been proposed to shorten the delay of offloading data. Compared with the cloud, the mobile edge is much closer to the user and thus has much low latency or response time. However, compared with conventional MCCs, MECs are constrained by computing capacity, especially under the scenario of dense population.

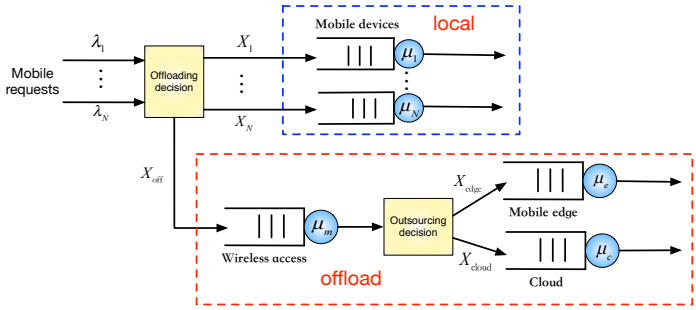


Fig. 11: A hybrid scheme of where to offload (Mobile edge or Cloud) [52]

Therefore, we should consider both the advantages of the Mobile edge and Cloud, when making offloading decisions on where to offload. Hybrid schemes of where to offload (either Mobile edge or Cloud) has been proposed in recent research works. In [52], the authors proposed a Cloud Assisted Mobile Edge computing (CAME) framework as shown in Fig. 11, in which cloud resources were leased to enhance the system computing capacity while mobile edge resources were used to reduce the latency. Specifically, the system delay was analyzed by modeling the CAME system as a queuing network. Considering the heterogeneity of computation resources and mobile tasks, offloading decisions were made which optimized the usage of cloud resources and balances the workloads between the cloud and the mobile edge [52].

Energy saving from mobile cloud offloading is not guaranteed if the evoked data transfers via wireless networks consume an unpredictable amount of energy. Therefore, running a certain part of the application locally on the mobile device can be more advantageous and may save both energy and response time, especially in the presence of intermittent wireless connectivity. The offloading inference engine proposed in [1] can adaptively make decisions at runtime, dynamically partition an application and offload part of the application execution to a powerful nearby surrogate like Cloudlet, MCM or edge server. Some recent works [80], [16] developed dynamic offloading decision algorithms for mobile users when taking the nearby cloudlet or middleware into account. They derived an adaptive offloading decision algorithm based on Lyapunov optimization, which determines where to perform each application task (locally, cloud or cloudlet) such that the energy consumption is minimized with a low delay penalty.

## VI. How to Offload

Offloading can be performed statically or dynamically via different wireless networks like WLAN and cellular networks. Since the transmission techniques differ in energy requirements and speeds, we should determine how to leverage the complementary strength of WiFi and cellular networks by choosing heterogeneous wireless interfaces for offloading, i.e., to find the right way to offload.

### A. Heterogeneous Wireless Environments

| | Cellular | WiFi |
|---|---|---|
| Delay | High | Low |
| Availability | High | Low |
| Energy-Efficiency | Low | High |

Fig. 12: Comparison of WiFi and cellular networks

Mobile devices, such as smartphones usually have multiple wireless interfaces (e.g., WiFi and cellular networks) with varying availability, delay, and energy cost for data transfer. The differences between them are summarized in Fig. 12. Thus, the cellular interface usually has higher availability than WiFi and can providenearly ubiquitous coverage for mobile devices in a wide area, but it has lower data transmission rate and is less energy-efficient than the WiFi interface for transmitting the same quantity of data [38].

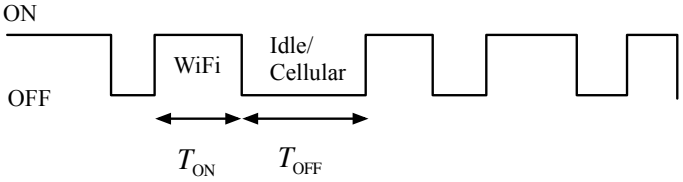


Fig. 13: The WiFi network availability model

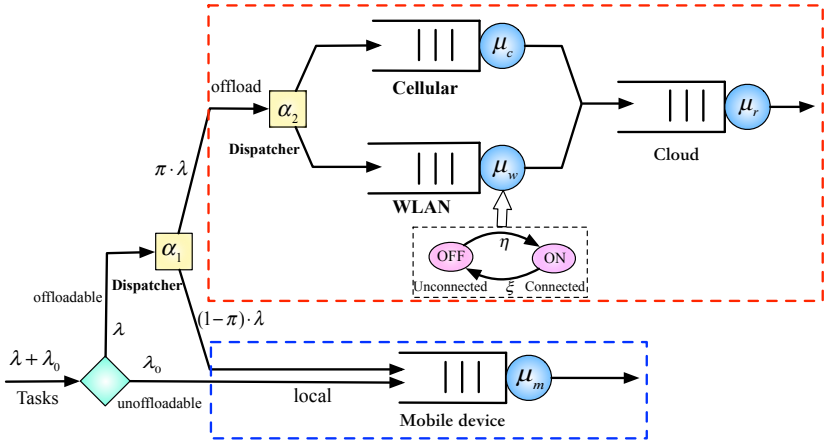


Fig. 14: A queueing model for mobile cloud offloading systems [87]

To facilitate the analysis of decision-making, most research works [82], [47], [83] assume that cellular networks are always available to mobile users, whereas the availability of WiFi networks depends on the location. Mobile users move in and out of a WiFi coverage area. The time variation of the WiFi connection state can be modelled by the ON-OFF alternating renewal process $\left(T_{\text{ON}}^{(i)},\ T_{\text{OFF}}^{(i)}\right),\ i \geq 1$, as shown in Fig. 13 [49]. The ON periods represent the presence of the WiFi connectivity, while the OFF periods denote the interruption of the WiFi connectivity [84]. During the latter periods data is either not transmitted (the interface is idle) or it is transmitted only through the cellular network. The duration of each ON period $T_{\text{ON}}^{(i)}$, is assumed to be an exponentially distributed random variable and independent of the duration of other ON or OFF periods [61]. Further, the WiFi availability ratio ($AR$) can be defined as $AR = \frac{\mathbb{E}[T_{\text{ON}}]}{\mathbb{E}[T_{\text{ON}}]+\mathbb{E}[T_{\text{OFF}}]}$.

Mobile cloud offloading migrates heavy computation from mobile devices to powerful cloud servers using one or more of possibly several available wireless networks. There are several ways to offload tasks to a dedicated resource, either using a cellular connection or via an intermittently available WLAN hotspot [47]. The unstable connectivity of wireless links, which is caused by the mobile nature of mobile devices, plays an important role in the offloading decision-decision making process. A weak and even intermittent wireless network affects the offloading process seriously and raise power consumption on the mobile device. Therefore, how to offload tasks through different wireless channels to achieve an overall optimal object is worth studying.

### *B. The Queueing Model*

Many recent works [47], [85], [84], [49] consider a queueing model for mobile offloading systems as depicted in Fig. 14. The mobile device, the cloud, and the wireless networks are represented as queueing nodes to capture the resource contention and delay on these systems [86].

As indicated in Fig. 14, job arrivals at the mobile device are assumed to follow a Poisson process with an average arrival rate of $\lambda+\lambda_0$, where $\lambda$ and $\lambda_0$ are the rates of offloadable and unoffloadable jobs, respectively. The arrival rate is based on the behavior of the application. The unoffloadable jobs with an arrival rate $\lambda_0$ are unconditionally executed locally on the mobile device. As for the offloadable ones with an arrival rate $\lambda$, the mobile device chooses to offload each job with a probability $0 \leq \pi \leq 1$. In the extreme cases, if $\pi = 0$ all the offloadable jobs are executed locally, and if $\pi = 1$ they are all offloaded to the cloud. According to the properties of the Poisson distribution [88], the jobs are offloaded to the cloud following a Poisson process with an average arrival rate of $\lambda_c = \pi \cdot \lambda$, the offloading rate. Similarly, jobs that are proceed locally instead of being offloaded follow a Poisson process with rate $\lambda_m = (1-\pi) \cdot \lambda$.

There are two dispatchers: $\alpha_1$ is used to allocate offloadable jobs either to the cloud or the mobile device, while $\alpha_2$ is to offload the jobs either via a cellular connection or a WLAN network to the cloud. The total cost, in terms of energy or response time for processing all offloadable jobs, is composed of remote costs (sending some jobs to the cloud and waiting for the cloud to complete them), and local costs (processing the remaining jobs locally on the mobile device) [87].

Offloading decisions can be made in a fixed, static manner while others are able to perform offloading in accordance with the dynamic behavior of the application [89].

Some researchers apply different offloading policies (static and dynamic), where arriving jobs are processed either locally on the mobile device or remotely on a cloud server. The dynamic offloading policy considers the increase in each queue and the change in a metric that newly arriving jobs bring in should they be assigned to that queue, while the static policy does not capture the dynamic increase [90]. A stochastic model for dynamic offloading has been developed in [47] using various performance metrics and also intermittently available access links.

Some researchers develop different offloading strategies (uninterrupted and interrupted): (a) the uninterrupted offloading strategy uses WiFi whenever possible, but switches to a cellular interface if no WiFi connection exists [61], and data is continuously transmitted while switching between different channels; (b) the interrupted offloading strategy assign jobs upon arrival to one of two parallel queues which describe cellular or WiFi transmission. Data transmission of the WiFi queue can be interrupted for short periods when the connection

is lost [91]. A comparative analysis of these strategies has been performed.

## VII. Conclusion

This paper presents a comprehensive survey of current research works conducted on decision-making for mobile cloud offloading. Offloading decisions play a crucial role in improving the performance of applications executing on the resource-constrained mobile devices and saving energy at the side of mobile devices.

The resource heterogeneity of mobile devices and cloud services, the complexity of mobile applications, the interruption of heterogeneous wireless networks and the characteristic of transferring a large amount of data have seriously prevented mobile cloud offloading from being widely adopted.

To address these challenges, the time- and energy-aware offloading decisions have to be made based on multiple perspectives. A good offloading decision is made by determining the right time to offload (*when to offload*) under different conditions of the device, such as available bandwidth, amount of data to be transferred, and energy to execute; choosing the right component to offload by splitting a specific application into local and remote parts (*what to offload*); finding the right place in which to be offloaded (*where to offload*) under different cloud resource conditions and determining the right path to offload (*how to offload*) by balancing different communication networks.

## Acknowledgment

This work was supported by Huawei Innovation Research Program (HIRP) grant funded by the Huawei Technologies Co. Ltd (No. HIRPO2017050307).

## References

[1] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading for pervasive computing," *Pervasive Computing, IEEE*, vol. 3, no. 3, pp. 66–73, 2004.

[2] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Unleashing the power of mobile cloud computing using thinkair," *arXiv preprint arXiv:1105.3232*, 2011.

[3] F. Xia, F. Ding, J. Li, X. Kong, L. T. Yang, and J. Ma, "Phone2cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing," *Information Systems Frontiers*, vol. 16, no. 1, pp. 95–111, 2014.

[4] K. Lee and I. Shin, "User mobility-aware decision making for mobile computation offloading," in *Cyber-Physical Systems, Networks, and Applications (CPSNA), 1st International Conference on*, pp. 116–119, IEEE, 2013.

[5] R. Beraldi, K. Massri, M. Abderrahmen, and H. Alnuweiri, "Towards automating mobile cloud computing offloading decisions: An experimental approach," in *The Eighth International Conference on Systems and Networks Communications*, 2013.

[6] A. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Communications Surveys & Tutorials, Accepted For Publications*, vol. 16, no. 1, pp. 393–413, 2014.

[7] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1294–1313, 2013.

[8] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 337–368, 2014.

[9] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 369–392, 2014.

[10] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[11] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, pp. –, 2017.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, pp. –, 2017.

[13] K. Yang, S. Ou, and H.-H. Chen, "On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications," *Communications Magazine, IEEE*, vol. 46, no. 1, pp. 56–63, 2008.

[14] M. Segata, B. Bloessl, C. Sommer, and F. Dressler, "Towards energy efficient smart phone applications: Energy models for offloading tasks into the cloud," in *Communications (ICC), IEEE International Conference on*, pp. 2394–2399, 2014.

[15] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *Communications Workshops (ICC), International Conference on*, pp. 728–732, IEEE, 2013.

[16] H. Wu, *Analysis of offloading decision making in mobile cloud computing*. PhD thesis, Freie Universität Berlin, 2015.

[17] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: from concept to practice and beyond," *Communications Magazine, IEEE*, vol. 53, no. 3, pp. 80–88, 2015.

[18] M. P. S. Nir, *Scalable resource augmentation for mobile devices*. PhD thesis, Carleton University Ottawa, 2014.

[19] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[20] H. Wu, W. Knottenbelt, K. Wolter, and Y. Sun, "An optimal offloading partitioning algorithm in mobile cloud computing," in *International Conference on Quantitative Evaluation of Systems*, pp. 311–328, Springer, 2016.

[21] U. Nandhini, L. Tamilselvan, U. S, and S. Nancy, "Client aware opportunistic framework of rich mobile applications in mobile cloud environment," *International Journal of u- and e- Service, Science and Technology*, vol. 10, no. 1, pp. 281–288, 2017.

[22] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3329, 2015.

[23] S. Ou, K. Yang, A. Liotta, and L. Hu, "Performance analysis of offloading systems in mobile wireless environments," in *Communications. ICC'07. IEEE International Conference on*, pp. 1821–1826, 2007.

[24] A. Rahmati and L. Zhong, "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *Proceedings of the 5th international conference on Mobile systems, applications and services*, pp. 165–178, ACM, 2007.

[25] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *Parallel and Distributed Processing (IPDPS), International Symposium on*, pp. 1–8, IEEE, 2008.

[26] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: enabling mobile phones as interfaces to cloud applications," in *Middleware*, pp. 83–102, Springer, 2009.

[27] B.-G. Chun and P. Maniatis, "Dynamically partitioning applications between weak devices and clouds," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, ACM, 2010.

[28] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 49–62, ACM, 2010.

[29] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?," *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[30] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 255–270, ACM, 2010.

[31] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*, pp. 301–314, ACM, 2011.

[32] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proceedings of the INFOCOM*, pp. 945–953, IEEE, 2012.

[33] D. Kovachev and R. Klamma, "Framework for computation offloading in mobile cloud computing," *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 1, no. 7, pp. 6–15, 2012.

[34] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 6, pp. 1991–1995, 2012.

[35] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: an efficient code partition algorithm for mobile cloud computing," in *Cloud Networking (CLOUDNET), 1st International Conference on*, pp. 80–86, IEEE, 2012.

[36] J. Luzuriaga, J. C. Cano, C. Calafate, and P. Manzoni, "Evaluating computation offloading trade-offs in mobile cloud computing: A sample application," in *the Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 138–143, 2013.

[37] R. Niu, W. Song, and Y. Liu, "An energy-efficient multisite offloading algorithm for mobile devices," *International Journal of Distributed Sensor Networks*, 2013.

[38] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, Y. Qu, and B. Li, "eTime: energy-efficient transmission between cloud and mobile devices," in *Proceedings of the INFOCOM*, pp. 195–199, IEEE, 2013.

[39] Y.-D. Lin, E.-H. Chu, Y.-C. Lai, and T.-J. Huang, "Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds," *Systems Journal, IEEE*, vol. 9, no. 2, pp. 393–405, 2013.

[40] K. Fekete, K. Csorba, B. Forstner, T. Vajk, M. Feher, and I. Albert, "Analyzing computation offloading energy-efficiency measurements," in *Communications Workshops (ICC), International Conference on*, pp. 301–305, IEEE, 2013.

[41] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proceedings of the INFOCOM*, pp. 190–194, IEEE, 2013.

[42] J. Niu, W. Song, and M. Atiquzzaman, "Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications," *Journal of Network and Computer Applications*, vol. 37, pp. 334–347, 2014.

[43] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Energy and performance-aware task scheduling in a mobile cloud computing environment," in *Cloud Computing (CLOUD), IEEE 7th International Conference on*, pp. 192–199, 2014.

[44] F. Mehmeti and T. Spyropoulos, "Is it worth to be patient? Analysis and optimization of delayed mobile data offloading," in *Proceedings of the INFOCOM*, pp. 2364–2372, IEEE, 2014.

[45] S. G. Ahmad, C. S. Liew, M. M. Rafique, E. U. Munir, and S. U. Khan, "Data-intensive workflow optimization based on application task graph partitioning in heterogeneous computing systems," in *Big Data and Cloud Computing (BdCloud), IEEE Fourth International Conference on*, pp. 129–136, 2014.

[46] U. Nandhini and L. TamilSelvan, "Computational analytics of client awareness for mobile application offloading with cloud migration," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 8, no. 11, pp. 3916–3936, 2014.

[47] E. Hyytiä, T. Spyropoulos, and J. Ott, "Offload (only) the right jobs: Robust offloading using the Markov decision processes," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 16th International Symposium on a*, pp. 1–9, IEEE, 2015.

[48] C. Ragona, C. Fiandrino, D. Kliazovich, F. Granelli, and P. Bouvry, "Energy-efficient computation offloading for wearable devices and smartphones in mobile cloud computing," in *IEEE Global Communications Conference*, 2015.

[49] F. Mehmeti and T. Spyropoulos, "Performance analysis of mobile data offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 482–497, 2016.

[50] U. Shaukat, E. Ahmed, Z. Anwar, and F. Xia, "Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges," *Journal of Network and Computer Applications*, vol. 62, pp. 18–40, 2016.

[51] U. Nandhini, L. Tamilselvan, S. Nancy, and U. Shanmugam, "A secure client aware certification for mobile cloud offloading decision," *International Journal of Intelligent Engineering and Systems*, vol. 10, no. 2, pp. 106–115, 2017.

[52] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Quality of Service (IWQoS), 2017 IEEE/ACM 25th International Symposium on*, pp. 1–10, 2017.

[53] J. Barrameda and N. Samaan, "A novel statistical cost model and an algorithm for efficient application offloading to clouds," *IEEE Transactions on Cloud Computing*, pp. –, 2017.

[54] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Infrastructure for pervasive computing: Challenges.," in *GI Jahrestagung (1)*, pp. 214–222, 2001.

[55] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[56] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pp. 4–4, USENIX Association, 2010.

[57] A. Gember, C. Dragga, and A. Akella, "ECOS: practical mobile application offloading for enterprises," in *Proc. Int. Conf. Mobile Systems, Applications And Services*, pp. –, 2012.

[58] X. Ge, S. Tu, T. Han, Q. Li, and G. Mao, "Energy efficiency of small cell backhaul networks based on gauss-markov mobile models," *IET Networks*, vol. 4, no. 2, pp. 158–167, 2015.

[59] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy cost models of smartphones for task offloading to the cloud," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 3, pp. 384–398, 2015.

[60] X. Ge, J. Yang, H. Gharavi, and Y. Sun, "Energy efficiency challenges of 5G small cell networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 184–191, 2017.

[61] F. Mehmeti and T. Spyropoulos, "Performance analysis of "on-the-spot" mobile data offloading," in *Global Communications Conference (GLOBECOM)*, pp. 1577–1583, IEEE, 2013.

[62] H. Wu and K. Wolter, "Dynamic transmission scheduling and link selection in mobile cloud computing," in *Analytical and Stochastic Modeling Techniques and Applications*, pp. 61–79, Springer, 2014.

[63] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G wireless backhaul networks: Challenges and research advances," *IEEE Network*, vol. 28, no. 6, pp. 6–11, 2014.

[64] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5G ultra-dense cellular networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, 2016.

[65] S. Ou, K. Yang, and A. Liotta, "An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems," in *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, pp. 116–125, IEEE Computer Society, 2006.

[66] P. Di Lorenzo, S. Barbarossa, and S. Sardellitti, "Joint optimization of radio resources and code partitioning in mobile cloud computing," *arXiv preprint arXiv:1307.3835*, 2013.

[67] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Computation offloading for mobile cloud computing based on wide cross-layer optimization," in *Future Network and Mobile Summit*, pp. 1–10, IEEE, 2013.

[68] B. Hendrickson and T. G. Kolda, "Graph partitioning models for parallel computing," *Parallel computing*, vol. 26, no. 12, pp. 1519–1534, 2000.

[69] K. Ali and O. Lhoták, "Application-only call graph construction," in *ECOOP 2012–Object-Oriented Programming*, pp. 688–712, Springer, 2012.

[70] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222–1239, 2001.

[71] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, 2013.

[72] H. Wu and K. Wolter, "Software aging in mobile devices: Partial computation offloading as a solution," in *Software Reliability Engineering Workshops (ISSREW), International Symposium on*, IEEE, 2015.

[73] L. Yang, J. Cao, S. Tang, D. Han, and N. Suri, "Run time application repartitioning in dynamic mobile cloud environments," *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 336–348, 2016.

[74] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *Wireless Communications, IEEE Transactions on*, vol. 12, no. 9, pp. 4569–4581, 2013.

[75] H. Wu, "Analysis of mhealth systems with multi-cloud computing offloading," in *Mobile Health*, pp. 589–608, Springer, 2015.

[76] M. Whaiduzzaman, A. Gani, N. B. Anuar, M. Shiraz, M. N. Haque, and I. T. Haque, "Cloud service selection using multicriteria decision analysis," *The Scientific World Journal*, 2014.

[77] H. Wu, Q. Wang, and K. Wolter, "Optimal cloud-path selection in mobile cloud offloading systems based on QoS criteria," *International Journal of Grid and High Performance Computing (IJGHPC)*, vol. 5, no. 4, pp. 30–47, 2013.

[78] V. X. Tran, H. Tsuji, and R. Masuda, "A new QoS ontology and its QoS-based ranking algorithm for web services," *Simulation Modelling Practice and Theory*, vol. 17, no. 8, pp. 1378–1398, 2009.

[79] M. Dağdeviren, S. Yavuz, and N. Kılınç, "Weapon selection using the AHP and TOPSIS methods under fuzzy environment," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8143–8151, 2009.

[80] H. Flores and S. N. Srirama, "Mobile cloud middleware," *Journal of Systems and Software*, vol. 92, pp. 82–94, 2014.

[81] H. Wu and D. Huang, "Modeling multi-factor multi-site risk-based offloading for mobile cloud computing," in *Network and Service Management (CNSM), 10th International Conference on*, pp. 230–235, IEEE, 2014.

[82] Y. Kim, K. Lee, and N. B. Shroff, "An analytical framework to characterize the efficiency and delay in a mobile data offloading system," in *Proceedings of the 15th international symposium on Mobile ad hoc networking and computing*, pp. 267–276, ACM, 2014.

[83] F. Mehmeti and T. Spyropoulos, "Stay or switch?: Analysis and comparison of delays in cognitive radio networks with interweave and underlay spectrum access," in *Proceedings of the 14th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac'16, pp. 9–18, ACM, 2016.

[84] H. Wu, Y. Sun, and K. Wolter, "Analysis of the energy-response time tradeoff for delayed mobile cloud offloading," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, pp. 33–35, Sept. 2015.

[85] H. Wu and K. Wolter, "Analysis of the energy-performance tradeoff for delayed mobile offloading," in *Performance Evaluation Methodologies and Tools (VALUETOOLS), 9th International Conference on*, ICST, 2015.

[86] V. Cardellini, V. D. N. Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. L. Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, 2013.

[87] H. Wu, W. Knottenbelt, and K. Wolter, "Analysis of the energy-response time tradeoff for mobile cloud offloading using combined metrics," in *Teletraffic Congress (ITC 27), 27th International*, pp. 134–142, IEEE, 2015.

[88] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[89] M. A. Khan, "A survey of computation offloading strategies for performance improvement of applications running on mobile devices," *Journal of Network and Computer Applications*, vol. 56, pp. 28–40, 2015.

[90] H. Wu and K. Wolter, "Tradeoff analysis for mobile cloud offloading based on an additive energy-performance metric," in *Performance Evaluation Methodologies and Tools (VALUETOOLS), 8th International Conference on*, pp. 90–97, ICST, 2014.

[91] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, pp. –, 2017.