Stochastic Analysis of Delayed Mobile Offloading in Heterogeneous Networks

Huaming Wu and Katinka Wolter

Abstract—Mobile cloud offloading that migrates heavy computation from mobile devices to powerful cloud servers through communication networks can alleviate the hardware limitations of mobile devices thus providing higher performance and saving energy. Different applications usually give different relative importance to response time and energy consumption. If a delay-tolerant job is deferred up to a given deadline, or until a fast and energy-efficient network becomes available, the transmission time will be extended, which can save energy because a more energy-efficient communication channel and a less energy-restricted computation platform may become available. However, if the reduced service time fails to cover the extra waiting time, this policy may not be competitive. In this paper, we investigate two types of delayed offloading policies, the partial offloading model where jobs can leave from the slow phase of the offloading process and be executed locally on the mobile device, and the full offloading model, where jobs can abandon the *WiFi Queue* and be offloaded via the *Cellular Queue*. In both models, we minimize the Energy-Response time Weighted Product (ERWP) metric. Not surprisingly, we find that jobs abandon the queue often when the availability of the WiFi network is low. In general, for delay-sensitive applications the partial offloading model is preferred under a suitable reneging rate, while for delay-tolerant applications the full offloading model shows very good results and outperforms the other offloading model when selecting a large deadline. From the perspective of energy consumption, the full offloading model will always be best, even if the deadline must be extremely long. Only if job response time is of high importance an optimal deadline to abort offloading in the partial offloading model or the WiFi transmission in the full offloading model can be found. For reduction of the energy consumption it will always be better to wait longer rather than compute locally or use the cellular network.

Index Terms—Mobile Cloud Computing, Mobile Offloading, Heterogeneous Networks, Energy-Performance Tradeoff, Queueing Model.

1 Introduction

Besides light-weight Internet applications, there is an increasing demand from mobile users for computation-heavy and energy-hungry applications that are being deployed to mobile devices. Running complex applications on such devices is however challenging due to the strict constraints on their resources, e.g., the limited computational capacity, battery lifetime and network connectivity. Offloading computation-intensive tasks from mobile devices to a capable cloud server via wireless networks is an effective way to alleviate a tussle between resource-constrained mobile devices and resource-hungry mobile applications, and thus boosting the device's performance. Potential benefits obtained from offloading include reducing the job response time as well as decreasing the amount of energy needed to process a job.

Mobile offloading is most beneficial for applications that require heavy computation on a rather small amount of data. More precisely, in this paper we consider applications that upload considerably more data than they download. Any image recognition application would be an example for such an application, e.g., optical text recognition or object recognition in images.

- *H. Wu is with the Center for Applied Mathematics, Tianjin University, P. R. China, 300072. Email: whming@tju.edu.cn.*
- *K. Wolter is with the Institut für Informatik, Freie Universität Berlin, Germany, 14195. Email: katinka.wolter@fu-berlin.de.*

Different types of applications usually give different relative importance to both factors of response time and energy consumption.

- *Delay-Tolerant Applications*: many mobile applications (e.g., iCloud, social networking, mobile healthcare and urban tomography) deal with video, audio, sensor data, or access large databases on the Internet, which are less sensitive to network delays. Participatory sensing applications are a good example of data-intensive yet delay-tolerant applications. The collective sampling of sensor data acquired by a number of sensor nodes creates a body of knowledge on parameters such as personal resource consumption, dietary habits and urban documentation [1]. Data is uploaded from a smartphone to a back-end cloud server either through the cellular network or any available WiFi network. Some of the sensor information is not time-critical and its submission to the server may be delayed until the device enters an energy-efficient network [2]. Users can browse or search the obtained archives through a website at the server side. As a daily-life scenario when traveling outside the normal network coverage area (e.g., abroad, where the cellular contract is not valid) a user may wish to trigger a job, in order not to forget, but require the result only upon return to the workplace. In general terms, for delay-tolerant applications, response time is less critical and optimizing energy usage is more relevant.

- *Delay-Sensitive Applications*: when running delay-sensitive applications (e.g., face recognition, video conferencing, vehicular communications, authentication) on mobile devices, mobile users desire a fast response which is comparable to their normal cognitive capabilities. Thus, for good user experience the response time of cognitive applications should be low. Task offloading can be exploited to use cognitive applications ubiquitously by executing them remotely on computing nodes. Fast response is a primary concern for these applications. The offloading scheme in which cloud services are available after short network latency (e.g., WiFi networks) can serve such applications in a better way and lead to a low response time.

Mobile users are easily subject to dynamically changing network conditions due to their mobility, which makes it hard to make good offloading decisions in mobile environments [3]. Mobile network environments have a great influence on the performance of offloading systems. Therefore, taking a high-quality offloading decision requires a good understanding of the network condition.

Mobile devices usually have multiple radio interfaces for data transfer, such as 3G/4G and WiFi with different availability, delay and energy cost. Thus, there are several ways to offload tasks to the cloud, e.g., via a costly cellular connection or via intermittently available WiFi [4]. By delaying offloading until WiFi becomes available, it is possible to reduce the transmission time at the expense of some extra waiting time. The reduced transmission time at a later point in time directly translates to saving battery power of the mobile device [5]. However, delayed offloading is still a matter of debate, since it is not known to what extent users would be willing to delay a transmission [6]. In this paper, we try to give an overall recommendation of how to balance the time and energy investment for different types of scenarios, i.e. delay-tolerant and delay-sensitive applications. We develop a theoretical framework to capture the energy-performance tradeoff by using queueing models with impatient jobs and service interruptions. The models can be used to predict the average performance and energy consumption of mobile offloading under a given network environment deployment condition.

The main contributions of this paper are as follows:

- we propose two analytical queueing models for delayed mobile cloud offloading systems: the partial offloading model and the full offloading model. A non-delayed offloading model [7] is also introduced and used for comparison.
- we develop an analytical framework for analyzing queueing models with reneging and service interruptions. From our framework, we obtain closed-form formulas for key performance metrics in the delayed offloading system such as Energy-Response time Weighted Product (ERWP), which combines the advantages of other previously studied metrics.
- we aim at answering the following questions: (i) Given a deadline, what are the expected response time and expected energy consumption as a function of network parameters and job arrival rate? (ii) How should the deadline be optimally chosen in order to achieve different energy-delay tradeoffs for specific applications? (iii) Among different offloading models, which one is best at achieving a performance gain according to the ERWP metric?

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes system models for the delayed offloading models as well as the considered metrics. Sections 4 and 5 present the mathematical models and their analysis for the partial and the full offloading model. The partial offloading model based on the ERWP metric is analyzed in Section 4. The full offloading model is proposed and analyzed in Section 5. Section 6 evaluates metrics and models using numerical examples based on real traces of mobile networks. The paper concludes in Section 7.

2 Related Work

This section discusses related work on offloading systems to reduce job completion time, energy usage and a combination of both, each in the respective subsection. The issues of time and energy saving on mobile devices are becoming increasingly relevant. Many research efforts have been devoted to offloading computation to remote servers in order to shorten execution time or save energy.

2.1 Reducing Job Completion Time

Mobile cloud offloading is sometimes implemented as a two-step procedure where the cloud is reached via a close-by cloudlet which is in turn connected to powerful cloud servers. Satyanarayanan *et al*. [8] proposed a VM-based cloudlet in mobile computing, to which a smartphone connects over WLAN. The assumption is that connection to the Cloud imposes higher latency and lower bandwidth than the Cloudlet. In essence, cloudlets make use of mobile devices simply as a thin-client to access local resources, rather than using the mobile device's resources directly and offloading only when required.

A stochastic model for dynamic offloading has been developed in [4] using various performance metrics and also intermittently available access links. The mobile nature of mobile devices and unstable connections of wireless links affect the predictability of the performance of a pervasive service running under the control of offloading systems. Ou *et al*. [9] analyzed the performance of offloading systems in mobile wireless environments when considering system failure and recovery. However, how to take offloading decisions was not considered. In [10] a framework using estimated bandwidth to take offloading decisions was investigated. The authors formulated decision problems for computational offloading systems according to the bandwidth prediction for the local and remote systems. The assumption is that network reliability is not an issue, while in a realistic scenario, the network may even not be available at all.

2.2 Saving Energy

Mobile offloading systems have been built over the past years for the purpose of reducing the energy consumption

of mobile devices. MAUI [11] was a system that enables energy-aware offloading of mobile code to the infrastructure. Its main aim was 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. The system decided at runtime which methods should be executed remotely as to save most energy under the mobile device's current connectivity constraints.

Kumar *et al*. [12] argued that offloading could potentially save energy for mobile users, but not all applications were energy-efficient when migrated to the cloud. It depended on whether the computational cost saved due to offloading outperforms the extra communication cost. A large amount of communication combined with a small amount of computation should preferably be performed locally on the mobile device, while little communication for a large amount of computation should preferably be executed remotely.

Some authors consider a response time constraint for the application when partitioning application tasks for execution on mobile devices and cloud servers. This deadline is an important issue for many interactive applications. To save energy while satisfying a given application deadline, dynamic offloading algorithms were presented in [13], [14]. They showed low complexity to solve the offloading decision-making problem (i.e., to determine which software components to execute remotely under mobile network environments).

2.3 Saving Time and Energy Combined

Both concerns, job response time and energy consumption have been addressed by several authors. CloneCloud [15] used a combination of static analysis and dynamic profiling to partition applications automatically at a fine granularity while optimizing execution time and energy usage for a target computation and communication environment. However, static application partitioning [16], [17] were not suitable in situations with heavily changing network condition. Intermittent connectivity may exist due to heterogeneous wireless environments, device mobility and cloud resource unavailability. Unstable connections in mobile networks have a great impact on the offloading decisions. High communication latency and energy consumption can make local execution more advantageous in certain circumstances.

Seamless offloading operation by switching between several transmission technologies has been proposed in [18]. They have examined the tradeoff between energy consumption for WiFi search and transmission rate when the WiFi network was only intermittently available. Energy-efficient delayed network selection has been suggested in [1], [19] 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. Moreover, the use of "delayed offloading" has been suggested in case no WiFi connection is available. Then (some) traffic can be delayed up to a chosen deadline, or until WiFi becomes available [20]. An online scheduling policy for delayed mobile offloading was proposed in [21], where the amount of offloaded data by using WiFi has been maximised to the extent possible. The scheme would only use cellular networks after expiry of a deadline.

This work is motivated by the above interesting efforts to investigate the intermittent network connectivity problem in a mobile cloud environment, aiming at balancing different objectives like minimum response time and minimum energy consumption. We explicitly consider the mobile nature of both user and application behavior and study how delayed offloading can tackle these heterogeneity problems by using a combined metric based on our previous work [22], [23], [24].

3 System Models

In this section, we define two different delayed offloading models based on a network availability model and propose a new metric to capture the energy-performance tradeoff.

3.1 The Network Model

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

Fig. 1. Comparison of WiFi and cellular networks

Figure 1 shows the assumptions we make. We assume that the cellular interface has higher availability than WiFi and can provide nearly ubiquitous coverage for mobile devices in a wide area, but it has lower data transmission rate and consumes more transmission energy than the WiFi interface [19]. In other words, we assume that WiFi is much faster and more energy-efficient than the cellular interface for transmitting the same quantity of data. To facilitate the analysis of the mobile offloading systems, we assume that a cellular network is available to mobile users all the time while the availability of a WiFi network depends on the location. Mobile users move in and out of a WiFi coverage area. These assumptions seem realistic enough in many situations to constitute the basis of the work presented in this paper.

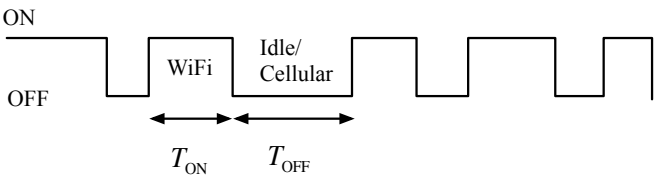


Fig. 2. The WiFi network availability model [25]

We model the time variation of the WiFi connection state by the ON-OFF alternating renewal process $(T_{\text{ON}}^{(i)}, T_{\text{OFF}}^{(i)})$, $i \geq 1$, as shown in Fig. 2. The ON periods represent the presence of the WiFi connectivity, while the OFF periods denote the interruption of the WiFi connectivity. 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 [6]. Further, the WiFi availability ratio (AR) can be defined as $AR = \frac{\mathbb{E}[T_{\text{ON}}]}{\mathbb{E}[T_{\text{ON}}]+\mathbb{E}[T_{\text{OFF}}]}$.

3.2 The Delayed Offloading Models

According to the network availability model depicted in Fig. 2, we define two types of delayed offloading models, namely, the partial offloading model and the full offloading model as follows. In delayed offloading, a deadline is associated with each data transfer and the data transfer is resumed whenever the device reaches the coverage of WiFi until the transfer is completed [5] or the deadline expires, whichever comes first. If the data transfer does not finish until the deadline expires, the task will either be executed locally (partial offloading model) or a cellular network will complete the data transfer (full offloading model).

- **Partial Offloading Model**: we employ a single queue with two phases (fast: WiFi network and slow: cellular network) to offload jobs to the cloud server. When there is a WiFi connection available, all the offloadable jobs are sent over the WiFi network; otherwise, they are sent over the cellular interface as the cellular network is always available. We set a reneging deadline in the cellular network, if the deadline expires before the job switched over to some WiFi AP, then it is executed locally on the mobile device rather than remotely in the cloud. By doing this, we have partial jobs offloaded to the cloud and the remaining ones processed locally.
- **Full Offloading Model**: when there is a WiFi connection available, all the offloadable jobs are sent over the WiFi network; otherwise, they can be delayed up to a given deadline, or until WiFi becomes available [20]. If the deadline expires before the job can be transmitted over some WiFi AP, then it is offloaded through the cellular network. In this way, we have all the offloadable jobs offloaded to the cloud via the cellular or WiFi network.

We consider a queueing system for delayed offloading. The mobile device, the cloud and the wireless networks are represented as queueing nodes to capture the resource contention and delay on the system. The mobile device executes an application with offloadable jobs that can be processed either locally on the processor of the mobile device, or remotely in a cloud infrastructure through offloading. The mobile device, the cellular and WiFi connections are modeled as $M/M/1$-FCFS queues, and the remote cloud is modeled as an $M/M/\infty$ queue, i.e., as a delay center [26]. We denote $1/\mu_m$ and $1/\mu_r$ the expected execution time of jobs on the mobile device and the cloud, respectively. The expected rates to transfer data to the cloud over the cellular network and WiFi are μ_c and μ_w, respectively.

The delayed offloading models involve queueing with reneging and service interruptions. In queueing, reneging means that a job will leave the queue and join another queue after the deadline expires. Service interruption literally means unwilling discontinuity of service in the queue, and this models connection and disconnection periods of a mobile device to WiFi networks in the system [27]. The essential difference between our delayed computation offloading model and the delayed data offloading in [5] is that not only data is transmitted to the cloud but the task itself is executed remotely on the cloud server. Usually, the offloaded data needs to be further analyzed or simple processed before it can show up.

3.3 Metrics

In this section, we define the metrics which will be used to evaluate and optimize the tradeoff between the job completion time and the energy usage.

3.3.1 ERWS

The Energy-Response time Weighted Sum (ERWS) metric is a cost function defined as the weighted sum of the respective expected values:

$$ERWS = \omega\mathbb{E}[\mathcal{E}] + (1-\omega)\mathbb{E}[T], \tag{1}$$

where $\mathbb{E}$ is used for expectation of a random variable, $\mathbb{E}[T]$ and $\mathbb{E}[\mathcal{E}]$ are expected response time and expected energy consumption, respectively, and ω (ranging between 0 and 1) is a weighting parameter that represents the relative significance of energy consumption and response time for the mobile device. To focus on performance, ω should be less than 0.5; to focus on energy consumption, ω should be greater than 0.5. When ω is exactly 0.5, the focus is on both increasing performance and reducing energy consumption. The expectation will be determined by its estimate, the mean value.

The ERWS metric has the advantage of being analytically well tractable since the expectation is additive over time and thus can be optimized via a Markov decision process [28]. From the view of minimization, this metric allows comparing arbitrary offloading policies to the optimal offloading policy. However, it has the disadvantage of being a linear combination of two metrics on different scales.

3.3.2 ERP

The Energy-Response time Product (ERP) is widely accepted as a suitable metric to capture the energy-performance tradeoff and it is defined as:

$$ERP = \mathbb{E}[\mathcal{E}] \cdot \mathbb{E}[T]. \tag{2}$$

Minimizing the ERP metric can be seen as maximizing the 'performance-per-joule', with performance being defined as jobs per time unit [28].

While the ERWS metric implies that a reduction in mean response time by one unit has the same value as the reduction of the mean energy usage by one unit. In contrast, the ERP metric being a product, does not combine two linear functions that live on totally different scales [28]. In other words, the ERP metric does not suffer from comparison of different scales. However, since ERP is the product of two mean values, it is a difficult metric to treat analytically.

3.3.3 ERWP

To overcome the aforementioned disadvantages, we propose a new metric called Energy-Response time Weighted Product (ERWP), which combines the strengths of ERWS and ERP in that it handles metrics on different scales well and is analytically tractable. The ERWP metric has been introduced in [29], [30]. It is defined as:

$$ERWP = \mathbb{E}[\mathcal{E}]^{\omega} \cdot \mathbb{E}[T]^{1-\omega}. \tag{3}$$

We can rewrite (3) as: $ERWP = e^{\omega \cdot \ln(\mathbb{E}[\mathcal{E}]) + (1-w) \cdot \ln(\mathbb{E}[T])}$, which inherits the characteristics of the ERWS metric that assigns different importance weights to energy consumption and response time, and has the advantage of being analytically tractable since the logarithmic expectation is additive over time. Meanwhile, the mean energy consumption and mean response time have equal importance when $\omega = 0.5$, also in (3): $ERWP = \sqrt{\mathbb{E}[\mathcal{E}] * \mathbb{E}[T]}$, which indicates that the ERWP metric has the advantages of the ERP metric that is insensitive to difference of scales.

4 The Partial Offloading Model

In this section, we analyze the partial offloading model with service reneging, which means that jobs give up upon expiry of the deadline. We first formulate the analytical model based on our network availability model, then we use queueing analysis to derive the ERWP metric.

4.1 The Model

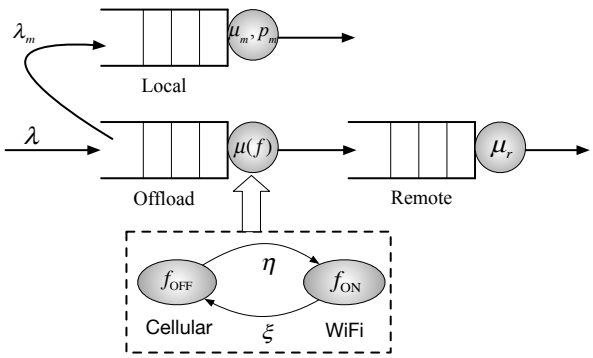


Fig. 3. The partial offloading model

Figure 3 depicts a delayed offloading model based on the network availability model. We consider a modulated $M/M/1$ queue in a two-phase (fast and slow) Markovian random environment, with impatient jobs, who may abort offloading to be processed locally. The jobs are offloaded either via a cellular connection or a WiFi network to the cloud. The single-server queueing system that oscillates between two feasible phases is denoted by f_{ON} and f_{OFF}, meaning that the WiFi connection can be operational or not. In the model the persistence of the system at any phase is governed by a random mechanism [31]: if the system operates in phase f_{ON}, it switches to phase f_{OFF} after random time of mean duration $1/\xi$; if the system operates in phase f_{OFF}, it switches to the other phase after random time of mean duration $1/\eta$. We assume that offloading jobs arrive at the system according to a Poisson process with rate λ, and the modulating process $f \in \{f_{\text{ON}}, f_{\text{OFF}}\}$ determines the service rates:

$$\mu(f) = \begin{cases} \mu_c, & \text{if } f = f_{\text{OFF}} \\ \mu_w, & \text{if } f = f_{\text{ON}} \end{cases}. \tag{4}$$

We assume that the mean job size is $\mathbb{E}[X]$, the transmission speed of the fast phase (WiFi network) is s_w with service rate $\mu_w = s_w/\mathbb{E}[X]$, and its operating power is p_w when serving jobs and zero whenever idle. Similarly, the corresponding speed for the slow phase (cellular network) is s_c with service rate $\mu_c = s_c/\mathbb{E}[X]$ $(\mu_c \leq \mu_w)$, and operating power p_c. When in the slow phase, jobs may become impatient. A reneging deadline T_d, is associated with each job in this phase. That is, upon arrival each job activates an individual timer which is exponentially distributed with a reneging rate r. If the system does not change its environment from the slow phase to the fast phase before the deadline expires, the job will be removed from the *Offload Queue* and is assumed to be executed locally on the mobile device rather than being offloaded to the cloud [32].

Figure 3 demonstrates that the delayed offloading model consists of an *Offload Queue* (with two alternating phases of cellular and WiFi), a *Local Queue* denoting the local processing on the mobile device and a *Remote Queue* representing the remote processing on the cloud.

The *Offload Queue* alternates its service by means of mutual resets according to the availability of WiFi, which is governed by an interrupted Poisson Process (IPP) with exponentially distributed ON-OFF periods. We model the intermittent availability of WiFi hotspots as a FCFS queue with occasional server break-down [4], either in the ON-state where the WiFi network is processing the existing jobs, or in the OFF-state during which jobs are served by the cellular network (cellular connectivity is assumed to be always available). However, when the job stays in the cellular network for too long, it abandons the *Offload Queue* and is then processed locally on the mobile device. If the job in the *Offload Queue* is completely transmitted before the assigned deadline has expired, we say that the job is successfully offloaded. If offloading fails, the job leaves the *Offload Queue* and join the *Local Queue* on the mobile device for immediate local processing. We call such an event a reneging event [27].

Since there is no waiting time before entering service, the $M/M/\infty$ queue of the cloud is occasionally referred to as a delay (sometimes pure delay) station, the probability distribution of the delay being that of the service time.

Especially, if we set the service rates μ_m and μ_r to ∞, the mobile computation offloading model in Fig. 3 reduces to mobile data offloading, i.e., there is no further execution for the arrival job. Therefore, the queueing model in Fig. 3 encompasses both mobile computation offloading and mobile data offloading.

4.2 Queueing Analysis

Given the previously stated assumptions, the partial offloading model can be modeled with a 2D Markov chain, as shown in Fig. 4.

The states with cellular network are denoted $\{c, i\}$, and the states with WiFi connectivity are denoted $\{w, i\}$. The

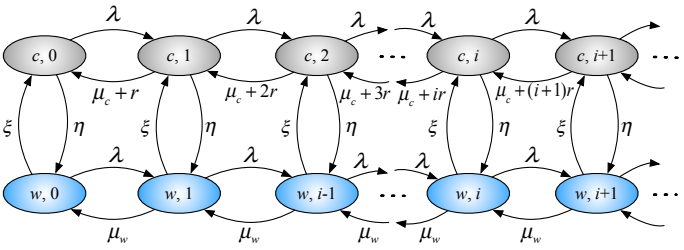


Fig. 4. The Markov chain for the partial offloading model

parameter i corresponds to the number of jobs in the system (queueing and in service). During the WiFi phase, the system depletes at rate μ_w and during the cellular phase, the system serves at rate $\mu_c + i \cdot r$ since any of the i queued jobs can abandon the *Offload Queue* [20]. Writing the balance equations for the cellular and WiFi phases gives:

$$(\lambda+\eta)\pi_{c,0} = (\mu_c + r)\pi_{c,1} + \xi\pi_{w,0} \tag{5a}$$

$$(\lambda+\eta+\mu_c+ir)\pi_{c,i} = \lambda\pi_{c,i-1} + \big(\mu_c+(i+1)r\big)\pi_{c,i+1} + \xi\pi_{w,i}\ (i>0) \tag{5b}$$

$$(\lambda+\xi)\pi_{w,0} = \mu_w\pi_{w,1} + \eta\pi_{c,0} \tag{5c}$$

$$(\lambda+\xi+\mu_w)\pi_{w,i} = \lambda\pi_{w,i-1} + \mu_w\pi_{w,i+1} + \eta\pi_{c,i}\ (i>0) \tag{5d}$$

The steady-state probability of finding the offloading system in some region with WiFi unavailable (with only cellular access) is $\pi_c = \frac{\mathbb{E}[T_{\text{OFF}}]}{\mathbb{E}[T_{\text{ON}}]+\mathbb{E}[T_{\text{OFF}}]} = \frac{\xi}{\eta+\xi}$. Similarly, the steady-state probability for the periods with WiFi available is $\pi_w = \frac{\mathbb{E}[T_{\text{ON}}]}{\mathbb{E}[T_{\text{ON}}]+\mathbb{E}[T_{\text{OFF}}]} = \frac{\eta}{\eta+\xi}$, which equals to the availability ratio AR.

The probability generating functions for both cellular and WiFi states are defined as:

$$G_c(z) = \sum_{i=0}^{\infty}\pi_{c,i}z^i \text{ and } G_w(z) = \sum_{i=0}^{\infty}\pi_{w,i}z^i, \quad |z|\le 1. \tag{6}$$

By multiplying each equation for i in (5c-5d) by z^n, respectively, summing over i and rearranging terms we obtain [32]:

$$G_w(z)\beta(z) = \eta z G_c(z) - \mu_w(1-z)\pi_{w,0},$$

where $\beta(z) = (\lambda z - \mu_w)(1-z) + \xi z$. The roots z_1, z_2 of the quadratic polynomial $\beta(z) = -\lambda(z-z_1)(z-z_2)$ are

$$z_{1,2} = \frac{\lambda+\mu_w+\xi \mp \sqrt{(\lambda+\mu_w+\xi)^2 - 4\lambda\mu_w}}{2\lambda}.$$

4.2.1 The General Case

We consider the partial offloading model as depicted in Fig. 3 when assuming the reneging rate $r \neq 0$. According to [32], we obtain:

$$\pi_{c,0} = \frac{rS\xi\kappa_2(1)}{\mu_c(\xi+\eta)(SV-TU)}, \tag{7}$$

$$\pi_{w,0} = -\frac{rT\kappa_2(1)}{\mu_w(\xi+\eta)(SV-TU)}, \tag{8}$$

where we define $S = \int_0^{z_1}\frac{\kappa_1(x)}{\beta(x)}dx$, $T = \int_0^{z_1}\frac{\kappa_1(x)}{x}dx$, $U = \int_{z_1}^{1}\frac{\kappa_2(x)}{\beta(x)}dx$ and $V = \int_{z_1}^{1}\frac{\kappa_2(x)}{x}dx$. Accordingly, $\kappa_1(z)$ and $\kappa_2(z)$ are represented as follows:

$$\kappa_1(z) = e^{-\frac{\lambda z}{r}}z^{\frac{\mu_c}{r}}(z_1-z)^{\frac{\eta}{r}\frac{z_1(z_2-1)}{z_2-z_1}}(z_2-z)^{-\frac{\eta}{r}\frac{z_2(z_1-1)}{z_2-z_1}}, z\le z_1,$$

$$\kappa_2(z) = e^{-\frac{\lambda z}{r}}z^{\frac{\mu_c}{r}}(z-z_1)^{\frac{\eta}{r}\frac{z_1(z_2-1)}{z_2-z_1}}(z_2-z)^{-\frac{\eta}{r}\frac{z_2(z_1-1)}{z_2-z_1}}, z\ge z_1.$$

By the definitions of $\kappa_1(z)$, $\kappa_2(z)$ and $\beta(z)$, it follows that $T, U, V > 0$ and $S < 0$. Therefore, $\pi_{c,0}$ and $\pi_{w,0}$ are positive. One can show formally that the system is ergodic. Intuitively, we see that the system is always stable since, with any set of parameters $\lambda \ge 0, \mu_c \ge 0, \mu_w > 0, \xi > 0, \eta > 0$ and $r > 0$. The abandonment process, whose overall rate increases with the number of jobs, prevents the explosion of the queue length [32]. Alternatively, the system is stable if and only if $\pi_{c,0}$ and $\pi_{w,0}$ are positive, which always holds for the above set of parameters.

Let μ be defined as: $\mu = \pi_c \cdot \mu_c + \pi_w \cdot \mu_w$. According to [32], we obtain:

$$\mathbb{E}[N_c] = \frac{\lambda - \mu + \mu_c\pi_{c,0} + \mu_w\pi_{w,0}}{r}, \tag{9}$$

$$\mathbb{E}[N_w] = \frac{\eta(\lambda-\mu) + r(\lambda-\mu_w)\pi_w + \eta\mu_c\pi_{c,0} + \mu_w(\eta+r)\pi_{w,0}}{\xi r}. \tag{10}$$

As shown in Fig. 4, the expected number of jobs served per unit of time in the slow phase and fast phase are $\mu_c(\pi_c - \pi_{c,0})$ and $\mu_w(\pi_w - \pi_{w,0})$, respectively [33]. Therefore, the rate of abandonment due to impatience in the slow phase, λ_{aband}, is given by:

$$\begin{aligned}\lambda_{\text{aband}} &= \lambda - \mu_c(\pi_c - \pi_{c,0}) - \mu_w(\pi_w - \pi_{w,0}) \\ &= \lambda - \mu + \mu_c\pi_{c,0} + \mu_w\pi_{w,0} \\ &= r\cdot\mathbb{E}[N_c],\end{aligned} \tag{11}$$

where the abandonment rate is proportional to the reneging rate and the mean number of jobs in the cellular phase.

The rate at which jobs are executed locally on the mobile device must be equal to the abandonment rate, i.e., $\lambda_m = \lambda_{\text{aband}}$. The probability that an arbitrary job arriving at the *Offload Queue* will leave and join the *Local Queue*, i.e., it will be executed locally and will never be offloaded again, is defined as:

$$\Pr\{\text{abandon}\} = \frac{\lambda_{\text{aband}}}{\lambda} = \frac{\lambda - \mu + \mu_c\pi_{c,0} + \mu_w\pi_{w,0}}{\lambda}, \tag{12}$$

where Pr denotes the probability operation.

We recollect that the utilization of the service station is represented by: $\rho = 1 - (\pi_{c,0} + \pi_{w,0})$.

4.2.2 An Extreme Case

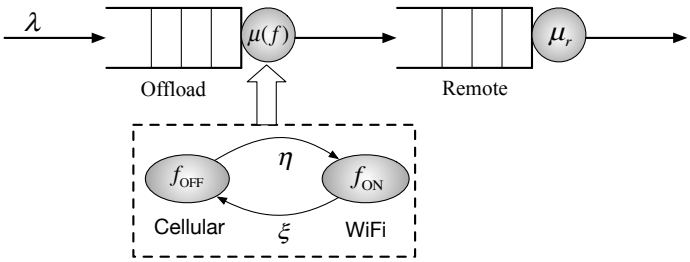


Fig. 5. The non-delayed offloading model

The partial offloading model in Fig. 3 will reduce to the non-delayed offloading model as depicted in Fig. 5 when $r \to 0$. Since in this case the reneging rate is zero, there will be no *Local Queue* in this model. When the network is available, all jobs are offloaded immediately no matter what the network quality is. Since also the channel with poor

quality is used, this offloading model may waste energy [5] and it is analyzed in comparison with the delayed offloading models.

After solving the equations of (5) when setting $r = 0$, we have [34]:

$$g(z)G_c(z) = \pi_{w,0}\xi\mu_w z + \pi_{c,0}\mu_c\big[\xi z + \lambda z(1-z) - \mu_w(1-z)\big],$$

where $g(z) = \lambda^2 z^3 - \lambda(\eta + \xi + \lambda + \mu_c + \mu_w)z^2 + (\eta\mu_w + \xi\mu_c + \mu_c\mu_w + \lambda(\mu_c + \mu_w))z - \mu_c\mu_w$, and it is proven that $g(z)$ has only one root z_0 in the interval $(0, 1)$.

After some algebraic manipulations, we obtain:

$$\pi_{c,0} = \frac{\xi(\mu-\lambda)z_0}{\mu_c(1-z_0)(\mu_w - \lambda z_0)}, \quad (13)$$

$$\pi_{w,0} = \frac{\eta(\mu-\lambda)z_0}{\mu_w(1-z_0)(\mu_c - \lambda z_0)}. \quad (14)$$

Once the values of $\pi_{c,0}$ and $\pi_{w,0}$ have been established, the probability generating functions can be calculated as:

$$G_c(z) = \frac{\xi(\mu-\lambda)z + \pi_{c,0}\mu_c(1-z)(\lambda z - \mu_w)}{g(z)}, \quad (15)$$

$$G_w(z) = \frac{\eta(\mu-\lambda)z + \pi_{w,0}\mu_w(1-z)(\lambda z - \mu_w)}{g(z)}. \quad (16)$$

By using $\mathbb{E}[N_i] = \sum_{n=0}^{\infty} n\pi_{i,n} = \mathrm{d}G_i(z)/\mathrm{d}z|_{z=1}$, we get the average number of jobs in the system [34]:

$$\begin{aligned}\mathbb{E}[N] &= \mathbb{E}[N_c] + \mathbb{E}[N_w] \\ &= \frac{\lambda}{\mu-\lambda} + \frac{\mu_c(\mu_w-\lambda)\pi_{c,0} + \mu_w(\mu_c-\lambda)\pi_{w,0}}{(\xi+\eta)(\mu-\lambda)} \\ &\quad - \frac{(\mu_c-\lambda)(\mu_w-\lambda)}{(\xi+\eta)(\mu-\lambda)}. \end{aligned} \quad (17)$$

4.3 Metric-Based Analysis

The total cost, in terms of energy or response time for processing all jobs is composed of the remote cost (sending some offloadable jobs to the cloud, idly waiting for the cloud to complete them and sending the computation result back to the mobile device), and the local cost (processing the remaining jobs locally on the mobile device). Since the delay caused by the transmission in the uplink usually dominates the transmission cost we neglect the cost in the downlink here.

4.3.1 The Mean Response Time

By Little's Law, $\mathbb{E}[N] = \lambda\mathbb{E}[T]$, the mean response time can be calculated as:

$$\begin{aligned}\mathbb{E}[T] &= \mathbb{E}\big[\mathbb{E}[T_i]\big] \\ &= \sum_{i\in\{c,w,m,r\}} \frac{\lambda_i}{\lambda}\mathbb{E}[T_i] \\ &= \frac{1}{\lambda}\sum_{i\in\{c,w,m,r\}} \mathbb{E}[N_i], \end{aligned} \quad (18)$$

where $i \in \{c, w, m, r\}$ represents the cellular phase, the WiFi phase, the mobile device and the remote cloud, respectively. $\mathbb{E}[N_c]$ and $\mathbb{E}[N_w]$ are the average number of jobs in the cellular network and WiFi network as obtained in (9) and (10), respectively.

Since the arrival rate to the *Local Queue* equals the abandonment rate of the *Offload Queue* for local processing we have $\lambda_m = r \cdot \mathbb{E}[N_c]$. For an ordinary $M/M/1$-FCFS queue, the average number of jobs on the mobile device is given by:

$$\mathbb{E}[N_m] = \frac{\rho_m}{1-\rho_m}. \quad (19)$$

where $\rho_m = \lambda_m/\mu_m$ is the utilization.

Since there is no waiting time before entering into remote service in the cloud, for an $M/M/\infty$ queue, the average number of jobs in the *Remote Queue* can be calculated as:

$$\mathbb{E}[N_r] = \frac{\lambda_r}{\mu_r}, \quad (20)$$

where $\lambda_r = \lambda - \lambda_m$ is the arrival rate to the *Remote Queue*.

4.3.2 The Mean Energy Consumption

We assume that each server operates at a constant power p_i, $\big(i \in \{c, w, m\}\big)$ whenever it is busy, i.e., the mobile device consumes energy only when there are jobs in the system. Since $\mathbb{E}[P] = \lambda\mathbb{E}[\mathcal{E}]$ is the mean power consumption, we can calculate the mean energy consumption for the partial offloading model can be calculated as follows:

$$\begin{aligned}\mathbb{E}[\mathcal{E}] &= \mathbb{E}\big[\mathbb{E}[\mathcal{E}_i]\big] \\ &= \sum_{i\in\{c,w,m\}} \frac{\lambda_i}{\lambda}\mathbb{E}[\mathcal{E}_i] \\ &= \frac{1}{\lambda}\sum_{i\in\{c,w,m\}} \mathbb{E}[P_i]. \end{aligned} \quad (21)$$

The application jobs that are remotely executed om the cloud server do not consume CPU energy on the local device.

For $i \in \{c, w, m\}$, the corresponding average power consumption can be calculated as:

$$\mathbb{E}[P_i] = p_i \cdot \Pr\{N_i > 0\} = p_i \cdot \rho_i. \quad (22)$$

Since the utilization of the queue is the probability that the server is busy, we have $\Pr\{N_i > 0\} = \rho_i$, i.e., the energy cost is only incurred during the fraction of the time the server is busy.

The energy consumed due to local execution depends on the processing speed of the mobile device. Since the service on the mobile device is always available, we have:

$$\mathbb{E}[P_m] = p_m \cdot \Pr\{N_m > 0\} = p_m \cdot \rho_m. \quad (23)$$

The mean energy consumption due to offloading via cellular or WiFi network depends on the transmission power and speed. We have:

$$\mathbb{E}[P_c] = p_c \cdot \Pr\{N_c > 0\} = p_c \cdot \rho_c, \quad (24)$$

$$\mathbb{E}[P_w] = p_w \cdot \Pr\{N_w > 0\} = p_w \cdot \rho_w, \quad (25)$$

where ρ_c and ρ_w are the utilizations of the cellular and WiFi networks, which are equal to the probability that the corresponding network is busy. According to Fig. 4, they can be calculated separately as follows:

$$\rho_c = \pi_c - \pi_{c,0}, \quad (26)$$

$$\rho_w = \pi_w - \pi_{w,0}. \quad (27)$$

4.3.3 The ERWP Metric

The energy-response time-weighted-product combines both, an energy metric and a performance metric in their weighted product. Our objective is to minimise the mean energy consumption and the mean response time. Further, by substituting (18) and (21) into (3), we can formulate the explicit expressions of the ERWP metric for the delayed offloading model:

$$r^* = \arg\min_r ERWP, \tag{28}$$

we seek the reneging rate r^* such that $ERWP$ is minimised. Remember, that reneging means that an impatient job gives up an offloading attempt and switches to local computation.

5 The Full Offloading Model

In this section, we discuss the full offloading model in which all jobs are offloaded. If possible jobs are offloaded via the WiFi connection, otherwise the cellular network is used.

5.1 The Model

As depicted in Fig. 6, the full offloading model consists of two coupled queues used for offloading from a mobile device, called *WiFi Queue* and *Cellular Queue*. Both queues are served by a FIFO (first-in-first-out) discipline. All jobs arriving to the system are by default sent to the WiFi interface for offloading. When a job is offloaded to the cloud via the WiFi network, there is queueing due to the not always sufficient transmission speed of the WiFi link. We model the intermittent availability of WiFi hotspots as a FCFS queue with occasional server break-down. The server is either in the ON-state processing the existing jobs, or in the OFF-state during which no job receives service. We assume that jobs will abandon the queue during periods without WiFi connectivity.

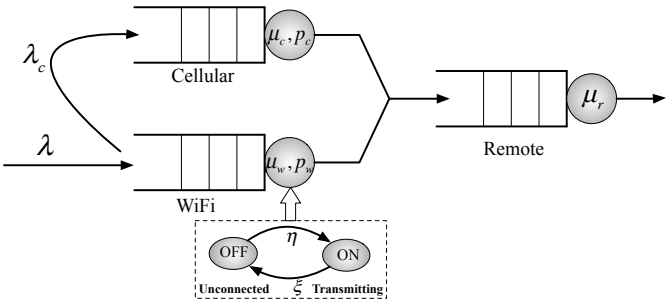


Fig. 6. The full offloading model

We assign a reneging deadline to each job. The deadline is drawn from an exponential distribution. Jobs are served in the FCFS order depending on their remaining time until expiry of the deadline (either while queued or while at the head of the queue, but waiting for the WiFi interface). When the server of the WiFi queue is in the OFF-state, jobs may become impatient. That is, each job, upon arrival, activates an individual timer, exponentially distributed with a reneging rate r. If the network does not change its environment from the OFF-state to the ON-state before the deadline expires, the job abandons the *WiFi Queue* to be offloaded via a cellular network [20] instead. If the job in the *WiFi Queue* is completely transmitted through the WiFi network before the assigned deadline has expired, we say that the job has been successfully offloaded. If offloading fails, the job leaves the *WiFi Queue* and joins the *Cellular Queue* in the mobile device for immediate transmission through the cellular network. We call such an event a reneging event.

When the job is offloaded to the cloud via a cellular network, there is queueing due to the not always sufficient transmission speed of the cellular link. Costs in terms of transmission delays (queueing and actual transmission time) and transmission energy consumption incur. Some degree of service is always available since the cellular connection always exists.

The *Remote Queue* is a pure delay station at which jobs spend an exponentially distributed amount of time with mean equal to $1/\mu_r$ time units.

5.2 Queueing Analysis

The *WiFi Queue* refers to offloading jobs from the mobile device to the cloud via a WLAN network, which is modeled as an $M/M/1$-FCFS queue with intermittently available service. When a failed server recovers, it continues to serve the job whose service has been interrupted, i.e., the work already completed is not lost (cf. data transfers with resume) [4]. We make the common and not too unrealistic assumption that the service fails from time to time and resumes its operation after a random interval. The Markov chain for the *WiFi Queue* is depicted in Fig. 7, which is equivalent to assuming that $\mu_c = 0$, $\pi_{\text{ON}} = \pi_w$ and $\pi_{\text{OFF}} = \pi_c$ in Fig. 4.

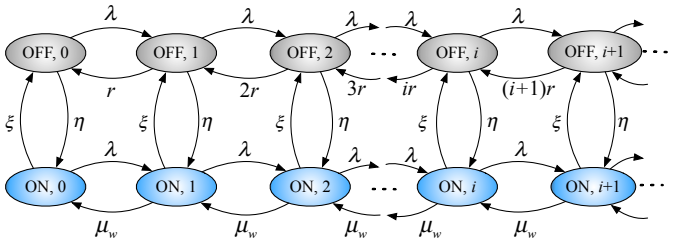


Fig. 7. The 2D Markov chain for the WiFi queue

The states with WiFi connectivity are denoted $\{\text{ON}, i\}$, and the states without WiFi connectivity are denoted $\{\text{OFF}, i\}$. During the ON-state, the system serves at rate μ_w and during the OFF-state, it serves at rate $i \cdot r$ since any of the i queued jobs can abandon the *WiFi Queue* [20]. Writing the balance equations for this chain gives:

$$\begin{aligned}
(\lambda+\eta)\pi_{\text{OFF},0} &= \xi\pi_{\text{ON},0} + r\pi_{\text{OFF},1} \\
(\lambda+\eta+ir)\pi_{\text{OFF},i} &= \lambda\pi_{1,i-1} + (i+1)r\pi_{\text{OFF},i+1} + \xi\pi_{\text{ON},i} \\
(\lambda+\xi)\pi_{\text{ON},0} &= \eta\pi_{\text{OFF},0} + \mu_w\pi_{\text{OFF},1} \\
(\lambda+\xi+\mu_w)\pi_{\text{ON},i} &= \lambda\pi_{\text{ON},i-1} + \mu_w\pi_{\text{ON},i+1} + \eta\pi_{\text{OFF},i}
\end{aligned}$$

After substituting $\mu_c = 0$ into $\kappa_1(z)$ and $\kappa_2(z)$, yields:

$$\kappa_1(z) = e^{-\frac{\lambda z}{r}}(z_1 - z)^{\frac{\eta}{r}\frac{z_1(z_2-1)}{z_2-z_1}}(z_2 - z)^{-\frac{\eta}{r}\frac{z_2(z_1-1)}{z_2-z_1}}, z \le z_1,$$
$$\kappa_2(z) = e^{-\frac{\lambda z}{r}}(z - z_1)^{\frac{\eta}{r}\frac{z_1(z_2-1)}{z_2-z_1}}(z_2 - z)^{-\frac{\eta}{r}\frac{z_2(z_1-1)}{z_2-z_1}}, z \ge z_1.$$

According to [25], we obtain:

$$\pi_{\text{OFF},0} = -\frac{S\xi\kappa_2(1)}{(\xi+\eta)U\kappa_1(0)}, \tag{29}$$

$$\pi_{\text{ON},0} = \frac{r\kappa_2(1)}{\mu_w(\xi+\eta)U}. \tag{30}$$

We further have $\mu = \pi_c \cdot \mu_c + \pi_w \cdot \mu_w = \pi_{\text{ON}}\mu_w$. After substituting the above expressions in (9) and (10), we derive the mean number of jobs in *WiFi Queue* as:

$$\mathbb{E}[N_{\text{OFF}}] = \frac{\lambda - \mu_w(\pi_{\text{ON}} - \pi_{\text{ON},0})}{r}, \quad (31)$$

$$\mathbb{E}[N_{\text{ON}}] = \frac{\eta\lambda - \mu_w(\eta + r)(\pi_{\text{ON}} - \pi_{\text{ON},0}) + \lambda r \pi_{\text{ON}}}{\xi r}. \quad (32)$$

Therefore, the average number of jobs in the *WiFi Queue* can be calculated as:

$$\mathbb{E}[N_w] = \mathbb{E}[N_{\text{OFF}}] + \mathbb{E}[N_{\text{ON}}]. \quad (33)$$

In Fig. 7 the expected number of jobs served per unit of time in the *WiFi Queue* is $\mu_w(\pi_{\text{ON}} - \pi_{\text{ON},0})$. Therefore, the rate of abandonment due to impatience in the OFF periods, λ_{aband}, is given by:

$$\lambda_{\text{aband}} = \lambda - \mu_w(\pi_{\text{ON}} - \pi_{\text{ON},0}) = r \cdot \mathbb{E}[N_{\text{OFF}}]. \quad (34)$$

where the abandonment rate is proportional to the reneging rate and the mean number of jobs in the queue during the OFF-state.

The rate of jobs sent to the cellular network λ_c must be equal to the abandonment rate, i.e., $\lambda_c = \lambda_{\text{aband}}$. The probability that an arbitrary job arriving to the *WiFi Queue* will abandon it, i.e., it will be offloaded over a *Cellular Queue*, is defined as:

$$\Pr\{\text{renege}\} = \frac{\lambda_{\text{aband}}}{\lambda} = \frac{\lambda - \mu_w(\pi_{\text{ON}} - \pi_{\text{ON},0})}{\lambda}. \quad (35)$$

5.3 Metric-Based Analysis

In this subsection we will derive expressions in the full offloading model for our metrics of interest, the mean response time, the mean energy consumption and the tradeoff metric based on the former two, the energy-response time-weighted product.

5.3.1 Mean Response Time

By Little's Law, $\mathbb{E}[N] = \lambda\mathbb{E}[T]$, the mean response time can be calculated as:

$$\mathbb{E}[T] = \mathbb{E}\big[\mathbb{E}[T_i]\big] = \sum_{i\in\{c,w,r\}} \frac{\lambda_i}{\lambda}\mathbb{E}[T_i] = \frac{1}{\lambda}\sum_{i\in\{c,w,r\}} \mathbb{E}[N_i], \quad (36)$$

where $\mathbb{E}[N_w]$ is the average number of jobs in the *WiFi Queue* as obtained in (33).

The *Cellular Queue* refers to offloading jobs from the mobile device to the cloud via a cellular network, which is modeled as an $M/M/1$-FCFS queue. Since the arrival rate to the *Cellular Queue* equals to the abandonment rate of the *WiFi Queue*, i.e., $\lambda_c = r \cdot \mathbb{E}[N_{\text{OFF}}]$. The average number of jobs in this queue is given by:

$$\mathbb{E}[N_c] = \frac{\rho_c}{1 - \rho_c}, \quad (37)$$

where $\rho_c = \lambda_c/\mu_c$ is the probability that the *Cellular Queue* is busy.

Since all the jobs are offloaded to the remote server in the cloud, for an $M/M/\infty$ queue the average number of jobs on the cloud server can be calculated as:

$$\mathbb{E}[N_r] = \frac{\lambda}{\mu_r}. \quad (38)$$

5.3.2 Mean Energy Consumption

The mean energy consumption can be calculated as:

$$\mathbb{E}[\mathcal{E}] = \mathbb{E}\big[\mathbb{E}[\mathcal{E}_i|i]\big] = \sum_{i\in\{w,c\}} \frac{1}{\lambda}\mathbb{E}[P_i] = \frac{1}{\lambda}\sum_{i\in\{w,c\}} p_i \cdot \Pr\{N_i > 0\} = \frac{1}{\lambda}\sum_{i\in\{w,c\}} p_i \cdot \rho_i, \quad (39)$$

where ρ_w is the fraction of time that WiFi is available to process jobs, and it can be calculated as:

$$\rho_w = \pi_{\text{ON}} - \pi_{\text{ON},0}, \quad (40)$$

as the recovery rate $\eta \to \infty$, the availability of WiFi $\pi_{\text{ON}} = AR = \frac{\eta}{\xi+\eta}$ tends to be 1.

5.3.3 ERWP Metric

In our analysis we wish to optimize the ERWP metric. By substituting (36) and (39) into (3), we can formulate the optimization of the ERWP metric for the offloading assignment as: $r^* = \arg\min_r ERWP$.

6 Performance Evaluation

In this section, we compare the analytical results by using the proposed delayed offloading models according to a realistic offloading scenario. In order to obtain realistic results we estimate model parameters from experiments. The offloading process includes a communication model and a remote execution model. We will conduct some experiments in order to use realistic communication parameters in our models.

6.1 Mobile Network Traces

The data transmission rate in real wireless networks is mostly not constant over time. It is affected by the changing signal quality and the presence of other users. We collect real network traces in mobile environments by using network and energy profilers. Those traces are then fed into the offloading model.

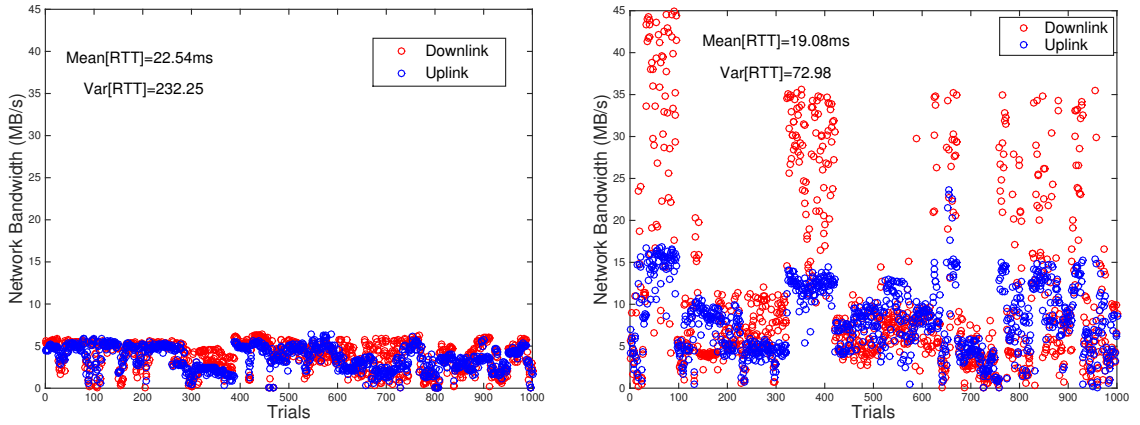
6.1.1 Network Profiler

A network profiler collects information about wireless connection status and available bandwidth. It measures the network characteristics at initialization, and continuously monitors environmental changes. Network throughput can be obtained by measuring the time duration when sending a certain amount of data as in [15]. Due to the mobile nature, the status of a wireless connection could change frequently (e.g., the user moves to other location). Fresh information about a wireless connection is critical for the optimizer to make correct offloading decisions.

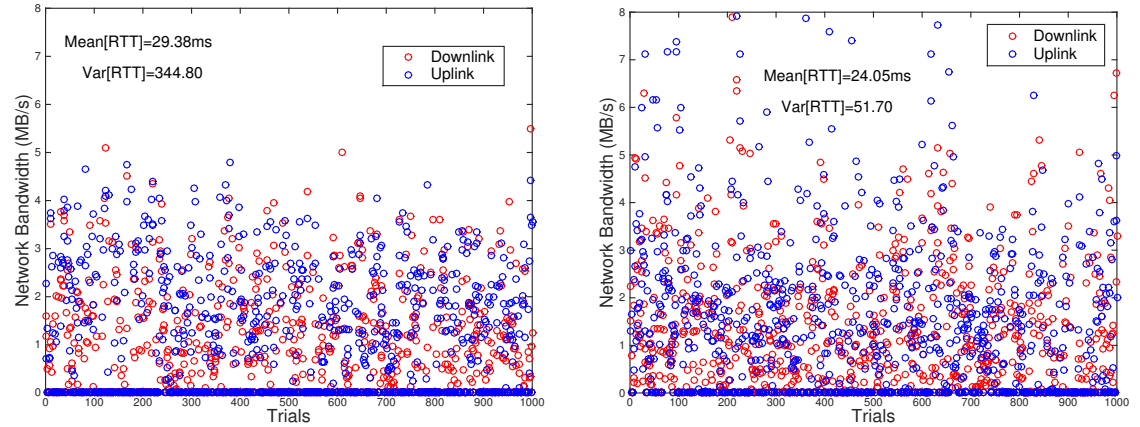
The profiler tracks several parameters for the WiFi and 3G interfaces, including the number of packets transmitted and received per second, and the receiving and transmitting data rate [35]. These measurements enable a better estimate of the currently achieved network performance.

TABLE 1
Mobile Device Specifications

Device	CPU	Memory	Communication Method	Technology
Xiaomi Redmi 2	Quad-core 2.1 GHz Cortex-A57	1GB RAM	WiFi	IEEE 802.11g
			3G	HSPAP/HSUPA
			4G	LTE
Samsung Galaxy S6	Quad-core 1.2 GHz Snapdragon 410	3GB RAM	WiFi	IEEE 802.11g
			3G	HSPAP/HSUPA
			4G	LTE

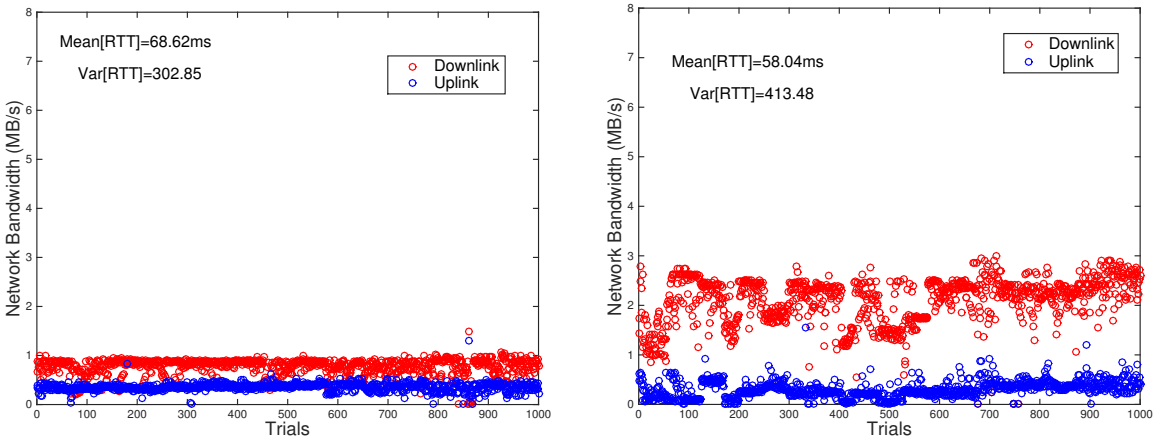


(a) Xiaomi Redmi 2 (indoor) (b) Samsung Galaxy S6 (indoor)



(c) Xiaomi Redmi 2 (outdoor) (d) Samsung Galaxy S6 (outdoor)

Fig. 8. The downlink and uplink bandwidth of WiFi in indoor and outdoor (walk) environments



(a) 3G with Samsung Galaxy S6 (b) LTE with Samsung Galaxy S6

Fig. 9. The downlink and uplink bandwidth of cellular networks in mobile (walk) environments

We use Speedtest[1] to measure the mobile network bandwidth. Actual devices (see Table 1) are applied in a mobile cloud environment with various mobile communication networks. Here, we measure wireless bandwidth statistics under the representative scenarios as shown in Table 2. Specifically, during one week in May 2015 we stayed inside some buildings or randomly walked around our campus, carrying two smartphones (Xiaomi Redmi 2 and Samsung Galaxy S6) equipped with WiFi and cellular interfaces. During this period the data has been sampled.

The measured mobile network traces are depicted in Figs. 8-9. We find that the bandwidth of both WiFi and cellular networks (3G and LTE) vary considerably over time and are highly unpredictable. Indoor WiFi, which has a good coverage, is stable and fast. But even in the same scenario, different mobile devices may record different levels of transmission speed. For example, the Samsung S6 has much higher bandwidth than the Xiaomi Redmi 2 in the indoor environment. This is because the two devices contain different hardware and software. The mobility of users has a significant impact on the network connection bandwidth quality. Outdoor WiFi wireless networks experience varying signal strength and suffer from frequent intermittent connectivities which make them unavailable from time to time. On the contrary, cellular networks are much more stable and also provide near-ubiquitous connectivity. Further, cellular connections can suffer from high latencies or round-trip time (RTT) and slow data transfers when compared with WiFi. We notice in general that the bandwidth of the downlink is higher at most times in most settings, sometimes by a considerable margin.

6.1.2 Energy Profiler

There are two ways to estimate the energy consumption, namely, software and hardware monitors. Some works [11], [36] used a power meter attached to the smartphone's battery to build an energy profile. Power Monitor (e.g., Monsoon monitor) is a device that measures energy consumption when data is transmitted from the mobile device to the cloud server by supplying a certain level of power to the mobile device. We use PowerTutor[2] to measure the power consumption of the applications. Although PowerTutor does not give as accurate results as a hardware power monitor, the results are still reasonable and provide some insight. PowerTutor provides detailed energy consumption information for each hardware component.

In Fig. 10 both energy consumption and transmission time increase in proportion to the transferred file sizes. When the same volume of data was transmitted, WiFi has relatively lower energy consumption than 3G. Moreover, the device's energy consumption via each communication network is proportional to its data transmission time.

1. A free connection analysis tool, which shows real-time download and upload graphs, stores results both locally and on the Internet for sharing, http://www.speedtest.net/

2. PowerTutor is an application for Android phones that provides accurate, real-time power consumption estimates for power-intensive hardware components, http://ziyang.eecs.umich.edu/projects/powertutor/

TABLE 2
Network Trace Data Sampling

Scene	Place	Mobility
Indoor (Static)	Office, Library, Classroom, Kitchen, Washing Room, Meeting Room, Student Cafeteria, Laboratory	Low
Outdoor (Dynamic)	Walk around the campus	Medium

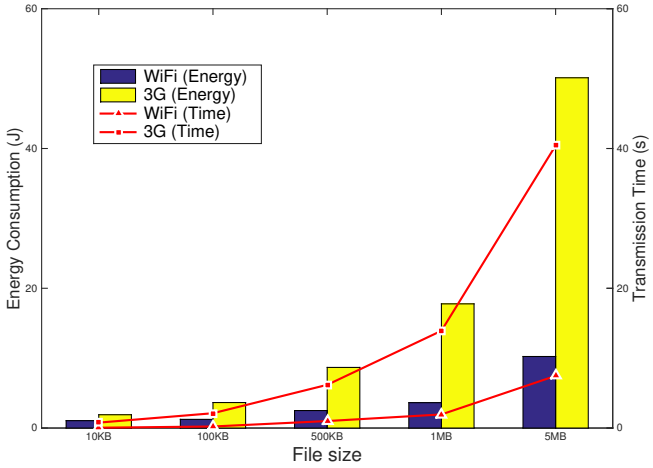


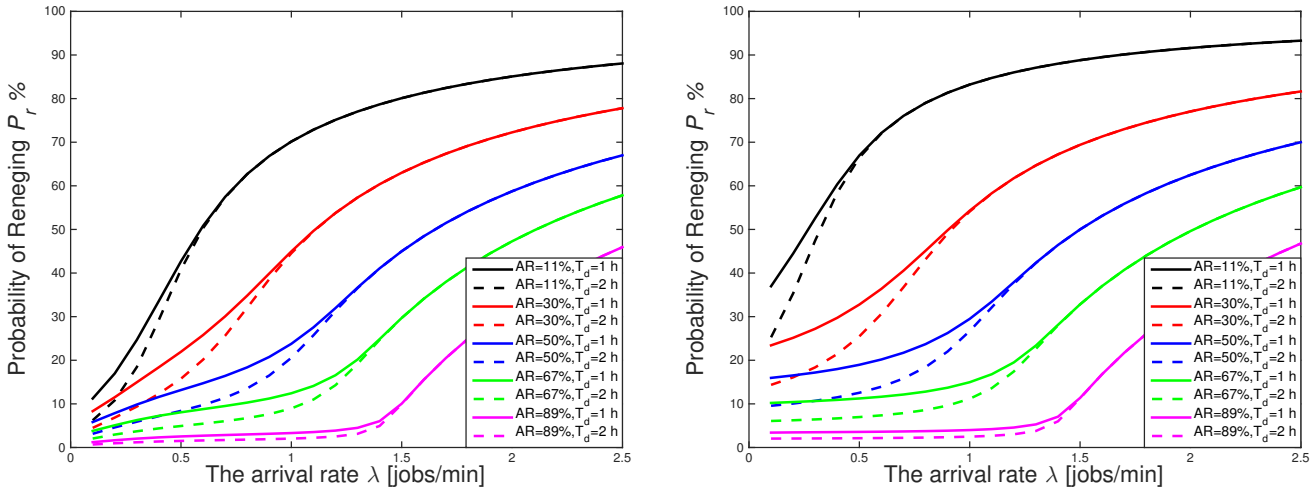
Fig. 10. The energy cost and transmission time with Xiaomi Redmi 2

6.2 Numerical Analysis

In this section, we will first derive the needed parameters from our experimental measurement results, and then we will analyze the models using those parameters.

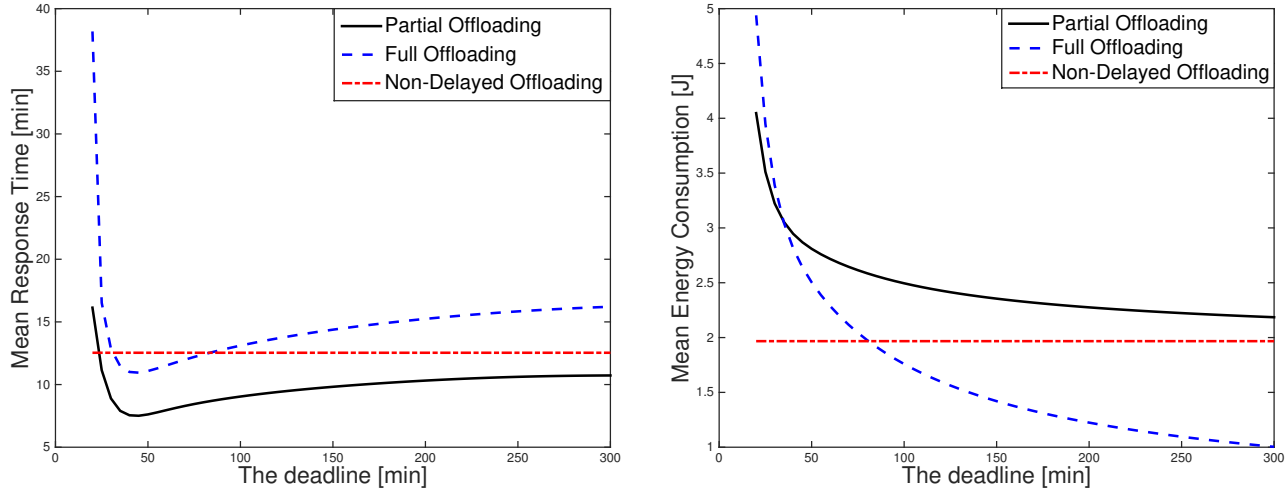
Different wireless network interfaces vary in many ways, which we have to capture in simplified form in just a few parameters. According to the mobile data traces collected above, we consider here a simple scenario where the transmission rate of the cellular network is lower than that of WiFi, i.e., $s_c < s_w$ and the power consumption when transmitting jobs via the cellular link is higher than when using the WiFi link, i.e., $p_c > p_w$. Using measurements from real traces in [5], the average data rates of the cellular and WiFi networks are set to $s_c = 200$ Kbps and $s_w = 2$ Mbps, respectively. The average duration of WiFi availability period is 52 min ($\xi = 1/52$ min^{-1}), while the average period duration with only cellular network coverage is 25.4 min ($\eta = 1/25.4$ min^{-1}). The availability ratio is thus 67%. The mean job size is assumed to be 10 MB. According to the power models developed in [37], we set the power coefficients to $p_c = 2.5$ W, $p_w = 0.7$ W and $p_m = 2$ W, respectively. In addition, suppose that the total job arrival rate is $\lambda = 0.5$ packet/min, the mobile service rate is $\mu_m = 0.2$ and the cloud service rate is $\mu_r = 1$.

We first analyze the probability of reneging for the two delayed offloading models. An availability ratio of 11% has been reported in [38]. Fig. 11 shows that as the availability ratio (AR) of the WiFi network increases, the fraction of jobs that abandon the *Offload Queue* (for the partial offloading model, refer to Fig. 11(a)) or the *WiFi Queue* (for the full offloading model, refer to Fig. 11(b)) declines rapidly. However, the full offloading model has much higher reneging (offload abandonment) probability than the partial offloading



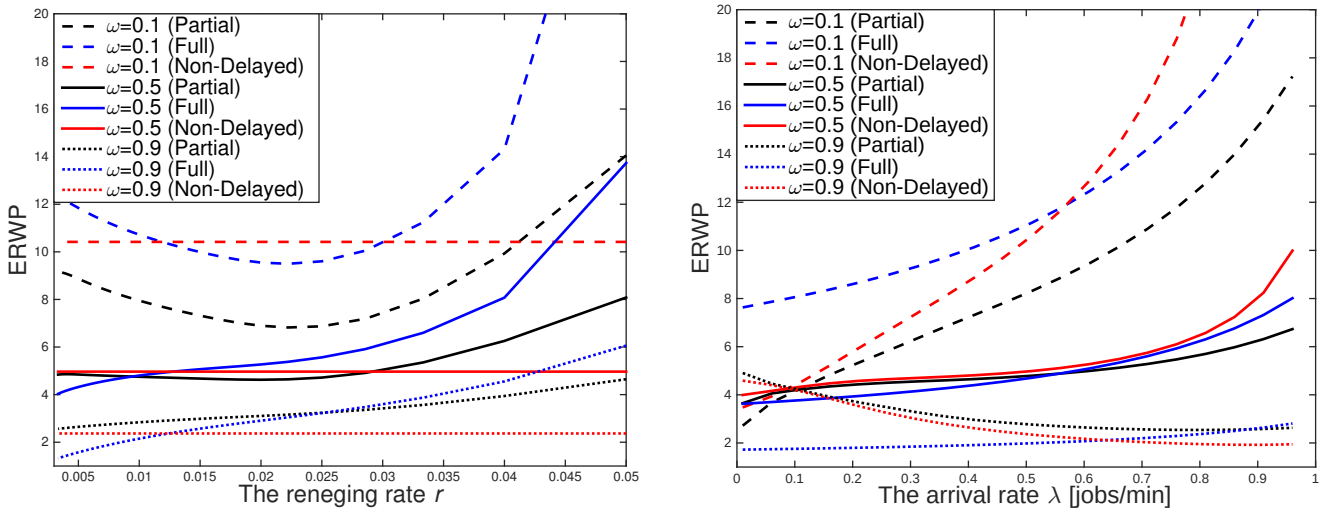
(a) Partial offloading model (b) Full offloading model

Fig. 11. The reneging probabilities for the delayed offloading models



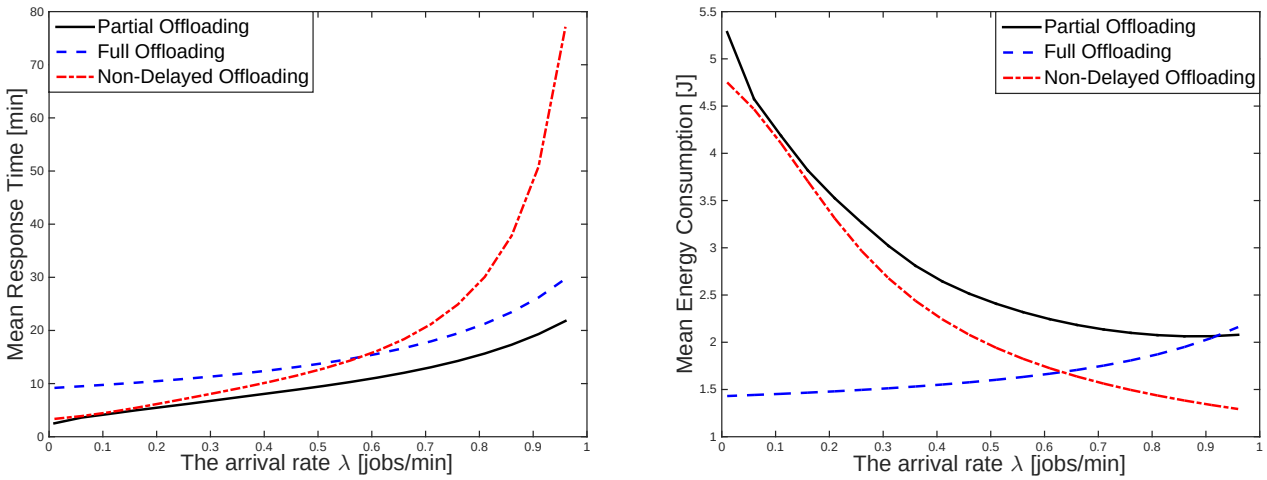
(a) Mean Response Time (b) Mean Energy Consumption

Fig. 12. Mean response time and energy consumption of the offloading models under different deadlines



(a) Reneging rate (b) Arrival rate

Fig. 13. Comparison of ERWP for the offloading models under different reneging rates and arrival rates



(a) Mean Response Time

(b) Mean Energy Consumption

Fig. 14. Mean response time and energy consumption of the offloading models under different arrival rates

model under the same deadline T_d. This can be explained by the fact that the partial offloading model can use the cellular network to transmit data and thus the number of jobs waiting in the *Offload Queue* is reduced. On the other hand, as the reneging deadline increases from 1 h to 2 h, jobs have a higher chance to be offloaded via the WiFi network, and therefore the reneging probability decreases for lower arrival rates. However, at high arrival rates, the reneging probability stays the same irrespective of the deadline.

The mean response time includes the queueing as well as the service time. From Fig. 12(a), it can be seen that the partial offloading model has the lowest average response time since it makes full use of the slow phase of the cellular network while WiFi is unavailable. For the lower deadlines ($T_d < 40$ min), the mean response time decreases as the deadline increases since jobs with higher deadlines have a better chance to transmit over the fast WiFi network, leading to shorter response time. However, the mean response time increases for higher deadlines, since jobs with lower deadlines leave the queue earlier, leading to smaller queueing delays. From Fig. 12(b), we can observe that when the reneging deadline is small, the non-delayed offloading model achieves the lowest mean energy consumption among the three models, but as the deadline increases, the full offloading model is much better. This is due to the fact that the WiFi network is much faster and more energy-efficient than the cellular network. The reduced service time can lead to lower energy consumption on the mobile device.

Please note that the inverse reneging rate corresponds to the mean deadline. Therefore the minimum in Fig. 12(a) for a deadline of ≈ 40 corresponds to the minimum ERWP in Fig. 13(a) at a reneging rate of 0.025.

Different applications usually assign different importance to the relative energy usage and performance. We use the ERWP metric to compare the three offloading models according to their energy-performance tradeoff. It can be observed from Fig. 13(a) that when ω is small, the partial offloading model can achieve the smallest ERWP value by optimally choosing the reneging rate r. This indicates that when considering response time more important (for delay-sensitive applications), it is better to use the partial offloading model. Otherwise, when considering energy consumption more important than response time (for delay-tolerance applications), the full offloading model should be preferred. The latter translates the reduced transmission time from the fast WiFi network into lower usage of battery power for the mobile device. As shown in Fig. 13(b), when the weighting parameter ω is small, as the arrival rate of the offloadable jobs λ increases, all the three offloading models perform worse. However, the non-delayed offloading model is more sensitive to the job arrival rates. The partial offloading model can always achieve the lowest ERWP value. This means that when considering response time more important, it is better to use the partial offloading model. Otherwise, when considering energy consumption more important than response time, the full offloading model should be preferred at low job arrival rate λ. While at higher arrival rate, the non-delayed offloading model performs better according to the ERWP metric.

We then fix the reneging deadline to 2 h and compare the mean response time and energy consumption under different values of the job arrival rate λ. As shown in Fig. 14(a), the mean response time increases with the increase of λ due to the queueing effects. The partial offloading model performs much better than the other two models since it fully uses the unavailability periods of WiFi by offloading jobs over a cellular network. This in turn leads to high energy consumption as shown in Fig. 14(b). The full offloading model is much more energy-efficient than the non-delayed offloading model at low arrival rate λ, while at high λ, the non-delayed offloading model can save much more energy. This can be drawn from Fig. 11(b)) since as λ increases, more jobs are abandoned from the *WiFi Queue*. Those jobs are then offloaded via the costly cellular network, which result in higher energy consumption.

6.3 Offloading Experiments

To confirm the insights we gained from analyzing the models we have run experiments using different deadlines

and different connectivity scenarios. For the experiments we chose a real-world setup which consisted of a predefined route during which we followed a predefined sequence of file uploads with specified file sizes. We conducted these experiments on a Google Nexus 5 device and had multiple runs where each run used a different deadline. The objective was to measure the upload performance and energy usage for different file sizes under changing circumstances regarding WiFi coverage using varying deadlines.

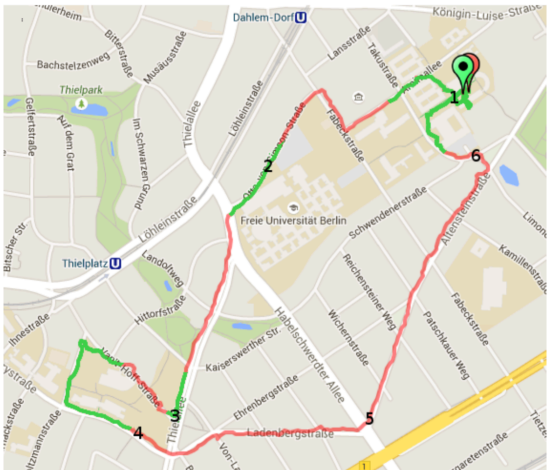


Fig. 15. The walking route around FUB campus (red line: WiFi unavailable, green line: WiFi available)

The walking route can be seen in Fig. 15, the route is marked with different colors showing WiFi availability, where green sections indicate that WiFi was available and red sections show that no WiFi was available. The positions at which the next file upload was issued are shown using numbers. The corresponding scheduled files are as follows: 50 MB, 1 MB, 10 MB, 1 MB, 10 MB, 50 MB. The length of the route was 5km and with an average walking speed of 5.5 km/h, it had a duration of 55 min. We used several test runs with different delays: no delay, 30 sec, and 30 min.

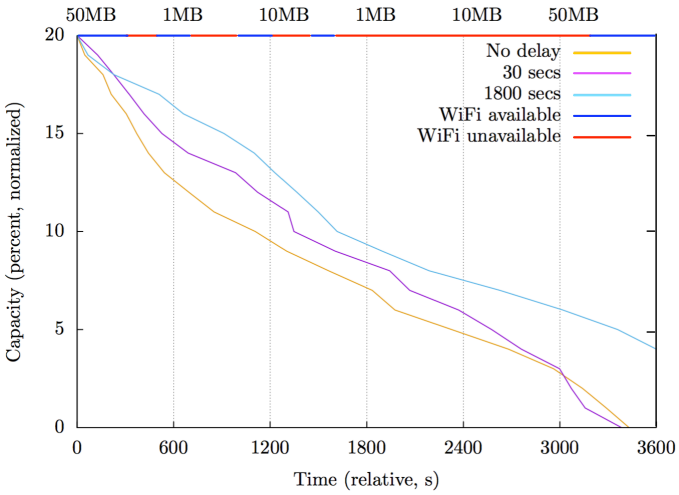


Fig. 16. Capacity discharge over time for different executions

Figure 16 shows the capacity discharge over time for different executions. This is the major performance evaluation metric, as the main goal of delayed offloading is to reduce the energy consumption. The x-axis shows the time in seconds relative to the start of the test. The normalized capacity discharge is shown on the y-axis in percent. On top of the diagram, the uploaded files are shown with their corresponding file size as well as the WiFi availability (blue=available; red=unavailable). This is a rather undisturbed test run that shows a decreasing energy consumption for increasing deadlines very well. Taking the case of the 30 sec deadline as an example, there is an obvious drop in battery capacity after 30 sec for the first upload of the 10 MB file, which clearly shows the high energy consumption of the 3G interface.

7 Conclusions

In this paper, we have developed analytical queueing models for delayed mobile cloud offloading to leverage the complementary strength of WiFi and cellular networks by choosing heterogeneous wireless interfaces for offloading. We have carried out optimality analysis of the energy-performance tradeoff for mobile cloud offloading systems based on the ERWP metric. This metric captures both, energy and performance characteristics. Our analysis even included intermittently available access links.

We find that when the availability ratio (AR) of the WiFi network is relatively low, the percentage of jobs that abandon the queue is very high. We can optimally choose the reneging deadline to achieve different energy-performance tradeoffs by optimizing the ERWP metric. For delay-sensitive applications, the partial offloading model is preferred when setting an intermediate deadline, while for delay-tolerant applications, the full offloading model shows very good results and outperforms the other offloading models when using a large deadline. In general one can say that the partial offloading policy is faster, while the full policy uses less energy.

When optimising the energy consumption the full offloading model will always be best, even if the deadline must be extremely long. Only if job response time is of high importance reasonable results for the tradeoff, captured in the ERWP metric, can be obtained. Then an optimal deadline to abort offloading in the partial offloading model or the WiFi transmission in the full offloading model can be found. For reduction of the energy consumption it will always be better to wait longer rather than compute locally or use the cellular network. The proposed queueing models can be used to describe complex and realistic offloading systems.

References

[1] 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.

[2] M. H. Cheung and J. Huang, "Dawn: Delay-aware Wi-Fi offloading and network selection," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 6, pp. 1214–1223, 2015.

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

[4] 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), 2015 IEEE 16th International Symposium on a*, pp. 1–9, IEEE, 2015.

[5] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can WiFi deliver?," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 2, pp. 536–550, 2013.

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

[7] F. Rebecchi, M. D. De Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti, "Data offloading techniques in cellular networks: A survey," *Communications Surveys and Tutorials, IEEE Communications Society*, vol. 17, no. 2, pp. 580–603, 2015.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] L. Wang and M. Franz, "Automatic partitioning of object-oriented programs for resource-constrained mobile devices with multiple distribution objectives," in *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on*, pp. 369–376, IEEE, 2008.

[17] L. Lei, Z. Zhong, K. Zheng, J. Chen, and H. Meng, "Challenges on wireless heterogeneous networks for mobile cloud computing," *Wireless Communications, IEEE*, vol. 20, no. 3, pp. 34–44, 2013.

[18] 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.

[19] 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 *INFOCOM, 2013 Proceedings IEEE*, pp. 195–199, IEEE, 2013.

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

[21] H. Deng and I.-H. Hou, "Online scheduling for delayed mobile offloading," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1867–1875, IEEE, 2015.

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

[23] 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.

[24] 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), 2014 8th International Conference on*, pp. 90–97, ICST, 2014.

[25] 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.

[26] 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*, vol. 157, no. 2, pp. 421–449, 2016.

[27] 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 ACM international symposium on Mobile ad hoc networking and computing*, pp. 267–276, ACM, 2014.

[28] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," *Performance Evaluation*, vol. 67, no. 11, pp. 1155–1171, 2010.

[29] 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), 2015 27th International*, pp. 134–142, IEEE, 2015.

[30] H. Wu and K. Wolter, "Analysis of the energy-performance tradeoff for delayed mobile offloading," in *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2015 9th International Conference on*, pp. 250–258, ICST, 2015.

[31] S. Balsamo, G.-L. dei Rossi, and A. Marin, "Queueing networks and conditional product-forms," in *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, pp. 204–213, ICST, 2013.

[32] N. Perel and U. Yechiali, "Queues with slow servers and impatient customers," *European Journal of Operational Research*, vol. 201, no. 1, pp. 247–258, 2010.

[33] U. Yechiali, "Queues with system disasters and impatient customers when system is down," *Queueing Systems*, vol. 56, no. 3-4, pp. 195–202, 2007.

[34] U. Yechiali and P. Naor, "Queuing problems with heterogeneous arrivals and service," *Operations Research*, vol. 19, no. 3, pp. 722–734, 1971.

[35] 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 *INFOCOM, 2012 Proceedings IEEE*, pp. 945–953, IEEE, 2012.

[36] J.-A. Hong, S. Seo, N. Kim, and B.-D. Lee, "A study of secure data transmissions in mobile cloud computing from the energy consumption side," in *Information Networking (ICOIN), 2013 International Conference on*, pp. 250–255, IEEE, 2013.

[37] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pp. 280–293, ACM, 2009.

[38] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 209–222, ACM, 2010.

Huaming Wu received the B.E. and M.S. degrees from Harbin Institute of Technology, China in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science at Freie Universität Berlin, Germany in 2015. He is currently an assistant professor in the Center for Applied Mathematics, Tianjin University. His research interests include model-based evaluation, wireless and mobile network systems, mobile cloud computing and deep learning.

Katinka Wolter received her PhD degree from Technische Universität Berlin in 1999. She has been Assistant professor at Humboldt-University Berlin and lecturer at Newcastle University before joining Freie Universität Berlin as a professor for dependable systems in 2012. Her research interests are model-based evaluation and improvement of dependability, security and performance of distributed systems and networks.