Tradeoff Analysis for Mobile Cloud Offloading Based on an Additive Energy-Performance Metric

Huaming Wu
Institute for Computer Science
Freie Universität Berlin
14195 Berlin, Germany
huaming.wu@fu-berlin.de

Katinka Wolter
Institute for Computer Science
Freie Universität Berlin
14195 Berlin, Germany
katinka.wolter@fu-berlin.de

ABSTRACT

Mobile offloading migrates heavy computation from mobile devices to powerful cloud servers. It is a promising technique that can save energy of the mobile device while keeping job completion time low when cloud servers are available and accessible. The benefit obtained by offloading greatly depends on whether it is applied at the right time in the right way. In this paper, we use queueing models to minimize a weighted sum of energy consumption and performance expressed in the Energy-Response time Weighted Sum (ERWS) metric. We consider different offloading policies (static and dynamic), where arriving jobs are processed either locally or remotely in the cloud. Offloading can be performed via WLAN or via a cellular network. The transmission techniques differ in energy requirement and speed. We find that the dynamic offloading policy derived from the tradeoff offloading policy (TOP) outperforms other policies like the random selection of transmission channel by a significant margin. This is because the dynamic offloading policy considers the increase in each queue and the change in metric that newly arriving jobs bring in should they be assigned to that queue. The ERWS metric can be reduced more by considering either energy consumption or response time and it is minimal when optimising only energy consumption.

Keywords

Energy-Performance Tradeoff, Queueing Model, Offloading Policies, Mobile Cloud Computing

1. INTRODUCTION

Mobile cloud computing aims at combining the strength of cloud computing and the convenience of mobile terminals. However, mobile devices have to master many challenges in order to effectively use cloud resources. Limited radio resources or limitations of other communication channels as well as lack of sufficient battery power may significantly impede the improvement of service quality [1] anticipated by using cloud services. Nonetheless computation offloading, which migrates computation-intensive tasks from mobile devices to a remote cloud infrastructure via a network is a popular approach to alleviate the burden of resource-constrained mobile devices. Since offloading an application to the cloud is not always possible or effective, one may consider sometimes executing a program locally and to offload only when available networks seem favorable to the desired metric.

A typical objective of computation offloading is to minimize the application response time (i.e., latency or delay). Mobile devices usually have multiple wireless interfaces for data transfer, such as 3G/EDGE and WiFi with different performance and energy consumption. Recently, in addition to response time, energy consumption of mobile devices has become an important criterion for network selection. Rahmati et al. [2] suggested on-the-spot network selection by examining the tradeoff between energy consumption for WiFi search and transmission efficiency when a WiFi network was intermittently available. A stochastic model for that matter has been developed in [3] using various performance metrics and also intermittently available access links. Some studies [4], [5] suggested energy-efficient delayed network selection to optimise the tradeoff between transmission power of heterogeneous network interfaces (e.g., 3G, WiFi) and transmission delay.

While we have previously studied the delay-energy tradeoff by deciding whether or not and how to offload entire applications [6], in this paper we assume that an application can consist of several components, or jobs, that are treated separately, and thus offloading decisions should be made for every job. The general cost metric includes energy consumption related costs in addition to the usual performance metrics such as the response time. We use queueing theory to model the offloading systems and employ the Energy-Response time Weighted Sum (ERWS) as metric to capture energy-performance tradeoffs. The ERWS metric has the advantage of being analytically tractable since the expectation is additive over time [7]. It has the disadvantage of a linear combination of two metrics on different scales and as a consequence it is always beneficial to the metric to concentrate efforts on reducing energy consumption.

Our queueing model is a fork-join model, similar to the one formulated in [8] for minimising subtask dispersion. However, in this paper we only analyse a submodel and do not exploit the fork-join structure.

The remainder of this paper is as follows. In Section 2, we introduce a queueing model for classic offloading systems and define an ERWS metric. Static and dynamic offloading policies under the ERWS metric are proposed in Section 3. Section 4 gives some numerical examples. And finally, the paper is concluded in Section 5.

2. SYSTEM OVERVIEW

2.1 The Queueing Model

We consider a queueing model for mobile cloud offloading systems as depicted in Fig.1. The mobile device, the cloud and the wireless networks are represented as queueing nodes to capture the resource contention and delay on these systems. The mobile device executes an application with different types of jobs that can be classified into the following two classes. Each time

VALUETOOLS 2014, December 09-11, Bratislava, Slovakia

DOI 10.4108/icst.valuetools.2014.258222

a job is executed, a decision must be taken into which class it belongs:

- **Unoffloadable**: some jobs should be unconditionally processed locally in the mobile device, either because transferring the relevant information would take more time and energy or because these tasks must access local devices (e.g., sensors, user interface, etc.) [9]. Local processing consumes battery power of the mobile device. Luckily, there are no communication costs or delays.
- **Offloadable**: some jobs are flexible tasks that can be processed either locally in the processor of the mobile device, or remotely in a cloud infrastructure through computation offloading. Many tasks fall into this category, and the offloading decision depends on whether the communication costs outweigh the local processing costs [10].

The problem of taking offloading decisions correctly does not exist for unoffloadable jobs. However, for offloadable ones, the mobile device should judiciously make decisions that optimise the considered metric.

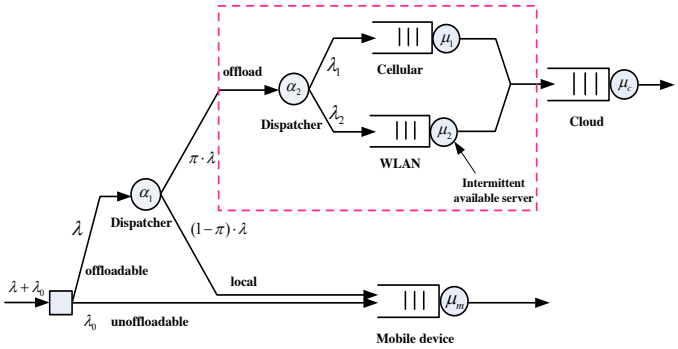


Figure 1: A queueing model for mobile cloud offloading systems

As depicted in Fig.1, job arrivals in the mobile device are assumed to follow a Poisson process with an average arrival rate of $\lambda + \lambda_0$. The arrival rate is based on the behavior of the application. The unoffloadable jobs with arrival rate λ_0 are unconditionally executed locally. As for the offloadable ones with arrival rate λ, the mobile device chooses to offload each job with probability π. As in [11], jobs are offloaded to the cloud following a Poisson process with an average arrival rate of $\pi \cdot \lambda$, which is called the offloading rate. There may be several ways to offload computation to the cloud, e.g., via a costly cellular connection (2G or 3G), or via intermittently available WLAN hotspots. The cellular interface can provide a ubiquitous coverage for mobile devices in a wide area, but it has lower data transmission rate and needs more transmission energy than the WiFi interface. The mobile device, the cellular and WLAN connections are modeled as $M/G/1$-FCFS queues, and the remote cloud is modeled as an $M/G/\infty$ queue, i.e., as a delay center [12]. We denote $1/\mu_m$ and $1/\mu_c$ 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 WLAN are μ_1 and μ_2, respectively.

2.2 Objectives

As shown in Fig.1, there are two dispatchers: α_1 is used to allocate the offloadable jobs either to the cloud or to the mobile device, while α_2 is to offload jobs either via a cellular connection or a WLAN network to the cloud. The total cost for offloading a job is composed of the cost for sending the job to the cloud, idly waiting for the cloud to complete the job, and receiving the result back from the cloud. Since the delay caused by the transmission in the uplink usually dominates the total cost, our analysis focuses on the dispatcher inside the red dotted block, which is responsible for selecting the best transmission channel. It is a fork-join queue where incoming jobs are split on arrival for service by two servers and joined before departure. Only when all the jobs are transmitted and have rejoined can the cloud processing start.

The objective is to minimize a weighted sum of the mean energy consumption and response time under the assumption that the dispatcher is aware of the remaining service time of each job in the system, including that of the arriving job [13].

2.3 Metrics

Energy consumption and response time are two primary aspects for mobile cloud systems that must be considered when making offloading decisions. The response time is the time between arrival of a job until it completes service and departs. The energy consumption is the energy spent on the mobile device in that period. We study the tradeoff between the mean energy consumption and mean response time, which is a non-trivial multi-objective optimization problem. It is addressed by setting the cost function as the weighted sum of both average values, i.e., the ERWS metric:

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

where $\mathbb{E}[T]$ is the mean response time, $\mathbb{E}[\mathcal{E}]$ is the mean energy consumption, and $0 \leq \omega \leq 1$ is a weighting parameter used to share relative importance between the mean energy consumption and mean response time.

For (1), the mean time and energy are additive terms over time, and thus we can optimize the ERWS metric via Markov Decision Processes [7]. From the view of minimization, this metric allows comparing arbitrary offloading policies to the optimal offloading policy in our work.

Since the consumed power is $P = \lambda\mathcal{E}$ [14], and by Little's Law, $\mathbb{E}[N] = \lambda\mathbb{E}[T]$, the objective of optimizing the ERWS metric in (1) can be more conveniently expressed as:

$$PQWS = \frac{1}{\lambda}\{\omega\mathbb{E}[P] + (1-\omega)\mathbb{E}[N]\} \quad (2)$$

where N is the number of jobs in the queueing system.

Thus, to analyze the energy-performance tradeoff, instead of optimizing the ERWS metric directly, we can also minimize a Power-Queue length Weighted Sum (PQWS) metric as shown in (2), which is a weighted sum of the mean power consumption and mean queue length (or average number of jobs) in a queueing system.

3. OFFLOADING POLICIES

In this section, we derive three offloading policies under the ERWS metric, defining different strategies to determine λ_1 and λ_2, i.e., to assign jobs upon arrival to one of two parallel queues which describe cellular or WLAN transmission.

3.1 Model and Problem Formulation

The interface selection problem in offloading systems is modeled as the decision to which queue an arriving job should be assigned. In this decision the offloading dispatcher takes both, performance and energy costs, into account. This seems natural for heterogeneous servers where a job needs a different

amount of energy and time to be served by different servers [15]. For example, whereas assigning each job to a low power server would be beneficial from the energy consumption perspective at low loads, such a policy may end up in difficulties at higher loads as the response time increases rapidly [13]. Indeed, the energy-performance tradeoff takes on different explicit expressions and under different offloading policies. Our objective is to minimize the ERWS metric under static and dynamic offloading policies.

As shown in Fig.2, we consider a queuing model that consists of two parallel queues of cellular and WLAN with work conserving queuing disciplines. λ_1 and λ_2 are the mean rates of jobs into Queue 1 and Queue 2. Since the original offloading jobs arrive at the system according to a Poisson process with rate $\lambda_c = \pi \cdot \lambda$, one policy would be random assignment into Queue i ($i \in \{1, 2\}$), which results in independent Poisson processes with rate λ_i, and we have $\lambda_1 + \lambda_2 = \lambda_c$.

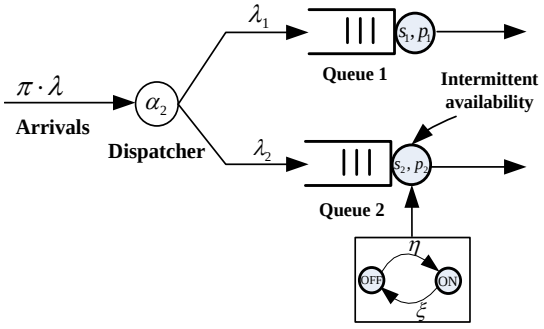


Figure 2: A queueing model for offloading assignment decisions

The two queues have the following behavior:

- **Queue 1**: When a job is offloaded to the cloud via a cellular network (2G/3G), there is queueing due to the transmission speed of the cellular link. Costs arise in terms of transmission delays (queueing and actual transmission time) and transmission energy consumption. We assume that the service speed (or transmission rate) is s_1, and the operating power for Server 1 is p_1 when serving jobs and zero whenever idle. Further, Server 1 is always available since the cellular connection is always on.

- **Queue 2**: When a job is offloaded to the cloud via a WLAN network, there is queueing due to the transmission speed of the WLAN link. We assume that Server 2 runs at speed s_2, and its operating power is p_2 when serving jobs and zero whenever idle. We model the intermittent availability of hotspots as a FCFS queue with occasional server break-down [3]. The availability of Server 2 is governed by an interrupted Poisson Process (IPP) with exponentially distributed ON-OFF periods. Specifically, the server is either in ON-state processing the existing jobs, or in OFF-state during which no job receives service. We assume that the sojourn time in a hotspot and the time to move from one hotspot to another are exponentially distributed with parameters ξ (failure rate), and η (recovery rate), respectively.

Different wireless network interfaces vary in many ways, which we have to capture in simplified form in just few parameters. Cellular networks such as EDGE and 3G, usually have much higher availability than WiFi, but the transmission rate of WiFi is higher (we do not consider 4G LTE networks here, which can achieve much faster speed than common WiFi router, but are still not widely available). Cellular networks transmit at hundreds of *Kbps* for EDGE to a few *Mbps* for 3G, WiFi at ten or more *Mbps*. Besides, the WiFi interface is more energy-efficient than the cellular interface [5]. These imply that Queue 2 is usually much faster and more energy-efficient than Queue 1 for transmitting the same quantity of data. Therefore, we just assume that Queue 1 has lower service rate and higher energy consumption than Queue 2, i.e., $s_1 < s_2$ and $p_1 > p_2$.

In minimizing the ERWS metric, it seems favorable to assign jobs to Queue 2 rather than to Queue 1. However, when taking the WLAN link's intermittent availability in Queue 2 into account, the optimal assignment has to be reconsidered.

Upon arrival of a job an assignment decision is made and the job is placed into the corresponding queue according to the ERWS metric denoted as:

$$ERWS = \frac{1}{\lambda_c} \sum_{i=1}^{2} \lambda_i \Big\{ \omega \mathbb{E}[\mathcal{E}_i] + (1-\omega)\mathbb{E}[T_i] \Big\} \tag{3}$$

where λ_c is the total job arrival rate for offloading, λ_i is the mean rate of jobs into Queue i, $\mathbb{E}[\mathcal{E}_i]$ and $\mathbb{E}[T_i]$ are the mean energy consumption and the mean response time in Queue i, respectively.

A key assumption in our work is that each server operates at a constant power p_i whenever the server is busy. Since $P_i = \lambda_i \mathcal{E}_i$ is the consumed power, further by Little's Law, $\mathbb{E}[N_i] = \lambda_i \mathbb{E}[T_i]$, the ERWS metric in (3) can be more conveniently expressed by the PQWS metric:

$$\begin{aligned} PQWS &= \frac{1}{\lambda_c} \sum_{i=1}^{2} \Big\{ \omega \mathbb{E}[P_i] + (1-\omega)\mathbb{E}[N_i] \Big\} \\ &= \frac{1}{\lambda_c} \sum_{i=1}^{2} \Big\{ \omega p_i \Pr\{N_i > 0, e_i = 1\} + (1-\omega)\mathbb{E}[N_i] \Big\} \end{aligned} \tag{4}$$

where Pr is the probability operation, $e_i = 1$ indicates that Server i is available and N_i is the number of jobs in Queue i.

Since the number of jobs in Queue i can be dependent of the state of Server i, we have:

$$\Pr\{N_i > 0, e_i = 1\} = \Pr\{N_i > 0 | e_i = 1\} \cdot \Pr\{e_i = 1\}. \tag{5}$$

Further, since Server 1 is always available, we have $\Pr\{e_1 = 1\} = 1$ and $\Pr\{N_1 > 0 | e_1 = 1\} = \Pr\{N_1 > 0\}$. The fraction of time that Server 2 is available to process jobs is:

$$\Pr\{e_2 = 1\} = \frac{\eta}{\xi + \eta} \triangleq \gamma \tag{6}$$

where as the recovery rate $\eta \to \infty$, the availability of Server 2 tends to be 1.

Since the probability that the corresponding server is busy is equal to the work load [14], we have:

$$\Pr\{N_i > 0\} = \rho_i \tag{7}$$

where ρ_i is the work load of Queue i.

We can further formulate the optimization of the PQWS metric for the offloading assignment as:

$$\lambda_i^* = \arg\min_{\lambda_i} PQWS \tag{8}$$

and we find the arrival rate λ_i^* to Queue i such that PQWS is minimised when both queues are in operation.

3.2 Static Offloading Policy

The static offloading policy is to assign arriving jobs using the optimal assignment scheme which corresponds to the smallest possible cost in the PQWS metric. We always assign the offloading jobs according to that scheme.

The expected number of jobs in Queue 1 is given by the Pollaczek-Khinchine formula:

$$\mathbb{E}[N_1] = \lambda_1 \mathbb{E}[S_1] + \frac{\lambda_1^2 \mathbb{E}[S_1^2]}{2(1-\rho_1)}. \tag{9}$$

Note, that a job of size X served at speed s will be completed at time X/s. Since $\mathbb{E}[S_i] = \mathbb{E}[X/s_i] = \mathbb{E}[X]/s_i$, the work load for Queue 1 during the busy period is $\rho_1 = \lambda_1 \mathbb{E}[S_1] = \lambda_1 \mathbb{E}[X]/s_1$. Especially, if the job size X is exponentially distributed, according to $\mathbb{E}[X^2] = 2\mathbb{E}^2[X]$ and $\mathbb{E}[S_i^2] = \mathbb{E}[(X/s_i)^2] = \mathbb{E}[X^2]/s_i^2$, we further have:

$$\begin{aligned} \mathbb{E}[N_1] &= \lambda_1 \mathbb{E}[X]/s_1 + \frac{2\lambda_1^2 \mathbb{E}^2[X]/s_1^2}{2(1-\rho_1)} \\ &= \rho_1 + \frac{2\rho_1^2}{2(1-\rho_1)} \\ &= \frac{\rho_1}{1-\rho_1}. \end{aligned} \tag{10}$$

Similarly, if the server in Queue 2 is always available like Queue 1, we can have $\Pr\{N_2 > 0 | e_2 = 1\} = \lambda_2 \mathbb{E}[S_2] = \lambda_2 \mathbb{E}[X]/s_2$. However, Queue 2 refers to offloading jobs from the mobile device to the cloud via a WLAN network, which is modelled as an $M/G/1$-FCFS queue with intermittently available service. When a server recovers, it continues to serve the customer whose service has been interrupted, i.e., the work already completed is not lost (cf. data transfers with resume). The mean response time for Queue 2 is then given by [3]:

$$\mathbb{E}[T_2] = \mathbb{E}[Y] + \frac{\lambda_2 \mathbb{E}[Y^2]}{2(1-\rho_2)} + \frac{\kappa(1+\lambda_2/\eta)}{1+\lambda_2\kappa} \tag{11}$$

where $\kappa = \frac{\xi/\eta}{\lambda_2+\xi+\eta}$, $\mathbb{E}[Y] = \mathbb{E}[S_2]/\gamma$ and $\mathbb{E}[Y^2] = \mathbb{E}[S_2^2]/\gamma^2 + 2\xi/\eta^2 \mathbb{E}[S_2]$.

Similarly, the work load for Queue 2 during the busy period is $\rho_2 \triangleq \lambda_2 \mathbb{E}[Y] = \lambda_2 \mathbb{E}[S_2]/\gamma$. Since we require $\rho_2 < 1$ for the queue to be stable, it yields: $\lambda_2 \mathbb{E}[S_2] < \gamma$.

According to Little's Law, the expected number of jobs in Queue 2 is $\mathbb{E}[N_2] = \lambda_2 \mathbb{E}[T_2]$. Especially, if X is exponentially distributed, $\mathbb{E}[X^2] = 2\mathbb{E}^2[X]$, we further have:

$$\begin{aligned} \mathbb{E}[N_2] &= \lambda_2 \mathbb{E}[Y] + \frac{\lambda_2^2 \mathbb{E}[Y^2]}{2(1-\rho_2)} + \frac{\kappa(\lambda_2 + \lambda_2^2/\eta)}{1+\lambda_2\kappa} \\ &= \rho_2 + \frac{\lambda_2^2 \mathbb{E}[S_2^2]}{2\gamma^2(1-\rho_2)} + \frac{\frac{2\xi}{\eta^2}\lambda_2^2 \mathbb{E}[S_2]}{2(1-\rho_2)} + \frac{\kappa(\lambda_2 + \lambda_2^2/\eta)}{1+\lambda_2\kappa} \\ &= \rho_2 + \frac{2\lambda_2^2 \mathbb{E}^2[X]/s_2^2}{2\gamma^2(1-\rho_2)} + \frac{\frac{\xi}{\eta^2}\lambda_2^2 \mathbb{E}[X]/s_2}{1-\rho_2} + \frac{\kappa(\lambda_2 + \lambda_2^2/\eta)}{1+\lambda_2\kappa} \\ &= \rho_2 + \frac{\rho_2^2}{1-\rho_2} + \frac{\xi\lambda_2^2 \mathbb{E}[X]}{\eta^2 s_2(1-\rho_2)} + \frac{\kappa(\lambda_2 + \lambda_2^2/\eta)}{1+\lambda_2\kappa} \\ &= \frac{\rho_2}{1-\rho_2} + \frac{\xi\lambda_2^2 \mathbb{E}[X]}{\eta^2 s_2(1-\rho_2)} + \frac{\kappa(\lambda_2 + \lambda_2^2/\eta)}{1+\lambda_2\kappa}. \end{aligned} \tag{12}$$

After substituting (5), (10) and (12) into (8), we obtain a new offloading policy called *Tradeoff Offloading Policy*, and we bring in two previous policies named *Random Offloading Policy* and *Load-balanced Offloading Policy* as a comparison. All the three offloading policies used in our analysis are listed as follows:

- **Random Offloading Policy (ROP)**: arriving jobs are randomly assigned to the two queues (Bernoulli split), assuming that each queue has the same probability of being chosen. Therefore, for the queueing model in Fig. 2, we have job arrival rate $\lambda_1 = \lambda_2 = \lambda_c/2$. This is a static policy that randomly chooses the transmission channel.

- **Load-balanced Offloading Policy (LOP)**: since the service rate in Queue 2 is much faster than that in Queue 1, more jobs can be served in Queue 2 in the same time. However, since Server 2 is sometimes unavailable, the actual service rate could in fact be lower. When considering the service rate and the availability of servers in different queues, jobs are allocated to Queue i in a Poisson process with parameters $\lambda_1 = \frac{s_1}{s_1+s_2\gamma}\lambda_c$ and $\lambda_2 = \frac{s_2\gamma}{s_1+s_2\gamma}\lambda_c$. Thus, under this policy all the queue loads are equal and given by $\rho_1 = \rho_2 = \frac{\lambda_c \mathbb{E}[X]}{s_1+s_2\gamma}$. This is a static policy that balances the load across the queues.

- **Tradeoff Offloading Policy (TOP)**: arriving jobs are assigned to Queue 1 and Queue 2 according to the optimized objective of the PQWS metric defined in (8), minimizing the weighted sum of mean energy consumption and mean response time. This is the proposed static policy that considers the tradeoff between energy consumption and performance according to the ERWS metric.

3.3 Dynamic Offloading Policy

The dynamic offloading policy inserts a newly arriving job tentatively into each of the queues and chooses the one corresponding to the smallest increase in the PQWS metric when also counting the jobs that have been already in the system. We adopt a value function defined in [16] to assign jobs dynamically.

We assume a size-aware system, where the service time of jobs becomes known upon arrival. Let $\Delta_j^{(i)}$ denote the remaining service time of job j ($j = 1, 2, \cdots, n$) in Queue i ($i \in \{1, 2\}$), where jobs are ordered such that job 1 receives service first, then job 2, and so forth. The state of the server is known. Since Server 1 is always available, the state of Queue 1 can be denoted by vector $\vec{z}^{(1)}$:

$$\vec{z}^{(1)} = \big(\Delta_1^{(1)}, \Delta_2^{(1)}, \cdots, \Delta_n^{(1)} |\, e_1 = 1\big). \tag{13}$$

Similarly, the state of Queue 2 is denoted by vector $\vec{z}^{(2)}$:

$$\vec{z}^{(2)} = \big(\Delta_1^{(2)}, \Delta_2^{(2)}, \cdots, \Delta_n^{(2)} |\, e_2 \in \{0, 1\}\big). \tag{14}$$

The backlog, i.e., the amount of service time for unfinished jobs in Queue i, is denoted by $U_{\vec{z}^{(i)}} = \sum_{j=1}^{n} \Delta_j^{(i)}$, and the cumulative response time during time interval $(0, t)$ [17] is expressed as:

$$V_{\vec{z}^{(i)}}(t) \triangleq \int_0^t N_{\vec{z}^{(i)}}(\tau) d\tau \tag{15}$$

where $N_{\vec{z}^{(i)}}(t)$ is the number of jobs in Queue i at time t.

The relative value is the expected difference in the cumulative costs between a system initially in state $\vec{z}$ and a system initially in equilibrium [17]:

$$v_{\vec{z}^{(i)}} \triangleq \lim_{t\to\infty} \mathbb{E}\Big[V_{\vec{z}^{(i)}}(t) - \mathbb{E}[N_{\vec{z}^{(i)}}] \cdot t\Big]. \tag{16}$$

According to (4), the cumulative cost of the PQWS metric in

Queue i is:

$$\omega p_i \Pr\{N_{\vec{z}^{(i)}} > 0, e_i = 1\} + (1-\omega)\mathbb{E}[N_{\vec{z}^{(i)}}] \tag{17}$$

where the first term corresponds to the mean energy consumption and the second corresponds to the mean response time.

The difference $\vec{z}^* - \vec{z}$ that characterizes the expected difference in the future costs between states $\vec{z}^*$ and $\vec{z}$ is the quantity on which job assignment is based. When a new job is admitted to one of the two queues, it tends to increase the total cost of the PQWS metric. The complete understanding of the future costs is summarized in the relative value $v_{\vec{z}}$. For a fixed policy resulting in a stable system, the relative value $v_{\vec{z}} - v_{\vec{0}}$ gives the expected difference in the infinite horizon of cumulative costs between an arbitrary state $\vec{z}$, and an empty system initially in state $\vec{0}$ with no jobs.

PROPOSITION 1. *For Queue i, the difference of relative values with respect to the weighted sum of the energy consumption and response time in (17) can be calculated by:*

$$v_{\vec{z}^{(i)}} - v_{\vec{0}^{(i)}} = \omega\big(v^E_{\vec{z}^{(i)}} - v^E_{\vec{0}^{(i)}}\big) + (1-\omega)\big(v^T_{\vec{z}^{(i)}} - v^T_{\vec{0}^{(i)}}\big). \tag{18}$$

PROOF. The difference of relative values with respect to the weighted sum of the energy consumption and response time can be decomposed into:

$$\begin{aligned} v_{\vec{z}^{(i)}} - v_{\vec{0}^{(i)}} &= \big[\omega v^E_{\vec{z}^{(i)}} + (1-\omega)v^T_{\vec{z}^{(i)}}\big] - \big[\omega v^E_{\vec{0}^{(i)}} + (1-\omega)v^T_{\vec{0}^{(i)}}\big] \\ &= \omega\big(v^E_{\vec{z}^{(i)}} - v^E_{\vec{0}^{(i)}}\big) + (1-\omega)\big(v^T_{\vec{z}^{(i)}} - v^T_{\vec{0}^{(i)}}\big) \end{aligned}$$

from which the result follows directly. □

It can be seen from (18) that the difference of relative values with respect to the PQWS metric in state $\vec{z}$ can be decomposed into the difference of relative values with respect to energy consumption and the difference of relative values with respect to response time, which can be treated separately as follows.

PROPOSITION 2. *For Queue i with a work conserving queuing discipline, the difference of relative values with respect to energy consumption can be calculated as [13]:*

$$v^E_{\vec{z}^{(i)}} - v^E_{\vec{0}^{(i)}} = \Pr\{e_i = 1\} \cdot p_i U_{\vec{z}^{(i)}}. \tag{19}$$

PROOF. We assume identical arrivals to a single-server system initially in state $\vec{z}$ and to an empty single-server reference system. When all the initial work $U_{\vec{z}^{(i)}}$ in the queue i is served, both systems are empty. Thus, the difference of energy consumption between the two systems is $p_i U_{\vec{z}^{(i)}}$, which is simply the difference in the energy needed to serve the initial work in the systems. Further, since energy is consumed only when the server is available, we derive (19). □

According to **Proposition 2**, since Server 1 is always available, the difference of relative values with respect to energy consumption is given by:

$$v^E_{\vec{z}^{(1)}} - v^E_{\vec{0}^{(1)}} = \Pr\{e_1 = 1\} \cdot p_1 U_{\vec{z}^{(1)}} = p_1 U_{\vec{z}^{(1)}}. \tag{20}$$

Similarly, since Server 2 is intermittently available, the difference of relative values with respect to energy consumption is given by:

$$v^E_{\vec{z}^{(2)}} - v^E_{\vec{0}^{(2)}} = \Pr\{e_2 = 1\} \cdot p_2 U_{\vec{z}^{(2)}} = \gamma \cdot p_2 U_{\vec{z}^{(2)}}. \tag{21}$$

PROPOSITION 3. *For the stable $M/G/1$-FCFS Queue 1, the difference of relative values with respect to response time is given by [16]:*

$$v^T_{\vec{z}^{(1)}} - v^T_{\vec{0}^{(1)}} = \frac{\lambda_1 U^2_{\vec{z}^{(1)}}}{2(1-\rho_1)} + \sum_{j=1}^{n}(n+1-j)\Delta^{(1)}_j. \tag{22}$$

And for the stable $M/G/1$-FCFS Queue 2 with intermittently available service, the difference of relative values with respect to the response time is given by [3]:

$$\begin{aligned} v^T_{\vec{z}^{(2)}} - v^T_{\vec{0}^{(2)}} = \frac{\lambda_2 U^2_{\vec{z}^{(2)}}}{2\gamma^2(1-\rho_2)} + \frac{1}{\gamma}\sum_{j=1}^{n}(n+1-j)\Delta^{(2)}_j \\ + \frac{1-e_2}{\eta}\Big[n + \frac{\lambda_2 U_{\vec{z}^{(2)}}}{\gamma(1-\rho_2)}\Big]. \end{aligned} \tag{23}$$

From (23), we note that the terms with $(1-e_2)$ as factor correspond to an additional penalty due to the currently unavailable Server 2.

According to **Proposition 1**, after substituting (20-23) into (18), we can get the differences of relative values with respect to the PQWS metric for Queue 1 and Queue 2 as follows, respectively:

$$\begin{aligned} v_{\vec{z}^{(1)}} - v_{\vec{0}^{(1)}} = \omega p_1 U_{\vec{z}^{(1)}} + (1-\omega)\Big[\frac{\lambda_1 U^2_{\vec{z}^{(1)}}}{2(1-\rho_1)} \\ + \sum_{j=1}^{n}(n+1-j)\Delta^{(1)}_j\Big] \end{aligned} \tag{24}$$

$$\begin{aligned} v_{\vec{z}^{(2)}} - v_{\vec{0}^{(2)}} = \omega\gamma p_2 U_{\vec{z}^{(2)}} + (1-\omega)\Bigg\{\frac{\lambda_2 U_{\vec{z}^{(2)}}}{2\gamma^2(1-\rho_2)} \\ + \frac{1}{\gamma}\sum_{j=1}^{n}(n+1-j)\Delta^{(2)}_j + \frac{1-e_2}{\eta}\Big[n + \frac{\lambda_2 U_{\vec{z}^{(2)}}}{\gamma(1-\rho_2)}\Big]\Bigg\}. \end{aligned} \tag{25}$$

We assume that n jobs have already arrived to the queuing system, for a newly arriving job of size x^* with index $(n+1)^{\text{th}}$, the service time of the new job $\Delta^{(i)}_{n+1} = x^*/s_i$ is inserted according to the queuing discipline used in Queue i. Therefore, the mean additional increase in future costs to assign the new job is $v_{\vec{z}^{(i)}_{\text{new}}} - v_{\vec{z}^{(i)}}$, where $\vec{z}^{(i)}_{\text{new}} = \vec{z}^{(i)} \oplus \Delta^{(i)}_{n+1}$ is the new state if the arriving job with service time $\Delta^{(i)}_{n+1}$ is added to Queue i.

PROPOSITION 4. *The relative increment of the PQWS metric with the admission cost of a job with service time $\Delta^{(i)}_{n+1}$ in Queue i is as follows:*

$$\begin{aligned} v_{\vec{z}^{(1)}_{new}} - v_{\vec{z}^{(1)}} = \omega p_1 \Delta^{(1)}_{n+1} \\ + (1-\omega)\left[\frac{\lambda_1 \Delta^{(1)}_{n+1}(2U_{\vec{z}^{(1)}} + \Delta^{(1)}_{n+1})}{2(1-\rho_1)} + U_{\vec{z}^{(1)}} + \Delta^{(1)}_{n+1}\right] \end{aligned} \tag{26}$$

$$\begin{aligned} v_{\vec{z}^{(2)}_{new}} - v_{\vec{z}^{(2)}} = \omega\gamma p_2 \Delta^{(2)}_{n+1} + (1-\omega)\Bigg\{\frac{\lambda_2 \Delta^{(2)}_{n+1}\big(2U_{\vec{z}^{(2)}} + \Delta^{(2)}_{n+1}\big)}{2\gamma^2(1-\rho_2)} \\ + \frac{1}{\gamma}\big(U_{\vec{z}^{(2)}} + \Delta^{(2)}_{n+1}\big) + \frac{1-e_2}{\eta}\Big[1 + \frac{\lambda_2 \Delta^{(2)}_{n+1}}{\gamma(1-\rho_2)}\Big]\Bigg\}. \end{aligned} \tag{27}$$

PROOF. According to (24) and (25), we have the relative

values of the PQWS metric in the new state $\vec{z}^{(i)}_{\text{new}}$:

$$v_{\vec{z}^{(1)}_{\text{new}}} - v_{\vec{0}^{(1)}} = \omega p_1 \big(U_{\vec{z}^{(1)}} + \Delta^{(1)}_{n+1}\big) + (1-\omega)\Big[\frac{\lambda_1 \big(U_{\vec{z}^{(1)}} + \Delta^{(1)}_{n+1}\big)^2}{2(1-\rho_1)} + \sum_{j=1}^{n+1}(n+2-j)\Delta^{(1)}_j\Big] \quad (28)$$

$$v_{\vec{z}^{(2)}_{\text{new}}} - v_{\vec{0}^{(2)}} = \omega\gamma p_2 \big(U_{\vec{z}^{(2)}} + \Delta^{(2)}_{n+1}\big) + (1-\omega)\Big\{\frac{\lambda_2 \big(U_{\vec{z}^{(2)}} + \Delta^{(2)}_{n+1}\big)^2}{2\gamma^2(1-\rho_2)} + \frac{1}{\gamma}\sum_{j=1}^{n+1}(n+2-j)\Delta^{(2)}_j + \frac{1-e_2}{\eta}\Big[n+1+\frac{\lambda_2\big(U_{\vec{z}^{(2)}} + \Delta^{(2)}_{n+1}\big)}{\gamma(1-\rho_2)}\Big]\Big\}. \quad (29)$$

The relative increment of the PQWS metric can be decomposed into the difference of relative values as:

$$v_{\vec{z}^{(i)}_{\text{new}}} - v_{\vec{z}^{(i)}} = \big[v_{\vec{z}^{(i)}_{\text{new}}} - v_{\vec{0}^{(i)}}\big] - \big[v_{\vec{z}^{(i)}} - v_{\vec{0}^{(i)}}\big]. \quad (30)$$

After substituting (24) and (28) into (30), we can derive (26), and then after substituting (25) and (29) into (30), we further derive (27). □

According to (26) and (27), we obtain the dynamic offloading assignment policy that assigns the newly arriving job to Queue i^* according to:

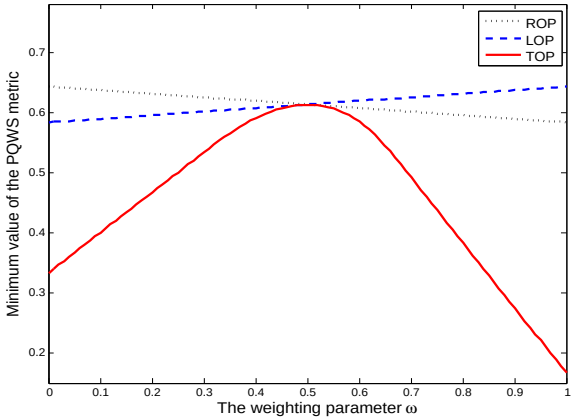
$$i^* = \arg\min_i \big\{v_{\vec{z}^{(i)}_{\text{new}}} - v_{\vec{z}^{(i)}}\big\}. \quad (31)$$

4. NUMERICAL EXAMPLES

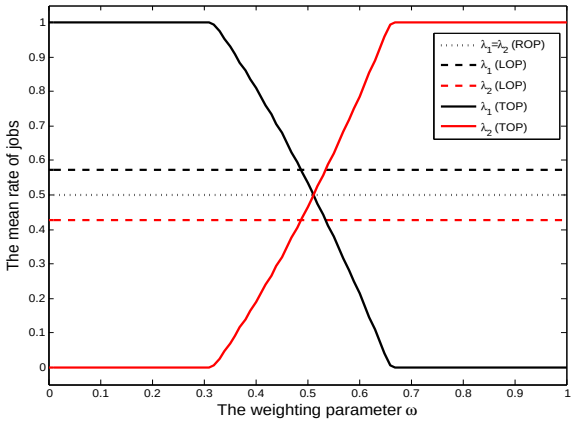
In this section we evaluate the assignment policies defined in the previous sections using the model with two queues to represent the offloading scheme. We set the parameters for Server 1 to $s_1 = 400Kbps$, $p_1 = 4W$ and for Server 2 to $s_2 = 600Kbps$, $p_2 = 2W$. This seems reasonable since the transmission rate of the cellular network is smaller than that of WLAN, i.e., $s_1 < s_2$ and the power consumption when transmitting jobs via the cellular link is larger than the WLAN link, i.e., $p_1 > p_2$. Besides, suppose that the total job arrival rate for offloading is $\lambda_c = 1$ *packet/s*, both the failure rate ξ and recovery rate η of Server 2 are equal to 1, and the packet sizes are exponentially distributed with $X \sim \text{Exp}(100)$ Kb.

One can see in Fig.3(a) that the proposed tradeoff offloading policy (TOP) performs significantly better than the random offloading policy (ROP) and the load-balanced offloading policy (LOP). Interestingly, all three policies achieve the same minimum value of the PQWS metric when $\omega = 0.5$, where the mean energy consumption and mean response time are equally important. At low values of ω, the response time is always more important than the energy consumption and dominates the queue selection. At high values, the energy consumption component becomes the decisive factor. The smallest PQWS metric is obtained when $\omega = 1$ and the TOP assignment is used. This shows that TOP outperforms other policies most in the case when only the energy consumption is considered in the offloading system.

The corresponding assignments under different policies are depicted in Fig.3(b). For the ROP and LOP schemes, the assignment is insensitive to the weight parameter ω. As expected, the random policy assigns half of the jobs to either queue. However, for the TOP scheme, at low values of ω, all the jobs are directed to Queue 1, and as ω increases, slowly more and more jobs are assigned to Queue 2 to balance the tradeoff between



(a) Minimum value of the PQWS metric



(b) Corresponding dispatching results

Figure 3: Comparison of different static offloading policies

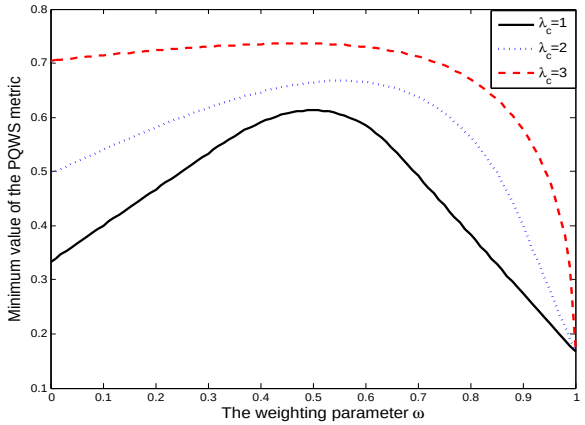
energy consumption and performance.

The impact of the total job arrival rate λ_c is shown in Fig.4(a), where we only use the static offloading policy TOP. The effect mentioned above, that optimising one element of the combined metric minimises PQWS holds for different values of the arrival rate, but is much more pronounced at high load. In fact, for high load focussing on energy consumption ($\omega = 1$) gives much lower PQWS than only regarding the job response time ($\omega = 0$). The corresponding assignment results under different λ_c are depicted in Fig.4(b). It should be noted that for higher load the point where jobs are assigned to both queues in equal shares is when $\omega > 0.5$, i.e., when energy cost are weighed more than response time in the metric.

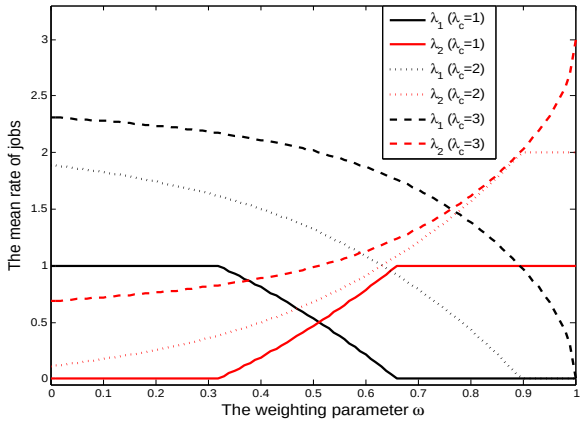
In Fig.5 we study the recovery rate of Server 2. Obviously, the faster recovery is, the lower becomes the PQWS metric under TOP policy. This is because as $\eta \to \infty$, Server 2 is always available and then the preferred scheduling target.

To illustrate the dynamic offloading policy, we assume that the length of a newly arriving job is $x^* = 100Kb$. As shown in Fig.6, the axes represent the existing work $U_{\vec{z}^{(i)}}$ in Queue i, the coloured lines denote the threshold under different ω where the arriving job has equal probability to join both queues. The areas above the thresholds refer to choosing Queue 1, while the areas under the threshold refer to choosing Queue 2 for the new job.

We dynamically assign the job into one of the two queues based



(a) Minimum value of the PQWS metric



(b) Corresponding dispatching results

Figure 4: The proposed tradeoff offloading policy (TOP) under different total job arrival rate λ_c

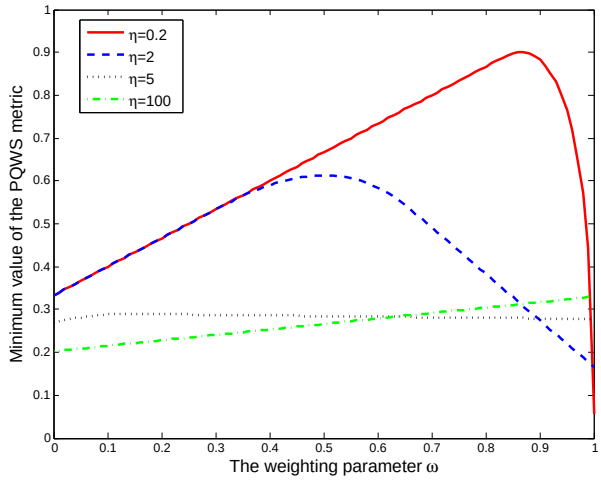
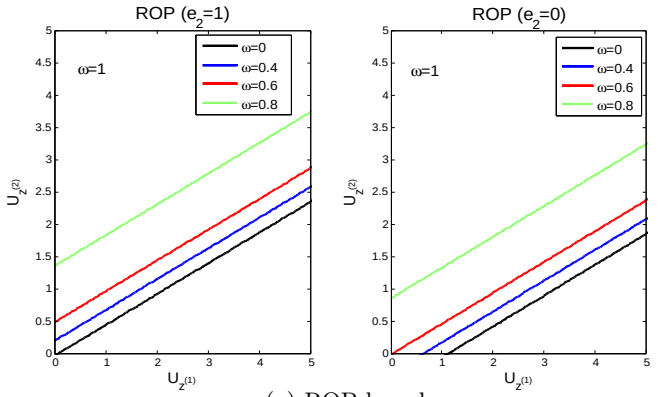
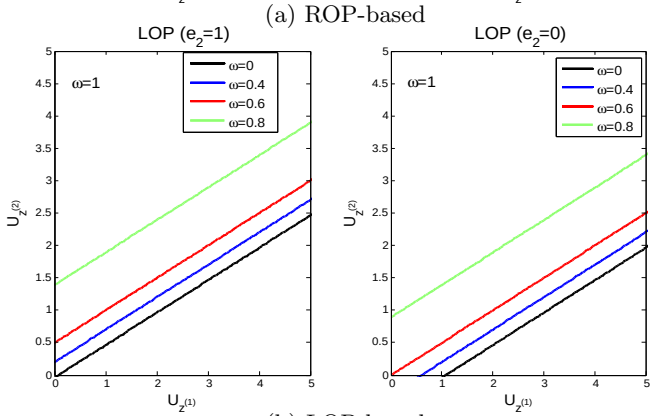


Figure 5: The proposed tradeoff offloading policy (TOP) under different η (recovery rate of Server 2)

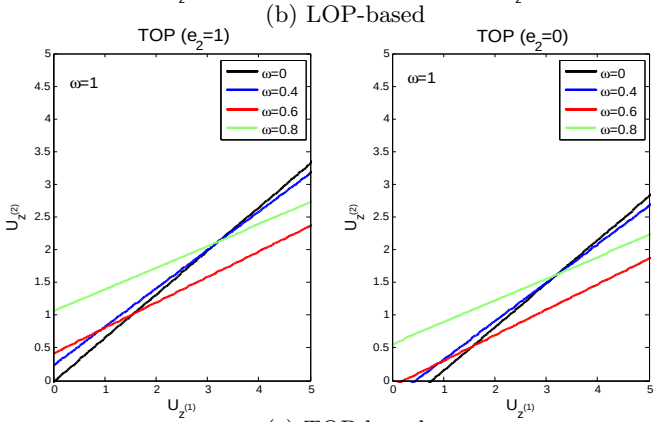
on the three static policies. In Fig.6, the left side is the situation when Server 2 is available ($e_2 = 1$), and on the right side it is unavailable ($e_2 = 0$). Obviously, the areas that correspond to assignment of new jobs to Queue 2 when Server 2 is available are much larger compared to the cases when it is unavailable. For the ROP and LOP policies, the thresholds are always parallel under different values of $\omega \in [0, 1]$. Whereas assigning the job to a low power server (Queue 1) would be beneficial from the energy consumption perspective at low loads, such a policy may end up in difficulties at higher loads as the response time



(a) ROP-based



(b) LOP-based

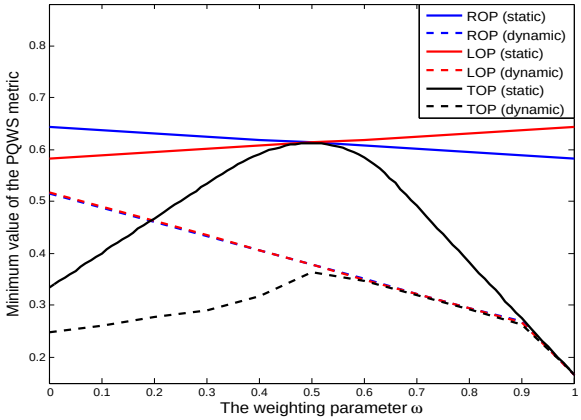


(c) TOP-based

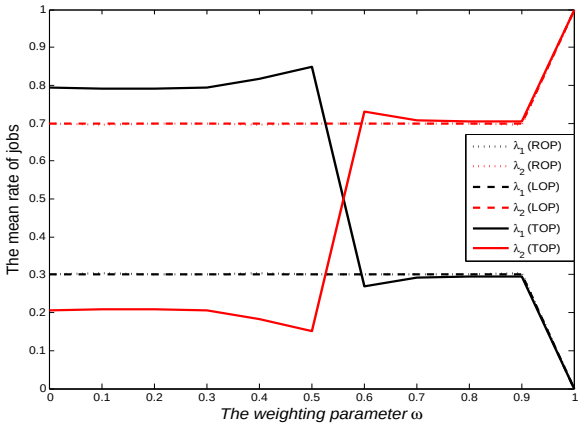
Figure 6: Dynamical dispatching decisions to choose Queue i under different static offloading policies

grows quickly. However, for the TOP scheme, the assignment thresholds cross under different ω. That is because the TOP scheme tries to dynamically balance the allocated jobs in order to minimize the objective value of the PQWS metric.

Fig.7(a) compares the dynamic offloading schemes and the different static policies. The dynamic offloading scheme under the TOP policy achieves the best performance, while the ROP- and LOP-based schemes have almost the same PQWS value when jobs are dynamically assigned to the two queues. Further, the TOP-based dynamic offloading scheme yields a gain over the schemes where only the static policies are adopted, e.g., they are up to 70% better than the static policy of ROP. This is because the dynamic offloading policy considers effectively the dynamic increase in each queue that newly arriving jobs bring in. As shown in Fig.7(b), for the TOP-based dynamic offloading scheme, there exists a turning point in the middle where allocating the jobs to Queue 2 begins to outweigh Queue 1. There is a leap when $\omega = 1$, since $U_{\vec{z}^{(2)}} < U_{\vec{z}^{(1)}}$ at that point, the new arrival jobs are always assigned to Queue 2.



(a) Minimum value of the PQWS metric



(b) Corresponding dynamic dispatching results

Figure 7: Comparison of different static and dynamic offloading policies

5. CONCLUSIONS

This paper addresses the optimality analysis of the energy-performance tradeoff for mobile cloud offloading systems based on a queueing model, which captures both energy and performance metrics and also intermittently available access links. The optimal assignment policy can find an appropriate tradeoff between the minimising energy costs and minimising delay. In the numerical experiments, the proposed dynamic policy combined with the tradeoff offloading policy (TOP) shows very good results and outperforms other policies by a significant margin.

This simple queueing model can be used to describe the complex real offloading system. Although our derivation of the results are based on the assumption that arriving jobs to the system following a Poisson process and the job sizes are exponentially distributed, the approach can be adjusted to other arrival processes with arbitrary distributions of the job sizes.

In the future we may use general fork-join models to analyse mobile offloading systems and we will derive policies based on the Energy-Response time Product (ERP) [7] to see whether this metric will be symmetric across the weighting factor, i.e., whether it will give equal importance to both contributing metrics, energy consumption and delay.

6. REFERENCES

[1] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[2] 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.

[3] E. Hyytiä, T. Spyropoulos, and J. Ott, "Optimizing offloading strategies in mobile cloud computing," 2013.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] T. Pesu and W. J. Knottenbelt, "Dynamic subtask dispersion reduction in heterogeneous parallel queueing systems," in *Proc. of UKPEW 2014*, (Newcastle.upon.Tyne,UK), September 2014.

[9] 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.

[10] 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.

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

[12] V. Cardellini, V. De Nito Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," tech. rep., Technical Report, available online at http://www. optimizationonline. org/DB HTML/2013/08/3981. html, 2013.

[13] A. Penttinen, E. Hyytiä, and S. Aalto, "Energy-aware dispatching in parallel queues with on-off energy consumption," in *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International*, pp. 1–8, IEEE, 2011.

[14] L. L. Andrew, M. Lin, and A. Wierman, "Optimality, fairness, and robustness in speed scaling designs," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, pp. 37–48, ACM, 2010.

[15] X. Lu *et al.*, "Energy-aware performance analysis of queueing systems," 2013.

[16] E. Hyytiä, S. Aalto, A. Penttinen, and J. Virtamo, "On the value function of the m/g/1 fcfs and lcfs queues," *Journal of Applied Probability Evaluation*, vol. 49, no. 4, pp. 1052–1071, 2012.

[17] E. Hyytiä, J. Virtamo, S. Aalto, and A. Penttinen, "M/m/1-ps queue and size-aware task assignment," *Performance Evaluation*, vol. 68, no. 11, pp. 1136–1148, 2011.