Joint Resource Scheduling and Computation Offloading for Energy Harvesting Communications

Ibrahim Fawaz, Mireille Sarkiss, Philippe Ciblar

Joint Resource Scheduling and Computation Offloading for Energy Harvesting Communications

Ibrahim Fawaz[1,2], Mireille Sarkiss[1], and Philippe Ciblat[2]
[1] CEA, LIST, Communicating Systems Laboratory PC 173, 91191 Gif-sur-Yvette, France
[2] LTCI, Télécom ParisTech, Université Paris-Saclay, F-75013, Paris, France
Email: ibrahim.fawaz@cea.fr, mireille.sarkiss@cea.fr, philippe.ciblat@telecom-paristech.fr

***Abstract*—This paper studies the joint optimization of resource scheduling and computation offloading for mobile networks where energy harvesting (EH)-enabled devices are wirelessly connected to nearby base stations (BSs), which can be endowed with some computational capabilities. We consider that a mobile device may run its applications either locally or remotely at its serving base station. We also consider that its applications are strict delay constraints. Our objective is to minimize the packets' loss due to buffer overflow or delay violation of the queued packets at the mobile device. We formulate this problem as a Markov Decision Process (MDP) and exhibit an optimal deterministic offline scheduling-offloading policy. This policy makes decision on the processing location (either local or offloading) and on the number of processed packets by relying on the knowledge on the current channel, the past data and energy arrivals as well as the harvested energy available in the battery. We show through numerical results that the proposed policy can significantly improve the successfully received packets' rate and the energy consumption compared to other policies, such as immediate scheduling or only local processing or only offloading policies.**

I. Introduction

Nowadays, the unprecedented growth of mobile communications driven by the huge number of connected devices and new mobile applications is significantly increasing the demands for high-volume delay-sensitive data traffic, requiring thus intensive computation and leading to high energy consumption. However, this expansion of wireless services is still restrained by mobile terminals limitations in terms of processing capacity, storage and energy. Recently, Mobile-Edge Computing (MEC) [1], [2] and Energy Harvesting (EH) [3], [4] have been proposed as promising technologies to improve mobile devices computing capabilities and extend their battery lives.

On one hand, MEC enables offloading computation tasks from mobile devices to remote cloud servers or to nearby base stations with more energy and computation resources. This allows to reduce locally the consumed energy at mobile devices. Optimizing the transmission strategies in MEC has attracted considerable attention during the past decade. For instance, in [5], the energy consumption of the mobile device was minimized by jointly optimizing the radio resource scheduling and the computation offloading under average delay constraints using Dynamic Programming (DP) techniques. In [6], a delay-optimal task scheduling policy for single-user systems was developed using Markov Decision Process (MDP) formulation.

On the other hand, EH from surrounding environments allows to extend wireless devices lifetime by exploiting alternative renewable energy sources, such as solar power. Resource scheduling with EH-enabled devices has been widely studied during last years. The most related to our work are [7] and [8]. In [7], the weighted packet loss rate is minimized under an average delay constraint in wireless sensor networks. Based on a Constrained MDP framework, the authors proposed a distributed energy allocation algorithm with multi-level water-filling structure based on local system states, using approximate MDP and online stochastic learning. In [8] however, the expected total transmitted data is maximized. The authors proposed online learning and offline policies depending on the information available at the transmitter.

To take advantage of both technologies, recent studies have integrated EH capabilities into MEC systems. This new field opens great opportunities to improve the performance of mobile devices but also brings new challenges in designing optimal and efficient policies, taking into account both radio and computation resources under energy harvesting constraints. A very recent work studying EH-MEC systems proposed a dynamic computation offloading policy for mobile devices using Lyapunov optimization techniques [9]. In [10], the resource management problem is also considered for EH-MEC but from the server side.

In this paper, we address joint resource scheduling and computation offloading for a single EH mobile user served by a base station (BS). We force the system here to respect a strict delay constraint as in our previous work [11]. Then, we minimize the number of discarded packets due to delay violation and buffer overflow, assuming random data and energy future arrivals, as well as time-varying channel where the current realization is known. The problem is formulated as an MDP and solved using Policy Iteration (PI) algorithm. We compare the proposed policy with one performing immediate scheduling, and two additional ones executing either only local processing or only offloading decisions.

The remainder of the paper is organized as follows. In Section II, we describe the system model. In Section III, we

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675891.

formulate the MDP problem by defining its states, actions and transition probabilities and we propose a PI algorithm to solve it. We provide and analyze numerical results in Section IV. Finally, we give some concluding remarks and future perspectives in Section V.

II. System Model

We consider a MEC scenario where an EH mobile user is wirelessly connected to a BS endowed with some cloud resources. The mobile user stores the harvested energy from an external source in a limited-capacity battery and the data packets arriving from the upper layer in a finite buffer. The communication is slotted into consecutive epochs of equal duration T_s. Type of processing (locally or remotely) as well as the number of packets to be processed are determined at the beginning of each time slot, depending on current channel state and previous data and energy arrivals. In the sequel, we describe the data, energy and channel models, as well as the different types of scheduling decisions and their corresponding consumed energy.

A. Data queue model and strict delay constraint

The mobile user receives data packets and stores them in a buffer of size B_d packets. We model the data arrival process as an independent identically distributed (i.i.d.) Poisson distributed process with an average arrival rate λ_d. We assume that all packets are of the same size L bits. At the beginning of time slot n, let a_n denote the number of received packets with the following probability distribution

$$p(a_n = a) = e^{-\lambda_d}.\frac{(\lambda_d)^a}{a!}.$$

A packet is discarded from the buffer if there is a

- **buffer overflow**, i.e., new packets are discarded if there is no space in the buffer to store them;
- **delay violation**, i.e., packets are discarded if they are stored in the queue more than the maximal delay K_0.

To describe the buffer configuration, the age of the packets, i.e., the time spent by the packets, within the buffer is necessary. Therefore, we introduce the notation $k_i(n)$ corresponding to the age of the i-th packet at time n in the buffer. By definition, we have $k_i(n) \in \{-1, ..., K_0\}, \forall i, n$ and $k_i(n) = -1$ stands for an empty space in the buffer (i.e., when the i-th packet does not exist). In Fig. 1, we provide a buffer state at time n, where q_n is the queue length in the buffer. Notice that $k_j(n) \leq k_i(n)$, $\forall i \leqslant j$.

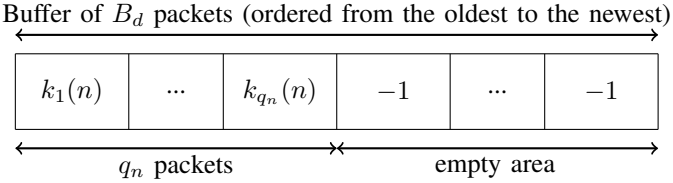

Fig. 1: Buffer configuration at time slot *n*.

B. Energy model

We assume that the energy arrives in multiple packets of energy units (e.u) of $\mathcal{E}_U$ Joules (J). The received energy is stored in a battery of limited capacity B_e, and is lost when it exceeds B_e. Then, the energy arrival process, counting as a number of the energy units, is modeled as an i.i.d. Poisson distributed process with an average arrival rate λ_e. Thus, let e_n denote the harvested energy at the beginning of time slot n. Its probability distribution is given by

$$p(e_n = e) = e^{-\lambda_e}.\frac{(\lambda_e)^e}{e!}.$$

Let b_n denote the energy level of the battery at the beginning of time slot n after receiving the harvested energy e_n. We have $b_n \in \{0, ..., B_e\}$. Let E_n be the energy consumed to execute packets during time slot n. By construction, $E_n \leqslant b_n \ \forall n$. In addition, we suppose causal Energy State Information at the Transmitter (ESIT), i.e., b_n is known when scheduling at time slot n.

C. Channel model

We consider the communication channel between the mobile device and its BS as a flat-fading channel with bandwidth W_{DL} (Hz) in the downlink and W_{UL} (Hz) in the uplink. The additive white Gaussian noise has power spectral density N_0. During time slot n, the channel response remains constant with complex-valued amplitude h_n, and varies independently along time slots. We define the channel gain as $x_n = |h_n|^2$. We assume $\{x_n\}_n$ is an i.i.d. uniform distributed process into a finite set $\mathcal{X}$. We also assume current Channel State Information at the Transmitter (CSIT), i.e., x_n is known when scheduling at time slot n.

D. Scheduling decisions and related consumed energy

At the beginning of time slot n, three scheduling decisions are possible:

- **Local processing:** The mobile device uses its own processor to execute u packets from the buffer ($u \leqslant q_n$) during time slot n. We assume that processing one packet locally consumes a power P_ℓ. Then the consumed energy, expressed as an integer multiple of the energy unit, is given by

$$E_\ell(u) = \left\lceil u.P_\ell.\frac{T_s}{\mathcal{E}_U} \right\rceil. \tag{1}$$

- **Offloading:** The mobile device transmits u packets to be executed at its BS and then receives the results. The mobile device thus consumes energy to send data, to wait for the remote processing and to receive the result. The energy consumption, expressed as an integer multiple of the energy unit, is given by

$$E_o(x_n, u) = \left\lceil \frac{u}{\mathcal{E}_U} \left(\frac{L.P_t}{W_{UL}.\log_2\left(1 + \frac{P_t.x_n}{W_{UL}.N_0}\right)} + T_w.P_w + \frac{L_{DL}.P_r}{W_{DL}.\log_2\left(1 + \frac{P_s.x_n}{W_{DL}.N_0}\right)} \right) \right\rceil \tag{2}$$

where L_{DL} is the size in bits of the computation result, P_t is the transmission power of the mobile device, P_r is the power consumed by the mobile device to receive the result, and P_s is the power used by the BS to send the result. While waiting for the packets processing, the transmitter consumes a power P_w. Finally, T_w is the time for the BS to execute one packet. In addition, we require that this offloading procedure lasts at most one time slot leading to the following constraint

$$u\Bigg(\frac{L}{W_{UL}.\log_2\left(1+\frac{P_t.x_n}{W_{UL}.N_0}\right)}+T_w+\frac{L_{DL}}{W_{DL}.\log_2\left(1+\frac{P_s.x_n}{W_{DL}.N_0}\right)}\Bigg)\leqslant T_s. \quad (3)$$

Notice that W_{DL}, W_{UL}, N_0, L_{DL}, P_s, T_w are pre-fixed parameters. Forcing equality in Eq. (3) enables us to find P_t which so depends on u and the channel realization x_n and is time-varying along with the time slots.

- **Idle:** The mobile device does not execute any packet and decides to wait for the next time slot. We assume that the device electronic circuitry is sleeping and so the consumed energy is given by

$$E_I = 0. \quad (4)$$

III. Problem Formulation and Resolution

Now, we aim at finding an optimal policy μ that minimizes the number of discarded packets due to buffer overflow and delay violation. The policy μ is a sequence of actions that specify the processing decisions (local processing, offloading or staying idle) and the number of packets *u* to be scheduled at each time slot, based on the past system states and actions. In this section, we characterize the appropriate states and actions and show that our problem can be formulated as an MDP. We define then the transition matrix and the reward of this MDP and propose an offline policy iteration algorithm to solve it.

A. State Space

The state space $\mathcal{S}$ is the set of $\mathbf{s} = (\mathbf{k}, b, x)$ where

- $\mathbf{k} = [k_1, \cdots, k_{B_d}]$ is the vector indicating the age of each packet in the data buffer,
- b is the battery level, and
- x is the channel gain.

Due to the strict delay constraint, we describe the data buffer states using $\mathbf{k}$ instead of the queue length q used in the state-of-the-art [5], [7], [9]. Actually, q can be extracted from $\mathbf{k}$

$$q_n = \max\left\{i \mid k_i(n) \geqslant 0\right\}. \quad (5)$$

The state space is finite, and the total number of possible states $|\mathcal{S}|$ is upper-bounded by $(K_0+2)^{B_d}.|B_e+1|.|\mathcal{X}|$. By assuming that packets are queued in an decreasing order of their age in the buffer, i.e. $k_1(n) \geq k_2(n) \geq \cdots \geq k_{q_n}(n)$, we can significantly reduce the state space by removing all impossible combination of components in $\mathbf{k}$. For instance, with $B_d = 8$, $K_0 = 3$, $B_e = 4$ and $|\mathcal{X}| = 5$, our system has only 12375 states compared to the upper-bound of 9765625.

B. Action Space

The action space $\mathcal{V}$ denotes the processing decisions (local processing, offloading or staying idle) and the number of packets u that the mobile device can schedule during a slot. Let U_ℓ be the maximum number of packets that can be processed locally during a time slot and U_o be the maximum number of packets that can be offloaded during a time slot. U_ℓ is limited by the capacity of the internal processor of the mobile device and U_o is obtained from Eq. (3) with equality, using the maximum transmit power $P_{\max}$ and the best channel gain $x_{\max} = \max_{x\in\mathcal{X}} x$. Finally, the action space is finite with cardinality $V = |\mathcal{V}| = U_\ell + U_o + 1$. The actions are ordered and the m-th action is as follows:

- if $m = 0$, idle processing is considered and $u = 0$.
- if $m = m_\ell$ with $m_\ell \in \{1, \cdots, U_\ell\}$, local processing is applied and $u = m_\ell$.
- if $m = m_o$ with $m_o \in \{U_\ell + 1, \cdots, V-1\}$, offloading is applied and $u = m_o - U_\ell$.

At time slot n, $\nu_n \in \{0, , \cdots, V-1\}$ corresponds to the decided action, and u_{ν_n} is the number of processed packets, either locally or remotely.

C. Markov Decision Process

During time slot n, $w_n = \max(u_{\nu_n}, m_n)$ packets leave the buffer due to local or remote processing and/or discarded, where m_n is the number of packets reaching the maximal delay (K_0) in the buffer. The age of the remaining packets in the buffer is incremented by 1. Moreover, a_{n+1} new packets arrive to the buffer with age 0. Therefore, the vector $\mathbf{k}$ can be updated from slot n to slot $n+1$ according to the following rule.

1: **for** $i = 1$ **to** $q_n - w_n$ **do**
$k_i(n+1) = k_{w_n+i}(n) + 1$
end for
2: **for** $i = q_n - w_n + 1$ **to** $q_n - w_n + a_{n+1}$ **do**
$k_i(n+1) = 0$
end for
3: **for** $i = q_n - w_n + a_{n+1} + 1$ **to** B_d **do**
$k_i(n+1) = -1$
end for

At the same time, e_{n+1} e.u are harvested and stored in the battery while E_n e.u are used to schedule u_{ν_n} packets according to Eqs. (1), (2), or (4). Therefore, at time slot $n+1$, the battery state is updated according to

$$b_{n+1} = \min\left\{b_n - E_n + e_{n+1}, B_e\right\}. \quad (6)$$

We thus remark that $(\mathbf{k}_{n+1}, b_{n+1})$ only depends on previous state $(\mathbf{k}_n, b_n)$, action ν_n (which provides E_n since x_n is, known), and external perturbation (a_{n+1}, e_{n+1}), confirming that the problem boils down to MDP.

D. Transition Matrix

The state transition probability of the MDP is defined by $p(\mathbf{s}'|\mathbf{s}, \nu)$ denoting the transition probability to fall in the future state $\mathbf{s}' = (\mathbf{k}', b', x')$ after taking action ν in the current state $\mathbf{s} = (\mathbf{k}, b, x)$. Assuming that the buffer, battery and channel

states are independent to each other and channel states are not time-correlated, the transition probability satisfies the following equation

$$p(\mathbf{s}'|\mathbf{s},\nu) = p(\mathbf{k}'|\mathbf{k},b,\nu).p(b'|b,x,\nu).p(x'), \quad (7)$$

where $p(x')$ is the distribution of the channel states, $p(\mathbf{k}'|\mathbf{k},b,\nu)$ indicates the probability transitions between buffer states, and $p(b'|b,x,\nu)$ indicates the probability transitions between battery states.

We first exhibit the set of possible actions $\mathcal{A}(\mathbf{s}) \subset \mathcal{V}$ for each state $\mathbf{s}$. We have $\mathcal{A}(\mathbf{s}) = \mathcal{A}_0(\mathbf{s}) \cap \mathcal{A}_1(\mathbf{s}) \cap \mathcal{A}_2(\mathbf{s})$, where $\mathcal{A}_0(\mathbf{s})$, $\mathcal{A}_1(\mathbf{s})$ and $\mathcal{A}_2(\mathbf{s})$ are defined through their complementary sets as follows: *i*) the set $\overline{\mathcal{A}_0}(\mathbf{s})$ is composed by the actions of offloading using a transmit power $P_t > P_{\max}$ according to Eq. (3); *ii*) the action ν belongs to $\overline{\mathcal{A}_1}(\mathbf{s})$ by satisfying at least one of the following conditions

1: $u_\nu > q$ **or** $k'_i > k_i + 1$ **or** $q' < q - w$
2: $k'_i \neq k_{i+u_\nu} + 1$ **and** $k_{i+u_\nu} \neq -1$
3: $k'_i > 0$ **and** $k_{i+u_\nu} = -1$
4: $q = B_d$ **and** $u_\nu \neq 0$ **and** $k'_i > 0, \forall i \in \{q - w + 1, ..., B_d\}$

iii) the action ν belongs to $\overline{\mathcal{A}_2}(\mathbf{s})$ by satisfying at least one of the following conditions

1: $0 > b - E$
2: $b' < b - E$

where E is the energy consumed when the action ν is applied.

Secondly, when $\nu \in \mathcal{A}(\mathbf{s})$, the transitions are as follows

1: **if** $q' < B_d$ **then**
$p(k'_i|k_i,b,\nu) = e^{-\lambda_d}.\frac{(\lambda_d)^{q'-q+w}}{(q'-q+w)!}$
2: **else**
$p(k'_i|k_i,b,\nu) = 1 - \mathrm{Q}(B_d - q + w, \lambda_d)$,

and

1: **if** $b' < B_e$ **then**
$p(b'|b,x,\nu) = e^{-\lambda_e}.\frac{(\lambda_e)^{b'-b+E}}{(b'-b+E)!}$
2: **else**
$p(b'|b,x,\nu) = 1 - \mathrm{Q}(B_e - b + E, \lambda_e)$.

where Q is the regularized gamma function.

E. Reward

In this paper, we focus on infinite horizon MDP problem. We consider thus time-averaged cost, where at a given time slot $n \in \{0, \cdots, N\}$, the system state is denoted by $\mathbf{s}_n = (\mathbf{k}_n, b_n, x_n)$ and $\mu(\mathbf{s}_n) = \nu_n$ is the action. Our objective is to minimize the average number of discarded packets under policy μ. Hence, we define the cost function as

$$\overline{D}(\mu) = \lim_{N\to+\infty} \frac{1}{N} \mathbb{E}^{\mu}\left[\sum_{n=1}^{N} \Big(\varepsilon_d(\mathbf{s}_n,\nu_n) + \varepsilon_o(\mathbf{s}_n,\nu_n)\Big)\right], \quad (8)$$

where $\mathbb{E}^{\mu}$ is the expectation with respect to the policy μ and where $\varepsilon_d(\mathbf{s}_n,\nu_n)$ and $\varepsilon_o(\mathbf{s}_n,\nu_n)$ are the instantaneous number of discarded packets due to delay violation and buffer overflow, respectively.

At a given slot n, when the system state is $\mathbf{s}_n$ and the performed action is ν_n, the number of discarded packets due to delay violation is given by

$$\varepsilon_d(\mathbf{s}_n,\nu_n) = \begin{cases} 0 & \text{if } m_n = 0 \text{ or } m_n \leqslant u_{\nu_n} \\ m_n - u_{\nu_n} & \text{otherwise.} \end{cases} \quad (9)$$

The buffer overflow occurs when $q_n - w_n + a_{n+1} > B_d$, thus the number of discarded packets due to buffer overflow is obtained as follows

$$\begin{aligned} \varepsilon_o(\mathbf{s}_n,\nu_n) = & \sum_{a=B_d-q_n+w_n+1}^{+\infty} (q_n - w_n + a - B_d).e^{-\lambda_d}.\frac{(\lambda_d)^a}{a!} \\ = & \ \lambda_d.(1 - \mathrm{Q}(B_d - q_n + w_n, \lambda_d)) \\ + & \ (q_n - w_n - B_d) \\ \times & \ (1 - \mathrm{Q}(B_d - q_n + w_n + 1, \lambda_d)). \end{aligned} \quad (10)$$

We can then state the MDP optimization problem as

$$\mu^\star = \arg\min_{\mu} \overline{D}(\mu) \quad (11)$$

F. Offline Dynamic Programming Approach

We propose an offline dynamic programming approach to solve this problem. We present in Algorithm 1 the optimal policy computation using PI algorithm [12]. This optimal offline deterministic policy consists in a one-to-one mapping from the state space $\mathcal{S}$ to the action space $\mathcal{V}$, performing a unique action ν whenever a state $\mathbf{s}$ is visited. We remind that it only depends on energy arrival and data arrival *a priori* distributions and current channel states at the mobile device.

IV. Numerical Results

We evaluate numerically the optimal policy devised for the scheduling-offloading problem. We consider the system described in Section II and fix its characteristics as: the slot duration is $T_s = 1$ ms, the channel state x takes 5 possible values from the finite set $\mathcal{X} = \{-5.41, -1.59, 0.08, 1.42, 3.18\}$ dB with equal probabilities. The noise power spectral density is $N_0 = -87$ dBm/Hz and the allocated bandwidth is $W_{DL} = 5$ MHz in the downlink and $W_{UL} = 500$ kHz in the uplink. Data packets have equal size $L = 5000$ bits and are stored in a buffer of size $B_d = 8$ packets. The resulting packets from remote processing have also equal size $L_{DL} = 500$ bits. The maximum delay of packets in the buffer is $K_0 = 3$ (i.e., in absolute time $K_0 T_s = 3$ ms). Energy arrivals are stored as energy units in a battery of size $B_e = 4$ e.u, where $\mathcal{E}_U = 40$ nJ. The maximum available power at the transmitter is $P_{\max} = 2$ mW. During a time slot, a maximum of $U_\ell = 2$ packets can be processed locally or a maximum of $U_o = 4$ packets can be offloaded. The remaining values are chosen as: $P_\ell = 30$ μW, $P_r = 0.2$ mW, $P_s = 1.6$ kW, $P_w = 0.1$ mW, and $T_w = 0.1$ ms.

In Fig. 2, we illustrate the average number of discarded packets for the optimal policy versus the number of iterations with various energy arrival rates λ_e and a data arrival rate $\lambda_d = 1.5$. We can notice that it takes only a few hundreds iterations to the PI algorithm to converge. In addition, if a

Algorithm 1 Optimal policy computation

1: Select arbitrary policy μ^0
Let $\beta^0 = 0$
Fix a tolerance parameter $\epsilon > 0$
Set $j = 1$

2: Given the policy μ^{j-1}
Compute the transition matrix $\mathbf{T}$ of size $|\mathcal{S}| \times |\mathcal{S}|$ when policy μ^{j-1} is applied according to Section III-D
Compute the cost vector $\mathbf{c}$ of length $|\mathcal{S}|$

$$\varepsilon_d(\mathbf{s}, \mu^{j-1}(\mathbf{s})) + \varepsilon_o(\mathbf{s}, \mu^{j-1}(\mathbf{s}))$$

for each state $\mathbf{s}$ in $\mathcal{S}$.

3: Let a scalar β^j and a vector $\mathbf{v}^j$ of length $|\mathcal{S}|$ be the solutions of

$$\beta^j + (I - \mathbf{T})\mathbf{v}^j = \mathbf{c}$$

$$\sum_{\mathbf{s} \in \mathcal{S}} v^j(\mathbf{s}) = 0$$

where $v^j(\mathbf{s})$ is the component of $\mathbf{v}^j$ corresponding to the $\mathbf{s}$ in $\mathcal{S}$.

4: Find a policy μ^j by computing an optimal action for each state $\mathbf{s}$ using

$$\mu^j(\mathbf{s}) = \arg\min_{\nu \in \mathcal{V}} \varepsilon_d(\mathbf{s}, \nu) + \varepsilon_o(\mathbf{s}, \nu) + \sum_{\mathbf{s}' \in \mathcal{S}} p(\mathbf{s}'|\mathbf{s}, \nu) v^j(\mathbf{s}')$$

5: If $\mu^j(\mathbf{s}) = \mu^{j-1}(\mathbf{s})$ for any state $\mathbf{s}$ or $\left|\beta^j - \beta^{j-1}\right| < \epsilon$, then $\hat{\mu} = \mu^j$ is the optimal policy estimate and stop; else $j = j + 1$ and go to step 2.

large amount of energy can be harvested, λ_e increases, and the system is able to schedule more packets reducing hence considerably the average number of discarded packets.

In Fig. 3, we show the percentage of discarded packets versus the data arrival rate λ_d for $\lambda_e = 0.5$ (small) and $\lambda_e = 1.5$ (large) energy arrival rates. We compare the performance of the optimal policy to three different policies, namely immediate scheduling, local processing and offloading policies. The *immediate* policy processes, locally or remotely, the maximum number of packets whenever energy is available in the battery. The *local* and *offload* policies however, are obtained from our offline PI algorithm but take only local decisions and only offload decisions, respectively. We can observe that the proposed deterministic offline policy outperforms the other policies as it can adapt its processing to the energy and data arrivals as well as channel conditions. Yet, when the data arrival rate λ_d increases, the number of discarded packets of all the policies increases because the buffer overflow can happen more often. We can also see that for both λ_e values, the *local* policy discards the highest number of packets due to the limited capacity of the mobile device processor. Moreover, when λ_e is small, the *offload* policy is able to sustain more efficiently the system than the *immediate* policy by scheduling more packets depending on the channel states. This situation is reversed when λ_e is large since the scavenged energy is sufficiently available to process the maximum number of packets by the immediate policy irrespective of channel conditions.

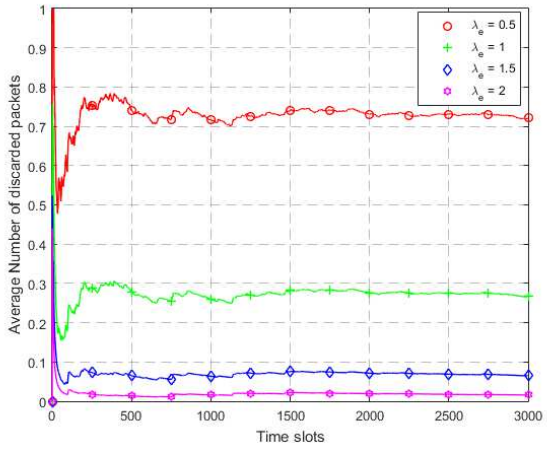

Fig. 2: Convergence analysis for the average rate of discarded packets for different energy arrival rates.

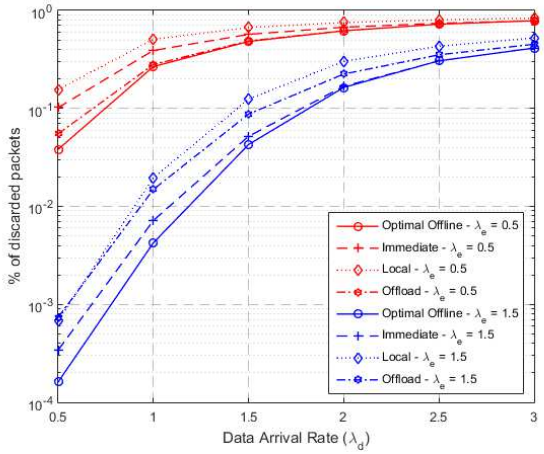

Fig. 3: Percentage of the discarded packets versus data arrival rate for energy arrival rates $\lambda_e = 0.5$ and 1.5.

In Fig. 4 and Fig. 5, we plot respectively the average consumed energy and the average battery state versus the data arrival rate λ_d for energy arrival rates $\lambda_e = 0.5$ and 1.5. We can observe that *local* and *immediate* policies experience the highest energy consumption since processing packets locally consume more energy, draining thus the battery level. The optimal proposed policy consumes approximately the same energy amount as the *offload* policy while sending more packets. Indeed, it ensures a better sustainable communication with less number of discarded packets by optimally using the available energy, leading hence to a higher energy level in the battery.

In Fig. 6, we display the percentage of processing decisions for the optimal and *immediate* policies in Fig. 6(a) and (c) respectively at $(\lambda_d = 1, \lambda_e = 1)$, and in Fig. 6 (b) and (d) respectively at $(\lambda_d = 2, \lambda_e = 2)$. As we can see, when the

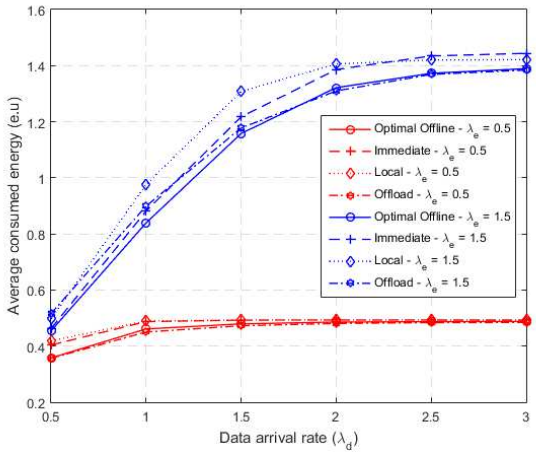

Fig. 4: Average consumed energy versus data arrival rate for energy arrival rates $\lambda_e = 0.5$ and 1.5.

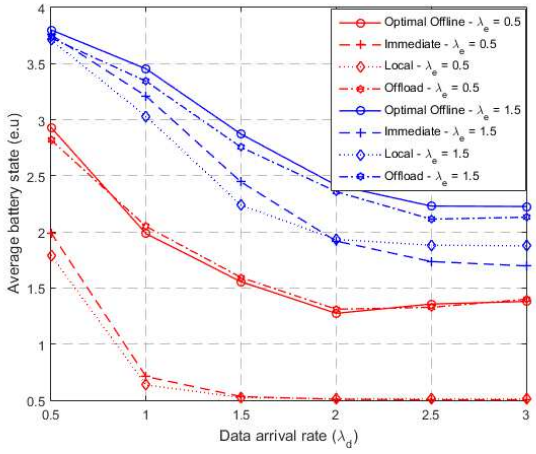

Fig. 5: Average battery state versus data arrival rate for energy arrival rates $\lambda_e = 0.5$ and 1.5.

data arrival rate increases, the system schedules more packets either locally or remotely to minimize the number of discarded packets, which decreases *Idle* mode events. When more energy is available in the battery, the *immediate* policy processes the maximum number of packets, hence can offload more packets regardless of channel states. Therefore, energy shortage can occur more often forcing the system to enter *Idle* mode more than with optimal policy.

V. Conclusion

We have addressed computation offloading problem from an energy harvesting mobile device to its serving resourceful base station under strict delay constraint. We have proposed an optimal policy to minimize the packet loss rate using MDP framework and dynamic programming techniques. By leveraging on the knowledge of the available energy in the battery, the data and energy arrivals as well as the channel states, the optimal offline policy decides to process locally

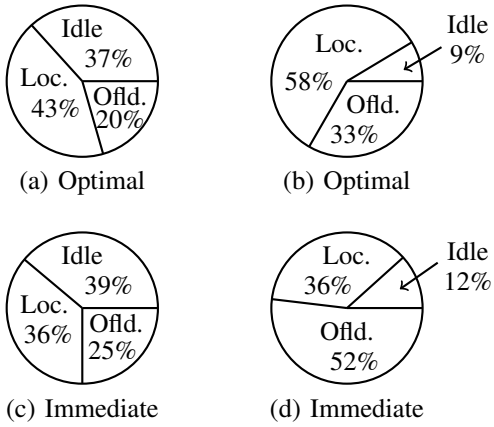

Fig. 6: Percentage of processing decisions at $(\lambda_d = 1, \lambda_e = 1)$ (a) and (c), and $(\lambda_d = 2, \lambda_e = 2)$ (b) and (d).

or remotely while specifying the number of packets to be processed. For future work, we aim at (i) investigating the case of unknown channel to the mobile device, (ii) focusing on online approaches instead of offline.

References

[1] K. Kumar, J. Liu, Y-H. Lu and B. Bhargava, "A Survey of Computation Offloading for Mobile Systems," Mobile Networks and Application, vol. 18, no. 1, pp. 129-140, February 2013.

[2] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628-1656, Third Quarter 2017.

[3] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover and K. Huang, "Energy Harvesting Wireless Communications: A Review of Recent Advances," IEEE Journal on Selected Areas in Communications, vol. 33, no. 3, pp. 360-381, March 2015.

[4] M. L. Ku, W. Li, Y. Chen and K. J. R. Liu, "Advances in Energy Harvesting Communications: Past, Present, and Future Challenges," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1384-1412, February 2016.

[5] W. Labidi, M. Sarkiss and M. Kamoun, "Energy-Optimal Resource Scheduling and Computation Offloading in Small Cell Networks," IEEE International Conference on Telecommunications (ICT), April 2015.

[6] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," IEEE International Symposium on Information Theory (ISIT), July 2016.

[7] L. Lei, Y. Kuang, X. Shen, K. Yang, J. Qiao and Z. Zhong, "Optimal Reliability in Energy Harvesting Industrial Wireless Sensor Networks," IEEE Transactions on Wireless Communications, vol. 15, no. 8, pp. 5399-5413, August 2016.

[8] P. Blasco, D. Gunduz and M. Dohler, "A Learning Theoretic Approach to Energy Harvesting Communication System Optimization," IEEE Transactions on Wireless Communications, vol. 12, no. 4, pp. 1872-1882, April 2013.

[9] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590-3605, December 2016.

[10] J. Xu, L. Chen and S. Ren, "Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing," IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 3, pp. 361-373, September 2017.

[11] I. Fawaz, P. Ciblat and M. Sarkiss, "Energy Minimization based Resource Scheduling for Strict Delay Constrained Wireless Communications," IEEE Global Conference on Signal and Information Processing (GLOBALSIP), December 2017.

[12] D. P. Bertsekas, "Dynamic Programming and Optimal Control,", vol.2, Fourth Edition.