Performance Modeling of Delayed Offloading in Mobile Wireless Environments With Failures

Huaming Wu, *Member, IEEE*

***Abstract*— Mobile devices and wireless network links are sometimes unreliable and unstable owing to user mobility in mobile wireless environments. As a result, failures may occur during the offloading process, which has a huge influence on the offloading performance. Worse still, it is challenging to make high-level offloading decisions since they closely rely on an accurate predicting model, especially when encountering with offloading failures. Unfortunately, due to the high mobility and heterogeneous network environments, such a model is currently not available to researchers and practitioners. To fill that gap, this letter develops an analytical queueing model for delayed offloading systems with intermittent connectivity. We set a deadline to detect offloading failures when taking the impatient jobs and service interruptions into account. Furthermore, for different types of mobile applications, a tradeoff analysis between energy saving and latency reducing is analyzed on the basis of an energy–response time weighted product metric. The proposed offloading model can be effectively used for describing complex and realistic offloading systems with the presence of failures.**

***Index Terms*— Mobile device, wireless network, offloading, failure, queueing model.**

I. Introduction

ALONG with the development of Mobile Cloud Computing (MCC) and Mobile Edge Computing (MEC), offloading is becoming a promising approach to relieve computing pressure, by migrating tasks from resource-constrained mobile devices either to a resourceful cloud server or a nearby edge server over wireless networks. Thus, it can effectively enhance the performance of mobile applications and energy saving on the mobile devices [1]. However, wireless environments between the mobile devices and the cloud/edge servers are not stable, and as a result, mobile devices are prone to dynamically changing network conditions on account of user mobility [2].

Disconnection of mobile devices from wireless networks and unstable cloud/edge servers can cause failures in execution of offloaded tasks, which could severely affect the performance of mobile offloading systems [3]. Once a failure occurs, a mobile device has to wait for the failure to recover, for example, the link is reconnected from disconnection or the server is available from outage. Checkpointing was used in [4] to realize fault-tolerance by periodically saving the state of a process during the failure-free execution. During the operation of failure recovery, the offloading system loaded the latest checkpoint data and recovered the application execution. MAUI [5] attempted to handle crash failures by employing a timeout mechanism. If the mobile device is out of contact with the MAUI server while the server is executing an offloaded task, then the mobile device can either execute this task locally or attempt to find an alternate server and then offloads the task to this new server.

Since wireless interfaces usually have different availabilities, delays and energy costs, tasks can be offloaded to the cloud/edge servers either via costly cellular networks or intermittently available WiFi networks [6]. By deferring task offloading until some high-speed wireless network becomes available, the data transfer time can be reduced at the expense of some extra waiting time. The reduced transfer time may directly translate into energy saving on mobile devices [7]. Offloading failures may occur once in a while since the mobile network environment between the mobile side and the server side is not stable. Unfortunately, existing research on stochastic modeling of delayed offloading systems is lacking, especially when considering offloading failures occur in the mobile device.

The aim of this letter is to solve the dilemma of waiting for the repair or starting a local execution after the failure. We cope with offloading failures that appear on the mobile side owing to wireless link failures or on the server side owing to server failures, e.g., the server is shutdown or the server becomes unreachable because of the mobile device's movement. Offloading failures are detected by a setting deadline. As motivated, we try to solve the following issues: (i) In case the server is not available or in link failures, how to balance the offloading benefits, e.g., execution time reduction and energy saving? (ii) Once a failure occurs during the offloading process, the mobile user has to wait for the failure to recover, but to what extent the user is willing to defer a transmission and how long should it wait? (iii) How to set an optimal deadline for latency-sensitive or latency-tolerant applications when paying attention to time saving or energy reducing?

II. Delayed Offloading Model

Figure 1 shows a delayed offloading model with the presence of failures, which are listed as follows:

- Connection failures: Link failures occur frequently due to the high mobility of mobile devices, which will ultimately lead to connection failures [8]. When the mobile devices get suddenly disconnected from the cloud/edge servers, severe performance degradation will incur for mobile applications since the offloading request is dropped. During the process of offloading execution, a disconnection between a mobile device and a cloud/edge server can be treated as a failure. Besides, the servers sometimes are inaccessible due to node failure or unstable link.
- Server failures: Because the reliability is typically ensured through Service Level Agreements (SLAs)

Manuscript received July 18, 2018; accepted August 17, 2018. Date of publication August 22, 2018; date of current version November 12, 2018. This work was supported in part by the Huawei Innovation Research Program (Grant Number: HIRPO2017050307), in part by the National Natural Science Foundation of China (Grant Number: 61801325), and in part by the Natural Science Foundation of Tianjin City (Grant Number: 18JCQNJC00600). The associate editor coordinating the review of this letter and approving it for publication was J. Ben Othman.

The author is with the Center of Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2018.2866585

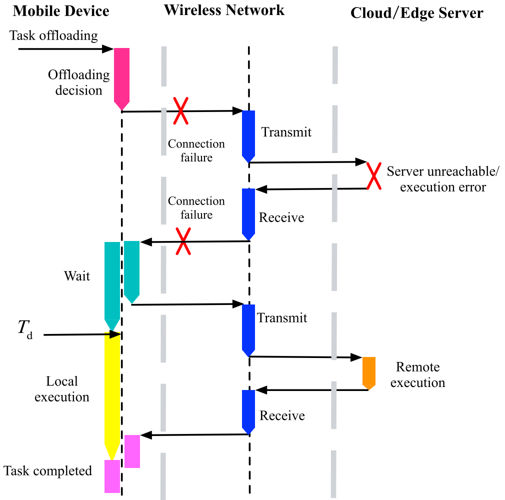

Fig. 1. The delayed offloading model with failures [10].

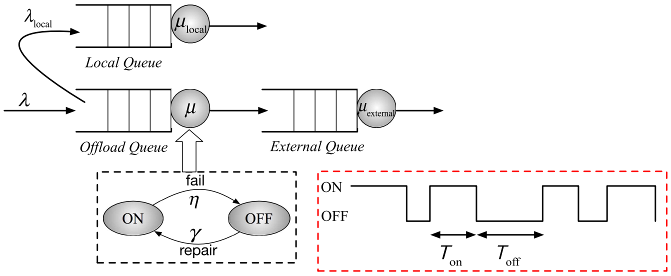

Fig. 2. A queueing model for mobile offloading systems with failures.

negotiated with the service providers, cloud/edge servers are not perfectly reliable, especially when data center outages occur, they will not be accessible. A recent survey [9] reported that 93% of enterprises that suffered from a data center downtime for more than 10 days, filed for bankruptcy within a year of the outage. Execution errors in the servers can also happen and return some wrong results. This can be another cause of the connection failures.

In addition, there exist crash failures especially when disastrous breakdowns (e.g., running out of battery, abnormal shutdown).

In Fig. 1, the offloading process will be interrupted in the event of a connection or server failure, and the task has to be re-executed from the scratch. The mobile device is unaware that how long it should wait for the failure recovery. In order to tackle this issue, we define an exponential distributed deadline T_{d}. Using the memoryless property of exponential random variables, the analysis process can be greatly simplified [7]. On the one hand, if the failure is repaired in a period of time ($\leq T_{\text{d}}$), then the task will be offloaded again; on the other hand, if the failure does not recover before the deadline expires, then the task will be processed locally instead of being offloaded to the server.

As shown in Fig. 2, we can treat such mobile system as an $M/M/1$ modulated *Offload Queue*, where it occasionally suffers a disastrous breakdown during the offloading phase. Once a failure occurs, this queue declines all existing jobs, and a repair process begins immediately, referred as the repair phase. When the connection is down and undergoing a repair process, new arrivals become impatient: each individual job, upon arrival, activates an ‘impatience timer’, exponentially distributed with an abandonment rate ξ $(= 1/T_{\text{d}})$ [10]. If the failure is repaired before the assigned deadline T_{d} has expired, then we retry offloading, i.e., the job will be migrated to *External Queue* with cloud/edge server for external execution. If the mobile environment remains unchanged, the queue does not change from the repair phase to the offloading phase before T_{d} has expired, then the job is scheduled to *Local Queue* for local execution immediately instead of being offloaded to the nearby edge server or remote cloud server.

TABLE I
NOTATIONS

Notations	Meanings
T_{d}	The assigned deadline
ξ	The abandonment rate
η	The failure rate
γ	The recovery rate
ω	The weighting parameter
λ	The job arrival rate
μ	The job service rate
π_{on}	The steady probability when the queue is in the on-state
π_{off}	The steady probability when the queue is in the off-state
λ_{local}	The job arrival rate in the local queue
μ_{external}	The job service rate in the external queue
ρ_{local}	The utilization of the local queue
ρ_{on}	The utilization during the on-state
ρ_{off}	The utilization during the off-state
p_{local}	The power consumption for local processing
p_{idle}	The power consumption for being idle
p_{tr}	The power consumption for transferring data

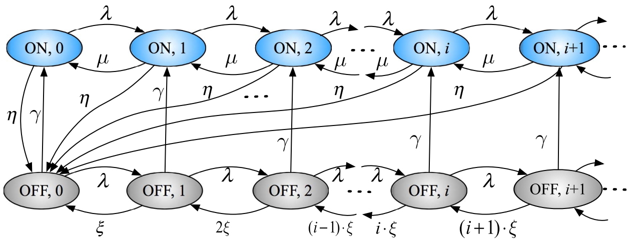

Fig. 3. The 2D Markov chain for the delayed offloading model with failures.

Assuming that jobs arriving at *Offload Queue* is according to a Poisson process with a rate of λ. The mobile system suffers disastrous breakdowns, occurring when the server is at its offloading phase, at a Poisson rate η [10]. When the server fails, all jobs present are rejected and lost. Once an offloading failure happens, a repair process will begin immediately. The repair times are exponentially distributed random variables with a rate of γ. Accompanied with offloading failures, the *Offload Queue* is then modeled with a 2D Markov chain, as illustrated in Fig. 3.

During the on-state, the queue drains at rate μ and during the off-state, the queue drains at rate $i \cdot \xi$ since any of the i queued jobs can abandon the *Offload Queue* and then move to *Local Queue* for local execution on the mobile device [11]. The balance equations can be written as:

$$(\lambda+\gamma)\pi_{\text{off},0} = \xi\pi_{\text{off},1}+\eta\pi_{\text{on}} \tag{1a}$$

$$(\lambda+\gamma+i\xi)\pi_{\text{off},i} = \lambda\pi_{\text{off},i-1}+(i+1)\xi\pi_{\text{off},i+1}\ (i>0) \tag{1b}$$

$$(\lambda+\eta)\pi_{\text{on},0} = \gamma\pi_{\text{off},0}+\mu\pi_{\text{on},1} \tag{1c}$$

$$(\lambda+\mu+\eta)\pi_{\text{on},i} = \lambda\pi_{\text{on},i-1}+\mu\pi_{\text{on},i+1}+\gamma\pi_{\text{off},i}\ (i>0) \tag{1d}$$

The steady probability when the offloading queue is in the off-state can be written as $\pi_{\text{off}} = \frac{\mathbb{E}[T_{\text{off}}]}{\mathbb{E}[T_{\text{on}}]+\mathbb{E}[T_{\text{off}}]} = \frac{\eta}{\gamma+\eta}$. In a similar way, the steady probability when the offloading queue is in the on-state is $\pi_{\text{on}} = \frac{\mathbb{E}[T_{\text{on}}]}{\mathbb{E}[T_{\text{on}}]+\mathbb{E}[T_{\text{off}}]} = \frac{\gamma}{\gamma+\eta}$.

According to [12], after some algebraic manipulations in Eq. (1), we can calculate:

$$\mathbb{E}[N_{\text{off}}] = \frac{\lambda}{\gamma+\xi}\pi_{\text{off}}, \tag{2}$$

$$\mathbb{E}[N_{\text{on}}] = \frac{\lambda}{\eta(\gamma+\xi)}(\xi\pi_{\text{on}}+\gamma) - \frac{\mu}{\eta}(\pi_{\text{on}}-\pi_{\text{on},0}), \tag{3}$$

where $\pi_{\text{off},0} = K\eta\pi_{\text{on}}/\xi$, $\pi_{\text{on},0} = \frac{\gamma z_0}{\mu(1-z_0)}G(z_0)$,

$$G(z) = \pi_{\text{off},0} \cdot \left[1 - \frac{\int_0^z (1-s)^{\frac{\gamma}{\xi}-1}\mathrm{e}^{-\frac{\lambda}{\xi}s}\mathrm{d}s}{K}\right](1-z)^{-\frac{\gamma}{\xi}} \cdot \mathrm{e}^{\frac{\lambda}{\xi}z},$$

$z_0 = \frac{\lambda+\mu+\eta-\sqrt{(\lambda+\mu+\eta)^2-4\lambda\mu}}{2\lambda}$, $K = \int_0^1 (1-s)^{\frac{\gamma}{\xi}-1}\mathrm{e}^{-\frac{\lambda}{\xi}s}\mathrm{d}s$.

As shown in Fig. 3, this queue suffers from two types of losses:

- Rejected jobs due to some disastrous failures occurring in the offloading system. The expected number of lost jobs lost per unit of time is denoted as $\eta\mathbb{E}[N_{\text{on}}]$.
- Abandonments of impatient jobs during the repair phase. The expected number of jobs served per unit of time is calculated as $\mu(\pi_{\text{on}}-\pi_{\text{on},0})$.

An arbitrary job arriving at the *Offload Queue* may depart and join the *Local Queue* because of impatience in the repair process. Thus, the rate of jobs processed locally, λ_{local}, is given by:

$$\begin{aligned}\lambda_{\text{local}} &= \lambda - \eta\mathbb{E}[N_{\text{on}}] - \mu(\pi_{\text{on}}-\pi_{\text{on},0}) \\ &= \lambda - \frac{\lambda}{\gamma+\xi}(\xi\pi_{\text{on}}+\gamma) \\ &= \xi \cdot \mathbb{E}[N_{\text{off}}].\end{aligned} \tag{4}$$

Further, the probability of abandoning the *Offload Queue* is defined as:

$$\Pr\{\text{abandonment}\} = \frac{\lambda_{\text{local}}}{\lambda} = \frac{\xi}{\gamma+\xi}\pi_{\text{off}}, \tag{5}$$

where Pr denotes the probability operation. From Eq. 5, we can find that the abandonment probability is not related with the job arrival rate λ.

III. Metric-Based Analysis

A. Mean Response Time

In case of offloading failures, the total offloading cost is composed of the cost for sending jobs to the cloud/edge server, the waiting cost for repairing the failure, the cost for local processing on the mobile device and the idly waiting cost for the cloud/edge server to complete the jobs.

Applying Little's Law, $\mathbb{E}[N] = \lambda\mathbb{E}[T]$, we can calculate the mean response time:

$$\begin{aligned}\mathbb{E}[T] &= \mathbb{E}\big[\mathbb{E}[T_j]\big] \\ &= \sum_{j\in\{\text{on,off,local,external}\}} \frac{\lambda_j}{\lambda}\mathbb{E}[T_j] \\ &= \frac{1}{\lambda}\sum_{j\in\{\text{on,off,local,external}\}} \mathbb{E}[N_j],\end{aligned} \tag{6}$$

where $j \in \{\text{on}, \text{off}, \text{local}, \text{external}\}$ represents the offloading phase, the repair phase, the local processing phase (*Local Queue*) and the external processing phase (*External Queue*), respectively.

The average number of jobs in the *Local Queue* can be calculated as:

$$\mathbb{E}[N_{\text{local}}] = \frac{\rho_{\text{local}}}{1-\rho_{\text{local}}}, \tag{7}$$

where $\rho_{\text{local}} = \lambda_{\text{local}}/\mu_{\text{local}}$ is the utilization of the *Local Queue*.

Further, the average number of jobs executed on the cloud/edge server in the *External Queue* is calculated as:

$$\mathbb{E}[N_{\text{external}}] = \frac{\lambda-\lambda_{\text{local}}}{\mu_{\text{external}}}. \tag{8}$$

B. Mean Energy Consumption

According to the relationship between energy and power consumption $\mathbb{E}[P] = \lambda\mathbb{E}[\mathcal{E}]$, the mean energy consumption for the delayed offloading model with the presence of failures can be calculated by:

$$\begin{aligned}\mathbb{E}[\mathcal{E}] &= \mathbb{E}\big[\mathbb{E}[\mathcal{E}_j]\big] \\ &= \sum_{j\in\{\text{on,off,local}\}} \frac{\lambda_j}{\lambda}\mathbb{E}[\mathcal{E}_j] \\ &= \frac{1}{\lambda}\sum_{j\in\{\text{on,off,local}\}} \mathbb{E}[P_j],\end{aligned} \tag{9}$$

Since the probability that the server is busy equals to the utilization of the queue, i.e., $\Pr\{N>0\} = \rho$, the corresponding mean power consumption can be calculated by:

$$\mathbb{E}[P] = p \cdot \Pr\{N>0\} = p \cdot \rho. \tag{10}$$

We assume that the mobile system consumes p_{local} for local processing, p_{idle} for being idle, and p_{tr} for transferring data. Then we have:

$$\mathbb{E}[\mathcal{E}] = \frac{p_{\text{local}} \cdot \rho_{\text{local}} + p_{\text{idle}} \cdot \rho_{\text{off}} + p_{\text{tr}} \cdot \rho_{\text{on}}}{\lambda}, \tag{11}$$

where $\rho_{\text{on}} = \pi_{\text{on}} - \pi_{\text{on},0}$ and $\rho_{\text{off}} = \pi_{\text{off}} - \pi_{\text{off},0}$ are the utilizations during the on-state and off-state, respectively.

C. The ERWP Metric

We use a metric named Energy-Response time Weighted Product (ERWP) [13]:

$$\mathit{ERWP} = \mathbb{E}[\mathcal{E}]^{\omega} \cdot \mathbb{E}[T]^{1-\omega}, \tag{12}$$

where $0 \le \omega \le 1$ is a weighting parameter that balances the relative importance of energy consumption and response time according to the type of application. A larger weighting factor ($0.5 < \omega \le 1$) means more importance is put on energy saving, such as for latency-tolerant applications; on the contrary, a smaller weighting factor ($0 \le \omega < 0.5$) means that we should focus more on time saving, such as for latency-sensitive applications.

Further, by substituting Eqs. (6) and (9), into Eq. (12), the optimization problem according to the ERWP metric can be formulated as:

$$\xi^* = \arg\min_{\xi}\ \mathit{ERWP}, \tag{13}$$

where we attempt to find out the abandonment rate ξ^* that minimizes the ERWP value.

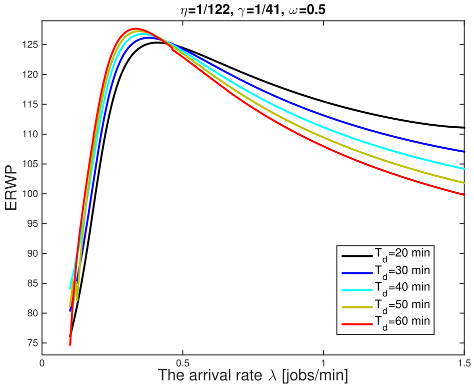

Fig. 4. Comparison of ERWP values for different deadlines as job arrival rate increases.

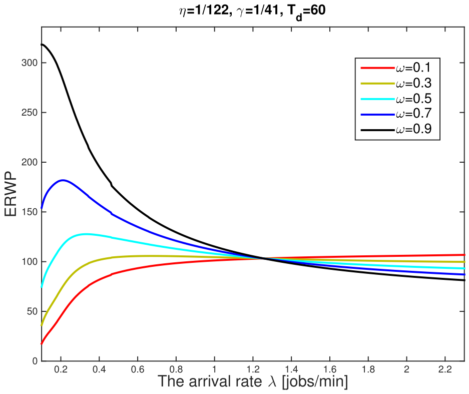

Fig. 5. Comparison of ERWP values for different weighting parameters as job arrival rate increases.

IV. Performance Evaluation

As a classic example, we consider an offloading scenario with connection failures when using WiFi networks. Using measurements from real traces in [7], the average data rate of WiFi is 1.28 Mbps, the average duration of WiFi availability period is 122 min ($\eta = 1/122$ min^{-1}), while the average duration with WiFi unavailability is 41 min ($\gamma = 1/41$ min^{-1}) [14]. The mean job size is assumed to be 50 MB. According to [15], when using a SAMSUNG Galaxy Nexus with dual-core 1.2 Ghz CPU and 1 GB of RAM, the power coefficients are set as: $p_{\text{m}} = 2.4$ W, $p_{\text{tr}} = 1.5$ W and $p_{\text{idle}} = 0.05$ W, respectively. Besides, we set the local service rate as $\mu_{\text{local}} = 0.3$ and the external service rate as $\mu_{\text{external}} = 1$.

As shown in Fig. 4, at lower job arrival rates, the ERWP value arises as the deadline T_{d} increases, but at higher arrival rates, the ERWP value starts to drop as the T_{d} increases. All arrival jobs are offloaded to a nearby edge server or a remote cloud server through WiFi when the WiFi network is available; otherwise, if the deadline expires, they are processed locally on the mobile device. Since jobs accompanied with higher deadlines are prone to be transmitted over the WiFi network, resulting in smaller response time; while with lower deadlines, jobs leave the queue earlier, leading to smaller queueing delays. Thus, the proposed delayed offloading model accompanied with failures can obtain the smallest ERWP value by selecting an optimal T_d.

As shown in Fig. 5, when considering response time as a more important factor (e.g., for latency-sensitive applications), the delayed offloading model is recommended at lower arrival rates; while at higher ω when considering energy consumption more important (e.g., for latency-tolerant applications), it is preferred to adopt the delayed offloading model at higher arrival rates.

V. Conclusion

In this letter, we presented a novel methodology for modeling offloading failures in mobile wireless environments with intermittently available access links. We have conducted research for mobile offloading systems with the presence of failures according to the ERWP metric, which takes both energy and delay characteristics into account.

On the basis of the achieved numerical results, we find that the abandonment probabilities remain constant across different job arrival rates. By optimally choosing a reneging deadline in the minimization of the ERWP metric, different energy-delay tradeoffs can be achieved for different kinds of applications. More importantly, we can use the proposed queueing model to capture the dilemma between offloading retry and local re-execution when failures occur in complex and realistic offloading systems.

References

[1] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 461–474, Feb. 2018.

[2] K. Lee and I. Shin, "User mobility-aware decision making for mobile computation offloading," in *Proc. IEEE 1st Int. Conf. Cyber-Phys. Syst., Netw., Appl. (CPSNA)*, Aug. 2013, pp. 116–119.

[3] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "An online algorithm for task offloading in heterogeneous mobile clouds," *ACM Trans. Internets Technol.*, vol. 18, no. 2, 2018, Art. no. 23.

[4] S. Ou, Y. Wu, K. Yang, and B. Zhou, "Performance analysis of fault-tolerant offloading systems for pervasive services in mobile wireless environments," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2008, pp. 1856–1860.

[5] E. Cuervo *et al.*, "Maui: Making smartphones last longer with code offload," in *Proc. ACM 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 49–62.

[6] E. Hyytiä, T. Spyropoulos, and J. Ott, "Offload (only) the right jobs: Robust offloading using the Markov decision processes," in *Proc. IEEE 16th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2015, pp. 1–9.

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

[8] P. V. Krishna, S. Misra, V. Saritha, D. N. Raju, and M. S. Obaidat, "An efficient learning automata based task offloading in mobile cloud computing environments," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[9] A. Carroll. *Shocking Statistics About Data Center Downtime*. Accessed: Apr. 2013. [Online]. Available: https://lifelinedatacenters.com/data-center/data-center-downtime/

[10] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

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

[12] N. Perel and U. Yechiali, "Queues with slow servers and impatient customers," *Eur. J. Oper. Res.*, vol. 201, no. 1, pp. 247–258, 2010.

[13] H. Wu, Y. Sun, and K. Wolter, "Analysis of the energy-response time tradeoff for delayed mobile cloud offloading," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, pp. 33–35, Sep. 2015.

[14] F. Mehmeti and T. Spyropoulos, "Performance modeling, analysis, and optimization of delayed mobile data offloading for mobile users," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 550–564, Feb. 2017.

[15] X. Liu, S. Guo, and Y. Yang, "Task offloading with execution cost minimization in heterogeneous mobile cloud computing," in *Proc. Int. Conf. Mobile Ad-Hoc Sensor Netw.* Singapore: Springer, 2017, pp. 509–522.