International Journal of Current Sciences and Engineering Intelligence
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: *https://ijcsi.com*
Volume 01, Issue 02, October 2026
OPEN ACCESS
E-ISSN :

Deadline Aware Edge Task Offloading Under Network and Queue Uncertainty

Mayank Singh[1], Mohit Malhotra[2], Riya Tripathi[3], Shubham Pandey[4]
[1,2,3] Students, Government Engineering College, Modasa
[4] Assistant Professor, Government Engineering College, Modasa

Article Info

Article History:
Published: 14 Nov 2026

Publication Issue:
Volume 01, Issue 02
October 2026

Page Number:
1-5

Corresponding Author:
Mayank Singh

Abstract:
Edge computing can reduce the energy consumed by mobile devices, but offloading decisions become unreliable when average latency is treated as a guarantee that a task will finish on time. This paper develops an analytical framework that chooses between two local processor settings and a remote edge server. The framework combines a device energy model, transmission time, an exponential queueing model, and a requirement that each selected operating mode meet a deadline with at least 95 percent probability. A constructed workload is evaluated under three wireless conditions and several server arrival rates. At an arrival rate of 80 tasks per second, the probability constrained policy consumes 0.11425 joules per task and achieves a calculated deadline success probability of 99.50 percent. A policy based only on mean latency consumes less energy but achieves 93.44 percent success. These are model calculations rather than measurements from deployed equipment. The analysis explains why queue uncertainty, local fallback capacity, and the boundary of the energy accounting must be considered together when designing an offloading controller.
Keywords: Edge computing; computation offloading; deadline probability; queueing theory; energy optimization; mobile computing; resource allocation.

Introduction

Mobile applications increasingly execute computation that competes with limited battery capacity and processor resources. Image analysis, local classification, and interactive data processing can benefit from nearby computational infrastructure. However, sending a task to an edge server introduces communication and waiting time. A fast processor at the destination does not remove the delay caused by a congested transmission link or a busy service queue. The practical decision therefore concerns the complete route from task arrival to usable output.

A common simplification compares average local execution time with average remote execution time. This comparison is useful for capacity planning, but it does not directly answer a deadline question. A system with a mean response time below 300 milliseconds may still finish a substantial fraction of tasks after 300 milliseconds. For interactive applications, that distinction changes the value of an apparently attractive energy saving.

The research question addressed here is whether a small, interpretable decision rule can expose this tradeoff without requiring a learned controller or a large simulation platform. The proposed approach evaluates a finite set of execution modes and rejects modes whose calculated probability of meeting the deadline is too low. It then selects the lowest expected device energy among the remaining modes.

The contribution is an auditable comparison of decision criteria under explicit assumptions. It is not a claim that probability constrained offloading is a new research field. The model isolates one mechanism, the difference between mean queue delay and its upper tail, so that the reason for a routing decision can be checked directly.

Detailed Literature Review

Shi and colleagues [1] describe edge computing as computation placed near data sources and discuss the resulting system challenges. Satyanarayanan [2] explains the architectural importance of proximity for applications that depend

Copyright © 2026 The Author(s) : This is an open access article under the CC BY license (http://creativecommons.org/licenses/by-nc/4.0/)

on responsive interaction. Together, these works motivate examining communication distance and resource placement, while leaving open the application specific question of how an individual task should be routed.

Mao and colleagues [3] review the joint management of radio and computational resources. Mach and Becvar [4] similarly examine edge architectures and computation offloading. These surveys establish that communication conditions and server resources should be considered jointly. They also show why a model that treats remote execution as a fixed, universally faster operation would omit central aspects of the problem.

MAUI [5] demonstrates fine grained energy aware code offloading, while CloneCloud [6] investigates elastic execution between mobile devices and cloud infrastructure. Their system contributions involve program execution and partitioning mechanisms that the present analysis does not implement. Here, a complete task is assumed to be available for either local execution or remote execution, with migration overhead summarized by a fixed term.

Mao, Zhang, and Letaief [7] study dynamic offloading with energy harvesting devices. Mao and colleagues [8] examine a power and delay tradeoff in a multiuser system. Liu and colleagues [9] formulate delay optimal scheduling using system state. These studies support the importance of adaptive decisions, but their optimization settings differ from the finite action comparison developed here.

The queueing foundation follows standard analysis of the M/M/1 system [10]. Under stationary Poisson arrivals, exponential service times, first come first served processing, and a stable service rate, the system sojourn time has an exponential distribution. This distribution permits an explicit deadline probability. The present framework uses that tractable result as a conditional model, rather than assuming that every practical edge server satisfies those statistical conditions.

Detailed Methodology & Framework

The study uses a constructed task requiring 200 million local processor cycles and an upload of one million bits. The local processor can operate at either 0.5 or 1.0 gigahertz. Local execution time equals the cycle requirement divided by processor frequency. Dynamic device energy is modeled as a coefficient of 10 to the power minus 27 multiplied by the cycle count and the square of frequency. These assumptions give local execution times of 400 and 200 milliseconds, with energies of 0.05 and 0.20 joules respectively.

$$\mathrm{Tlocal} = \frac{\mathrm{C}}{\mathrm{f}}\,;\ \mathrm{Elocal} = \kappa \mathrm{C} \mathrm{f}^{2}$$

$$\mathrm{Tedge} = \frac{\mathrm{b}}{\mathrm{R}} + \mathrm{t}_{0} + \mathrm{Q}\,;\ \mathrm{Q} \sim \mathrm{Exp}(\mu - \lambda)$$

$$\mathrm{P}(\mathrm{Tedge} \leq \mathrm{D}) = 1 - \exp[-(\mu - \lambda)(\mathrm{D} - \mathrm{b}/\mathrm{R} - \mathrm{t}_{0})]$$

$$\mathrm{Eedge} = 0.5(\mathrm{b}/\mathrm{R}) + 0.05[\mathrm{t}_{0} + 1/(\mu - \lambda)]$$

C is the cycle count, f the local processor frequency, κ the energy coefficient, b the upload size, R the uplink rate, t_0 the fixed overhead, μ the service rate, λ the arrival rate, and D the deadline. The probability expression applies when $D > b/R + t_0$; otherwise success probability is zero.

Remote execution uses three uplink rates: 20, 5, and 2 megabits per second. Their workload weights are 0.5, 0.3, and 0.2. Upload time is payload size divided by uplink rate. A fixed overhead of 20 milliseconds represents remaining communication and handling time. The edge service rate is 100 tasks per second, and the main comparison uses an aggregate arrival rate of 80 tasks per second.

The edge system time is modeled as exponentially distributed with rate equal to service rate minus arrival rate. Variability in effective server processing is represented by exponential service even though the local cycle requirement is fixed. The evaluated device is treated as a negligible probe of the aggregate queue, so changing its routing policy does not change the specified arrival rate. This assumption would require revision for coordinated routing by many devices.

Remote device energy includes upload power of 0.5 watts and idle waiting power of 0.05 watts. It excludes server electricity, network infrastructure energy, and battery conversion losses. Consequently, the results describe device energy rather than total environmental impact. Expected remote energy uses expected waiting time, whereas deadline feasibility uses the entire assumed delay distribution.

The deadline is 300 milliseconds. The probability constrained controller requires at least 95 percent deadline success for every selected link class. A mean based controller instead requires only expected completion time below the deadline. Both choose the lowest expected energy among feasible actions. Always local and always remote policies provide comparison baselines.

All reported values follow direct substitution into the equations and weighted averaging across the three link classes. No random sample, training dataset, or hardware experiment is represented as having been collected. Sensitivity analysis changes the aggregate arrival rate while holding the other parameters fixed. The framework diagram separates model estimation, feasibility screening, energy comparison, execution, and feedback, making the dependency of each decision visible.

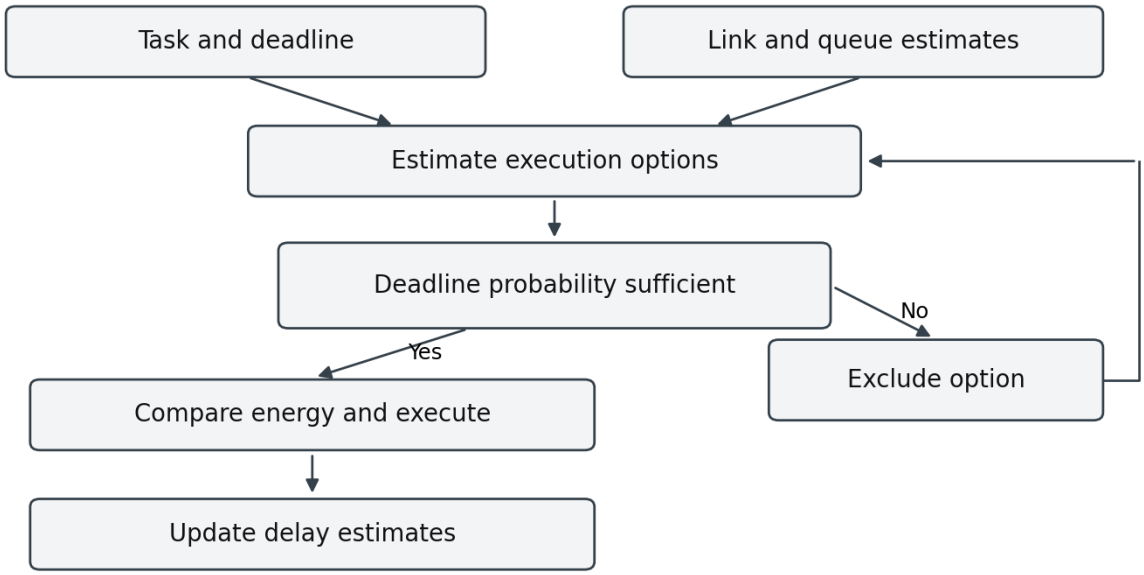


Figure 1. Deadline probability screening before energy minimization

Detailed Results and Analysis

At the main arrival rate, mean remote completion times are 120, 270, and 570 milliseconds for the good, moderate, and poor links. The corresponding deadline success probabilities are approximately 98.99, 79.81, and zero percent. The moderate link illustrates the central problem: its mean time is below the deadline, but approximately one fifth of its tasks are late under the assumed distribution.

The mean based policy therefore offloads on the good and moderate links and executes locally at the faster setting on the poor link. Its weighted mean completion time is 181 milliseconds and its energy is 0.08530 joules per task. Nevertheless, its weighted deadline success is only 93.44 percent. The probability constrained policy offloads only on the good link, giving 160 milliseconds, 0.11425 joules, and 99.50 percent success.

Table 1. Calculated policy comparison at 80 arrivals per second

Policy	Good / moderate / poor link	Mean time ms	Energy J/task	Deadline success
Slow local	Slow / Slow / Slow	400	0.05000	0.00%
Fast local	Fast / Fast / Fast	200	0.20000	100.00%
Always offload	Edge / Edge / Edge	255	0.09600	73.44%

Policy	Good / moderate / poor link	Mean time ms	Energy J/task	Deadline success
Mean based	Edge / Edge / Fast	181	0.08530	93.44%
Probability constrained	Edge / Fast / Fast	160	0.11425	99.50%

Note: Values are calculated from the stated analytical scenario.

Relative to always using the faster local processor, the probability constrained policy reduces modeled device energy by 42.88 percent. Its energy exceeds that of the mean based controller because it purchases greater reliability through additional local processing. The slower local baseline has the smallest energy of the listed fixed policies, but misses every deadline and is therefore unsuitable for the stated requirement.

At an arrival rate of 40 tasks per second, both adaptive policies select the same actions and consume approximately 0.08397 joules. Their deadline success is 99.75 percent. At 90 tasks per second, the probability constrained controller switches entirely to fast local execution. This sensitivity shows that offloading should not be assumed beneficial under every server condition.

The calculated probabilities depend on correctly specified queue parameters. If the controller uses an arrival rate of 80 while the actual rate is 90, its previously selected good link offloading decision no longer meets the per class requirement. The resulting weighted success is approximately 94.99 percent. Thus, a mathematically correct rule can still produce an unsuitable decision when its inputs are stale.

Future Enhancement

A practical extension should estimate delay distributions from timestamped task records and compare empirical deadline frequencies with model predictions. If arrivals are bursty or service times are not exponential, an empirical quantile model or a more suitable queueing distribution should replace the M/M/1 assumption. The evaluation should reserve later observations for checking prediction quality.

Further work can introduce task dependent payloads, processor contention, retransmissions, and the cost of changing execution location. Multiuser routing should account for the feedback between offloading decisions and server congestion. A controller could also optimize total system energy rather than device energy alone. These extensions would make the framework more realistic, but they should preserve an explicit distinction between average performance and deadline reliability.

Conclusion

The analytical comparison demonstrates that selecting offloading actions using mean latency can admit routes whose deadline success is inadequate. Under the stated workload, a probability constrained controller sacrifices some energy savings to raise calculated deadline success from 93.44 to 99.50 percent, while retaining a substantial energy advantage over fast local execution. The result follows from screening the queueing tail before comparing energy. Its applicability depends on accurate delay modeling, available local fallback capacity, and the specified accounting boundary. The framework provides a transparent starting point for measured validation rather than evidence of performance in a deployed network.

References

[1] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge Computing: Vision and Challenges. IEEE Internet of Things Journal, 3(5), 637–646. Source

[2] Satyanarayanan, M. (2017). The Emergence of Edge Computing. Computer. Source

[3] Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A Survey on Mobile Edge Computing: The Communication Perspective. arXiv:1701.01090. Source

[4] Mach, P., and Becvar, Z. (2017). Mobile Edge Computing: A Survey on Architecture and Computation Offloading. arXiv:1702.05309. Source

[5] Cuervo, E., Balasubramanian, A., Cho, D. K., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. (2010). MAUI: Making Smartphones Last Longer with Code Offload. MobiSys. Source

[6] Chun, B. G., Ihm, S., Maniatis, P., Naik, M., and Patti, A. (2011). CloneCloud: Elastic Execution between Mobile Device and Cloud. EuroSys. Source

[7] Mao, Y., Zhang, J., and Letaief, K. B. (2016). Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices. arXiv:1605.05488. Source

[8] Mao, Y., Zhang, J., Song, S. H., and Letaief, K. B. (2016). Power-Delay Tradeoff in Multi-User Mobile-Edge Computing Systems. arXiv:1609.06027. Source

[9] Liu, J., Mao, Y., Zhang, J., and Letaief, K. B. (2016). Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems. arXiv:1604.07525. Source

[10] Purdue University. The M/M/1 Queue. ECE 647 lecture material. Source