iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80816

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Optimization of Mobile Edge Computing and Networking for Ultra-Low Latency and Green Internet of Things Applications

Prasidhee Koli, Darshan Bhangale, Dhrumi Karia, Dhruv Chawda, Mayur Hingu, Prithvi Bhadra, Neha Panadi, Dr. Asha Durafe
Electronics and Computer Science Shah & Anchor Kutchhi Engineering College Mumbai, India

Abstract: ***IoT growth has quietly outpaced what cloud-centric network architectures were built to handle. When billions of sensors, actuators, and embedded controllers started producing continuous data streams, the assumption that round-trip latency toadistantserverwasanacceptablecoststoppedholding.In industrial automation lines, hospital monitoring loops, and vehicle-to-vehicle coordination, tens of milliseconds separate a correctly functioning system from a failed one.***

This paper examines Mobile Edge Computing (MEC) as a structural fix: computation moves physically closer to wheredataoriginates,cuttingpropagationdelayandrelievingbackhaul pressureatthesametime.Welookathowtaskoffloading,energy budgeting, and shared resource allocation actually behave under field conditions — varying wireless channels, heterogeneous workloads, energy supplies that aren't always predictable. The optimization framework we propose uses a Deep Q-Network (DQN) that learns from what the system does rather than from a pre-built model of what it's supposed to do, which matters when operating conditions drift.

One finding we want to highlight upfront: minimizing latency and minimizing energy are not genuinely competing objectives.A well-trained edge policy can satisfy timing requirements while simultaneously cutting power draw across the device, network, and server layers.

Keywords: MobileEdgeComputing(MEC),InternetofThings (IoT),Ultra-LowLatency,GreenComputing,ResourceOptimiza- tion, Task Offloading, Deep Q-Network, Edge Networking

I. INTRODUCTION

Think about what a large hospital network actually looks like in practice. Infusion pumps, ventilators, wearable monitors,environmentalsensors,andaccesscontrolsystemsall share the same infrastructure, yet they have vastly different timing requirements. Routine telemetry can wait seconds or minutes. A closed-loop medication delivery system or an emergencyalert cannot — the difference between a 15ms and a150ms response time carries direct clinical consequences in those cases.

The mismatch between these requirements and conven- tional cloud architecture is fundamentally a physics problem, not a software one. Optical fiber carries signals at roughly200km/ms.Thatplacesahardflooronround-triplatency to a geographically distant data center. When platforms like AWSwereoriginallydesigned,latencywasanannoyancethat affecteduserexperience;itwasnotasafety-criticalparameter. Sub-10ms application deadlines simply were not part of the design brief for centralized cloud infrastructure.

Multi-access Edge Computing (MEC) is the architectural response to that mismatch. Instead of sending computation to distant data centers, MEC co-locates processing with radio access infrastructure. Base stations already installed across urbanareasbecomehostingsitesforedgeservers,cuttingboth propagation delay and the volume of data that needs to travel over backhaul links. Reducing that data movement also brings a secondary benefit: measurable reductions in network-level energy consumption.

The rest of this paper covers: MEC hardware and software architecture;thetaskoffloadingproblemanditsinherentcom- plexity; energy efficiency across all system tiers; multi-tenant resourceallocation;latencydecomposition;emergingresearch directions;openchallenges;andourDQN-basedoptimization approach.

©IJRASET: All Rights are Reserved

II. MOBILE EDGE COMPUTING (MEC) ARCHITECTURE

A. Foundational Concepts

Every measurement an IoT device takes has to go some- where:processedlocally,offloadedtoanedgeserver,orsentto thecloud.Localprocessingworkswhenthedevicehasenough compute capacity and battery headroom. Cloud transmission works when deadlines are soft enough to tolerate the round- trip. MEC sits between these two — a server physically near the data source, often sharing a cabinet with the serving base station, capable of handing back results before hard deadlines expire. The device effectively borrows server-grade compute without absorbing the latency penalty of a data center round- trip.

This idea has roots in content delivery networks, whichhave been moving computation geographically closer to users for decades. MEC's distinguishing feature is its coupling with mobile radio infrastructure and its explicit targeting of latency andenergyconstraintsthatCDNswereneverdesignedaround.

B. HardwareandSoftwareArchitecture

MEC systems are organized into three layers. The *edge server*isthephysicalfoundation:computehardwareatatower or small-cell site, equipped with CPUs, optionalGPU acceler- ators for inference, local storage, and network interfaces. The *MEChost*referstothecompletephysicalnodeincludingradio access and backhaul connectivity. The *MEC platform* is the softwarelayerthathandlestaskreception,scheduling,resource allocation, and result delivery — essentially a miniaturized cloud controller operating at tower scale.

Userequipment—phones,sensors,cameras,vehicleunits— sits at the far edge generating the workloads that the MEC platform arbitrates.

Interoperability across vendor components remains a gen- uineprobleminpractice.AdevicecertifiedforoneMECplat- form does not automatically work with another, and standards bodies are still working through this gap.

C. TaskExecutionFlow

When a task arrives, the MEC platform checks local re- source availability. If capacity permits, the edge server pro- cesses it and returns a result — typically under 5ms. Heavier jobs that can't be handled locally, such as large-scale model inference, multi-site analytics queries, or requests that need archival data, get pushed up to cloud infrastructure.

There's an energy angle here that matters for battery- operateddevices.Movingacomputationallyexpensivetaskoff the device lets its processor coast at a lower utilization level, which cuts power draw significantly. Across a deployment lifetime measured in months or years between battery swaps, these per-task savings accumulate into a real extension of maintenance intervals.

D. ComparisonwithCloudComputing

Cloudinfrastructuremakessenseforaspecificclass ofworkloads:traininglargemodels,runningbatchana- lytics across sensor populations distributed across geogra-phy, archival processing where timing flexibility exists. The hard limit is propagation delay — a request traveling to a metropolitan-areadatacenterandbackcannotfitinsidea 10ms window under normal network conditions.

Edge servers are smaller and more constrained, but locally handled tasks come back in roughly 1–5ms. That's the range hard-deadline IoT applications actually need. The engineering challenge is building a classifier that correctly routes each arrivingtasktotherighttier,quickly,andwithlowoverhead.

III. GREEN COMPUTING: ENERGY EFFICIENCY ACROSS SYSTEM TIERS

InmuchoftheMECliterature,energyefficiencyappearsas a secondary constraint — something to optimize once latency targets have been met. At IoT scale, this framing gets the priority order wrong, and the numbers make clear why.

Take a network of one million sensors, each sending 1KB of raw data to cloud servers every 10 seconds. That generates 100MB/sof sustained backhaul traffic. Replace raw transmis- sionwithlocalpreprocessingand50-bytesummaries,andthat traffic drops by a factor of roughly 20. Every node along the path — routers, switches, optical amplifiers — draws powerin proportion to the traffic it carries. Across the size of IoT infrastructure today, the gap between these two approaches represents a meaningful share of total data center energy budgets.

A. Device-Level Power Consumption

On most battery-powered IoT devices, the radio transceiver is the dominant power consumer. Modern low-power micro- controllers handle simple computation efficiently; transmis- sion, especially over extended wireless paths, is expensive by comparison.

Adaptive offloading policy — choosing between raw transmission and local preprocessing based on current channel quality — can meaningfully extend battery life across deployment timescales when preprocessing costs less than what transmission would consume.

B. Network-LevelPowerConsumption

Trafficprocessedattheedgenevertouchesbackhaulinfras- tructure.Forhigh-volumeIoTapplicationsthisisasubstantial saving.Corenetworkpowerconsumptionscaleswithload, so reducing upstream traffic reduces energy expenditure allthe way up the chain to cloud data centers. Operational cost andenvironmentalcostargumentspointinthesamedirection: process at the edge when doing so is feasible.

C. Server-LevelPower Management

Edge servers need their own power management, particu- larly at sites running on renewable or battery backup power. Dynamicvoltageandfrequencyscalingletsserverstradeclock rate for power draw during quiet periods. Workload consoli- dation deactivates cores or entire machines when aggregate demand is low. Where solar generation is available at tower sites, the scheduler has to incorporate energy forecasts into allocation decisions, because power availability is no longer guaranteed.

D. EnergyHarvestingDevices

SomeIoTdeploymentsusedevicesthatextractenergyfrom the environment — photovoltaic cells, piezoelectric trans- ducers, thermoelectric generators, or RF scavenging. These devices can run for long periods without direct battery re- placement, but the energy they collect is not constant or predictable. An offloading policy that doesn't account for this uncertainty — one that behaves the same way on a sunny afternoon as during an overcast morning — will drain devices fasterthannecessaryordefertoomuchwork.Couplingenergy forecasting with offloading policy decisions in a principled wayis anopenproblemthat hasattractedattention butlacks a fully satisfying solution.

IV. LATENCY DECOMPOSITION

Reaching sub-10ms end-to-end requires knowing which components are actually consuming the latency budget, be- cause the effective interventions differ by component. Apply- ingthewrongfixtothewrongsourcewastesbothengineering effort and compute resources.

A. AirInterfaceDelay

The first delay sits between the device and the base sta-tion. In 4G LTE networks, the 1ms subframe structure com- bined with the multi-subframe scheduling pipeline typically contributes 15–20ms of radio round-trip — a number that surprises many application developers who assumed it was negligible.5GNRreducesthissubstantially;undercleanchan- nel conditions it can fall below 1ms. But cell-edge devices, physically obstructed links, and congested spectrum can push the effective air interface delay back up toward LTE levels regardless of the radio generation. This component is set by physics and antenna geometry, not by application software — there's no way to engineer around it in code.

B. ServerQueueDelay

Once a task reaches the edge server, it joins a queue. Under lightloadthiswaitisnearzero.Underburstarrivals—say, a factory floor anomaly triggers simultaneous alerts from a dozensensors—tasksarrivefasterthantheservercanprocess them and queue depth climbs. At that point, queuing delay dominates the latency budget. Importantly, allocating more CPUdoesn'tfixthis:whenarrivalrateexceedsservicerate,the queue grows regardless of per-task execution speed. The right interventionisadmissioncontrol—sheddingordeferringlow- priority tasks — not raw clock speed.

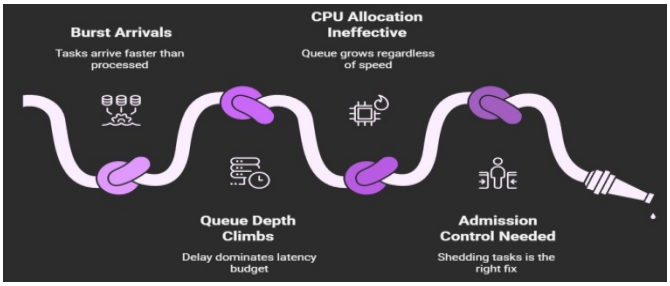


Fig.1.LatencyBottleneck

The time to actually execute a task varies by orders of magnitude.Asimplethresholdcomparisonorrunningaverage takesmicroseconds.Aconvolutionalnetworkprocessinga high-resolutionimagemightneed30–50msevenonhardware with dedicated inference accelerators. Higher CPU frequency shortens execution time but raises power draw super-linearly, somaximizingfrequencyforeverytaskisn'taviablestrategy; the scheduler must weigh execution speed against the power budget on a per-task basis.

C. DownlinkTransmissionDelay

Returning the result to the requesting device is usually the smallest contributor. Computation tends to compress informa- tion: the output of an image classifier is a label, not a feature tensor;theoutputofacontrolloopisasetpoint,notaraw sensor frame. Downlink delay only becomes significant when outputsarelarge—reconstructedvideostreams,modelweight updatesforfederatedlearning,high-fidelityARrendering.For mostIoTtasksit'snegligiblerelativetotheothercomponents. Onesystem-levelfailuremodedeservesexplicitmention: optimizingonecomponentinisolationcanworsenothers. Maximizing CPU frequency across all tasks to shrink compu- tationdelaywillexhausttheserver'sthermalenvelope,trigger- ing throttling that causes queuing delay to grow beyond whatamoremeasuredallocationwouldhaveproduced.Effective optimizationhastotreatthesystemasanintegratedwhole, notasasetofindependentsub-problems.

V. EMERGING RESEARCH DIRECTIONS

A. SplitInferenceArchitectures

Running a complete neural network on an IoT device is constrained by memory, compute, and the latency of local inference on battery-powered hardware. Sending raw sensor data to the cloud for inference incurs bandwidth costs and round-trip latency that hard-deadline applications cannot ab- sorb. Split inference partitions the network across the device- edge boundary: initial layers run on the device, producing intermediate feature representations that are smaller than the raw input, which are then transmitted to the edge for the remainder of the computation. The result comes back as a compact output.

The optimal partition point depends on current channel conditions, server load, and the specific network architecture. Determining this in real time without itself adding significant latency remains an open problem.

B. FederatedLearningattheEdge

Growing regulatory pressure around health records, loca- tion data, and behavioral profiles has made centralized data collection for model training harder to justify legally and operationally.Federatedlearningoffersapracticalalternative: devices keep their data local, run training locally, and con- tributeonlyparametergradients—smallnumericalvectors to an edge aggregation server. The server merges these contributions and redistributes an updated shared model.

Atedgescale,however,severaldifficultiesemergethatdon't appear in textbook federated learning setups. Devices dropout mid-round due to battery depletion or connectivity loss. Devicestrainingatdifferentspeedsproducestalegradientsthat candestabilizeaggregation.Anddevicesobservingfundamen- tally different data distributions — a scenario common when sensorsaredeployedacrossvariedphysicalenvironments make it harder for a single shared model to generalizewell across the population. Working through these practical complications, particularly in high-mobility 6G scenarios, isan active area.

C. UAV-AssistedEdgeCoverage

Fixed base station infrastructure leaves gaps in disaster response, temporary high-density events, remote agriculture, and maritime scenarios. Mounting edge compute hardware on UAVsallowsmobileedgecapacitytobedeployedwherefixed infrastructure is absent or damaged.

This makes the optimization problem harder. UAV position affects channel quality to served devices, coupling trajectory planning with offloading and resource allocation decisions. Flight competes with compute provisioning for onboard en- ergy. The joint three-dimensional optimization of trajectory, offloading, and resource allocation is drawing significant re- search attention.

D. ReconfigurableIntelligentSurfaces

Devicesphysicallyobstructedfromtheirbasestationssuffer channel degradation that limits offloading throughput regard- less of server availability. Reconfigurable Intelligent Surfaces (RIS) are arrays of passive reflective elements with software- controlled phase responses, positioned to redirect wireless signals around obstacles. No additional transmit power is required.

For MEC, improved channel quality between device and basestationdirectlyincreasesachievableoffloadingrates andreducesdevicetransmissionenergy.Jointoptimization of surface configuration, offloading decisions, and resource allocation is an active research area.

VI. OPEN RESEARCH CHALLENGES

A. TractableHeterogeneityModeling

A consistent simplification in published MEC literature is treating all devices as equivalent — same data rate classes, compute profiles, battery characteristics, deadline distribu- tions. Real deployments are far more varied. A smart building networkmightsimultaneouslyservepassiveHVACsensors generating bytes per minute and 4K video analytics nodes saturatinggigabitlinks.Existingoptimizationframeworkshandle eitherdeviceclasswellinisolation;theytendtodegradewhen both are present in the same system. Building frameworksthat accommodate genuine heterogeneity without collapsing tractability is an unresolved challenge.

B. PrivacyinMulti-TenantDeployments

MostMECoffloadingmodelsassumethattheedgeserveris a trusted party for all tenants using it. In practice, commercial edge deployments are multi-tenant: a single server handles requestsfromdevicepopulationsbelongingtodifferentorganizations,operatingunderdifferentdatagovernanceconstraints. Awearablehealthdevicethatoffloadsbiometricprocessingto such a server has no hardware-enforced guarantee that its data won't be accessed by co-located workloads. Homomorphic encryption is the theoretically clean answer — it enables computation on encrypted data without decryption — but the overhead it imposes today is two to three orders of magnitude beyond what MEC latency budgets can absorb. Hardware- based trusted execution environments are substantially faster, buttheycomewiththeirownvulnerabilitysurfaceanddepend on silicon features that aren't present in all deployed edge hardware. Neither option fully closes the gap between privacy requirements and practical constraints.

C. Long-HorizonPlanningUnderNon-Stationarity

Most deployed offloading algorithms are reactive: observe currentbatterylevel,channelquality,andserverload,thenpick an action. This works reasonably well when system statistics are stationary — when current observations are representative of near-future conditions. IoT environments often are not stationary.Energy-harvestingdevicesaretheclearestexample: a device that anticipates strong solar input in two hoursshouldmakedifferentoffloadingdecisionsnowcompared to one facing an overcast afternoon. Lyapunov optimization andmodel-predictivecontroladdresstemporalcouplingunder specificassumptionsaboutfuturestatisticalbehaviorthatfield conditionsroutinelyviolate.Robustlong-horizonplanningun- dernon-stationaryuncertaintyinresource-constrainedsystems is still an open problem.

D. Standardization Fragmentation

ETSI MEC, the OpenFog Consortium, and AKRAINO de- scribe overlapping but mutually incompatible visions of edge infrastructure. Offloading frameworks built for one platform requiresubstantialreworktodeployonanother.Forpractition- ers, this means research results — which are almost always platform-specific—cannotbestraightforwardlycompared or translated to production deployments. The widening gap between research capability and practical deployability is a real impediment to the field's progress.

VII. METHODOLOGY

A. SystemModel

We consider a network of NIoT devices and Medge serverscommunicatingthroughsharedbasestationinfrastructure.Device$i$generatestaskscontinuously.Eachtaskis characterized by a data volumed_i(bits), a computational requirementc_i(CPU cycles), and a completion deadlineT_i. Tasksthatdonotcompletewithintheirdeadlinesareclassified asfailures.Theoptimizationobjectivethereforeaddressesboth averagelatencyminimizationanddeadlinemissratereduction.

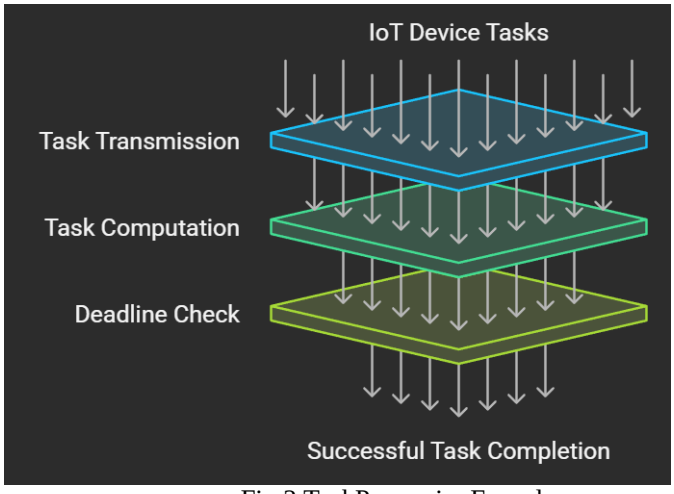


Fig.2.TaskProcessingFunnel

B. ProblemFormulation

We seek a joint policy that reduces total latency and energy expenditure across all tasks in a given time window. Four constraint classes bound the feasible solution space:

- Deadlineconstraints:Task$i$mustfinishexecutionand return its result within time $T_i$from submission.
- Server capacity constraints: At no time step may the aggregateCPUdemandassignedtoanysingleedgeserver exceed that server's physical processing capacity.
- Transmission power constraints:Radiotransmitpower on each device must stay within the hardware-specified limit throughout the offloading period.
- Assignment constraints**:** Every task runs on exactly one resource—thelocaldeviceoronedesignatededgeserver—withnomid-taskmigrationpermitted.

C. DQN-BasedOffloadingFramework

WeuseaDeepQ-Networktolearntheoffloadingpol- icy from experience rather than from a pre-specified system model. At each scheduling epoch, the agent observes three quantities from the current environment: the battery charge level of the requesting device, the estimated channel quality towardeachavailableedgeserver,andthecurrentqueuedepth ateachofthoseservers.Theseformthestatevectors_t.

Givens_t,theagentcomputesaQ-valueforeachcan- didateaction—runthetasklocally,oroffloadtoserver$k \in \{1,\ldots,M\}$—andtakestheaction withthehighestvalue. Afterexecution,theobservedoutcomesdeterminethereward:

$$r_t = -(\alpha \cdot L_t + \beta \cdot E_t) \quad (1)$$

where $L_t$is the measured task latency, $E_t$is the measured energy consumption, and α, β are weights that set the relative importanceofthetwoobjectives.Thenegativesignmeansthe agent is rewarded for minimizing both simultaneously.

The case for using RL here rests on one practical ob- servation: any analytically derived offloading policy encodes assumptions about channel statistics, arrival rates, and server behavior that hold at calibration time and erode afterward. A DQN policy doesn't freeze those assumptions — it keeps up- dating from what the deployed system actually does, retaining effectiveness as operating conditions evolve.

D. Algorithm

Algorithm1DQN-BasedTaskOffloading

1:InitializeDQNnetworkparametersθandedgeserver capacities
2:**repeat**
3: IoTdevicegeneratestask withparameters (d_i, c_i, T_i)
4: Observestates$_t$:batterylevel,channelquality,server queue depths
5: Selectaction$a_t = \arg\max_a Q(s_t, a; \theta)$

6: **if** a_t=offloadtoserver$k$ **then**
7: Transmittasktoserver$k$
8: Executetaskatedgeserver$k$
9: Transmitresulttodevice
10: **else**
11: Executetasklocallyondevice
12: **endif**
13: Measureresultinglatency$L_t$andenergyE_t
14: Computerewardr_t= $-(\alpha L_t+\beta E_t)$
15: Update DQN parameters using experience (s_t, a_t,r_t, s_{t+1})
16:**until**convergenceordeploymenttermination

VIII. RELATEDWORK

Table I summarizes relevant prior work, the problems ad- dressed, and identified limitations.

TABLEI
SUMMARYOFRELATEDWORK:PROBLEMSADDRESSEDANDLIMITATIONS

Sr.	Author(s)	ProblemAddressed	Limitations
1	Ke Zhang et al. (2018)	IoTrequireslow-latencydistributedpro-cessing	Limiteddeploymentscope:scalabilityunderlarge device populations not validated
2	ZhihaoRenetal.(2025)	AR/VRapplicationsdemandmicrosecond-scalelatencyguarantees	High infrastructure cost; integration complexitywith existing deployments
3	Seung-QueLeeetal.(2020)	Routingoverheadcausesunacceptablela-tency:URLLCrequiressub-1msresponse	Validationlimitedto5Gscenarios:scalabilitytoreal deployments unconfirmed
4	RaghuDhumpatietal.(2025)	Staticresourceallocationproduceslatencyspi kes under load variation	MLinferenceoverhead:privacyconcernsinshared infrastructure
5	MustafaErgenetal.(2024)	5Gedgecapacityinsufficientforanticipated futurecommunicationdemands	Early-stage:standardizationpendingacrossven-dor platforms
6	TienV.Thaietal.	MIMO-MECintegrationfor6Gscenarios inadequatelystudied	Surveyonly:noexperimentalvalidationpro-vided
7	AbdulmohsenAlmalawietal.	Inefficientedgecachingcausesunnecessarylat ency at access nodes	Algorithmvalidatedinlimitedscenarios:real-world deployment not verified
8	ShamimTaimooretal.	Fixededgeserverscannotcoverremoteor disaster-affectedareas	HighUAVenergyconsumption:trajectoryplan-ning is computationally intensive
9	ZouheirTrabelsietal.	Latency-sensitiveIoVapplicationsrequire intelligentoffloading	Fuzzy rule sets require manual tuning; general-ization across IoV scenarios
10	Laszlo Toka	DeployingURLLCapplicationsatedgelacks standard orchestration	unverifiedKubernetesoverheadmayconflictwithstr ict latencytargets
11	LucianoBaresietal.	Traditionaledgearchitectureslackflexibil-ityfordynamicworkloads	Cold-startlatencyinserverlessfunctions:lim-ited support for stateful task processing
12	SreenuBanothetal.	IoTdataprocessingrequireslow-latency architecturesproximatetodatasources	Survey-based:lacksperformancebenchmarkingacross architectural variants
13	AmitKumarMishra	Applicationssimultaneouslydemandlowla-tencyandhighthroughput	Theoreticalanalysisonly:nosimulationorex-perimental validation provided

IX. CONCLUSION

MEC works. The latency and energy reductions it produces are not theoretical projections — they follow from moving computationphysicallyclosertowheredataisgenerated, andthemechanismiswellunderstood.What'sharderto close is the gap between what MEC can do in a well- configureddeploymentandwhatitconsistentlydeliversunder the messy conditions of real infrastructure: dynamic traffic, heterogeneous devices, shifting channel statistics, and energy supplies that don't hold to schedule.

Threeproblemssitatthecenterofthatgap.First,offloading policiesthatperformwellincontrolledsettingstendtodegrade as operating conditions drift, because most of them embed assumptions about the environment that stop holding over time.Second,multi-tenantresourceschedulingonsharededgeserversrequireshandlinggenuineworkloadheterogeneity—not averaged abstractions — to avoid systematic unfairness or priority inversions at peak load. Third, energy management across device, network, and server tiers needs to be treated asa coupled problem, not a per-tier optimization.

The DQN-based framework described here targets the of- floadingpiece.Itlearnsfromexperienceratherthanfroma pre-calibrated model, which gives it the ability to track shifting conditions without manual re-tuning. On its own it doesn't solve the multi-tenant scheduling or cross-tier energy coupling problems, but it provides a foundation that can be extended.Plannednextstepsincludemulti-agentcoordination to handle competition among concurrent offloading decisions, integration of energy harvesting forecasts into the reward signal, and evaluation against mobility traces that include realistic handover sequences.

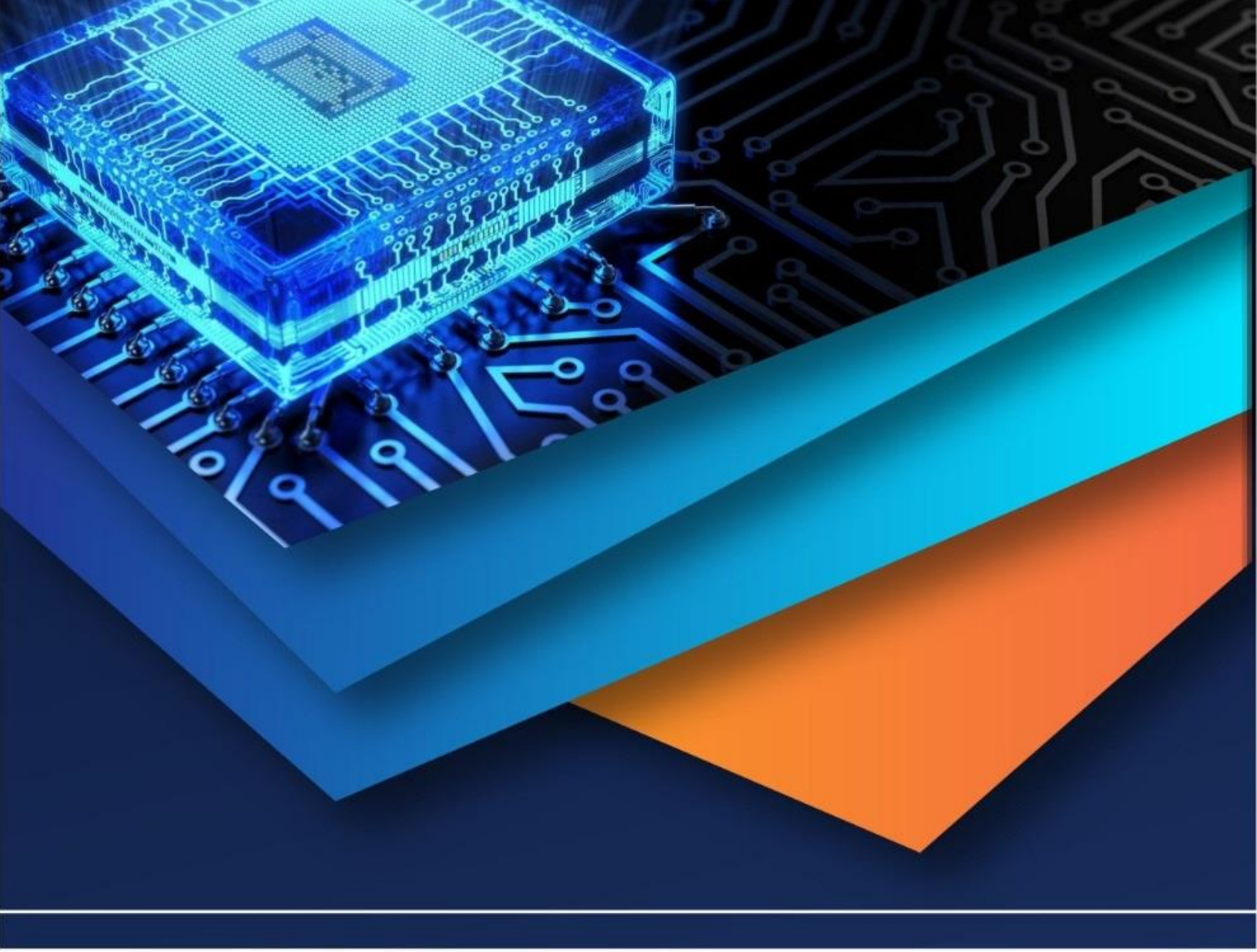
X. ACKNOWLEDGMENT

We are grateful to Shah & Anchor Kutchhi Engineering College for providing the research environment and facilities that supported this work. We also thank the anonymous reviewers whose comments strengthened the paper.

REFERENCES

[1] K.Zhang,Y.Mao,S.Leng,Q.Zhao,L.Zhang,X.Peng,L.Pan,
[2] S.Maharjan,andY.Zhang,"Energy-efficientoffloadingformobileedgecomputingin5Gheterogeneousnetworks,"IEEEAccess,vol.4,pp.5896–5907,2016.
[3] Z.Ren,Z.Liu,andY.Chen,"Ultra-lowlatencyedgecomputingforimmersive AR/VR applications," IEEE Trans. Mobile Comput., 2025.
[4] S.-Q. Lee, D.-H. Han, and J.-H. Kim, "URLLC-aware routing forindustrial IoT in 5G networks," in Proc. IEEE VTC, 2020.
[5] R. Dhumpati, A. Sharma, and K. Rao, "Adaptive resource allocation forheterogeneous MEC environments," IEEE Internet Things J., 2025.
[6] M. Ergen, B. Aydin, and T. Yilmaz, "Beyond 5G edge computing:Architectural considerations for 6G," IEEE Commun. Mag., 2024.
[7] T. V. Thai, N. H. Tran, and C. S. Hong, "MIMO-assisted mobile edgecomputing:Asurveyfor6Gsystems,"IEEECommun.Surv.Tut.,2023.
[8] A. Almalawi, X. Yu, Z. Tari, A. Fahad, and I. Khalil, "Intelligent edgecachingforlatency-sensitiveIoTservices,"IEEETrans.ParallelDistrib.Syst., 2023.
[9] S.Taimoor,M.Ali,andA.Khan,"UAV-assistedmobileedgecomputing:Energyandcoverageoptimization,"IEEEWirelessCommun.Lett.,2023.
[10] Z. Trabelsi, H. Afifi, and F. Zarai, "Fuzzy logic-based task offloadingfor Internet of Vehicles," in Proc. IEEE ICC, 2023.
[11] L. Toka, "Kubernetes-based orchestration for URLLC edge applica-tions," in Proc. IEEE INFOCOM, 2022.
[12] L.Baresi,D.Filgueira,andG.Quattrocchi,"Serverlessedgecomputing:Challenges and opportunities," IEEE Pervasive Comput., vol. 21, no. 2,pp.48–56,2022.
[13] S. Banoth and R. Thakur, "Survey of edge computing architectures forreal-time IoT data processing," IEEE Access, 2023.
[14] A. K. Mishra, "Joint latency and throughput optimization in multi-tieredge computing networks," IEEE Trans. Netw. Sci. Eng., 2023.
[15] P. Mach and Z. Becvar, "Mobile edge computing: A survey on archi-tectureandcomputationoffloading,"IEEECommun.Surv.Tut.,vol.19,no. 3, pp. 1628–1656, 2017.
[16] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimizedcomputation offloading performance in virtual edge computing systemsvia deep reinforcement learning," IEEE Internet Things J., vol. 6, no. 3,pp.4005–4018,2019.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)