

Bridging the Internet of Mirrors (IoM) and Internet of Sounds (IoS): An RL-Driven Multimodal Task Orchestration Approach

Haneen Fatima*, Muhammad Ali Imran*, Lina Mohjazi*, Ahmad Taha*

*James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, UK

Email: {haneen.fatima, muhammad.imran, lina.mohjazi, ahmad.taha}@glasgow.ac.uk

Abstract—The Internet of Mirrors (IoM) and Internet of Sounds (IoS) are IoT sub-ecosystems centred on distinct interfaces and application domains, with visual and audio sensing respectively as their primary modalities, whose complementary capabilities can support shared user-centric applications. Existing reinforcement learning (RL) formulations for IoT task offloading assume a single ecosystem, deriving a task’s eligible nodes, modality requirements, and urgency from generic system-state signals rather than from the context of an externally originating ecosystem. This paper presents a context-aware RL formulation for cross-ecosystem task orchestration, in which ecosystem origin and modality determine feasibility, while application context determines urgency. The formulation is motivated by an envisioned home-based vocal rehabilitation scenario, where an IoM smart mirror and a co-deployed IoS smart speaker jointly monitor the same recovery process, and is realised through a Double DQN agent with hard action masking and a mode-conditioned reward. Simulation results show the trained policy routes tasks consistently with this design, outperforms a cloud-offload baseline on latency and deadline satisfaction for live-interactive tasks as load increases, and sustains robust operation as audio-capable hardware becomes scarce by deferring infeasible tasks rather than failing outright, a condition the baseline has no mechanism to address. These results demonstrate that the IoM’s tiered infrastructure and the IoS’s audio-sensing specialisation can be jointly orchestrated to serve a shared scenario more effectively than either ecosystem operating alone.

Index Terms—Internet of Things, Internet of Mirrors, Internet of Sounds, Assistive living, Smart Mirrors, Context-Aware Systems

I. INTRODUCTION

A. Background

The Internet of Things (IoT) landscape is seeing the rise of a multitude of sub ecosystems, each built around smart, connected devices serving a particular sensing modality or application community [1]. As these ecosystems mature, they are increasingly co-deployed within the same physical environments, raising the question of how they interact with one another. The Internet of Mirrors (IoM) is one such ecosystem, composed of a diverse and heterogeneous network of smart mirrors with sensing and communication capabilities, delivering personalised health, beauty, and assisted-living services through a shared visual interface [2], [3]. The Internet of Sounds (IoS) is another example of an IoT sub-ecosystem comprised of networked “Sound Things”, devices that sense,

process, and act on audio-related information, spanning musical and non-musical applications [4], [5].

B. Motivation

Advanced, user-centric applications envisioned under the smart-cities vision, particularly in accessible and assistive living contexts, stand to benefit from combining the specialisations different ecosystems have independently developed for their own domain. The IoM’s tiered, visually-oriented sensing infrastructure and the IoS’s networked audio-sensing capabilities each serve different aspects of the same user-centric scenario, and bringing them together can enhance what either ecosystem offers on its own.

Realising this potential requires more than co-locating the two ecosystems in the same environment, it requires a mechanism capable of deciding, under live and dynamic conditions, how tasks originating from one ecosystem should be handled by the other’s infrastructure. Fixed or rule-based coordination schemes are ill-suited to this setting, since the load that such interactions introduce will vary in modality, urgency, and volume in ways that cannot be fully anticipated at design time. Reinforcement learning (RL), which learns adaptive decision-making policies directly from interaction with a dynamic environment, offers a natural mechanism for this kind of orchestration.

C. Related Work

Reinforcement learning has become a prominent paradigm for adaptive task offloading in IoT and edge-computing systems, with deep RL reported to improve latency, energy efficiency, and task completion across communication, computing, caching, and control [6], [7]. Value-based methods such as Double DQN are widely used for discrete offloading decisions [8], [9], while continuous-action and multi-agent variants address resource-allocation and cooperative-scheduling formulations elsewhere in the literature [10], [11]. A structurally similar tiered topology, end device, edge/fog, and cloud, recurs across this body of work, with agents mapped to tiers so that routing decisions reflect both local constraints and global objectives [12] [13].

These formulations are built around three assumptions that reflect the single ecosystem they were designed for. Tiers are defined by infrastructure capacity and network proximity –

what a node can compute, rather than by the role, specialisation, or origin of the task being routed to it [12], [14]. Data modality is typically treated as a factor shaping routing quality, enforced through reward shaping, traffic classification, or per-class QoS policy, rather than as a hard feasibility constraint on which nodes can process a task at all [15], [16]. Task urgency is derived from system-state signals such as queue depth or resource load [17], [18], and even where priority is modelled more explicitly, it is weighted through system-level or user-specified preference terms rather than the application-domain context in which a task is running [19], [20].

Each of these design choices is a reasonable fit when every task originates within the same ecosystem being orchestrated, since that ecosystem's own infrastructure design already implicitly encodes what its tasks need. An RL formulation built on these assumptions cannot simply be transferred to a cross-ecosystem setting, it must instead be adjusted and reformulated around the context and requirements of the specific ecosystems involved. An externally originated task's compatible processing tiers, its hardware and modality requirements, and its urgency are all determined by the context of the ecosystem it came from. These are not observable from system-state load alone, and not something the orchestrating ecosystem's own infrastructure was designed around. Reformulating an RL-based orchestration approach around this cross-ecosystem context is the gap this paper addresses.

D. Contribution

The main contributions of this work are as follows:

- A context-aware reinforcement learning formulation for cross-ecosystem task orchestration, in which the tiers, modality requirements, and urgency of an externally originated task are derived from the context of the ecosystem it originates from, rather than treated as generic system-state signals.
- An evaluation of IoM-IoS RL orchestration against a cloud-offload baseline through simulation, across varying load conditions and IoS modality availability.
- Grounding and demonstration of the approach through a home-based vocal rehabilitation scenario, in which the IoM orchestrates externally originated IoS audio-inference tasks alongside its own native visual workload.

II. SYSTEM ARCHITECTURE AND MODEL

Fig. 1 shows the overall architecture, spanning the IoM's own structure, the orchestration layer introduced in this paper, and its connection to the co-deployed IoS.

A. IoM Architecture

The IoM is deployed as a three-tier hierarchical network of nodes, reflecting increasing compute capability rather than a restriction on which nodes can communicate with one another:

- **Consumer tier** - home mirrors, retail kiosks, and personal devices
- **Professional tier** - clinics, salons, and pharmacies

- **Hub tier** - hospitals, enterprise servers, and service providers

Independently of this tier hierarchy, the IoM architecture is organised into three functional layers:

- **Perception layer** - sensing and user interaction through the smart-mirror interface
- **Network layer** - cross-tier connectivity, bandwidth allocation, and quality of service
- **Application layer** - service execution and delivery at the assigned tier node, spanning health, beauty, and assisted-living services

The tier hierarchy describes *where* computation happens; the functional layers describe *what kind* of work is done at each stage. This tiered structure and the latency-privacy trade-offs it introduces have been characterised in prior IoM work through an edge-cloud latency analysis [21] and a physical testbed evaluation of computational placement strategies across the tier hierarchy [22].

B. Orchestration Layer

The Intelligent Orchestration Layer is a control plane that coordinates task assignment across the tier hierarchy and functional layers shown in Fig. 1: it reads context signals and metrics from the Perception layer and network state from the Network layer, and writes its routing decision to the Application layer, where the task is then executed at the assigned tier node.

For single-modality tasks within the IoM hierarchy, prior work uses deterministic, context-aware allocation where decisions adapt to current conditions, but the allocation logic remains predefined rather than learned [22], [23]. Coordinating tasks that span multiple modalities and originate from an externally deployed ecosystem, as in Fig. 1, requires the orchestration layer to instead act as an adaptive control plane, capable of evaluating and routing tasks whose modality, origin, and requirements are not fixed in advance. This is the role the orchestration layer plays in the system, and its formal basis is defined in Section III.

C. IoS as an External Ecosystem

The IoS device in Fig. 1, a co-deployed smart speaker referred to here as a Sound Thing, is not a node in the IoM tier hierarchy and does not participate directly in the Perception/Network/Application structure. It connects to the orchestration layer through a single task-profile-and-request link, and any task it originates is, from that point on, handled like a native IoM task, assigned to a tier node, subject to the same constraints, scored by the same reward, with the result returned to the IoS device over the same link. In this context and initial setup, the IoS device performs no processing itself; it is a sensing and delivery endpoint, not a compute participant in the IoM hierarchy.

To visualise how the orchestration layer may interact with other ecosystems and Things under the IoT umbrella, the 'Internet of X' (IoX) is shown alongside the IoS in Fig. 1 as a general integration point. The IoM-IoS is one instance of this

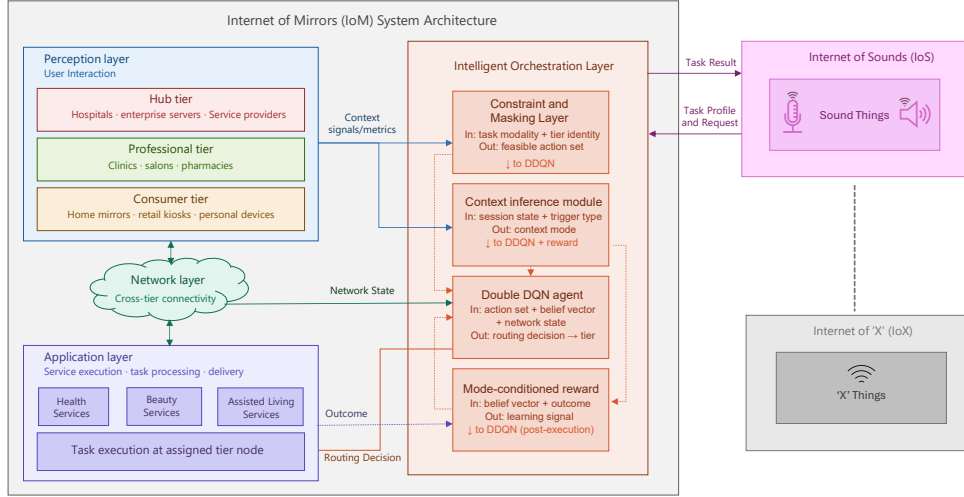


Fig. 1: IoM system architecture, showing the tier hierarchy and functional layers, the Intelligent Orchestration Layer, and its connection to the co-deployed IoS and the broader IoX vision.

broader IoX vision, and the orchestration layer's connection to it is structured accordingly, as an external link to a co-deployed ecosystem.

III. PROBLEM FORMULATION

This section formalises the orchestration problem introduced in Section II by separating deterministic context inference, hard feasibility constraints, learned routing, and mode-conditioned outcome evaluation.

Table I summarises the notation used throughout.

A. Task Profiles

A task profile is the load that the orchestration layer actually routes, it fixes a task's ecosystem origin, modality, operation type, and resource demands. Six representative task profiles are used in this work. The purpose of these profiles is to show the diversity of modality, ecosystem origin, and urgency that the orchestration layer must jointly handle within a single representative scenario, rather than testing the formulation against a narrow or homogeneous task set. They are defined along the following dimensions:

- **Ecosystem** - whether the task is native to the IoM or externally originated from the IoS.
- **Modality** - visual or audio, determining which nodes are hardware-eligible via ϕ .
- **Operation** - the type of processing performed (detection, tracking, scoring, classification, or reporting).
- **Trigger(s)** - which context mode(s) the profile can be invoked under.
- **Compute and latency demand** - the resource and timing requirements the profile places on its assigned node.

Table II specifies the six profiles spanning both ecosystems along these dimensions, three per ecosystem, together covering all three context modes.

Stringent and Tolerant latency map onto the deadlines defined in Section III-G: Stringent tasks carry the 200 ms (LI)

or 1000 ms (AM) budget, Tolerant tasks the 30 s (BB) budget. Compute (Low/Medium/High) is a qualitative descriptor of relative demand across the six profiles rather than a hard resource constraint, the only hard feasibility constraint enforced here is modality-hardware compatibility.

The metrics used to define these profiles are based on existing related applications. Pitch Scoring and Speech Clarity Classification are grounded in prior work on classical pitch-tracking methods and measured 5G-IoMusT link figures for the former [24], and compact audio-classification CNN architectures such as DS-CNN for the latter [25], [26]. The remaining profiles' compute demands are scaled relative to these two, reflecting the added cost of continuous tracking over single-frame detection, and of full-session aggregation over per-instance inference.

B. Representative Use Case

The task profiles above are intended to generalise across the diverse accessible use cases envisioned by the IoM and IoS, rather than being specific to any one application. To ground their use concretely, we instantiate and evaluate the formulation through one representative scenario: a home-based vocal rehabilitation session, a setting in which voice-assisted technology for home speech and voice therapy is an active and clinically motivated research direction [27]. The scenario is deliberately chosen to exercise both ecosystems together rather than in isolation since visual and vocal cues are complementary evidence of the same recovery process, and the session's structure naturally produces the mix of live, low-urgency, and post-session processing the orchestration layer must handle. Concretely, one session proceeds through three stages:

- **Active exercise repetitions**, during which an IoM smart mirror monitors the user's facial expression and lip movement while a co-deployed IoS smart speaker delivers a structured vocal exercise and captures the user's spoken

TABLE I: Notation Summary

Symbol	Meaning
$\mathcal{T}$	Task set
T	Decision horizon (number of steps)
N	Node set (Hub, Professional, Consumer tiers)
W	Set of links
w	A single link
$u(n)$	Utilisation of node n
$q(n)$	Queue depth of node n
$B(w)$	Total bandwidth of link w
$b(w)$	Currently-available bandwidth of link w
$\text{Load}(w)$	Current traffic load on link w
$R(w)$	Reliability of link w
E	Ecosystem set (open, extensible)
$e(t)$	Ecosystem origin of task t
M	Modality set (open, extensible)
$m(t)$	Modality of task t
ϕ	Processing-capability function
p_{aud}	Audio-provisioning probability
$\text{tier}(n)$	Tier membership of node n
$\sigma_r(t)$	Entry node for task t
Access	Permitted-tier access function
$\kappa(t)$	Trigger type of task t
$\text{mode}(t)$	Context mode of task t
$\mathcal{A}$	Action set
$\mathcal{A}(s_t)$	Feasible action set at state s_t
$n(t)$	Node assigned to task t
$\rho(w, t)$	Clamped utilisation of link w
$PL(w, t)$	Per-link packet loss
$PL(t)$	Full-path packet loss
θ_m	Packet-loss threshold for modality m
ξ	Congestion-sensitivity exponent
λ_{loss}	Packet-loss penalty weight
$\text{Path}(t)$	Service path for task t
$\ell(t)$	End-to-end latency of task t
$D(t)$	Deadline of task t
$\psi(\text{mode})$	Deadline-miss penalty magnitude
$\psi_{\text{defer}}(\text{mode})$	Defer penalty magnitude
C_{lat}	Clamp on the latency-ratio reward term
R_{lat}	Latency reward component
R_{dead}	Deadline-satisfaction reward component
R_{eff}	Resource-efficiency reward component
R_{loss}	Packet-loss reward component
$w_k(t)$	Mode-selected weight for component k
$w_k^{(i)}$	Per-mode weight for mode i
π	Policy
π^*	Optimal policy

response, both requiring immediate feedback to sustain the exercise loop.

- **Inter-repetition gaps**, during which the same sensing pipelines continue at a reduced rate, confirming the user is still present and audible rather than actively scoring an exercise.
- **Session close**, at which the accumulated visual and vocal data are compiled into summary reports for later clinician review, with no live user present.

TABLE III: Use-Case Pipeline Mapping

Pipeline step	Task profile invoked
Mirror watches the user's face during the exercise	Expression Detection (LI)
Mirror checks mouth shape against the target sound	Lip Movement Tracking
Speaker compares spoken pitch to the target	Pitch Scoring (LI)
Speaker judges whether speech came out clearly	Speech Clarity Classification
Between repetitions: confirms user still present/audible	Expression Detection + Pitch Scoring (AM, low-rate)
After the session trends compiled for clinician review	Session Summary Report + Vocal Trend Report

Table III maps each stage's constituent steps onto the task profiles defined above.

C. Modality-Hardware Capability and State Space

The IoM's existing tier hierarchy already represents heterogeneity in compute capacity. Accommodating externally originated IoS tasks introduces a second, structurally distinct axis, some nodes are physically incapable of processing a given modality regardless of available compute. Under the modelled capability assumptions, Consumer nodes are excluded from audio processing, while audio-processing capability at Professional and Hub nodes is provisioned independently at topology instantiation:

$$\phi(n, \text{vis}) = 1, \quad \forall n \in N \quad (1)$$

$$\phi(n, \text{aud}) = 0, \quad \forall n \in C \quad (2)$$

$$\phi(n, \text{aud}) \sim \text{Bernoulli}(p_{\text{aud}}), \quad \forall n \in P \cup H \quad (3)$$

where $\phi : N \times M \rightarrow \{0, 1\}$ indicates whether node n has the hardware pipeline for modality m . Visual processing is universal across all three tiers. The provisioning probability p_{aud} is not fixed since it is varied across evaluation runs (Section IV) to test how task completion and defer behaviour degrade as audio-processing capability becomes scarcer across the Professional and Hub tiers, directly probing the consequence of cross-ecosystem modality heterogeneity.

The state observed by the orchestration layer at each decision step combines this node-level information with system- and task-level information:

$$s_t = \langle \{u(n)\}, \{q(n)\}, \{b(w)\}, \{R(w)\}, \{\phi(n, \text{aud})\}_{n \in P \cup H}, m(t), e(t), \kappa(t), \text{mode}(t) \rangle \quad (4)$$

$u(n)$ and $q(n)$ give per-node utilisation and queue depth; $b(w)$ and $R(w)$ give available bandwidth and reliability per link. Ecosystem origin $e(t)$ and modality $m(t)$ are carried explicitly, rather than folded into a single task-type label, because a node's eligibility for an externally originated task depends on both – not on system load alone.

D. Context Mode Determination

Generic IoT offloading formulations typically derive task urgency from system-state signals such as queue depth or resource load, which describe how busy the system is rather than the consequence of delay. In a cross-ecosystem, multi-modal setting such as IoM-IoS, urgency can instead vary with the point in a session at which a task occurs. We therefore derive task urgency from application-domain context rather than system state.

The three context modes introduced here follow from two properties of a session that jointly determine urgency, whether a live user interaction is currently active, and, if so, how time-sensitive the task's outcome is. Table IV summarises the resulting structure. Each mode corresponds to a genuinely distinct operational phase of a live-interaction session, instantiated concretely for the vocal rehabilitation scenario in Table III.

TABLE II: Task Profile Categories and Characteristics

Task	Ecosystem	Modality	Operation	Trigger(s)	Compute	Latency
Expression Detection	IoM	Visual	Detection	Live + Ambient	Medium	Stringent
Lip Movement Tracking	IoM	Visual	Tracking	Live	Medium-High	Stringent
Session Summary Report	IoM	Visual	Reporting	Background	High	Tolerant
Pitch Scoring	IoS	Audio	Scoring	Live + Ambient	Medium	Stringent
Speech Clarity Classification	IoS	Audio	Classification	Live	Medium-High	Stringent
Vocal Trend Report	IoS	Audio	Reporting	Background	High	Tolerant

TABLE IV: Context Modes

Mode	Session active	Urgency	Example tasks
LI	Yes	High	Lip tracking, speech clarity
AM	Yes	Low	Low-rate expression/pitch checks
BB	No	Low	Session/vocal trend reports

Rather than being inferred by a learned classifier, the context mode is computed deterministically from the task's trigger type:

$$\alpha(t) = 1[\kappa(t) \in \{\text{live, ambient}\}] \quad (5)$$

$$\text{mode}(t) = \begin{cases} \text{LI} & \text{if } \alpha(t) = 1 \wedge \kappa(t) = \text{live} \\ \text{AM} & \text{if } \alpha(t) = 1 \wedge \kappa(t) = \text{ambient} \\ \text{BB} & \text{if } \alpha(t) = 0 \end{cases} \quad (6)$$

Because $\alpha(t)$ is itself a function of the already-observed $\kappa(t)$, $\text{mode}(t)$ is fully determined once the trigger is known. The mode determines deadline and reward priorities, not the destination node, routing remains the responsibility of the Double DQN after feasibility masking.

E. Action Space and Constraint Masking

Existing offloading formulations typically treat data modality as a soft factor shaping routing quality rather than a hard feasibility constraint. This does not extend to a cross-ecosystem setting, where an externally originated task's eligible nodes are limited by hardware capability that no amount of available compute can substitute for. At each decision step, the orchestration layer selects a specific processing node, or defers:

$$\mathcal{A} = N \cup \{\text{defer}\} \quad (7)$$

Two hard constraints define the feasible subset of $\mathcal{A}$ at state s_t : modality-hardware eligibility, and stakeholder-tier access:

$$N_m(t) = \{n \in N : \phi(n, m(t)) = 1\} \quad (8)$$

$$N_{acc}(t) = \{n \in N : \text{tier}(n) \in \text{Access}(\text{tier}(\sigma_r(t)))\} \quad (9)$$

The feasible action set is their intersection, with defer unioned in unconditionally so that $\mathcal{A}(s_t)$ is never empty:

$$\mathcal{A}(s_t) = (N_m(t) \cap N_{acc}(t)) \cup \{\text{defer}\} \quad (10)$$

$\sigma_r(t)$ is the entry node, the requesting IoM node for native tasks, or the co-deployed entry node for externally originated IoS tasks. Ecosystem origin $e(t)$ does not appear explicitly in the constraint equations, cross-ecosystem handling is concentrated entirely in how $\sigma_r(t)$ is set, so the same constraint structure applies regardless of a task's origin. This formulation is a state-wise constrained decision process, constraints are enforced by masking the action set directly, guaranteeing zero infeasible routing at every step, rather than through an expectation-budget constraint as in a classical constrained MDP.

F. Packet Loss and End-to-End Latency

Packet loss and latency are tracked explicitly because they are the primary channel through which the added IoS-IoM link, and the cross-ecosystem load it carries, are felt by the rest of the system. Packet loss on a link is derived from its reliability and its current congestion:

$$\rho(w, t) = \min\left(\frac{\text{Load}(w)}{B(w)}, 1\right) \quad (11)$$

$$PL(w, t) = 1 - R(w) \cdot (1 - \rho(w, t))^\xi \quad (12)$$

$$PL(t) = 1 - \prod_{w \in \text{Path}(t)} (1 - PL(w, t)) \quad (13)$$

$\rho(w, t)$ is clamped to $[0, 1]$ so that an overloaded link does not invert the loss calculation. ξ controls how sharply congestion-driven loss rises near saturation.

End-to-end latency is measured to delivery at the user-facing endpoint, over a path from source, to the processing node, to the delivery endpoint:

$$\text{Path}(t) = [\text{src}(t) \rightarrow \sigma_r(t)] \oplus [\sigma_r(t) \rightarrow n(t)] \oplus [n(t) \rightarrow \text{dst}(t)] \quad (14)$$

$$\ell(t) = \ell_{\text{uplink}}(t) + \ell_{\text{route}}(t) + \ell_{\text{proc}}(t) + \ell_{\text{downlink}}(t) \quad (15)$$

For a native IoM task, $\text{src}(t) = \text{dst}(t) = \sigma_r(t)$; for an IoS-originated task, both endpoints are the IoS device, with $\sigma_r(t)$ the co-deployed entry node.

Each mode carries a deadline $D(t)$ reflecting the differing consequence of delay. Live-Interactive tasks use a 200 ms end-to-end budget. ITU-T G.114 identifies approximately 150 ms one-way delay as a reference for interactive real-time communication [28], corresponding to approximately 300 ms for a symmetric round trip. The 200 ms budget adopted here is deliberately more stringent for the closed feedback loop between a user's live action and the system's response. Ambient Monitoring's 1000 ms budget and Background-Batch's 30 s budget are successively looser in accordance with their reduced urgency.

G. Reward Function

Task urgency, once derived from context mode, must also shape how outcomes are scored: a single fixed weighting cannot represent the differing priorities of a live interaction

versus a batch process. The reward is therefore composed of four signed components and conditioned on context mode:

$$R_{lat}(t) = -\min\left(\frac{\ell(t)}{D(t)}, C_{lat}\right) \quad (16)$$

$$R_{dead}(t) = -\psi(\text{mode}(t)) \cdot 1[\ell(t) > D(t)] \quad (17)$$

$$R_{eff}(t) = 1 - \frac{\text{rank}(n(t)) - 1}{2} \quad (18)$$

$$R_{loss}(t) = -\lambda_{loss} \cdot \max(0, PL(t) - \theta_{m(t)}) \cdot 1[\kappa(t) \in \{\text{live}, \text{ambient}\}] \quad (19)$$

where $\text{rank}(C)=1$, $\text{rank}(P)=2$, $\text{rank}(H)=3$, so R_{eff} rewards the lowest-capability tier that satisfies the task. R_{lat} is clamped at C_{lat} to bound the penalty for a repeatedly deferred or very late task. R_{loss} is structurally zero for Background-Batch tasks, formalising their exemption from packet-loss penalty.

The composite reward is piecewise, since a deferred task has no assigned node:

$$r_t = \begin{cases} \sum_k w_k(t) \cdot R_k(t) & \text{if } a_t \in N \\ -\psi_{defer}(\text{mode}(t)) & \text{if } a_t = \text{defer} \end{cases} \quad (20)$$

$$w_k(t) = w_k^{(\text{mode}(t))}, \quad w_k^{(i)} \geq 0, \quad \sum_k w_k^{(i)} = 1 \quad \forall i \quad (21)$$

A deferred task retains its mode and deadline, so deferral does not exempt it from urgency: ψ_{defer} is itself mode-scaled, and the accumulating wait is reflected in $\ell(t)$ once the task is eventually assigned. Table V gives the per-mode weight vectors used in this work.

TABLE V: Per-Mode Reward Weights

Mode	R_{lat}	R_{dead}	R_{eff}	R_{loss}
LI	0.35	0.35	0.10	0.20
AM	0.25	0.20	0.30	0.25
BB	0.10	0.10	0.80	0.00

These weight vectors are a designed prior rather than a derived optimum: the ordering within each mode follows directly from the mode semantics defined in Section III-D. Live-Interactive's tight deadline and closed feedback loop demand that latency and deadline-satisfaction dominate its weighting (0.70 combined), even at the cost of resource efficiency; Background-Batch's lack of real-time urgency instead allows efficiency to dominate, since there is no live feedback loop to protect. The exact magnitudes are tuned rather than derived from a first-principles argument. This follows standard practice in multi-objective RL, where linear scalarisation is the most widely used approach because it avoids the added complexity of multi-policy methods [29], at the cost of only representing the convex portion of the trade-off space those objectives define. Section IV's behavioural validation checks that the trained policy's actual behaviour matches each mode's intended weighting.

H. Optimisation Objective

The state, feasible action set, and reward combine into the orchestration layer's optimisation objective:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=1}^T r_t \right], \quad \text{s.t. } a_t \in \mathcal{A}(s_t) \quad \forall t \quad (22)$$

This states the orchestration layer's objective in full to find the policy π , a mapping from the current state s_t to a choice of action in $\mathcal{A}(s_t)$, that maximises the total reward accumulated over one session's sequence of tasks, in expectation, while never selecting an action outside the feasible set at any step. Because $\mathcal{A}(s_t)$ already excludes any node that cannot process the task's modality or violates access rules, learning affects *which* feasible node is best given context, not whether a chosen node is permitted.

T is the number of decision steps in one bounded session. An undiscounted, finite-horizon return is used, since each episode is a short, bounded session rather than an ongoing process. Both the feasible action set and the reward's weighting change with every task, the former with modality eligibility and the random provisioning of node capability, the latter with context mode, making this a moving target rather than a fixed assignment solvable once at design time. Section IV describes the Double DQN agent trained to solve this objective under these shifting conditions.

IV. METHODOLOGY AND SIMULATION DESIGN

This section describes how the formulation in Section III is realised and evaluated. This includes the training algorithm used to solve the optimisation objective, the episode and simulation structure it is trained within, the baseline it is compared against, and the design of that comparison.

A. Training Algorithm

The orchestration layer is trained using Double DQN [8], with two departures from a generic DQN loop. Firstly, hard masking is applied before action selection, and then the same mask is reused when computing the target. The agent maintains two networks: an online network $Q(s, a; \theta)$, parameterised by weights θ , used both to select actions and to compute the greedy next action a^* ; and a target network $Q(s, a; \theta^-)$, a periodically-synced copy of θ used only to evaluate that action when forming the training target y . Separating selection (online network) from evaluation (target network) removes the value-overestimation a single network would otherwise accumulate. Transitions are stored in and sampled from a replay buffer $\mathcal{B}$. Algorithm 1 summarises the training procedure, where $\gamma \in [0, 1]$ is the discount factor applied to the target value.

Algorithm 1 Double DQN with Action Masking

```

1: Initialise online network  $Q(s, a; \theta)$ , target network  $Q(s, a; \theta^-) \leftarrow Q(s, a; \theta)$ 
2: Initialise replay buffer  $\mathcal{B}$ 
3: for each training episode do
4:   Reset environment: redraw  $\phi(n, \text{aud})$  (Eq. 3); reset queues and link states
5:   for each task arrival  $t$  do
6:     Observe state  $s_t$ 
7:      $\text{mode}(t) \leftarrow$  deterministic mode from  $\kappa(t)$ 
8:      $\mathcal{A}(s_t) \leftarrow$  feasible actions via hard masking
9:      $a_t \leftarrow \epsilon$ -greedy over  $Q(s_t, \cdot; \theta)$ , restricted to  $\mathcal{A}(s_t)$ 
10:    Execute  $a_t$ ; observe  $r_t$  and  $s_{t+1}$ 
11:    Store  $(s_t, a_t, r_t, s_{t+1}, \mathcal{A}(s_{t+1}))$  in  $\mathcal{B}$ 
12:    Sample mini-batch from  $\mathcal{B}$ 
13:     $a^* \leftarrow \arg \max_{a \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a; \theta)$   $\triangleright$  online net selects
14:     $y \leftarrow r_t + \gamma \cdot Q(s_{t+1}, a^*; \theta^-)$   $\triangleright$  target net evaluates
15:    Update  $\theta$  by gradient descent on  $(y - Q(s_t, a_t; \theta))^2$ 
16:    Periodically sync  $\theta^- \leftarrow \theta$ 
17:   end for
18: end for

```

Masking the target’s max-over-actions matters: without it, the network would be trained to value actions it was never permitted to take, corrupting the learned values for infeasible state-action pairs. For a finite-horizon episode return, γ is set to 1. The agent was trained with ϵ -greedy exploration, ϵ decayed exponentially from 1.0 to a 0.05 floor, over 1,200 episodes, using a standard multi-layer perceptron as the Q-network; hyperparameters were selected via a small sensitivity comparison on a held-out validation seed split, distinct from the seeds used for reported results.

B. Episode Structure

An episode mirrors one vocal-rehabilitation session, an active exercise phase with inter-repetition gaps, followed by session close and post-session reporting. Task instances are drawn from the six profiles (Section III-B) according to their representative triggers: Live-Interactive tasks during exercise repetitions, Ambient-Monitoring instances during inter-repetition gaps, and Background-Batch tasks once at session close. Episode length is set to 15–20 minutes of simulated session time.

An episode terminates at session close, after Background-Batch tasks are generated and processed, there is no early termination on task failure, since failures are penalised through reward rather than by ending the episode. At reset, queues, node loads, and link states are reinitialised. Audio provisioning $\phi(n, \text{aud})$ is redrawn per episode during training, so the agent does not overfit to a single hardware layout, but held fixed per topology seed during evaluation, so that any performance difference between methods reflects the policy rather than a favourable draw of which nodes support audio.

C. Simulation Setup

The IoM is simulated as a three-tier network of IoM nodes deployed over a 5×5 km urban area, matching the topology used in prior IoM evaluation [22], [23], so that results remain comparable with that wider body of work rather than

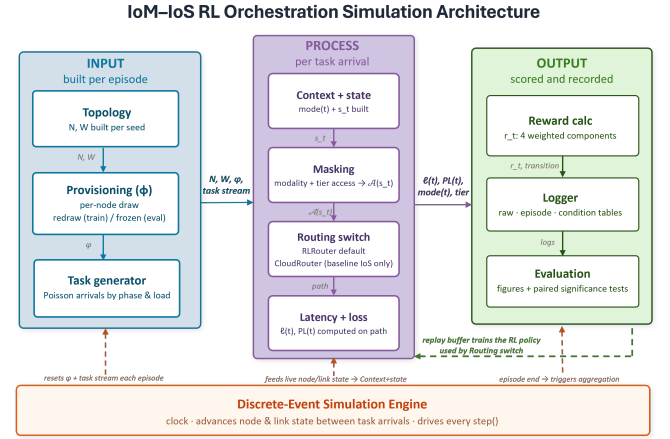


Fig. 2: Simulation architecture, showing the environment interface, constraint and reward modules, and the agent-training loop.

introducing a second, scenario-specific deployment scale. The home-based scenario in this paper is one entry point into this shared topology: the Consumer-tier mirror and co-deployed IoS speaker are not served by dedicated infrastructure of their own, but route into the same area wide Professional and Hub tiers as any other IoM deployment.

Fig. 2 shows the resulting simulation architecture, including the environment’s reset and step interface, the constraint and reward modules described in Section III, and the agent-training loop of Algorithm 1. The three stages map directly onto the episode structure above:

- INPUT is built once per episode reset: the topology, the provisioning draw $\phi(n, \text{aud})$, and the task-arrival stream for that session
- PROCESS runs once per task arrival, mirroring `step()`: state and context are assembled, the feasible action set is computed via hard masking, and the Routing Switch dispatches the task to the RL Router (the trained Double DQN agent, Algorithm 1) under the orchestrated condition, or to the Cloud Router under the baseline process.
- OUTPUT scores and logs the outcome of each task; during training, this transition is written to the replay buffer $\mathcal{B}$, which in turn trains the policy used by the Routing Switch, closing the learning loop shown at the bottom of the figure. A discrete-event simulation engine coordinates all three stages, owning the simulated clock and triggering the OUTPUT stage’s aggregation at each episode’s end.

D. Baseline

The comparison baseline is a cloud-offload configuration. IoS audio tasks are routed to a remote Google Cloud endpoint rather than into the IoM tier hierarchy, while native IoM visual tasks remain routed by the same trained agent in both conditions. This isolates the comparison to the decision this work is concerned with, whether to orchestrate IoS tasks through the IoM topology or offload them to the cloud.

The network latency segment of the cloud-offload path uses empirically measured round-trip latency from an actual Google

Cloud deployment accessed over Wi-Fi, following the same edge-cloud measurement methodology established in prior IoM work [21]. Both the orchestrated and cloud-offload paths are measured to the same user-delivery endpoint, and both face an identical task-arrival sequence within a given condition, since the task-generator seed is shared across methods rather than drawn independently, making this a paired comparison rather than two independent runs.

E. Evaluation Design

The evaluation proceeds in three stages. This is by first confirming the trained policy behaves as designed, then comparing it against the cloud-offload baseline, and finally stress-testing it on the axis with no baseline analogue.

Validation. Before any baseline comparison, the trained policy is checked against the behaviour the formulation intends by assessing the assigned-tier distribution by task profile, deadline-satisfaction and defer-rate by context mode as load increases.

Baseline Comparison Load in the form of task arrival rate is evaluated across low, medium, and high conditions, defined relative to the IoM topology’s empirically calibrated saturation point ($\lambda_{rel} = 30\%, 60\%, 85\%$) rather than as arbitrary task counts. Audio pipeline provisioning is held fixed at $p_{aud} = 1.0$ so this stage carries no modality-scarcity confound. This is the primary comparison against the cloud-offload baseline, using task completion rate, end-to-end latency, and packet-loss rate as key metrics, each additionally split by ecosystem origin.

Modality-Heterogeneity Sensitivity. Audio-provisioning probability p_{aud} is swept from 10% to 100% in 10% increments, with load held fixed, tracking overall, native-IoM, and IoS completion rate alongside defer rate as p_{aud} varies. This stage directly tests adaptation to data-modality heterogeneity.

V. RESULTS AND DISCUSSION

A. Validation

Before comparing the trained policy against the cloud-offload baseline, we first confirm that it behaves as intended.

Fig. 3 shows the assigned-tier distribution by task profile under the orchestrated condition at medium load, grouped by ecosystem origin. The two groups reveal the same underlying logic applied consistently across two very different task populations. Within native IoM tasks, routing tracks urgency. Session Summary Report, a background reporting task, lands on the Consumer tier the majority of the time, while the two live-interactive profiles, Expression Detection and Lip Movement Tracking, are pushed toward Professional and Hub nodes instead. Within IoS tasks, the pattern is starker, every audio-gated profile is entirely absent from the Consumer column, and each one clears Professional or Hub in over ninety-eight percent of instances. This is a stronger result than urgency-based routing alone, it demonstrates that the constraint and reward structure generalise across ecosystems without any ecosystem-specific routing rule. The policy did not need to be told that audio tasks behave differently from visual

ones, the modality-hardware constraint alone was sufficient to produce that separation, and the mode-conditioned reward alone was sufficient to produce urgency-appropriate placement within each ecosystem.

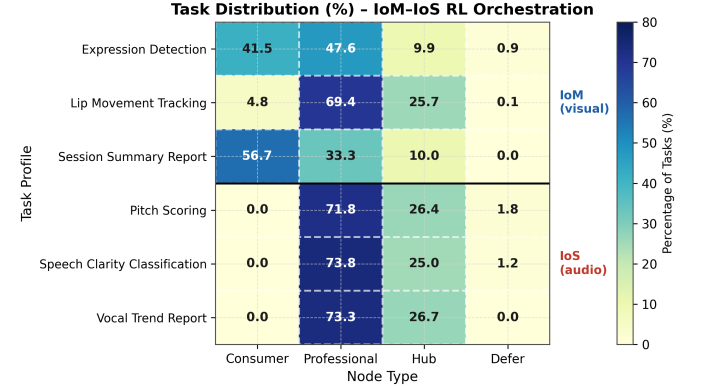


Fig. 3: Assigned-tier distribution by task profile under the orchestrated condition at medium load, grouped by ecosystem of origin.

The same separation holds at the level of outcomes. Background-Batch tasks never miss their deadline or defer, regardless of load. Ambient-Monitoring degrades only mildly. Live-Interactive is the only mode under real pressure, deadline satisfaction drops from 0.992 to 0.933 and defer rate rises more than fivefold between low and high load, confirming that the per-mode weighting translates into intended behaviour rather than remaining an untested prior e.g. LI’s combined 0.70 weight on latency and deadline is exactly the mode under strain, BB’s 0.80 weight on efficiency is exactly the mode that never strains. This sets the terms for what follows, a favourable baseline comparison is only meaningful once the policy has shown it treats tasks differently according to need.

B. Load vs. Baseline

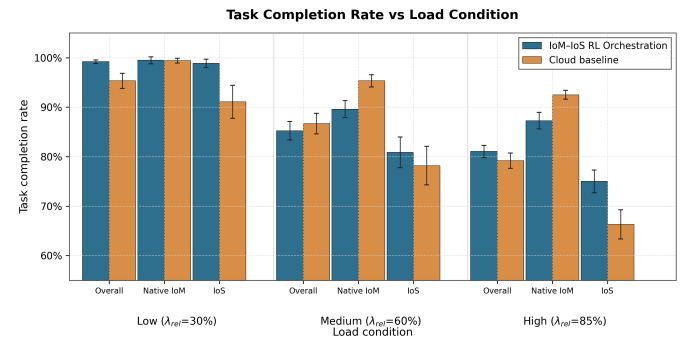


Fig. 4: Task completion rate versus load condition, orchestrated system compared against the cloud-offload baseline.

We now compare the trained policy against the cloud-offload baseline across low, medium, and high load, holding audio provisioning fixed so the comparison isolates load alone.

Fig. 4 reports completion rate split by ecosystem origin. At low load, orchestration achieves a higher completion rate (99.2% vs. 95.3%, $p < 0.001$), while native-IoM is statistically

Latency and Reliability for Critical Tasks vs Load Condition

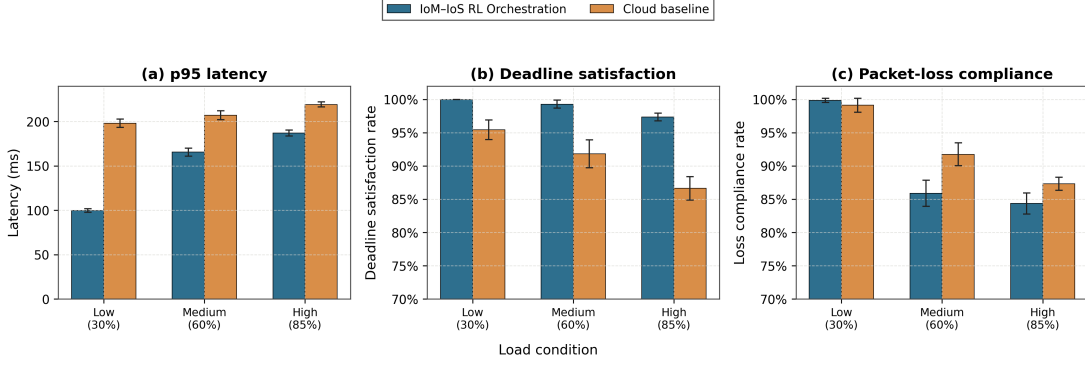


Fig. 5: Latency and reliability for Live-Interactive tasks versus load condition: p95 latency, deadline satisfaction, and packet-loss compliance.

indistinguishable between conditions (99.5% vs. 99.4%, $p = 0.815$), with no measurable native-IoM completion penalty at low load. As load rises, native-IoM completion becomes significantly worse under orchestration at both medium (89.6% vs. 95.3%, $p < 0.001$) and high load (87.3% vs. 92.5%, $p < 0.001$), since native tasks are routed identically in both conditions and the difference is congestion, not routing quality. IoS completion remains significantly higher under orchestration at every load (medium: +2.7 pp, $p = 0.018$; high: +8.7 pp, $p < 0.001$), the gap widening as load increases. Orchestration is buying IoS performance by spending native-task headroom, a trade worth making here since the vocal exercise IoS is scoring and the facial expression IoM is tracking are evidence of the same repetition, not two competing workloads.

This trade-off produces a pooled effect that is non-monotonic in load and worth stating explicitly, overall completion favours orchestration at low load (+3.9 pp), reverses in cloud’s favour at medium load (−1.4 pp, $p = 0.037$), then reverses back at high load (+1.9 pp, $p = 0.033$). This follows directly from the two component trends growing at different rates, between medium and high load the IoS advantage more than triples (2.7 to 8.7 pp) while the native disadvantage stays roughly flat (−5.7 to −5.2 pp), so once the IoS gain outweighs the native loss the pooled average swings back even though native performance has not recovered. The overall metric is a weighted composite of two ecosystems moving in opposite directions, not a trend in its own right.

Fig. 5 narrows the comparison to Live-Interactive tasks. Orchestration achieves lower p95 latency and higher deadline satisfaction than the cloud-offload baseline across all load conditions, with the deadline-satisfaction advantage becoming more pronounced as load increases. Packet-loss compliance follows the opposite trend: orchestration performs marginally better at low load but falls below the cloud baseline at medium and high load as contention increases within the shared IoM topology. These results highlight a trade-off between improved responsiveness and reduced packet-loss compliance under heavier load.

C. Modality-Heterogeneity Sensitivity

The final evaluation scales how many nodes can physically process audio, at fixed medium load, with no cloud-offload baseline, since a cloud path does not face a hardware-availability constraint of this kind.

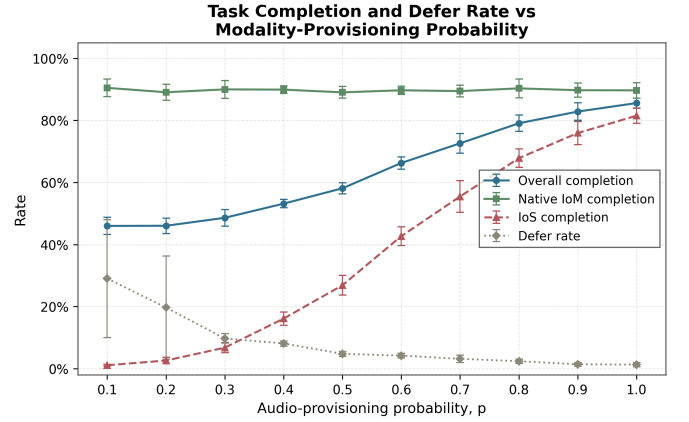


Fig. 6: Task completion and defer rate versus audio-provisioning probability, at fixed medium load.

Fig. 6 shows IoS completion rising monotonically from 1.1% at $p_{\text{aud}} = 0.1$ to 81.5% at $p_{\text{aud}} = 1.0$, while native-IoM completion stays flat throughout (89.0–90.3%) and overall completion rises correspondingly, from 46.0% to 85.6%. This is the modality-hardware constraint doing exactly what it was built to do: IoS tasks are gated by audio capability and native tasks are not, so scarcity on one side does not infringe on the other. Defer rate falls smoothly across the same range, from 29.1% at $p_{\text{aud}} = 0.1$ to 1.3% at $p_{\text{aud}} = 1.0$. A hard constraint could easily have produced a system that simply fails, or silently misroutes, whenever the hardware it depends on is scarce. Instead, the policy defers tasks it genuinely cannot place, and does so proportionally less often as the constraint binds less frequently, this is the reward treating a deferred task as an acceptable outcome rather than a failure, exactly as the mode-conditioned reward in Section III-G was designed to do.

One caveat applies at the scarce end of the sweep: defer-rate variance is high at $p_{\text{aud}} = 0.1$ (std 0.194 against a mean of

0.291), because provisioning at that point is bimodal across seeds, some topology draws end up with no audio-capable nodes at all, others with a few. This reflects two different operating regimes, not measurement noise, and is a property of the provisioning process rather than a weakness in the constraint itself.

VI. CONCLUSION

This paper formalised cross-ecosystem task orchestration as a problem of context, not just compute. Encoding that context as a hard action mask rather than a soft reward term proved to be the operative design choice, it is what let a single policy generalise across two structurally different task populations without any ecosystem-specific routing rule, and what let the system degrade by deferral rather than by failure as audio hardware grew scarce.

The cost of this benefit is visible in how orchestration improves IoS and Live-Interactive outcomes by drawing on native-IoM headroom, a trade that is favourable here because the two ecosystems are sensing the same underlying event, but would need re-examining in a setting where the co-deployed ecosystems serve unrelated workloads rather than complementary evidence of one process. The formulation's generality is correspondingly untested beyond the single scenario and single external ecosystem (IoS) evaluated here. Extending this framing to additional co-deployed ecosystems under the broader IoX vision, and validating it on physical IoM-IoS hardware rather than in simulation, are the natural next steps to establish how far the context-aware masking approach generalises beyond the vocal-rehabilitation case studied here.

REFERENCES

- [1] M. A. Iqbal, S. Hussain, H. Xing, and M. Imran, *Internet of Things (IoT) fundamentals*. Wiley Publishing, 2021, pp. 1–28.
- [2] H. Fatima, M. A. Imran, A. Taha, and L. Mohjazi, "Internet-of-Mirrors (IoM) for connected healthcare and beauty: A prospective vision," *Internet of Things*, vol. 28, p. 101415, Dec. 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2542660524003561>
- [3] H. Fatima, M. A. Alharbi, A. Taha, M. A. Imran, and L. Mohjazi, "Enabling Inclusive and Accessible Living Through the Internet of Mirrors (IoM)," *IEEE Internet of Things Magazine*, pp. 1–9, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/11059764>
- [4] L. Turchet, M. Lagrange, C. Rottondi, G. Fazekas, N. Peters, J. Østergaard, F. Font, T. Bäckström, and C. Fischione, "The Internet of Sounds: Convergent Trends, Insights, and Future Directions," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11 264–11 292, Jul. 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/10061604>
- [5] L. Turchet, G. Fazekas, M. Lagrange, H. S. Ghadikolaei, and C. Fischione, "The Internet of Audio Things: State of the Art, Vision, and Challenges," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 233–10 249, Oct. 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9099251>
- [6] X. Chen and others, "Deep Reinforcement Learning for Internet of Things: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1545–1580, 2021.
- [7] L. Lei and others, "Deep Reinforcement Learning for Autonomous Internet of Things: Model, Applications and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1721–1758, 2020.
- [8] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
- [9] A. Robles-Enciso and A. F. Skarmeta, "A multi-layer guided reinforcement learning-based tasks offloading in edge computing," *Computer Networks*, vol. 220, p. 109476, 2023.
- [10] Y. Chen, Y. Sun, C. Wang, and T. Taleb, "Dynamic Task Allocation and Service Migration in Edge-Cloud IoT System Based on Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 9, pp. 16 742–16 757, 2022.
- [11] C. Sun, X. Li, C. Wang, Q. He, X. Wang, and V. C. M. Leung, "Hierarchical Deep Reinforcement Learning for Joint Service Caching and Computation Offloading in Mobile Edge-Cloud Computing," *IEEE Transactions on Services Computing*, 2024.
- [12] P. Zhang and others, "Multi-Target-Aware Dynamic Resource Scheduling for Cloud-Fog-Edge Multi-Tier Computing Network," *IEEE Transactions on Intelligent Transportation Systems*, 2023.
- [13] S. Shahhosseini and others, "Online Learning for Orchestration of Inference in Multi-user End-edge-cloud Networks," *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 6, pp. 1–25, 2022.
- [14] K. Moghaddasi, S. Rajabi, and F. S. Gharehchopogh, "An advanced deep reinforcement learning algorithm for three-layer D2D-edge-cloud computing architecture for efficient task offloading in the Internet of Things," *Sustainable Computing: Informatics and Systems*, 2024.
- [15] T. Shi and others, "Multimodal Mobile Edge Computing: Multi-Objective Optimization With Synchronization Constraint," *IEEE*, 2025.
- [16] J. Liang, B. Ma, Z. Feng, and J. Huang, "Reliability-aware Task Processing and Offloading for Data-intensive Applications in Edge Computing," *IEEE Transactions on Network and Service Management*, 2023.
- [17] P. Liu, Z. Liu, J. Wang, Z. Wu, P. Li, and H. Lu, "Reinforcement learning empowered multi-AGV offloading scheduling in edge-cloud IIoT," *Journal of Cloud Computing*, vol. 11, no. 1, pp. 1–14, 2022.
- [18] M. R. Raju and S. K. Mothku, "Delay and energy aware task scheduling mechanism for fog-enabled IoT applications: A reinforcement learning approach," *Computer Networks*, vol. 224, p. 109603, 2023.
- [19] Y. Ma and others, "Multi-Agent Deep Reinforcement Learning for Joint Task Offloading and Resource Allocation in IIoT with Dynamic Priorities," *Sensors*, vol. 25, no. 19, p. 6160, 2025.
- [20] S. Shresthamali, M. Kondo, and H. Nakamura, "Multi-Objective Resource Scheduling for IoT Systems Using Reinforcement Learning," *Journal of Low Power Electronics and Applications*, vol. 12, no. 4, p. 53, 2022.
- [21] H. Fatima, M. A. Imran, L. Mohjazi, and A. Taha, "A Latency Analysis of Edge vs. Cloud Computing for Internet-of-Mirrors (IoM) Applications," in *2024 IEEE Middle East Conference on Communications and Networking (MECON)*, Nov. 2024, pp. 440–445. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/1088 2135>
- [22] H. Fatima, M. A. Imran, A. Taha, and L. Mohjazi, "Toward Real-Time Mirrors Intelligence: System-Level Latency and Computation Evaluation in Internet of Mirrors (IoM)," in *32nd International Conference on Telecommunications (ICT 2026)*, May 2026, thessaloniki, Greece, 20–22 May 2026 (Accepted for Publication). [Online]. Available: <https://eprints.gla.ac.uk/view/author/68145.html>
- [23] H. Fatima, L. Mohjazi, M. A. Imran, and A. Taha, "A Context-Aware Task Distribution (CATD) Framework for Internet of Mirrors (IoM) Applications," Feb. 2026. [Online]. Available: [https://www.authorea.com/users/1027514/articles/1387245-a-context-aware-task-distribution-c atd-framework-for-internet-of-mirrors-iom-applications](https://www.authorea.com/users/1027514/articles/1387245-a-context-aware-task-distribution-catd-framework-for-internet-of-mirrors-iom-applications)
- [24] L. Turchet and P. Casari, "Latency and reliability analysis of a 5g-enabled internet of musical things system," *IEEE Internet of Things Journal*, vol. 11, no. 1, pp. 1228–1240, 2024.
- [25] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.
- [26] P. M. Sørensen, B. Epp, and T. May, "A depthwise separable convolutional neural network for keyword spotting on an embedded system," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2020, no. 1, p. 10, 2020.
- [27] "Voice-assisted technology for people with parkinson's disease experiencing speech and voice difficulties: Co-designing solutions using design thinking," *JMIR Rehabilitation and Assistive Technologies*, vol. 13, p. e84364, 2026.
- [28] ITU-T, "One-way transmission time," International Telecommunication Union, Tech. Rep. ITU-T Recommendation G.114, 2003.
- [29] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.