Asynchronous Risk-Aware Multi-Agent Packet Routing for Ultra-Dense LEO Satellite Networks

Ke He[†§],, Thang X. Vu[†§],, Le He[‡], Lisheng Fan[‡], Symeon Chatzinotas[†] and Björn Ottersten[†]
[†]The Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg.
[‡]School of Computer Sciense and Cyber Engineering, Guangzhou University, Guangzhou, China.
[§]Corresponding authors: Ke He and Thang X. Vu ({ke.he, thang.vu}@uni.lu)

***Abstract*—The rise of ultra-dense LEO constellations creates a complex and asynchronous network environment, driven by their massive scale, dynamic topologies, and significant delays. This unique complexity demands an adaptive packet routing algorithm that is asynchronous, risk-aware, and capable of balancing diverse and often conflicting QoS objectives in a decentralized manner. However, existing methods fail to address this need, as they typically rely on impractical synchronous decision-making and/or risk-oblivious approaches. To tackle this gap, we introduce `PRIMAL`, an event-driven multi-agent routing framework designed specifically to allow each satellite to act independently on its own event-driven timeline, while managing the risk of worst-case performance degradation via a principled primal-dual approach. This is achieved by enabling agents to learn the full cost distribution of the targeted QoS objectives and constrain tail-end risks. Extensive simulations on a LEO constellation with 1584 satellites validate its superiority in effectively optimizing latency and balancing load. Compared to a recent risk-oblivious baseline, it reduces queuing delay by over 70%, and achieves a nearly 12 ms end-to-end delay reduction in loaded scenarios. This is accomplished by resolving the core conflict between naive shortest-path finding and congestion avoidance, highlighting such autonomous risk-awareness as a key to robust routing.**

***Index Terms*—Asynchronous communication, multi-agent system, satellite network, network routing, distributional reinforcement learning, risk-aware**

I. Introduction

The rapid development of Low Earth Orbit (LEO) satellite mega-constellations is driving a new era of global connectivity [1]–[3]. These networks construct a dynamic space-based backbone of thousands of satellites interconnected by Inter-Satellite Links (ISLs) at altitudes of 500 to 2000 kilometers. Commercial constellations from providers like Starlink, OneWeb, Kuiper, Qianfan and Telesat are now being actively deployed with the ultimate goal of providing ubiquitous high-bandwidth and low-latency internet access to every corner of the globe [4]. In order to realize this vision, a core challenge is designing a robust and adaptive packet routing mechanism for this dynamic infrastructure [3].

However, designing such an algorithm is challenging due to the massive scale, dynamic topology, and significant propagation delays inherent in LEO networks [3], [5], [6]. These characteristics make centralized control with a timely global network view impractical [1], [7]. Consequently, an effective routing algorithm must be decentralized, allowing each satellite to operate *asynchronously* and *independently* using only local information. Furthermore, imbalanced global traffic distribution causes unpredictable congestion, demanding a mechanism for *asynchronous risk-aware packet routing* [2], [3], [5]. Such a mechanism should be able to manage conflicting Quality of Service (QoS) objectives, like latency minimization and load balancing, in a decentralized manner. Addressing this need is the main focus of our work.

Related Works & Limitations. Prior work has explored both traditional and learning-based strategies [2], [8]. Traditional rule-based methods often use static topology snapshots or geographic principles to pre-calculate routes [9]–[12]. While some approaches avoid full global knowledge [13], they are fundamentally *risk-oblivious* and fail to handle dynamic events like traffic congestion [14]. To address these limitations, data-driven Deep Reinforcement Learning (DRL) has emerged as a promising direction [15]–[20]. While early single-agent RL frameworks exist [15], they suffer from scalability bottlenecks, shifting the focus to decentralized Multi-Agent Reinforcement Learning (MARL) where each satellite acts based on its local observation [16], [17], [19], [21].

However, many existing MARL methods are often misaligned with the physical reality of LEO satellite networks. Approaches based on cooperative MARL paradigms like MAPPO [17], [19] typically enforce action-synchronization by discretizing time into synchronous time-slots, where agents are constrained to make at most one decision per time slot. While individual satellites can achieve high-precision physical clock synchronization, this does not resolve the impracticality of this time-slotted decision paradigm. The core issue is that forcing the naturally event-driven packet routing decision process (i.e., asynchronous packet arrivals or departures) into a rigid time-stepped joint-action model introduces artificial delays and severe scalability bottlenecks, as all agents must wait for a "global tick" before acting [7], [22]. Motivated by this point, asynchronous MARL approaches such as continual DRL with Federated Learning (FedL) and asynchrnous QMIX were proposed to address the issue [20], [23]. However, the absence of synchronized cooperation of all agents can lead to conflicting decisions among agents. This highlights the need for a framework that can manage the risks arising from uncoordinated decentralized actions made by independent agents.

Existing attempts at risk-awareness often rely on heuristic reward shaping. This approach incorporates risk-awareness by engineering complex weighted reward functions that try to

balance objectives such as energy budgeting, latency minimization and load balancing [15], [19], [24], [25]. However, such methods lack formal guarantees, and more critically, they require extensive human-effort on trial-and-error based adjusting [26]. A more principled approach is Constrained Reinforcement Learning (CRL). However, recent CRL-based approaches for satellite packet routing like [17], while using a primal-dual method, were risk-myopic to constraining only the average values (neglecting tail-end risks) and relied on centralized coordinators [27], reintroducing the aforementioned scalability and synchronization problems.

Motivations & Contributions. Our work is motivated by the two critical gaps in existing research. First, the dominant synchronous paradigm in many MARL routing algorithms is poorly suited to LEO networks. They rely on cooperative MARL frameworks that require a centralized coordinator and action-synchronization across all LEOs, which contradicts the inherently asynchronous and event-driven nature of packet routing [22], [28]–[30]. Independent MARL approaches, such as [20], remove the need for coordination, but they risk performance degradation as agents may repeatedly interfere with each other's policies. This makes it more difficult to manage diverse and conflicting QoS objectives, reinforcing the need for an asynchronous risk-aware MARL framework.

Second, existing strategies lack robust risk awareness. Many are *risk-oblivious* [15], [19], [25], relying on heuristic reward shaping that lacks formal guarantees and significant human-efforts on coefficient adjusting [26]. Others are *risk-myopic* [17], optimizing only for average performance while ignoring high-impact tail-end events like sudden latency spikes [31].

To address these limitations, we propose **`PRIMAL`**, a novel *asynchronous* and *risk-aware* MARL framework. **`PRIMAL`** uses an event-driven design that allows each satellite to act asynchronously based on their local information. Our method includes a principled primal-dual approach which learns the full distribution of the interested risk metrics, and directly constrains worst-case performance risks via distributional reinforcement learning. To summarize this work, our core contributions are:

- We formulate the packet routing problem in satellite networks as an asynchronous MARL problem based on an **event-driven semi-Markov decision process**. Our model operates in continuous time, where each satellite agent acts independently and asynchronously based on its local event-driven timeline and observations. This approach eliminates the unrealistic synchronization and discrete time-step assumptions of prior RL-based routing methods, leading to a more realistic and efficient routing paradigm.
- We propose a **principled risk-aware routing algorithm using a distributional primal-dual framework**. Instead of learning only the expected costs, our agents learn the full conditional distribution of routing outcomes via quantile regression. By directly constraining the Conditional Value-at-Risk (CVaR) at a given risk level, our algorithm can effectively mitigate tail-end risks, ensuring the routing policy is robust against worst-case performance degradation.

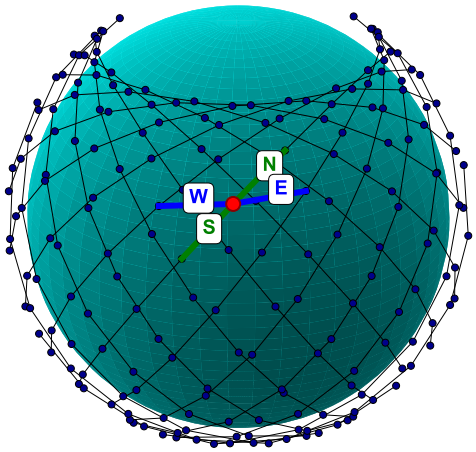

Fig. 1. Illustration of a Walker-Delta LEO constellation with grid topology, where each satellite connects two intra-plane neighbors (N+S) and two inter-plane neighbors (W+E). Note that we define the four directions w.r.t. the rotating direction of the orbit.

- We develop a **decentralized and synchronization-free scalable learning architecture** without reliance on a centralized coordinator that needs action-synchronization and global state information. This enables extensions such as online federated learning [20] that facilitates scalable and practical online adaption in real-world mega-constellations.

II. System Model & Problem Formulation

A. Network Model

As illustrated in Fig. 1, we model the LEO satellite network as a time-varying directed graph $\mathcal{G}_t = (\mathcal{N}, \mathcal{E}(t))$, where the node set $\mathcal{N}$ consists of satellites $\mathcal{N}_s$ and ground stations $\mathcal{N}_g$.

1) Nodes: The $|\mathcal{N}_s|$ satellites are arranged in a Walker constellation, forming a grid-like topology with Inter-Satellite Links (ISLs) [3]. Each satellite acts as a store-and-forward router with finite buffers. The $|\mathcal{N}_g|$ ground stations act as traffic entry/exit points, connecting to the network via Ground-to-Satellite Links (GSLs) with their respective *access satellites*.

2) Links, Packets, and Queues: The edge set $\mathcal{E}(t)$ includes dynamic ISLs with Free Space Optical (FSO) Lasers, and GSLs with Ka-Band radio frequency (RF) links, whose availability depends on line-of-sight (LoS) and minimum elevation angles [32]. Data is transmitted as packets, each defined by a tuple $p \triangleq \{s_p, d_p, L_p, \tau_p, \tau_p^{ttl}\} \in \mathcal{P}$, representing its source, destination, size, creation time, and a Time-to-Live (TTL) field initialized to the maximum TTL H, and $\mathcal{P}$ denotes the set of packets. Packets are processed in a First-In-First-Out (FIFO) manner and are dropped if TTL expires or if buffers are full.

B. Communication and Delay Model

The end-to-end (E2E) delay for a packet traversing the network is the sum of propagation, transmission, and queuing delays at each hop. To model these delays, we begin by defining a single, hop-indexed binary decision variable $x_{p,ij}^h = 1$ if packet $p \in \mathcal{P}$ traverses the link (i, j) as its h-th hop in its path and 0 otherwise, where $h \in \{0, 1, ..., H-1\}$. The delay

at hop h depends on the packet's arrival time at the starting node of the hop, and the arrival time is determined by the sum of delays from all previous hops (0 to $h-1$). For a packet p traversing a link (i,j) at time t, the single-hop delay is

$$D^h_{p,ij} = D^P_{ij}(\tau^h_p) + D^T_{ij}(L_p, \tau^h_p) + D^Q_{ij}(\tau^h_p), \tag{1}$$

where $D^P_{ij}(t)$ is the propagation delay, $D^T_{ij}(L_p,t)$ is the transmission delay, and $D^Q_{ij}(t)$ is the queuing delay of the sending node i at arrival time $t=\tau^h_p$. The arrival time τ^h_p at the start of hop h can be recursively defined as

$$\tau^h_p = \tau_p + \sum_{k=0}^{h-1} \sum_{(u,v)\in\mathcal{E}(\tau_{p,k})} x^p_{uv,k} \cdot D^p_{uv,k} \tag{2}$$

Now we are ready to introduce the models of each delay component.

1) Propagation Delay: The propagation delay $D^P_{ij}(t)$ is the time needed for a signal to travel from node i to node j via link (i,j) at time t. It is determined by the Euclidean distance $d_{ij}(t)$ between the nodes at time t, as well as the speed of light κ_c, which is given by

$$D^P_{ij}(t) = \frac{d_{ij}(t)}{\kappa_c}. \tag{3}$$

2) Transmission Delay: The transmission delay $D^T_{ij}(L_p,t)$ is the time required to send all the L_p bits of packet p onto communication link (i,j) at time t. It is a function of the packet size L_p and the link's achievable data rate $R_{ij}(t)$. For GSLs operating in the Ka-Band, the data rate is modeled by the Shannon-Hartley theorem, which depends on the link's bandwidth B_{ij} and its Signal-to-Noise Ratio (SNR):

$$R^{GSL}_{ij}(t) = B_{ij}\log_2(1+\text{SNR}_{ij}(t)). \tag{4}$$

The $\text{SNR}_{ij}(t)$ is determined by [19]:

$$\text{SNR}_{ij}(t) = \frac{P^T_{ij}G^T_{ij}G^R_{ij}}{L_{ij}(t)\kappa_B T_{ij} B_{ij}}, \tag{5}$$

where P^T_{ij} is the antenna transmission power, G^T_{ij} and G^R_{ij} are the transmitter and receiver antenna gains, κ_B is the Boltzmann's constant, L_{ij} is the Free Space Path Loss (FSPL) and T_{ij} is the system noise temperature in Kelvin. L_{ij} is a function of the distance $d_{ij}(t)$ and carrier frequency of Ka-Band f_c:

$$L_{ij}(t) = \left(\frac{4\pi d_{ij}(t) f_c}{\kappa_c}\right)^2. \tag{6}$$

For FSO Laser based ISLs, which rely on FSO communication, the data rate is modeled differently [13]:

$$R^{ISL}_{ij}(t) = \frac{\tilde{B}_{ij}}{2}\log_2\left(1+\kappa_1\cdot e^{-\kappa_2\cdot d_{ij}(t)}\right), \tag{7}$$

where $\tilde{B}_{ij}$ is the optical bandwidth. The parameters κ_1 and κ_2 are related to the average optical SNR and attenuation conditions [33], [34]. Consequently, the transmission delay for a packet p of size L_p over link (i,j) is given by

$$D^T_{ij}(L_p,t) = \frac{L_p}{R_{ij}(t)}, \tag{8}$$

where $R_{ij}(t)$ is either $R^{GSL}_{ij}(t)$ or $R^{ISL}_{ij}(t)$ depending on the link type.

3) Queuing Delay: The queuing delay $D^Q_{ij}(t)$ is the time a packet spends waiting in an output buffer before its transmission begins. In a FIFO output queue, this delay is the sum of the transmission delays of all preceding packets in the queue for the same outgoing link. If packet p arrives at node i at time t and is routed to next-hop j, its queuing delay can be approximately calculated as

$$D^Q_{ij}(t) = \sum_{q\in\mathcal{P}_{ij}(t)} \frac{L_q}{R_{ij}(t)}, \tag{9}$$

where $\mathcal{P}_{ij}(t)$ is the set of packets already in the output queue for link (i,j) at the time t. The queuing delay is a direct indicator of local congestion, as a congested node inherently results in longer packet queue. Consequently, the accumulated queuing delay of a packet's journey can serve as an ideal metric for evaluating the network's load balancing performance.

C. Problem Formulation

The packet routing problem can be formulated as a large-scale non-linear integer programming problem. The main objective is to find an optimal routing policy, defined by the set of $|\mathcal{P}|\times|\mathcal{E}|\times H$ decision variables $\{x^h_{p,ij}\}$, that minimizes the total E2E delay of each packet

$$D_p = \sum_{h=0}^{H-1} \sum_{(i,j)\in\mathcal{E}(\tau^h_p)} x^h_{p,ij}\cdot D^h_{p,ij}, \tag{10}$$

while subjects to several fundamental constraints that define valid routing paths.

First, a valid path must be contiguous. The destination node of any given hop must serve as the source node for the subsequent hop. This path connectivity constraint is enforced for all non-terminal nodes:

$$\sum_{i\in\mathcal{N}_k} x^h_{ik,p} = \sum_{j\in\mathcal{N}_k} x^{h+1}_{kj,p} \quad \forall p\in\mathcal{P}, k\in\mathcal{S}, h<H-1. \tag{C1}$$

In addition, each packet's journey must start at its source ground station s_p and eventually terminate at its destination ground station d_p. The source and destination constraints for packet delivery are

$$\sum_{j\in\mathcal{N}_{s_p}} x^1_{s_p,j,p} = 1 \quad \text{and} \quad \sum_{h=0}^{H-1}\sum_{i\in\mathcal{N}_{d_p}} x^h_{i,d_p,p} = 1 \quad \forall p\in\mathcal{P}. \tag{C2}$$

Moreover, to ensure that the path is simple and loop-free within a hop index, a packet can traverse at most one link for any given hop h:

$$\sum_{(i,j)\in\mathcal{E}(\tau^h_p)} x^h_{p,ij} \le 1 \quad \forall p\in\mathcal{P}, h\in\{0,1,...,H-1\} \tag{C3}$$

To manage congestion, we add a constraint on the maximum accumulated queuing delay:

$$D_p^Q = \sum_{h,(i,j)} x_{p,ij}^h \cdot D_{ij}^Q(\tau_p^h) \leq D_{max}^Q, \quad \forall p \in \mathcal{P} \tag{C4}$$

where D_{max}^Q is a predefined threshold. The full problem is:

$$\textbf{P1:} \min_{\{x_{p,ij}^h\}} \quad \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} D_p \quad \text{s.t. (C1), (C2), (C3), (C4)} \tag{11}$$

This problem is intractable for decentralized solutions due to its non-linear and interdependent nature (e.g., queuing delay couples all routing decisions) and the massive scale of the network, which makes centralized solvers infeasible [35]. This motivates our decentralized and scalable learning framework.

III. Principled Risk-Aware Independent Multi-Agent Learning

To overcome the limitations of synchronized MARL models discussed earlier [17], [19], we adopt an asynchronous event-driven perspective. We model the trajectory of a single packet as a *Partially-Observed Constrained Semi-Markov Decision Process* (POCSMDP), where satellite routers are independent decision-makers. This packet-centric view defines a finite-horizon learning episode for each packet's journey.

A. Event-Driven Semi-Markov Decision Process

The POCSMDP for a packet p is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \{c_k\}, \mathcal{O}, H, \gamma_r, \gamma_c \rangle_p$, where:

- $\mathcal{S}$ is the global network state space. A state $s_h \in \mathcal{S}$ is a snapshot of the network's physical status and packet-specific information at the h-th hop.
- $\mathcal{A}$ is the packet routing action space at a satellite, i.e., the set of four outgoing ISLs (NSWE).
- $\mathcal{T}(s', \tau, |s, a)$ is the transition probability to state s' after a variable duration τ (the single-hop delay), making this a *semi-Markov* process.
- $\mathcal{O}$ is the observation function yielding a local observation $o \sim \mathcal{O}(\cdot|s)$, which includes packet state and local node/neighbor statistics, making the process *partially observable*.
- $r(o, a, o')$ and $\{c_k(o, a, o')\}_{k=1}^K$ are the reward (primary objective) and QoS cost functions (e.g., load balancing) for a given state transition.
- $\gamma_r, \gamma_c \in (0, 1]$ are discount factors.

This event-driven formulation ensures that satellites react to packet arrivals in real-time based on current local information. While each packet defines a conceptual learning episode, the physical agents are the satellites. For scalability, we employ parameter sharing, where all satellites use a single, homogeneous policy $\pi_\theta(a|o)$, stored locally but shared across the network, enabling distributed and asynchronous execution.

B. Maximum Entropy Constrained Reinforcement Learning

Following the event-driven POCSMDP framework, we can solve the routing problem **P1** via CRL [36]. Since each satellite acts independently without coordination, maintaining a certain level of policy stochasticity is beneficial for exploration and for mitigating the non-stationary environment issue arising from the concurrent learning of other agents. This can be achieved by incorporating an additional constraint on the expected policy entropy during learning, known as maximum entropy RL [37]. For each packet p, we are interested in the two variables, the reward-return $Z_\pi^r = \sum_{h=0}^{H-1} \gamma_r^h r(o_h, a_h, o_{h+1})$, and the cost-return $Z_\pi^{c_k} = \sum_{h=0}^{H-1} \gamma_c^h c_k(o_h, a_h, o_{h+1})$, all induced by a fixed policy π. The overall objective is to find the optimal policy that solves the following constrained optimization problem:

$$\textbf{P2:} \quad \max_\pi \quad \mathcal{J}_r(\pi) = \mathbb{E}\left[Z_\pi^r\right] \tag{12a}$$

$$\text{s.t.} \quad \mathcal{J}_{c_k}(\pi) \leq D_k, \quad \forall k \in \{1, \ldots, K\} \tag{12b}$$

$$\mathcal{H}\left(\pi(\cdot|o_h)\right) \geq \bar{\mathcal{H}}, \quad \forall h \in \{1, \ldots, H\} \tag{12c}$$

where $\mathcal{H}\left(\pi(\cdot|o_h)\right) = \mathbb{E}_{a \sim \pi(\cdot|o)}\left[-\log \pi(a|o)\right]$ denotes the entropy of the policy, and $\bar{\mathcal{H}}$ is the desired minimum expected policy entropy. The term $\mathcal{J}_{C_k}(\pi) \leq D_k$ represents a placeholder for the k-th QoS constraint, which can be a constraint on the expected cost-return as

$$\mathcal{J}_{c_k}(\pi) = \mathbb{E}\left[Z_\pi^{c_k}\right] \leq D_k, \tag{13}$$

where D_k is the desired cost threshold.

In addition to constraining the expected costs, we further consider risk-averse probabilistic constraints based on Conditional Value-at-Risk (CVaR). Specifically, the Value-at-Risk (VaR) at risk level $\epsilon_k \in (0, 1)$ is the $(1 - \epsilon_k)$-quantile of the cost distribution defined as

$$\mathrm{VaR}_{\epsilon_k}(Z_\pi^{c_k}) = \inf\{z \in \mathbb{R} : F_{Z_\pi^{c_k}}(z) \geq 1 - \epsilon_k\}, \tag{14}$$

where $F_{Z_\pi^{c_k}}$ is the Cumulative Distribution Function (CDF) of $Z_\pi^{c_k}$. The CVaR is then defined as the expected cost in the worst ϵ_k tail of the distribution [31]:

$$\mathrm{CVaR}_{\epsilon_k}(Z_\pi^{c_k}) = \mathbb{E}_\pi\left[Z_\pi^{c_k} \middle| Z_\pi^{c_k} \geq \mathrm{VaR}_{\epsilon_k}(Z_\pi^{c_k})\right], \tag{15}$$

Then, $\mathcal{J}_{c_k}(\pi) \leq D_k$ can be a risk-averse constraint as

$$\mathcal{J}_{c_k}(\pi) = \mathrm{CVaR}_{\epsilon_k}\left(Z_\pi^{c_k}\right) \leq D_k, \tag{16}$$

which requires that the expected cost-return $Z_\pi^{c_k}$, conditioned on being in the worst ϵ_k-percentile of outcomes, does not exceed the threshold D_k. This allows for direct control over worst-case scenarios, moving beyond simple average performance.

Note that traditional methods for solving **P2** often rely on reward engineering, where a hand-crafted reward function is designed as a weighted sum of the main objective $\mathcal{J}_r$ and the cost components $\mathcal{J}_{c_k}$. This approach relaxes the constraints by incorporating them as penalties into the objective function, using a set of fixed coefficients. Though effective, it requires

manually adjusting these coefficients to balance multiple, often conflicting objectives, which is known to be notoriously challenging in practice [26].

In contrast, primal-dual learning introduces a more principled alternative that directly solves the CRL problem by learning the best multipliers for the constraints [31], [38], [39]. Being reward-agnostic and requiring no prior knowledge, the approach can handle a wide range of general constraints, which eliminates the significant human-efforts required on adjusting the penalty coefficients.

Hence, we propose the **P**rincipled **R**isk-aware **I**ndependent **M**ulti-**A**gent **L**earning (**PRIMAL**) framework to solve **P2** via primal-dual learning. Our approach first transforms the constrained problem into an unconstrained one via Lagrange multipliers. We then solve the resulting Lagrangian dual problem by extending the Soft Actor-Critic (SAC) algorithm to our multi-agent setting with discrete actions [37], [40]. We now introduce two variants of our algorithm: **PRIMAL-Avg**, which handles constraints on expected costs, and **PRIMAL-CVaR**, which addresses constraints on worst-case costs. Both the two variants work asynchronously and independently with only limited local information, ensuring scalability and robustness in large-scale systems.

C. PRIMAL-Avg: Routing with Expected Cost Constraints

To solve the constrained optimization problem **P2** with expectation constraints, we define the Lagrangian as:

$$\mathcal{L}(\pi, \boldsymbol{\lambda}, \alpha) = \mathcal{J}_r(\pi) + \alpha \left(\mathbb{E}_\pi[\mathcal{H}(\pi)] - \bar{\mathcal{H}}\right) - \sum_{k=1}^{K} \lambda_k \left(\mathcal{J}_{c_k}(\pi) - D_k\right), \tag{17}$$

where $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_K\}$ with $\lambda_k \geq 0$ and $\alpha \geq 0$ are the Lagrange multipliers for the K cost constraints and entropy constraint, respectively. This Lagrangian function combines the original objective with the constraints into a single equation. The core idea of the primal-dual method is to transform the original constrained problem, known as the primal problem, into an equivalent dual problem. We first define the dual function $g(\lambda, \alpha) = \max_\pi \min_{\boldsymbol{\lambda}\geq \mathbf{0}, \alpha \geq 0} \mathcal{L}(\pi, \boldsymbol{\lambda}, \alpha)$ as the maximum value of the Lagrangian with respect to the policy π for a fixed set of multipliers. The dual problem then involves finding the multipliers that *minimize* the dual function, i.e., $\min_{\lambda\geq 0, \alpha \geq 0} g(\lambda, \alpha)$. Note that primal-dual CRL has been proved to have strong duality for single-agent fully observable RL [26]. However, the constrained and partially observable MARL problem of our case is fundamentally more challenging and highly nonconvex, such that there exist none known strong duality guarantees [41]. Despite this fact, primal-dual approach still serves as a feasible and principled way to solve the problem approximately.

We approach this using an actor-critic framework with function approximation, i.e., neural networks, for the policy (actor) and the state-action value functions (Q-functions, as the critics). In the maximum entropy framework, the soft reward critic Q^r_ϕ, parameterized by ϕ, is defined as the expected sum of future rewards and entropy bonuses after taking action a in observation o and then following policy π_θ thereafter:

$$Q^r_\phi(o, a) = \mathbb{E}_{\pi_\theta}\Bigg[\sum_{h=0}^{H-1} \gamma_r^h r(o_h, a_h, o_{h+1}) + \alpha \mathcal{H}(\pi(\cdot|o_h)) \Big| o_0 = o, a_0 = a \Bigg], \tag{18}$$

Analogously, we have the k-th cost critic $Q^c_{\psi_k}$ as

$$Q^c_{\psi_k}(o, a) = \mathop{\mathbb{E}}_{\pi_\theta}\Bigg[\sum_{h=0}^{H-1} \gamma_c^h c_k(o_h, a_h, o_{h+1}) \Big| o_0 = o, a_0 = a \Bigg], \tag{19}$$

with parameters ψ_k. Then, we have the recursive Bellman equations as

$$Q^r_\phi(o, a) = r(o, a, o') + \gamma_r \mathbb{E}_{a' \sim \pi_\theta(\cdot|o')} \left[Q^r_\phi(o', a') - \alpha \log \pi(a'|o') \right], \tag{20}$$

$$Q^c_{\psi_k}(o, a) = c_k(o, a, o') + \gamma_c \mathbb{E}_{a' \sim \pi_\theta(\cdot|o')} \left[Q^c_{\psi_k}(o', a') \right] \tag{21}$$

For discrete actions, we use $Q^r_\phi(o) \in \mathbb{R}^{|\mathcal{A}|\times 1}$, $Q^c_\psi(o) \in \mathbb{R}^{|\mathcal{A}|\times 1}$ and $\pi_\theta(o) \in \mathbb{R}^{|\mathcal{A}|\times 1}$ to denote the per-action outputs. The reward and cost critics can be learned by comparing their predictions to a Temporal Difference (TD) target calculated from the experience $(o, a, r, \{c_k\}, o') \sim \mathcal{D}$ drawn from a replay buffer $\mathcal{D}$. Formally, the TD targets can be computed following the Bellman equations as

$$y^r_{\phi'} = r + \gamma_r \pi_\theta^\top(o') \left(Q^r_{\phi'}(o') - \alpha \log \pi(o') \right), \tag{22}$$

$$y^c_{\psi'_k} = c_k + \gamma_c \pi_\theta^\top(o') Q^c_{\psi'_k}(o'), \tag{23}$$

where $(\cdot)^\top$ denotes the matrix transpose. In practice, these targets are estimated via the corresponding target networks $Q^r_{\phi'}$ and $Q^c_{\psi'_k}$ with mirrored parameters ϕ' and $\{\psi'_k\}_{k=1}^K$. The SAC framework optimizes the reward and cost critics according to their TD errors as

$$\mathcal{L}_\phi = \mathbb{E}_{(o,a,r,o')\sim\mathcal{D}} \left[\frac{1}{2} \left(Q^r_\phi(o, a) - y^r_{\phi'} \right)^2 \right], \tag{24}$$

$$\mathcal{L}_{\psi_k} = \mathbb{E}_{(o,a,c_k,o')\sim\mathcal{D}} \left[\frac{1}{2} \left(Q^c_{\psi_k}(o, a) - y^c_{\psi'_k} \right)^2 \right], \tag{25}$$

For the policy update, the policy parameters θ are optmized by minimizing the KL-divergence between the policy and a target distribution derived from the Q-function. Specifically, the policy is updated towards the exponential of the Lagrangian-based state-action values [40]. The objective for the actor is to minimize the following loss function [37]:

$$\mathcal{L}_\theta = \mathop{\mathbb{E}}_{o\sim\mathcal{D}} \left[\pi_\theta^\top(o) \big(\alpha \log \pi_\theta(o) - Q^r_\phi(o) + \sum_k \lambda_k Q^c_{\psi_k}(o) \big) \right] \tag{26}$$

This update encourages the policy to select actions that have high reward Q-values and low cost Q-values (weighted by the Lagrange multipliers λ_k), while also maintaining high entropy to facilitate exploration.

The Lagrange multipliers, which are crucial for enforcing the constraints, are updated to minimize the Lagrangian. This results in simple update rules where each multiplier is adjusted based on the extent of its corresponding constraint violation.

$$\mathcal{L}_{\lambda_k} = \mathop{\mathbb{E}}_{o\sim\mathcal{D}}\left[\lambda_k\left(\pi_\theta^\top(o)Q^c_{\psi_k}(o) - D_k\right)\right], \tag{27}$$

$$\mathcal{L}_{\alpha} = \mathop{\mathbb{E}}_{o\sim\mathcal{D}}\left[\alpha\left(-\pi_\theta^\top(o)\log\pi_\theta(o) - \bar{\mathcal{H}}\right)\right]. \tag{28}$$

The update for λ_k in (27) increases the multiplier if the estimated cost exceeds the threshold D_k, thereby strengthening the penalty on costly actions in the actor's objective. Conversely, it decreases if the constraint is satisfied. Similarly, the entropy multiplier α is adjusted in (28) to ensure the policy's entropy remains close to the target level $\bar{\mathcal{H}}$.

D. PRIMAL-CVaR: Routing with Worst-Case Cost Constraints

While **`PRIMAL-Avg`** ensures that QoS constraints are met on average, it remains oblivious to low-probability but high-impact events, such as sudden latency spikes that cause cascading network congestion. For mission-critical infrastructure like LEO networks, such tail-end risks are unacceptable. To address this, we introduce **`PRIMAL-CVaR`**, a variant that directly controls the tail-end risk of the cost distribution by satisfying the CVaR constraints defined in (16). This requires moving beyond learning the mere expectation of the cost-return.

To be aware of the worst-case cost values, the critic should be able to learn the full conditional distribution of the cost-return $Z_\pi^{c_k}$. We achieve this using a powerful distributional RL method known as Implicit Quantile Networks (IQN) [39]. Unlike traditional methods that learn the expected value (i.e., the mean) of a return, IQN aims to capture its full distribution. Its core idea is to approximate the quantile function (the inverse of the CDF) by learning a mapping from a probability $\zeta \in [0,1]$ to the corresponding return value. This enables the network to implicitly model the entire distribution by being able to estimate any of its quantiles.

Specifically, we adapt IQN for discrete multi-agent SAC. This is realized by a network, $Q^c_{\psi_k}(o,a,\zeta)$, which takes a sample $\zeta \sim U(0,1)$ as an additional input to generate a value for that specific quantile. This technique provides a far richer and more accurate representation of the cost-return distribution, forming a solid foundation for risk-aware control [36], [38]. The IQN-based cost critic is trained by minimizing the quantile regression loss, guided by the distributional Bellman equation:

$$\begin{aligned} Z_\pi^{c_k}(o,a) &= \sum_{h=0}^{H-1}\gamma_c^h c_k(o_h,a_h,o_{h+1})|o_0=o,a_0=a \\ &\overset{D}{=} c_k(o,a,o') + \gamma_c Z_\pi^{c_k}(o',a'), \end{aligned} \tag{29}$$

where $o' \sim P(\cdot|o,a)$, $a' \sim \pi_\theta(\cdot|o')$ and $\overset{D}{=}$ denotes equality in distribution. To leverage this for network training, we employ a specific sampling strategy for each transition (o,a,c_k,o') drawn from the replay buffer $\mathcal{D}$. First, we sample N quantile fractions from the standard uniform distribution, i.e., $\{\zeta_i\}_{i=1}^N \sim \mathcal{U}(0,1)$, to evaluate the current quantile estimates $Q^c_{\psi_k}(o,a,\zeta_i)$. Second, we sample another N' quantile fractions $\{\zeta'_j\}_{j=1}^{N'} \sim \mathcal{U}(0,1)$ to construct the TD target using the target network $Q^c_{\psi'_k}$. For each pair (i,j), the TD error is:

$$\delta_{ij} = \left(c_k + \gamma_c \pi_\theta^\top(o')Q^c_{\psi'_k}(o',\zeta'_j)\right) - Q^c_{\psi_k}(o,a,\zeta_i), \tag{30}$$

where $Q^c_{\psi'_k}(o',\zeta'_j) \in \mathbb{R}^{|\mathcal{A}|\times 1}$ outputs the per-action per-sample quantile Q-values for discrete action spaces. The parameters ψ_k of the cost critic are then updated by minimizing the total quantile Huber loss, averaged over all samples:

$$\mathcal{L}_{\psi_k} = \frac{1}{N'}\sum_{i=1}^{N}\sum_{j=1}^{N'}\mathbb{E}\left[\rho_{\zeta_i}(\delta_{ij})\right], \tag{31}$$

where $\rho_{\zeta_i}(u) = |\zeta_i - \mathbb{I}(u<0)|\mathcal{L}_{\text{Huber}}(u)$ is the quantile Huber loss [39]. This asymmetrically weighted loss penalizes over-estimation and under-estimation differently for each quantile ζ_i, which forces the critic to learn an accurate representation of the entire distribution [31].

With a fully characterized cost distribution, we can directly estimate the worst-case costs. To estimate the CVaR_{ϵ_k} at a given risk level ϵ_k for a state-action pair, $\Gamma_{\epsilon_k}(o,a) = \text{CVaR}_{\epsilon_k}(Z_\pi^{c_k}(o,a))$, we draw N^k i.i.d. samples $\{\zeta_m\}_{m=1}^{N^k}$ from the re-parametrized uniform distribution $\mathcal{U}(1-\epsilon_k,1)$. The

Algorithm 1 The PRIMAL Algorithm

Input: Risk mode $\mathtt{M} \in \{\mathtt{Avg}, \mathtt{CVaR}\}$, $\eta \in (0,1]$
Initialize π_θ, Q^r_ϕ, $\{Q^c_{\psi_k}\}_{k=1}^K$ and their targets $\phi', \{\psi'_k\}$
$\phi' \leftarrow \phi$ and $\psi'_k \leftarrow \psi_k$
Initialize multipliers $\boldsymbol{\lambda}$, α, and shared replay buffer $\mathcal{D}$
for each event e **do**
  if e is a packet arrival event (p,h) **then**
    Get packet p observation $o_h \sim \mathcal{O}(\cdot|s_h)$
    Execute action $a_h \sim \pi_\theta(\cdot|o_h)$
  else if e is an action completion event $(o',r,\{c_k\})$ **then**
    Collect transition $(o,a,r,\{c_k\},o')$ and add to $\mathcal{D}$
  end if
  for each training step **do**
    Sample a mini-batch $(o,a,r,\{c_k\},o') \sim \mathcal{D}$.
    Update reward critic ϕ via (24)
    if $\mathtt{M} = \mathtt{Avg}$ **then**
      Update expected cost critic ψ_k via (25)
      Update actor θ via (26).
      Update cost multipliers $\boldsymbol{\lambda}$ via (27).
    else if $\mathtt{M} = \mathtt{CVaR}$ **then**
      Update distributional cost critic ψ_k via (31)
      Update actor θ via (33).
      Update cost multipliers $\boldsymbol{\lambda}$ via (34).
    end if
    Update entropy multiplier α via (28).
    Soft update reward target: $\phi' \leftarrow \eta\phi + (1-\eta)\phi'$
    Soft update cost target: $\psi'_k \leftarrow \eta\psi_k + (1-\eta)\psi'_k$
  end for
end for

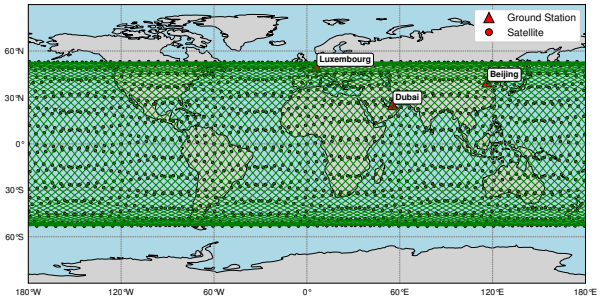

Fig. 2. Network topology used in simulations.

CVaR is then approximated by averaging the critic's output for these tail-end quantile fractions:

$$\Gamma_{\epsilon_k}(o,a) \approx \frac{1}{N^k}\sum_{m=1}^{N^k} Q^c_{\psi_k}(o,a,\zeta_m). \tag{32}$$

This estimate is then used to guide the actor, where the actor's objective function is modified to incorporate the CVaR estimate, averaged over the policy's action distribution:

$$\mathcal{L}_\theta = \mathop{\mathbb{E}}_{o\sim\mathcal{D}}\left[\pi_\theta^\top(o)\left(\alpha\log\pi_\theta(o) - Q^r_\phi(o) + \sum_k \lambda_k \Gamma_{\epsilon_k}(o)\right)\right], \tag{33}$$

where $\Gamma_{\epsilon_k}(o) \in \mathbb{R}^{|\mathcal{A}|\times 1}$ is a vector of per-action $\Gamma_{\epsilon_k}(o,a)$ estimates for discrete action space. The update for the Lagrange multiplier λ_k is also adjusted to reflect the CVaR constraint violation, ensuring the policy is pushed towards risk-averse behavior:

$$\mathcal{L}_{\lambda_k} = \lambda_k \mathop{\mathbb{E}}_{o\sim\mathcal{D}}\left[\pi_\theta^\top(o)\Gamma_{\epsilon_k}(o) - D_k\right]. \tag{34}$$

Note that the updates for the reward critic and the entropy multiplier α remain the same as in **PRIMAL-Avg**. Clearly, by replacing the standard cost critic with a distributional one and optimizing against CVaR, **PRIMAL-CVaR** provides a principled framework for building a robust routing policy that is explicitly sensitive to tail-end risks. This is critical for ensuring reliable performance in highly dynamic LEO networks. To summarize our proposed **PRIMAL** algorithm, we present its pseudo code in Algorithm 1 which integrates the two variants together for clarity. Note that we use a shared centralized replay buffer here by following the Centralized Training and Decentralized Execution (CTDE) paradigm during offline training. However, it can be easily extended to use private replay buffers via online federated learning, as shown in [20].

IV. Experiment

A. Environmental Settings

To empirically validate our **PRIMAL** framework, we developed an asynchronous event-driven high-fidelity simulator using Python and PyTorch [1]. Specifically, as shown in Fig. 2, we run simulations in an ultra-dense Walker-Delta LEO satellite with 22 satellites per orbit, and 72 evenly distributed orbits at the same altitude of 600 km in this Starlink-like constellation (1584 satellites). The inclination is 53° and the minimum elevation angle is 15°. We generate packets from three ground stations representing three cities on Earth: Luxembourg, Dubai, and Beijing. All cities have equal probability to be chosen as the source or destination node of a generated packet. We update the satellite positions every 100 ms. We set stable link data rates at 1000 Mbps for GSLs and 50 Mbps for ISLs. The node and link buffer sizes are both 16 Mbits. We set the traffic packet length following the same setting as in [20], with 80% being normal packets (64.8 Kbits) and the remaining 20% being small packets (16.2 Kbits). In addition, we set the maximum TTL as $H = 64$. We run the simulation by a 30-second training or evaluation epoch with a packet traffic rate of $10,000$ packets/s, totaling $300,000$ packets per run according to the Poisson process. We run training iteration once every 1 ms, and report training performance metrics every 2 seconds (2K iterations).

For the neural network implementation, the actor (π_θ) and critics (Q^r_ϕ, $Q^c_{\psi_k}$) use a shared backbone, a two-layer MLP with 512 hidden units, to process observations. Each component has a separate MLP output head. For the **PRIMAL-CVaR** variant, the cost critic $Q^c_{\psi_k}$ is an IQN suggested in [39] with two layers and quantile parameters $N = N' = N^k = 64$. All algorithms use a mini-batch size of 1024, a replay buffer size of 300000, and discount factors $\gamma_r = 0.99$ and $\gamma_c = 0.97$.

For the cost and reward functions, we define the cost function as the normalized queuing delay $c_h = D^Q_h/D_{norm}$, where D^Q_h is the queuing delay experienced when packet p is forwarded over a link at hop h, and $D_{norm} = 100$ ms is a predefined constant for normalization. This cost directly quantifies the level of local congestion. The reward function is designed to minimize delay and maximize delivery rates by combining a dense progressive reward and a terminal reward B_p. The per-hop reward is defined as:

$$r_h = \frac{\tau}{D_{norm}} - c_h + \Delta d + B_p \tag{35}$$

where τ is the total action delay, and Δd provides a dense reward for geographic progress towards the destination, which measured by the difference of Great Circle Distance (GCD). A large terminal reward B_p is added at the final hop to prioritize successful packet delivery:

$$B_p = \begin{cases} 1 + L_p, & \text{if } p \text{ is delivered,} \\ -\frac{5\tau^{ttl}_p}{D_{norm}} - \sum_{j=0}^{h}\Delta d_{GCD}, & \text{if } p \text{ is dropped,} \\ 0, & \text{else,} \end{cases} \tag{36}$$

which encourages maximizing the packet delivery rate. For comparison, we use a hand-crafted reward

$$\bar{r}(o,a,o') = r(o,a,o') + \sum_{k=1}^{K} c_k(o,a,o') \tag{37}$$

[1]Our code is available at https://github.com/skypitcher/risk_aware_marl

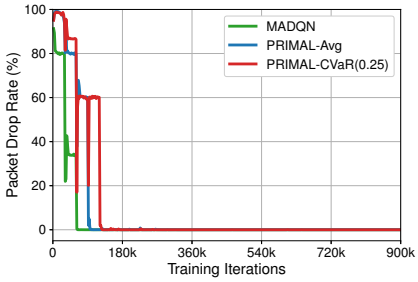

Fig. 3. Average packet drop rate versus training epochs for RL algorithms

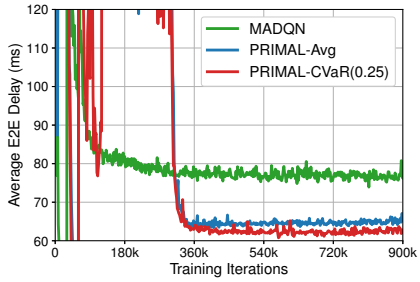

Fig. 4. Average E2E delay versus training epochs for RL algorithms

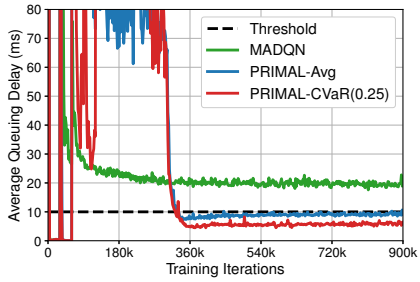

Fig. 5. Average queuing delay versus training epochs for RL algorithms

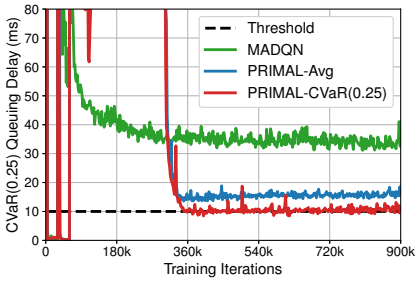

Fig. 6. $\mathrm{CVaR}_{0.25}$ Queuing delays versus training epochs for RL algorithms

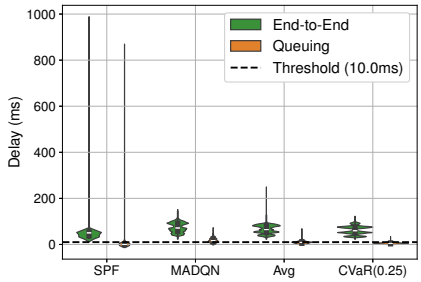

Fig. 7. Evaluated delay distribution comparison for the competing algorithms

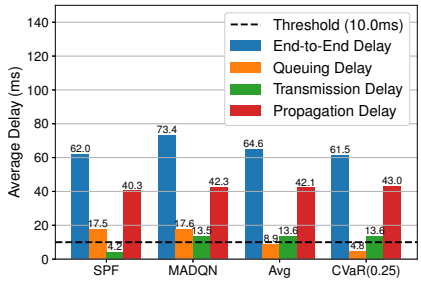

Fig. 8. Evaluated delay components comparison for the competing algorithms

as the reward function used by heuristic reward shaping approaches, which is equivalent to $\lambda_k = 1$ as all these components are well normalized. In addition, we set the minimum entropy as $\bar{\mathcal{H}} \approx 0.067$, which is a heuristic value considering the best action confidence to be 0.99 while the rest actions have the same probability of $\frac{1-0.99}{|\mathcal{A}|-1}$.

In all simulations, we set the threshold for the queuing delay cost as $D^Q_{max} = 10$ ms in total for each packet, and we compare the following algorithms:

- **`SPF`**: The Dijkstra's Shortest Path First (SPF) algorithm, assuming that routing table as precomputed based on the predictable orbital movements.
- **`MADQN`**: The multi-agent asynchronous DQN proposed in [20], using (37) as the effective reward function that relies on human prior knowledge.
- **`PRIMAL-Avg`** and **`PRIMAL-CVaR`**(ϵ): The two variants introduced in this paper, with a risk level $\epsilon \in [0, 1]$. Note that [17] uses a similar expectation based cost critic as in **`PRIMAL-Avg`**. However, [17] is not asynchronous and requires impractical synchronized joint-actions. While it can not work in our asynchronous simulator, we can treat **`PRIMAL-Avg`** as an asynchronous alternative for it.

B. Simulation Results

We now presents the empirical evaluation of our proposed PRIMAL algorithm against baseline methods. Figs.3-6 illustrate the training performance of the reinforcement learning agents. Specifically, Fig. 3 shows that all learning-based algorithms, **`MADQN`**, **`PRIMAL-Avg`**, and **`PRIMAL-CVaR`**, quickly learn to minimize the packet drop rate, achieving an almost-zero drop rate after approximately 150K training iterations. This indicates that all agents successfully learn the primary objective of delivering packets to their destinations. However, the algorithms show significant differences in their ability to manage network delay and constraints. As shown in Fig. 4, while all agents reduce the E2E packet delay over time, the **`PRIMAL`** variants achieve significantly better performance. **`PRIMAL-CVaR`** converges to the lowest average E2E delay of approximately 62 ms, followed by **`PRIMAL-Avg`** at around 66 ms, whereas **`MADQN`** stabilizes at a higher delay of about 77 ms.

The difference in performance is largely explained by how each algorithm handles the queuing delay constraint, which is a direct indicator for network congestion. Fig. 5 demonstrates that **`MADQN`** fails to respect the 10 ms queuing delay threshold, converging to a value consistently near 18 ms. In contrast, **`PRIMAL-Avg`**, which directly optimizes for the expected cost, successfully learns to keep the queuing delay at the 10 ms threshold. **`PRIMAL-CVaR`** goes one step further, reducing the average queuing delay to just 5 ms well below the threshold, by actively mitigating worst-case tail events. The unique capability of **`PRIMAL-CVaR`** is clearly validated in Fig. 6, which plots the $\mathrm{CVaR}_{0.25}$ of the queuing delay. We can find that only **`PRIMAL-CVaR`** successfully reduces the tail-end risk, bringing the $\mathrm{CVaR}_{0.25}$ of the queuing delay down to the 10 ms threshold, whereas both **`MADQN`** and **`PRIMAL-Avg`** exhibit a $\mathrm{CVaR}_{0.25}$ far exceeding this limit. This strongly confirms that **`PRIMAL-CVaR`** is highly risk-aware and is capable to effectively constrain worst-case congestion events.

The post-training evaluation results are summarized in

TABLE I
ROUTING PERFORMANCE COMPARISON FOR THE COMPETING ALGORITHMS.

	Throughput	Drop Rate	E2E Delay	Queuing Delay		Load Balancing Constraint	
				mean ± std	$\text{CVaR}_{0.25}$	Violation Rate[2]	Magnitude of violation[3]
SPF	27.0 Mbps	84.8%	62.0 ± 85.0 ms	**17.5** ± 80.2 ms	**70.1 ms**	85.7%	261.12 ± 176.60 ms
MADQN	542.7 Mbps	**0.00%**	73.4 ± 20.4 ms	**17.6** ± 10.1 ms	**31.1 ms**	75.5%	11.60 ± 8.10 ms
PRIMAL-Avg	542.9 Mbps	**0.00%**	64.6 ± 17.7 ms	**8.9** ± 5.3 ms	**16.0 ms**	38.6%	4.19 ± 3.35 ms
PRIMAL-CVaR	**543.0** Mbps	**0.00%**	**61.5** ± **18.2** ms	**4.8** ± 3.0 ms	**8.9 ms**	**5.8%**	**2.47** ± **2.38** ms

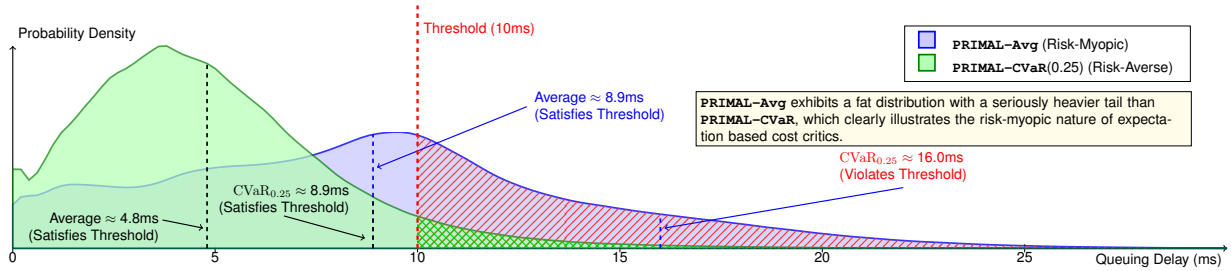

Fig. 9. Learned policy's queuing delay distribution comparison for the proposed **PRIMAL** variants. Data were directly drawn from well-trained models.

Table I and detailed in Figs. 7 and 8, averaged over five test runs with different random seeds. The static baseline **SPF** performs poorly, suffering an 84.8% drop rate and high E2E delay variance. This shows its inability to handle dynamic congestion. In contrast, all learning-based methods achieve high throughput and near-zero drop rates. Among them, **PRIMAL-CVaR** delivers the best overall performance: the highest throughput (543.0 Mbps), the lowest E2E delay (61.5 ± 18.2 ms), and minimal queuing delay (4.8 ± 3.0 ms). **PRIMAL-Avg** also satisfies the average queuing delay constraint (8.9 ± 5.3 ms) and outperforms **MADQN**, which violates the constraint with 17.6 ± 10.1 ms due to its heuristic risk handling. To assess risk, we report $\text{CVaR}_{0.25}$ of queuing delay: **MADQN** and **PRIMAL-Avg** incur 31.1 ms and 16.0 ms, while only **PRIMAL-CVaR** keeps it below the 10 ms threshold at 8.9 ms, indicating effective tail-risk mitigation. As a result, it also achieves the lowest violation magnitude, which outperforms all other methods.

Moreover, Figs. 7 and 9 provide a distributional view of the E2E and queuing delays. The results for SPF and MADQN show very high variance, indicating unpredictable performance. In contrast, the PRIMAL variants, particularly PRIMAL-CVaR, show much tighter delay distributions. In Fig. 9, while both the two **PRIMAL** variants keep the average queuing delay below the 10ms threshold, the distributions reveal a key difference in risk management. **PRIMAL-Avg** exhibits a wider distribution with a heavier tail, and its CVaR at a 25% risk level ($\text{CVaR}_{0.25}$) significantly violates the threshold. This demonstrates **PRIMAL-CVaR**'s ability to not only optimize for average performance but to also effectively mitigate high-delay tail events, resulting in a more predictable and robust routing policy.

Fig. 8 breaks down the average end-to-end delay into its constituent parts, offering crucial insights into the operational differences between the routing strategies. From the delay compositions of the **PRIMAL** variants, a well-known (and almost common sense) principle is clearly validated: the fastest path is not always the geographically shortest one. For example, **PRIMAL-CVaR** accepts a minor increase in propagation delay over **PRIMAL-Avg** (only 1 ms higher), indicating it selects physically longer routes. However, what makes this result interesting is not the validation of the principle, but how our algorithm effectively realizes this well-know trade-off. With the minimal detour cost, **PRIMAL-CVaR** achieves a massive 46% reduction in queuing delay comparing to **PRIMAL-Avg**, leading to more effective and balanced network load. This highlights that risk-aware congestion avoidance is the most critical factor influencing performance, far outweighing the marginal cost of a slightly longer path. By making this intelligent risk-aware trade-off, **PRIMAL-CVaR** effectively bypasses network hotspots to achieve the lowest overall end-to-end delay with almost-zero packet dropping rate.

V. CONCLUSION

In this paper, we proposed **PRIMAL**, an asynchronous risk-aware multi-agent packet routing framework tailored for the dynamic decentralized nature of LEO satellite networks. Our event-driven design enables each satellite to act independently with their own pace, while risk-awareness is achieved through primal-dual learning using distributional RL to capture routing cost distributions and constrain tail risks via CVaR. Empirical results show that **PRIMAL** effectively avoids traffic hotspots by trading off slightly longer paths for improved load balancing and overall performance, demonstrating a principled approach to risk-aware routing in highly dynamic networks.

[2]Packet dropping is considered as load balance violation as well.
[3]We only counts for packets that are successfully delivered.

REFERENCES

[1] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff, J. Querol, L. Lei, T. X. Vu, and G. Goussetis, "Satellite communications in the new space era: A survey and future challenges," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 1, pp. 70–109, 2021.

[2] H. Al-Hraishawi, H. Chougrani, S. Kisseleff, E. Lagunas, and S. Chatzinotas, "A survey on nongeostationary satellite systems: The communication perspective," *IEEE Commun. Surv. Tutorials*, vol. 25, no. 1, pp. 101–132, 2023.

[3] S. Li, Q. Wu, R. Wang, and R. Lu, "Toward networking and routing in 6G satellite-terrestrial integrated networks: Current issues and a potential solution," *IEEE Commun. Mag.*, vol. 63, no. 3, pp. 92–98, 2025.

[4] R. Zhang, H. Du, Y. Liu, D. Niyato, J. Kang, Z. Xiong, A. Jamalipour, and D. I. Kim, "Generative AI agents with large language model for satellite networks via a mixture of experts transmission," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 12, pp. 3581–3596, 2024.

[5] Y. Yang, X. He, J. Lee, D. He, and Y. Li, "Collaborative deep reinforcement learning in 6G integrated satellite-terrestrial networks: Paradigm, solutions, and trends," *IEEE Commun. Mag.*, vol. 63, no. 1, pp. 188–195, 2025.

[6] K. He, T. X. Vu, L. Fan, S. Chatzinotas, and B. Ottersten, "Spatio-temporal predictive learning using crossover attention for communications and networking applications," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 3, pp. 479–490, 2025.

[7] Z. Han, C. Xu, G. Zhao, S. Wang, K. Cheng, and S. Yu, "Time-varying topology model for dynamic routing in LEO satellite constellation networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3440–3454, 2023.

[8] Y. Lv, C. Xing, N. Xu, X. Han, and F. Wang, "Research of adaptive routing scheme for LEO network," in *Proc. IEEE ICCC*, 2019, pp. 987–992.

[9] G. Zheng, N. Wang, R. Tafazolli, X. Wei, and J. Yang, "Virtual data-plane addressing for SDN-based space and terrestrial network integration," in *Proc. IEEE HPSR*, 2021, pp. 1–6.

[10] E. Ekici, I. F. Akyildiz, and M. D. Bender, "A distributed routing algorithm for datagram traffic in LEO satellite networks," *IEEE/ACM Trans. Netw.*, vol. 9, no. 2, pp. 137–147, 2002.

[11] S. Li, Q. Wu, and R. Wang, "Dynamic discrete topology design and routing for satellite-terrestrial integrated networks," *IEEE/ACM Trans. Netw.*, vol. 32, no. 5, pp. 3840–3853, 2024.

[12] Y. Li, L. Liu, H. Li, W. Liu, Y. Chen, W. Zhao, J. Wu, Q. Wu, J. Liu, and Z. Lai, "Stable hierarchical routing for operational LEO networks," in *Proc. ACM MobiCom*, 2024, pp. 296–311.

[13] Q. Chen, L. Yang, Y. Zhao, Y. Wang, H. Zhou, and X. Chen, "Shortest path in LEO satellite constellation networks: An explicit analytic approach," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 5, pp. 1175–1187, 2024.

[14] Y. Huang, B. Feng, A. Tian, P. Dong, S. Yu, and H. Zhang, "An efficient differentiated routing scheme for MEO/LEO-based multi-layer satellite networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 1, pp. 1026–1041, 2024.

[15] W. Wei, L. Fu, H. Gu, X. Lu, L. Liu, S. Mumtaz, and M. Guizani, "Iris: Toward intelligent reliable routing for software-defined satellite networks," *IEEE Trans. Commun.*, vol. 73, no. 1, pp. 454–468, 2025.

[16] P. Zuo, C. Wang, Z. Yao, S. Hou, and H. Jiang, "An intelligent routing algorithm for LEO satellites based on deep reinforcement learning," in *Proc. IEEE VTC*, 2021, pp. 1–5.

[17] Y. Lyu, H. Hu, R. Fan, Z. Liu, J. An, and S. Mao, "Dynamic routing for integrated satellite-terrestrial networks: A constrained multi-agent reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 5, pp. 1204–1218, 2024.

[18] R. Zhang, H. Du, Y. Liu, D. Niyato, J. Kang, S. Sun, X. Shen, and H. V. Poor, "Interactive AI with retrieval-augmented generation for next generation networking," *IEEE Netw.*, vol. 38, no. 6, pp. 414–424, 2024.

[19] S. Li, Q. Wu, and R. Wang, "Efficient packet routing in ultra-dense LEO satellite networks via cooperative-marl with queuing theory model," in *Proc. IEEE WCNC*, 2025, pp. 1–6.

[20] F. Lozano-Cuadra, B. Soret, I. Leyva-Mayorga, and P. Popovski, "Continual deep reinforcement learning for decentralized satellite routing," *IEEE Trans. Commun.*, 2025.

[21] R. Zhang, K. Xiong, Y. Lu, P. Fan, D. W. K. Ng, and K. B. Letaief, "Energy efficiency maximization in RIS-assisted SWIPT networks with RSMA: A PPO-based approach," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 5, pp. 1413–1430, 2023.

[22] Y. Liang, H. Wu, and H. Wang, "ASM-PPO: Asynchronous and scalable multi-agent PPO for cooperative charging," in *Proc. AAMAS*, 2022, pp. 798–806.

[23] C. Le, T. X. Vu, and S. Chatzinotas, "Cooperative UAVs with asynchronous multi-agent learning for remote data collection," in *Proc. Globecom Wks*, 2024.

[24] K. He, T. X. Vu, D. T. Hoang, D. N. Nguyen, S. Chatzinotas, and B. E. Ottersten, "Risk-aware antenna selection for multiuser massive MIMO under incomplete CSI," *IEEE Trans. Wirel. Commun.*, vol. 23, no. 9, pp. 11 001–11 014, 2024.

[25] J. Song, Y. Ju, L. Liu, Q. Pei, C. Wu, M. A. Jan, and S. Mumtaz, "Trustworthy and load-balancing routing scheme for satellite services with multi-agent DRL," in *Proc. IEEE INFOCOM Workshop*, 2023, pp. 1–6.

[26] S. Paternain, L. Chamon, M. Calvo-Fullana, and A. Ribeiro, "Constrained reinforcement learning has zero duality gap," in *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[27] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan, "WCSAC: Worst-case soft actor critic for safety-constrained reinforcement learning," in *Proc. AAAI*, vol. 35, no. 12, 2021, pp. 10 639–10 646.

[28] Y. Xiao, W. Tan, and C. Amato, "Asynchronous actor-critic for multi-agent reinforcement learning," in *Adv. Neural Inf. Process. Syst.*, 2022.

[29] K. Menda, Y. Chen, J. Grana, J. W. Bono, B. D. Tracey, M. J. Kochenderfer, and D. H. Wolpert, "Deep reinforcement learning for event-driven multi-agent decision processes," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 1259–1268, 2019.

[30] C. Yu, X. Yang, J. Gao, J. Chen, Y. Li, J. Liu, Y. Xiang, R. Huang, H. Yang, Y. Wu, and Y. Wang, "Asynchronous multi-agent reinforcement learning for efficient real-time multi-robot cooperative exploration," in *Proc. AAMAS*, 2023, pp. 1107–1115.

[31] Q. Zhang, S. Leng, X. Ma, Q. Liu, X. Wang, B. Liang, Y. Liu, and J. Yang, "CVaR-constrained policy optimization for safe reinforcement learning," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 36, no. 1, pp. 830–841, 2025.

[32] S. Cakaj, "The parameters comparison of the Starlink LEO satellites constellation for different orbital shells," *Front. Commun. Netw.*, vol. 2, p. 643095, 2021.

[33] J. Lee, J. Park, M. Bennis, and Y. Ko, "Integrating LEO satellites and multi-UAV reinforcement learning for hybrid FSO/RF non-terrestrial networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3647–3662, 2023.

[34] A. Chaaban, Z. Rezki, and M. Alouini, "On the capacity of intensity-modulation direct-detection gaussian optical wireless communication channels: A tutorial," *IEEE Commun. Surv. Tutorials*, vol. 24, no. 1, pp. 455–491, 2022.

[35] M. Fonoberova and D. Lozovanu, "Optimal multicommodity flows in dynamic networks and algorithms for their finding," *Buletinul Academiei de Ştiinţe a Republicii Moldova. Matematica*, vol. 47, no. 1, pp. 19–34, 2005.

[36] W. Jung, M. Cho, J. Park, and Y. Sung, "Quantile constrained reinforcement learning: A reinforcement learning framework constraining outage probability," in *Adv. Neural Inf. Process. Syst.*, 2022.

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. ICML*, 2018, pp. 1861–1870.

[38] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. J. Spaan, "Safety-constrained reinforcement learning with a distributional safety critic," *Mach. Learn.*, vol. 112, no. 3, pp. 859–887, 2023.

[39] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *Proc. ICML*, vol. 80, 2018, pp. 1104–1113.

[40] H. Zhou, T. Wei, Z. Lin, J. Li, J. Xing, Y. Shi, L. Shen, C. Yu, and D. Ye, "Revisiting discrete soft actor-critic," *Trans. Mach. Learn. Res.*, vol. 2024, 2024.

[41] Z. Chen, Y. Zhou, and H. Huang, "On the duality gap of constrained cooperative multi-agent reinforcement learning," in *Proc. ICLR*, 2024.