Improving User Experience in Hybrid SATCOM with Deep Q-Learning

Matthieu Petrou
Viveris Technologies
Toulouse, FRANCE
matthieu.petrou@viveris.fr

Bastien Tauran
Viveris Technologies
Toulouse, FRANCE
bastien.tauran@viveris.fr

Laurent Chasserat
CNES
Toulouse, FRANCE
laurent.chasserat@cnes.fr

Matthieu Destruhaut
CNES
Toulouse, FRANCE
matthieu.destruhaut@cnes.fr

Moujahed Rebhi
Eutelsat Group
Paris, FRANCE
mrebhi@eutelsat.com

David Pradas
Viveris Technologies
Toulouse, FRANCE
david.pradas@viveris.fr

***Abstract*—In recent years, the number of Low Earth Orbit (LEO) satellites in orbit has drastically increased. Internet service providers are exploring the hybridization of LEO and GEO satellite links to enhance user connectivity. However, efficient hybridization requires intelligent routing management due to the distinct characteristics of these links: while LEO offers lower latency, it also experiences greater variability in performance, whereas GEO provides more stable but higher-latency connections. Additionally, accurately gauging traffic flows to their impact on user-perceived Quality of Experience (QoE) remains a complex challenge. To address this, we investigate reinforcement learning to optimize routing over dual satellite links (LEO and GEO). We train a Deep Q-Learning (DQL) agent, based on a LSTM model, to manage routing decisions with the objective of maximizing users' QoE for VoIP calls, YouTube streaming, web browsing, and data transfers. We compare our approach with the Minimum Sending Delay Scheduler and evaluate its real-time feasibility. Results show that our method provides a scalable routing solution, capable of efficiently handling a large number of flows while significantly improving user experience.**

***Index Terms*—SATCOM, QoE, QoS, reinforcement learning, deep learning, LEO, GEO, multi-path routing, hybridization**

I. Introduction

The satellite connectivity market is experiencing significant growth, driven by reductions in barriers related to bandwidth, latency, cost, and terminal availability. These advancements are enabling the market to expand at an unprecedented pace. By 2030, the value of the satellite connectivity market is expected to more than triple, increasing from 4.3 billion to 16 billion dollars [1].

New constellations of Low Earth Orbit (LEO) satellites providing high-speed access are currently being deployed, with some already operational, albeit with limited coverage or capacity. The contribution of non-geostationary satellites is projected to grow approximately 2.5 times faster than the overall market, accounting for nearly 50% of the market by 2030—primarily driven by LEO constellations.

In a multi-access integration context, satellite providers must consider link characteristics relative to key use cases. LEO satellites, orbiting between 200 and 2,000 km from Earth, offer low latency (20–50 ms) but experience greater performance variability due to rapid movement and frequent handovers, impacting stability and throughput. In contrast, GEO satellites, orbiting at approximately 35,786 km, deliver more stable, high-bandwidth connections at the cost of higher latency (500–600 ms), which degrades the performance of real-time applications such as VoIP and online gaming.

Satellite providers are exploring how to best combine different orbital systems to optimize access, balancing each system's advantages and limitations. The goal is full network integration with automated routing and unified terminals, ensuring seamless connectivity and high Quality of Experience (QoE) for users.

In this study, we focus on solutions at the ground segment level, where a network-layer routing approach offers simplicity, energy efficiency, and minimal intrusiveness. Routing is implemented at the IP layer, enabling dynamic flow selection between LEO and GEO links based on system needs. Although traffic classification (e.g., via DPI) is assumed to be available, conventional IP routing remains limited, relying only on basic Layer 3 information. Even with flow-specific routing tables based on ports, tags, or DPI-derived data, designing optimal policies for all flow types and network conditions quickly becomes complex, if not unfeasible.

To address this challenge, we investigate the feasibility of using Reinforcement Learning (RL) for routing management. As our objective is to improve overall QoE, we incorporate QoE metrics into the reward function. We assess the performance of our approach by comparing the QoE achieved with our solution against a well-known routing algorithm. Finally, we evaluate the practical viability of this approach by ensuring that decision-making can be performed in real-time within a realistic deployment. To the best of our knowledge, this is the first study to leverage RL for hybrid LEO-GEO satellite link management, making our contribution particularly relevant to satellite communication networks.

The rest of this paper is organized as follows: Section II

ISBN 978-3-903176-72-0

briefly reviews related work on multi-path hybridization and reinforcement learning for network optimization. Section III presents the methodology used for data collection, RL training, and system definition. Section IV evaluates the impact of RL-based routing control on users' QoE. Finally, Section V summarizes the findings and concludes the study.

II. Related Work

This section provides an overview of related work on multi-path hybridization. Additionally, it reviews previous studies on the application of RL in networking.

A. Link Hybridization

Several studies have explored hybrid link utilization and traffic routing to enhance network performance, particularly in multi-path environments.

Singh et al. [2] presents an analysis of different scheduling strategies. They showed that the simplest solution is the Minimum Sending Delay Scheduler (MSDS), which is used later in this paper. MSDS dynamically adjusts flow routing based on the delay of the one of the link, directing traffic to the other link when the first one experiences higher latency responsiveness.

To improve routing decisions for high-bandwidth applications, Kamboj et al. [3] proposed a QoS-aware dynamic multi-path routing scheme within SDN networks. Their method employs a greedy heuristic approach to optimize routing, focusing primarily on enhancing QoS parameters such as bandwidth allocation.

B. Reinforcement Learning for QoE Optimization

RL-based techniques have shown promise in optimizing network QoE. In SATCOM environments, Kebedew et al. [4] applied State-Action-Reward-State-Action (SARSA) and Q-learning for bandwidth allocation under constrained conditions. Wang et al. [5] proposed a QoE-driven path selection approach using ML prediction for application-aware routing. However, it requires prior knowledge of the best route during the training phase, which may not always be feasible in highly dynamic networks.

However, most existing RL solutions focus either on specific application types (e.g., VoIP or video streaming) or single performance metrics, limiting their generalizability. Furthermore, RL-based hybrid routing for diverse SATCOM traffic types remains underexplored.

III. System Description

This section describes the testbed used for data collection and RL agent training. It also presents the RL approach, detailing the system definition.

A. Tools

Our testbed relies on two emulated satellite links to connect users to the Internet and the server. To emulate these satellite links, we use OpenSAND [6], [7], an open-source, end-to-end satellite communication system emulator. OpenSAND provides a realistic representation of SATCOM systems [8]. A satellite link is composed of three main components: the satellite gateway (GW), the satellite terminal (ST), and the satellite itself (SAT). For both GEO and LEO satellite link emulations, we used real-world attenuation and latency traces collected from Eutelsat's own operational GEO and LEO satellite links.

We orchestrate and execute the tests using OpenBACH [9], an open-source network metrology test bench. OpenBACH enables the deployment of various applications, referred as jobs:

- For data transfer, we use iperf3 [10] to generate TCP traffic, employing the CUBIC congestion control algorithm.
- For VoIP flows, we leverage the functionality of D-ITG [11] to create VoIP streams.
- For web traffic, we employ both Apache and Firefox to generate web browsing activity.
- For YouTube traffic, we utilize Firefox to play YouTube videos. An OpenBACH script is used to collect YouTube-specific metrics via the SATboost plugin [12], which retrieves data from the YouTube interface — collected from the *"Stats for Nerds"* interface. It is important to note that we use only the YouTube data collection feature of SATboost and not its optimization capabilities.

To manage routing, we use algorithms that determine optimal paths, integrating our RL agent for decision-making, which is detailed in the next section. Selected routing paths are then communicated to OpenDaylight (ODL) [13], an open-source Software-Defined Networking (SDN) controller, via a Northbound REST API. ODL manages Open vSwitch, which replicates the functionality of a physical network switch in a virtualized environment, through the Southbound interface using the OpenFlow protocol. This architecture enables real-time dynamic routing adjustments for each flow.

B. Overall Architecture

Our objective is to use a testing platform that closely emulates a real-world system. To simplify the routing problem, we restrict return flows to the LEO link, focusing our analysis solely on the forward link. This choice ensures that bidirectional flows do not suffer from the high latency of GEO links in both directions. Moreover, GEO links typically have lower return link capacity compared to LEO links, making this approach a practical solution. A single LEO return link can support the combined return traffic of both forward links without becoming a bottleneck.

The testing platform consists of 15 virtual machines (VMs), including 13 VMs for network system emulation and 2 VMs for platform control. In detail, we have:

- 6 VMs emulate the satellite links using OpenSAND: GW, SAT, and ST for both GEO and LEO,
- 4 "Clients" and 1 "Server" generating traffic with the OpenBACH metrology testbed,
- 1 "SDN router" and 1 "Client Router" managing LEO/GEO hybridization,
- Tests are orchestrated by OpenBACH, and the controller is installed on a VM not used for system emulation,

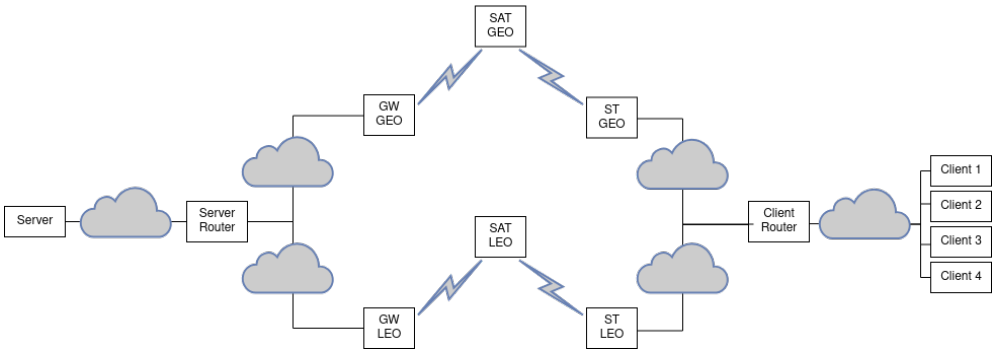


Fig. 1: Topology of the system

- The ODL manager is also deployed on a separate machine that communicates with the system but does not directly participate in network emulation.

Figure 1 illustrates the system topology.

C. Reinforcement Learning

We consider a standard reinforcement learning framework in which an agent interacts with an environment every one second. At each time step, the agent observes the current state, selects an action from the set of available actions in that state, and receives a reward as feedback. The environment then transitions to a new state based on the agent's action. This interaction can be summarized as a tuple (*state*, *action*, *next state*, *reward*), capturing the agent's experience at each step.

However, in our system, the reward is provided only for each combination of state and action at the end of the entire session. Therefore, the agent is trained using a replay buffer that stores the interaction tuples accumulated during the session, as real-time feedback is not available.

1) State: To effectively describe the environment to the agent, we selected a set of key metrics. First, we consider the number of flows for each flow type. The presence of numerous flows, or certain specific types, can significantly influence the agent's actions. Next, the buffer occupancy levels of the LEO and GEO links, which partially reflect congestion, are also deemed critical. Additionally, the average throughput, along with the number of packets per second in uplink, downlink, and total traffic, are important metrics to represent the utilization of each link. Finally, the RTT (Round Trip Time) for each link is included as a relevant metric to gauge congestion. Together, these categories constitute 28 metrics that characterize our system, which already constitutes a substantial state representation for the agent decision-making process.

2) Actions: To maintain a relatively simple agent while ensuring flexibility in its actions, we propose that for each flow type considered, the agent can decide to route 0%, 25%, 50%, 75%, or 100% of the flows through the LEO link. Using a discrete set of choices allows for the implementation of less complex agents. At the same time, having multiple levels of flow distribution provides enough flexibility to optimize the use of each link effectively.

3) Rewards: The reward system is designed to reflect the QoE perceived by users. Since different applications exhibit widely varying QoE under identical QoS conditions, each flow type is assigned a specific reward function. This ensures that the QoE characteristics of each flow are adequately represented.

To maintain fairness, it is essential to balance rewards across flow types. No flow should receive higher rewards than another when both achieve the same QoE. Balancing this reward distribution is a complex challenge, and our approach aims to mitigate this issue.

We adopt a similar approach to those proposed by Ko et al. [14] and Zhang et al. [15] to construct a reward system based on QoE for various types of flows. While their studies focus on a single type of flow, we extend the scope to include web browsing, VoIP calls, data transfers, and YouTube sessions. Consequently, significant modifications were made to adapt their methodology to our study.

To determine the coefficients in our reward system, we followed an empirical approach aimed at balancing the impact of different flow types on overall QoE. We first analyzed the contribution of each application to the global QoE and adjusted the coefficients to ensure fair prioritization, even under network congestion. The final values were chosen to reflect a trade-off that prevents one type of traffic from dominating the reward function while still aligning with realistic user experience expectations.

For data transfer, only throughput is considered to evaluate the users' QoE. The reward is computed using Equation 1, where $\alpha = 80$.

$$\mathrm{R}_{\mathrm{download}} = \alpha \cdot \mathrm{Throughput} \tag{1}$$

For web flows, only throughput is considered to evaluate the users' QoE. The reward for web flows is computed using Equation 2, where $\alpha = 40$.

$$\mathrm{R}_{\mathrm{web}} = \alpha \cdot \mathrm{Throughput} \tag{2}$$

For YouTube flows, stalling, initial delay, and resolution are considered to evaluate the users' QoE. The reward for

web flows is computed using Equation 3, where $\alpha = 30$, $\beta = -40$, and $\pi = -40$.

$$\mathrm{R_{yt}} = \alpha \cdot \text{Resolution} + \beta \cdot \text{Stalling} + \pi \cdot \text{Initial Delay} \quad (3)$$

For VoIP flows, we account for the Service Level Agreement (SLA) thresholds commonly adopted by satellite internet service providers. A key challenge lies in preventing VoIP rewards from disproportionately impacting other flows. If delay and packet loss are normalized similarly to other QoS metrics, extreme values dominate whenever SLA thresholds are exceeded. Conversely, normalizing solely against SLA values leads to excessively large normalized metrics in violation cases. To address this, we employ logarithmic scaling for metrics exceeding the SLA, while retaining linear normalization within the SLA limits. This ensures smoother transitions and balanced reward computations.

The reward for VoIP flows is computed using Equation 4, where $\alpha = 30$, $\beta = -40$, and $\pi = -40$. We define a generic penalty function f_x (Equation 5), where x represents either delay or loss. Each equation f_x accounts for the corresponding SLA threshold.

$$\mathrm{R_{VoIP}} = \alpha \cdot \text{Throughput} + \beta \cdot f_{\text{Delay}} + \pi \cdot f_{\text{Loss}} \quad (4)$$

$$f_{\mathrm{x}} = \begin{cases} \frac{\mathrm{x}}{\mathrm{SLA_x}}, & \text{if } \mathrm{x} \leq \mathrm{SLA_x}, \\ 1 + \log\left(1 + \frac{\mathrm{x} - \mathrm{SLA_x}}{\mathrm{SLA_x}}\right), & \text{if } \mathrm{x} > \mathrm{SLA_x}. \end{cases} \quad (5)$$

Despite these efforts, VoIP flows exhibit higher reward variance compared to other flow types. While VoIP is not the only flow type to give negative rewards, it has the most significant impact once SLA thresholds are breached. This behavior is necessary to ensure the agent prioritizes compliance with SLA requirements.

4) Agent Training: To train the agent, we employ the Deep Q-Learning (DQL) algorithm using the PyTorch framework [16]. This approach enables the agent to evaluate the optimal decision for a given system state while generalizing to unseen states. Additionally, DQL effectively captures the nonlinear relationships between system features and the corresponding rewards, making it well-suited for our use case.

In our implementation, the DQL algorithm is used to train a model based on a Long Short-Term Memory (LSTM) network. LSTMs are a specific type of Recurrent Neural Network (RNN) designed to retain information over extended periods. This capability is critical for handling long-term temporal dependencies that significantly affect the QoE. For instance, in the case of YouTube, where video buffering is involved, the QoE depends heavily on conditions preceding the video playback.

To train the agent, we perform randomized tests by launching combinations of 0–4 YouTube, 0–8 VoIP, 0–8 web browsing, and 0–8 iperf3 flows, with durations between 100 and 300 seconds. Throughout these experiments, we record the system state, as previously defined, and apply actions to adjust the distribution of each traffic type between the LEO and GEO links. At the end of each test, we compute the corresponding rewards to update the agent's learning model.

IV. Results

In order to evaluate the performance of the proposed solution, it is essential to compare it with an established and proven method. This section defines the test scenarios and compares the trained reinforcement learning agent, referred to as DQL, with the MSDS algorithm [2], as mentioned in Section II-A. We configure MSDS to redirect traffic to the GEO link when LEO experiences higher latency. In our case, we set the LEO delay threshold to 100 ms, prioritizing the rerouting of iperf3 flows first, followed by YouTube flows, and finally web flows. Lastly, we analyze the MOS obtained by VoIP users in the last two scenarios, comparing the routing decisions made by MSDS and DQL.

A. Test Scenarios for Performance Evaluation

Perfectly replicating a classical network is challenging due to the diversity of possible scenarios. We therefore compare the two solutions across five distinct cases, varying in network load and traffic type, as summarized in Table I. Since the number of flows was randomized during training, it is unlikely the agent encountered identical scenarios, reducing overfitting risks and promoting adaptation to diverse traffic conditions.

The *Low Load* scenario is designed to assess the agent's ability to optimize traffic routing when the network is lightly loaded. The *Medium Load* scenario simulates an average usage of the satellite links, with multiple users utilizing the network without overwhelming it.

To investigate further, we add the *Medium Load Without VoIP* and *Medium Load Without iperf* scenarios, which aim to achieve similar objectives but without two key applications. In particular, iperf3 uses TCP/CUBIC, making it more likely to dominate the network and affect other applications. Conversely, VoIP typically generates small traffic flows, making it vulnerable to network congestion. However, VoIP also generates the most negative rewards if delay and packet loss are high.

The final scenario, referred to as the *Overloaded Network* scenario, introduces more VoIP, iperf3, and web flows than the agent encountered during training. The objective of this scenario is to push the agent to its limits and evaluate its ability to generalize under extreme network conditions.

B. Performance Analysis

Table II summarizes the mean cumulative rewards obtained for each scenario and presents the differences between MSDS and DQL.

This table shows that the DQL solution slightly improves the overall QoE compared to MSDS in the Low Load scenario. More specifically, DQL primarily enhances the QoE of VoIP at the expense of either YouTube or Web flows. In the Medium Load scenario, DQL also improves the overall QoE compared

TABLE I: Overview of scenarios

Scenario (No. of Flows)	No. of YouTube Flows	No. of VoIP Flows	No. of iperf3 Flows	No. of Web Flows
Low Load (8)	2	2	2	2
Medium Load (15)	2	7	4	2
Medium Load Without VoIP (14)	3	0	6	5
Medium Load Without iperf (17)	4	6	0	7
Overloaded Network (40)	4	12	12	12

TABLE II: Mean of Cumulative Rewards for Each Scenario

Scenario	MSDS	DQL	Δ Rewards
Low Load	12,151	14,742	+2,591
Medium Load	17,354	22,052	+4,698
Medium Load Without VoIP	12,093	10,611	−1,482
Medium Load Without iperf	−43,618	2,830	+46,448
Overloaded Network	−133,880	−17,164	+116,716

to MSDS. It enhances the QoE of VoIP and iperf3 while slightly reducing the QoE of web applications.

Interestingly, in the Medium Load Without VoIP scenario, DQL reduces the overall QoE. The agent likely attempts to protect VoIP flows that it expects to arrive but never do. As a result, both YouTube and iperf3 applications experience worse QoE with DQL routing, whereas web applications achieve slightly better results. This suggests that the reward structure for VoIP may be overly punitive or that additional training without VoIP is necessary.

On the other hand, DQL significantly improves the overall QoE in the Medium Load Without iperf scenario compared to MSDS. The primary difference is the QoE of VoIP: MSDS struggles to handle bursty flows (YouTube and Web), as it reacts only to latency spikes, which often occur too late for effective mitigation.

Finally, both MSDS and DQL obtain cumulative negative rewards in the Overloaded Network scenario. However, DQL earn a smaller negative reward compared to MSDS. This scenario contains too many flows for the available links, and while DQL still results in negative rewards, the significant gap between DQL and MSDS demonstrates that DQL effectively mitigates network degradation.

Overall, the DQL approach outperforms MSDS in routing optimization. While the improvement is marginal in the Low Load scenario and ineffective in the Medium Load Without VoIP scenario, in all other cases, DQL significantly enhances cumulative rewards and, consequently, users' QoE.

C. Impact of DQL on VoIP Quality

The MOS is one of the most widely used metrics for assessing QoE. It ranges from 1 (lowest perceived quality) to 5 (highest perceived quality). Figures 2 present the MOS of each VoIP user in both the Medium Load Without Iperf3 (Figure 2a) and Overloaded Network (Figure 2b) scenarios. A key takeaway from these results is the significant improvement in VoIP MOS achieved by DQL.

Figure 2a highlights a clear improvement in VoIP MOS when using DQL, shifting most scores from a poor-to-fair range (1–3) to a fair-to-excellent range (3–4.5). This suggests that DQL effectively mitigates latency and packet loss, prioritizing VoIP traffic even under medium load conditions.

Similarly, Figure 2b shows that in a highly congested scenario, DQL improves MOS from a barely usable level (1–2) to a more acceptable quality (3–4). This indicates that, despite extreme network conditions, the RL agent succeeds in maintaining a more stable and usable VoIP service compared to MSDS.

These results confirm that DQL enhances VoIP experience, and could benefit other delay- and loss-sensitive applications.

D. Real-Time Decision

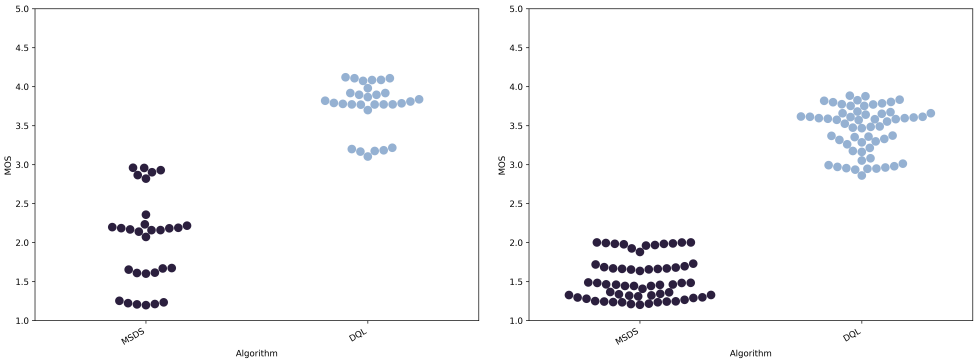
To assess the real-time feasibility of our agent, we measure its decision-making time under the Overloaded Network scenario to stress the system. To ensure a realistic assessment, the DQL agent runs on a constrained VM (2 CPU cores, 2 GB RAM, no GPU) to reflect the limited resources of real-world modems. This setup simulates the limited resources typical of real-world deployments.

Figure 3 presents the cumulative distribution function (CDF) of the measured decision times. The minimum recorded decision time is 1.6 ms, while the median decision time is 3.6 ms. Furthermore, 95% of decisions are made within 40 ms, and the maximum observed decision time is 688.3 ms.

In most cases, decision time stays well below one second, indicating real-time feasibility with a one-second decision interval. Additionally, if a decision is delayed, the previously selected action remains in effect until the next update. It is worth noting that the current implementation is a proof of concept: neither the model size nor the code execution has been optimized. Thus, further improvements could be achieved with system-level optimizations. Nonetheless, this aspect should be carefully considered in practical deployments.

V. Conclusion

This study explores the use of RL to hybridize LEO and GEO satellite links for QoE-optimized routing. For this purpose, we train a DQL agent based on a LSTM model to improve overall QoE across multiple applications, including VoIP, web browsing, YouTube streaming, and data transfer. The system operates with two emulated satellite links: a GEO



(a) Medium Load Without Iperf3 Scenario (b) Overloaded Network Scenario

Fig. 2: MOS of each VoIP users.

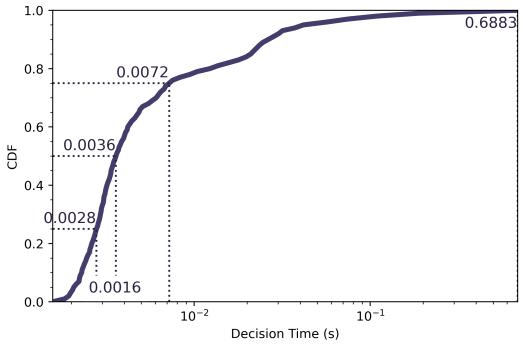


Fig. 3: CDF of Agent Decision Time (Logarithmic X-Axis).

and a LEO. The performance of the DQL-based approach is then compared to a well-known routing algorithm, MSDS.

Experimental results show that the DQL agent improves overall QoE in nearly all scenarios compared to MSDS. While MSDS aims to minimize delay and packet loss by prioritizing the LEO link, it struggles under high traffic and is sensitive to unexpected bursts. Additionally, we address the challenge of real-time deployment, verifying that the DQL agent can make routing decisions within practical time constraints.

These findings suggest that RL-based routing could be effectively integrated into next-generation satellite modems, capable of managing multiple links dynamically.

In future work, we aim to extend this approach to other applications, such as videoconferencing. Additionally, we plan to incorporate variations in link reliability, going beyond congestion management to account for dynamic link conditions. Furthermore, another objective is to integrate 5G Non-Terrestrial Networks by introducing a third link, leveraging RL-controlled routing decisions to enhance connectivity.

VI. Acknowledgments

This work was supported by Viveris Technologies, Eutelsat Group, and CNES.

References

[1] Eutelsat strategy update on the proposed combination with oneweb. 2025. [Online]. Available: https://www.eutelsat.com/en/news/press.html#/pressreleases/eutelsat-strategy-update-on-the-proposed-combination-with-oneweb-3210411

[2] A. Singh, C. Goerg, A. Timm-Giel, M. Scharf, and T.-R. Banniza, "Performance comparison of scheduling algorithms for multipath transfer," in *2012 IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 2653–2658.

[3] P. Kamboj, S. Pal, S. Bera, and S. Misra, "Qos-aware multipath routing in software-defined networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 723–732, 2023.

[4] T. M. Kebedew *et al.*, "Reinforcement learning for qoe-oriented flexible bandwidth allocation in satellite communication networks," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 305–310.

[5] L. Wang, X. Wu, and D. T. Delaney, "QoE-Oriented Routing Mixing Application KPIs and Link Metrics through Machine Learning," *IEEE Access*, pp. 1–1, 2024.

[6] OpenSAND. 2025. [Online]. Available: https://www.opensand.org/

[7] E. Dubois *et al.*, "OpenSAND, an open source satcom emulator," Kaconf, 2017.

[8] A. Auger, E. Lochin, and N. Kuhn, "Making Trustable Satellite Experiments: an Application to a VoIP Scenario," in *89th IEEE Vehicular Technology Conference*. Kuala Lumpur, Malaysia: IEEE, Apr. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02191751

[9] OpenBACH. 2025. [Online]. Available: https://www.openbach.org/

[10] iperf3. 2025. [Online]. Available: https://software.es.net/iperf/

[11] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-itg distributed internet traffic generator," in *First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings.*, 2004, pp. 316–317.

[12] SATboost add-on. 2025. [Online]. Available: https://github.com/CNES/satboost/

[13] OpenDaylight. 2025. [Online]. Available: https://docs.opendaylight.org/en/latest/index.html#

[14] K. Ko, S. Ryu, and J. W.-K. Hong, "Enhancing qoe of webrtc-based video conferencing using deep reinforcement learning," in *2023 24st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2023, pp. 195–200.

[15] H. Zhang *et al.*, "Onrl: improving mobile video telephony via online reinforcement learning," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3372224.3419186

[16] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.