Enhancing Network Intelligence with LLM-Based IBN and DRL: A Dynamic Approach for SAGIN Resource Management.

Sajid Alam
Dept. Electronics Engineering
Jeju national university
sajid.alam19801@gmail.com

Wang-Cheol Song
Dept. Computer Engineering
Jeju national university
philo@jejunu.ac.kr

***Abstract*—We propose an innovative Intent-Based Networking (IBN) system that leverages Large Language Models (LLMs) to translate high-level user intents into actionable network policies. By integrating an LLM-based translation engine with a Deep Reinforcement Learning (DRL) agent, our system enables dynamic and intelligent resource allocation within Space-Air-Ground Integrated Networks (SAGINs). The LLM interprets user requests expressed in natural language to generate precise network configurations, which the DRL agent uses to optimize resource utilization while meeting Quality of Service (QoS) requirements. Simulations demonstrate that our approach effectively reduces latency, enhances bandwidth utilization, minimizes QoS violations, and increases Virtual Network Request (VNR) acceptance rates compared to static allocation strategies. These results highlight the potential of combining LLMs with DRL in IBN systems to improve network adaptability, efficiency, and user satisfaction.**

***Index Terms*—IBN, deep Reinforcement Learning, Large language models.**

I. Introduction

Intent-Based Networking (IBN) enables defining network behaviors and objectives through high-level intents expressed in natural language, which are translated into actionable configurations by the Network Management System (NMS) [1]. This simplifies user interaction, enhances scalability, and reduces complexity [2]. Traditional structured formats like JSON and YAML lack the intuitiveness of natural language, which Large Language Models (LLMs) address by improving intent interpretation and translation [3], [4]. However, applying IBN to Space-Air-Ground Integrated Networks (SAGINs) poses unique challenges, including resource heterogeneity across satellite, aerial, and terrestrial networks, dynamic resource availability, and stringent Quality of Service (QoS) requirements for applications like Extended Reality (XR) [5], [6]. Advanced approaches are needed to manage resources adaptively while maintaining QoS under fluctuating conditions.

In this paper, we propose a novel framework integrating LLMs for translating intents into network policies with Deep Reinforcement Learning (DRL) for adaptive resource allocation in SAGINs. Our approach ensures efficient QoS management by dynamically optimizing resource usage, addressing the limitations of static resource allocation methods.

Our main contributions are:

- Developing an LLM-based translation engine to convert natural language intents into actionable network policies, enhancing usability and precision.
- Introducing an Assurance Module to monitor QoS and dynamically adjust policies via a DRL agent upon detecting violations.
- Demonstrating through simulations the effective reduction of QoS violations, improved resource utilization, and adaptation to dynamic network conditions.

II. System architecture

As illustrated in Fig. 1, the LLM-based IBN system in SAGIN addresses diverse applications like real-time video analytics, AR for industrial training, and cloud-based machine learning, each requiring substantial resources (e.g., 8 vCPUs, 16 GB memory, 10 Gbps bandwidth) with stringent latency and bandwidth constraints. The system translates high-level intents into precise resource allocation policies dynamically distributed across satellite, aerial, and terrestrial segments. Resource availability and compliance are verified through the Intent Validation Module, while the assurance mechanism continuously monitors performance and adjusts allocations to maintain QoS.

The SAGIN infrastructure, shown in Fig. **??**, is modeled as a multi-layered graph $G^S = \{N^S, E^S, R^S\}$, where N^S, E^S, and R^S represent nodes, edges, and resources, respectively. The space segment includes LEO satellites, the air segment comprises UAVs acting as aerial base stations, and the ground segment features base stations and MEC servers. Inter-segment edges facilitate seamless resource allocation, enabling the system to meet diverse QoS requirements efficiently across all layers. “‘

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

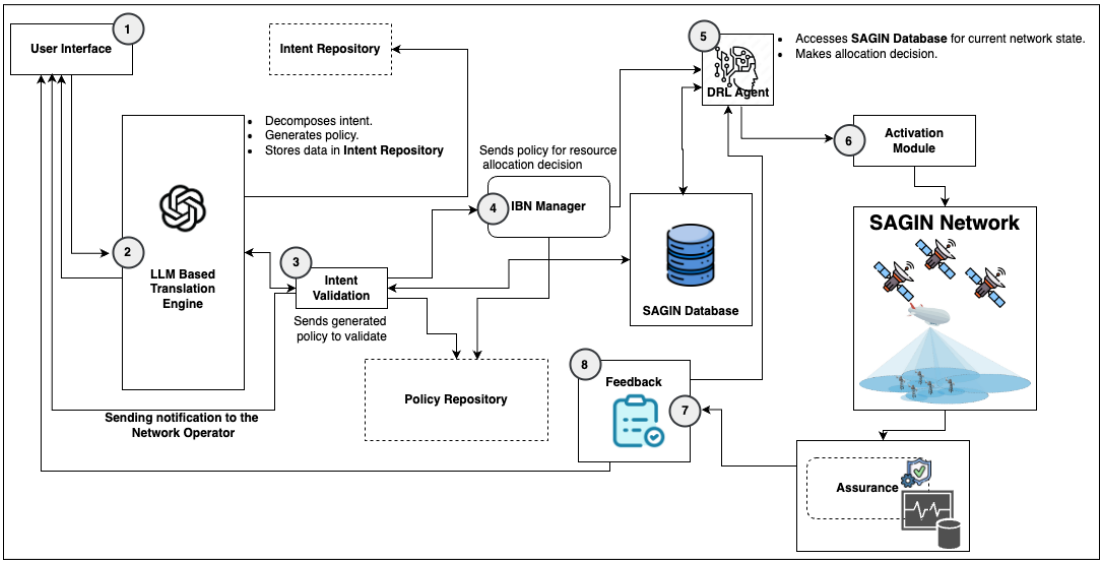

Fig. 1: LLM based IBN systm architecture

III. LLM Based IBN Mechanism

A. User Interface

The LLM-based IBN system, as illustrated in Fig. 1, provides an intuitive interface for network operators to seamlessly input resource requests using natural language. For instance, operators can specify requirements like "I need resources for three XR applications, each requiring 8 vCPUs, 16 GB memory, 5 Gbps bandwidth, and 10 ms latency." The system employs a Generative AI agent to clarify and confirm specifics such as CPU, memory, bandwidth, and latency requirements.

This interactive design avoids errors by providing real-time feedback and intelligent suggestions, ensuring precise resource requests. Before deployment, the system cross-verifies constraints to guarantee compliance with Virtual Network Request (VNR) specifications, enabling efficient resource allocation and enhancing operator confidence. "'

B. LLM-Based Translation Engine

The LLM-Based Translation Engine in our system model acts as a core component responsible for converting high-level intents expressed by network operators into detailed, actionable network configurations. This engine ensures that operators no longer need to understand or handle the intricate details of the underlying network configurations, making the orchestration of resources across the SAGIN network highly efficient and automated. As depicted in Fig. 1, the Translation Engine receives intents from the User Interface. These intents, expressed in natural language, include requirements for virtual network resources, such as CPU, memory, bandwidth, and latency. The LLM-based system decomposes these intents into machine-readable policies using a Generative AI-based model (e.g., GPT-like models). The process is broken down into key stages, ensuring that each intent is correctly interpreted, with no ambiguities, before proceeding to the next steps of validation and execution.

The core mathematical model of the Translation Engine can be described as a mapping function:

$$f_{\text{LLM}} : I_{\text{user}} \rightarrow P_{\text{network}} \tag{1}$$

Where I_{user} represents the set of user intents expressed in natural language and P_{network} is the set of network policies generated for resource allocation.

1) Steps in LLM Translation: The first step in the LLM-based translation process is **Intent Decomposition**. The Large Language Model (LLM) takes the user's input, I_{user}, and breaks it down into key resource requirements necessary for network allocation. These include the number of virtual CPUs (vCPU_{req}), memory requirements (Mem_{req}), bandwidth requirements (BW_{req}), and latency tolerance ($\text{Latency}_{\text{req}}$). The LLM interprets the user's natural language request and decomposes it into these specific elements, representing the process mathematically as:

$$P_{\text{generated}} = \text{LLM}(I_{\text{user}}) \tag{2}$$

Next, during **Policy Generation**, the Translation Engine formulates a policy based on the decomposed intent. This policy includes specific constraints to ensure efficient allocation of network resources. The policy is represented mathematically as:

$$P_{\text{network}} = \{\text{vCPU}_{\text{req}}, \text{Mem}_{\text{req}}, BW_{\text{req}}\text{Latency}_{\text{req}}\} \tag{3}$$

$$\text{vCPU}_{\text{req}} \geq \text{vCPU}_{\text{SAGIN}} \tag{4}$$

$$\text{Mem}_{\text{req}} \leq \text{Mem}_{\text{SAGIN}} \tag{5}$$

$$BW_{\text{req}} \geq BW_{\text{SAGIN}} \tag{6}$$

$$\text{Latency}_{\text{req}} \leq \text{Latency}_{\text{SAGIN}} \tag{7}$$

Once the policy is generated, it is passed to the **Validation Module**, where it is checked against the available resources in the SAGIN database to ensure that the requested resources can be allocated efficiently.

C. Intent Validation

The Intent Validation Module verifies the feasibility of generated policies by assessing resource availability across the SAGIN infrastructure, including vCPUs, memory, bandwidth, and latency requirements. It interacts with the SAGIN database to confirm resource sufficiency and, if unavailable, activates the Queue and Notification System to inform the network operator and queue the request until resources are available. Real-time updates keep operators informed, while the Intent Re-construction Module offers suggestions, such as reducing bandwidth or relaxing latency requirements, to adjust infeasible requests. This mechanism ensures efficient queuing, valid policy execution, and operator transparency through a feedback loop.

D. Queue and Notification System

The Queue System is integrated into the Intent Validation Module and automatically queues network resource requests if the available resources are insufficient. This system ensures that requests are processed as soon as resources become available. The Notification System then informs the network operator about the queue status and resource availability, reducing manual intervention.

Mathematically, the queue process is represented as:

$$Q = \begin{cases} P_{\text{operator}}, & \text{if resources available} \\ P_{\text{queue}}, & \text{if resources not available} \end{cases} \tag{8}$$

Once resources are available, the queued request is processed, and the operator is notified that their request is being deployed.

E. Mechanism of DDPG-Based DRL-Agent in LLM-Based IBN System

Once the network policies are validated, the DRL agent steps in to make intelligent decisions about how to allocate resources across satellite, aerial, and terrestrial network segments, ensuring that the generated policies are implemented effectively. The DRL agent operates using a Deep Deterministic Policy Gradient (DDPG) framework, which allows for continuous control and optimization. The state of the network, including real-time parameters such as available bandwidth, CPU resources, and latency, is fed into the agent, which then selects the best action to maximize the overall reward. The reward function is tightly coupled with the Assurance Module, which monitors network performance to ensure that QoS objectives are met. If any violations, such as latency breaches or bandwidth shortages, are detected, the Assurance Module triggers a feedback loop, prompting the DRL agent to adjust the resource allocation strategy and restore compliance. This adaptive mechanism enables the system to continuously refine its policies and provide optimal network performance, even under fluctuating conditions. The state parameter, s_t, encapsulates the real-time status of resources and performance across satellite, aerial, and terrestrial network segments, including available vCPUs, memory, bandwidth, and latency, as well as current VNRs. The state vector s_t is defined as:

$$\begin{aligned} s_t = \{&\text{vCPU}_S, \text{Mem}_S, BW_S, L_S, \text{vCPU}_A, \text{Mem}_A, BW_A, L_A, \\ &\text{vCPU}_T, \text{Mem}_T, BW_T, L_T, \text{Active VNRs}\} \end{aligned} \tag{9}$$

The action parameter, a_t, corresponds to the decisions made by the DRL agent regarding resource allocation. Actions involve dynamically assigning vCPUs, memory, bandwidth, and adjusting latency constraints across the network to satisfy the active VNRs. These actions ensure that the network resources are utilized efficiently and that the specific performance requirements of the VNRs are met. In terms of the action space, the DRL agent selects the optimal resource distribution at each timestep. The action a_t is a vector representing how much of each resource (vCPU, memory, bandwidth, etc.) will be allocated to each segment of the SAGIN network based on current and projected network demands. Formally, this can be represented as:

$$a_t = \{\text{Alloc vCPU}, \text{Alloc Mem}, \text{Alloc BW}, \text{Alloc Latency}\} \tag{10}$$

After the polices have been deployed through activation function, the Assurance Module monitors real-time performance metrics such as latency, bandwidth, and CPU utilization, ensuring that resource allocations comply with predefined QoS targets and triggering feedback loops to the DRL agent upon detecting any QoS violations. This integration maintains network adaptability and performance by enabling continuous adjustments to resource allocations, thereby ensuring consistent QoS across the SAGIN infrastructure.

F. Monitoring QoS Metrics

The system relies on the Assurance Module to track how well the current network performance aligns with

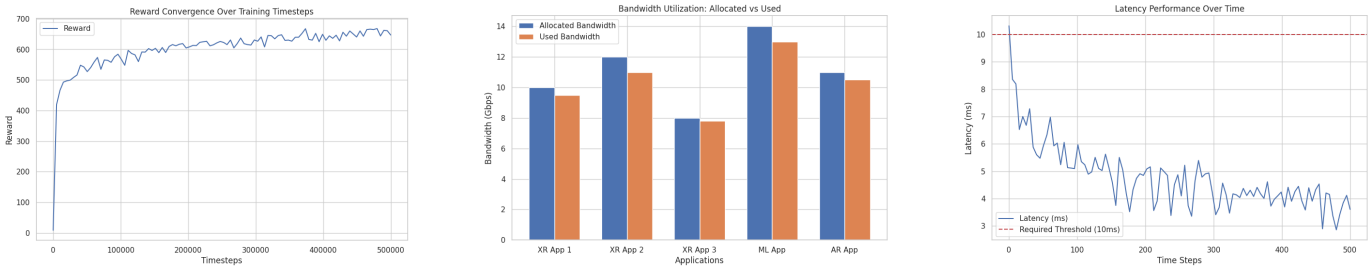

Fig. 2: (a) Reward over time, (b) Bandwidth utilization, (c) Latency, (d) QoS violation, (e) Resource utilization, and (f) VNR acceptance rate

the specified VNR requirements, which include performance targets such as latency and bandwidth. For instance, an XR application may require a latency of 10 ms and 5 Gbps of bandwidth. The Assurance Module continuously compares the real-time performance with these target metrics to calculate the QoS Achieved.

G. QoS Achieved

The QoS Achieved metric quantifies the alignment between real-time network performance and the target metrics specified in the VNR. A higher ratio indicates closer adherence to the desired performance standards.

$$QoS_{\text{Achieved}} = \frac{Perf_t}{Perf_{\text{target}}} \tag{11}$$

H. QoS Violations

QoS Violations occur when real-time network performance significantly deviates from target metrics, such as exceeding latency thresholds or dropping bandwidth below required levels. The degree of violation is calculated by the difference between target and actual performance, with larger deviations incurring heavier penalties.

$$\delta QoS = Perf_{\text{target}} - Perf_t \tag{12}$$

A violation is flagged if δQoS exceeds a predefined threshold ϵ. The penalty for QoS Violations is computed as:

$$P_{\text{violations}} = \beta \cdot \sum_{t=1}^{T} \max(0, \delta QoS_t - \epsilon) \tag{13}$$

I. Resource Wastage

Resource Wastage occurs when the DRL agent allocates more resources (vCPUs, memory, bandwidth) than utilized by the application. This inefficiency is quantified by the difference between allocated and utilized resources and penalized to promote optimal resource usage.

$$W_{\text{resource}} = R_{\text{allocated}} - R_{\text{utilized}} \tag{14}$$

The penalty for resource wastage is calculated as:

$$P_{\text{wastage}} = \gamma \cdot \sum_{t=1}^{T} W_{\text{resource}_t} \tag{15}$$

where γ is the penalty factor and T is the total number of time intervals. The reward function in the DRL agent is designed to maximize QoS Achieved while minimizing both QoS Violations and Resource Wastage. It provides positive reinforcement for maintaining or exceeding QoS requirements and penalizes the agent for violations and inefficient resource usage. The reward function at time t can be expressed as:

$$r_t = \alpha \cdot QoS_{\text{Achieved}_t} - \beta \cdot P_{\text{violations}_t} - \gamma \cdot P_{\text{wastage}_t} \tag{16}$$

J. Integration of the Assurance Module and Feedback Loop

The Assurance Module ensures compliance with predefined QoS targets by continuously monitoring real-time performance metrics such as latency, bandwidth,

and CPU utilization. Upon detecting QoS violations, it triggers a feedback loop, prompting the DRL agent to reassess and dynamically adjust resource allocation strategies. To address conflicting QoS requirements, the module prioritizes critical service-level objectives (SLOs) based on operator-defined weights, ensuring minimal user impact; for example, prioritizing latency for real-time applications like XR. Collaborating with the DRL agent, it reallocates resources in real time, addressing deviations and maintaining QoS compliance under dynamic conditions. A notification system informs operators of significant violations and corrective actions, enhancing transparency, adaptability, and efficient network performance across the SAGIN infrastructure.

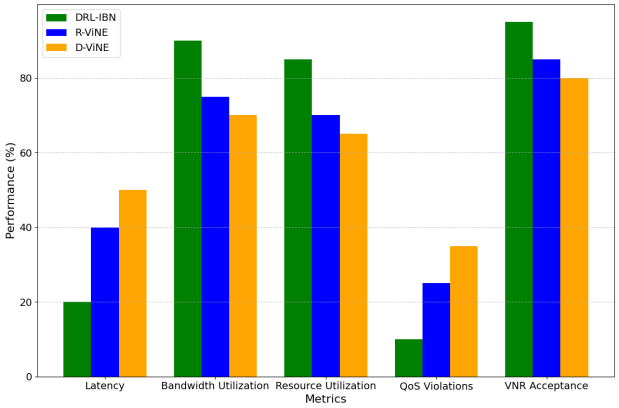

Fig. 3: Comparative Analysis: DRL-IBN System vs R-ViNE and D-ViNE Algorithms.

IV. EXPERIMENTAL RESULTS

We evaluated the proposed LLM-based IBN system integrated with a DRL agent through extensive simulations, focusing on metrics like latency, bandwidth utilization, resource utilization, QoS violations, and VNR acceptance rates. The experiments, conducted in a Python-based environment using TensorFlow, modeled dynamic SAGIN network conditions. The setup included an Intel Core i9-11900K CPU (3.50 GHz), 64GB RAM, and an NVIDIA RTX 3080 GPU. The DRL agent employed the Soft Actor-Critic (SAC) algorithm with hyperparameters: learning rate 1×10^{-4}, batch size 256, discount factor $\gamma = 0.99$, and ReLU-activated policy and Q-networks. A fine-tuned GPT model trained on 10,000 network queries achieved 92% translation accuracy and an 87% reduction in misinterpretations.

Training the DRL agent for 500,000 timesteps allowed optimization of resource allocation strategies. Fig. 4(a) shows reward convergence, indicating learned balance between resource allocation and QoS. Bandwidth utilization Fig. 4(b) remained high, preventing inefficiencies and degradation. Latency performance Fig. 4(c) consistently met a 10ms threshold. QoS violations, initially high, decreased significantly during training Fig. 4(d), demonstrating adaptive learning. Resource utilization Fig. 4(e) improved through dynamic allocation of vCPU and memory. The system achieved a high VNR acceptance rate Fig. 4(f), demonstrating robust dynamic resource management across varying network conditions.

V. COMPARATIVE ANALYSIS

The comparative analysis between the DRL-based system, R-ViNE and D-ViNE algorithms, as shown in the graph Fig. 3 across five key metrics: Latency, Bandwidth Utilization, Resource Utilization, QoS Violations, and VNR Acceptance. The DRL-IBN system significantly outperforms both R-ViNE and D-ViNE in most metrics, achieving the highest bandwidth utilization, resource utilization, and VNR acceptance rate, while maintaining the lowest latency and QoS violations. This demonstrates the DRL-IBN system's superior capability to dynamically manage resources and ensure QoS compliance, making it more effective in handling complex demands within the SAGIN framework compared to the baseline algorithms.

VI. CONCLUSION

In conclusion, the integration of an LLM-based translation engine with a DRL agent in an IBN framework presents a powerful solution for dynamic network resource management. The experimental results confirm that the proposed system can effectively optimize resource allocation, maintain QoS standards, reduce violations, and improve VNR acceptance rates. The agent's ability to learn and adapt to network dynamics makes it a valuable tool for network operators aiming to enhance service quality and operational efficiency. Future research will focus on deploying the system in real-world networks and evaluating its performance under more diverse and challenging conditions.

REFERENCES

[1] Arthur S Jacobs, Ricardo J Pfitscher, Rafael H Ribeiro, Ronaldo A Ferreira, Lisandro Z Granville, Walter Willinger, and Sanjay G Rao. Hey, lumi! using natural language for {intent-based} network management. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 625–639, 2021.

[2] Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

[3] Hocine Mahtout, Mariam Kiran, Anu Mercian, and Bashir Mohammed. Using machine learning for intent-based provisioning in high-speed science networks. In *Proceedings of the 3rd international workshop on systems and network telemetry and analytics*, pages 27–30, 2020.

[4] Elvis A Agüero, Luke Alventosa, Daniel M Harris, and Carlos A Galeano-Rios. Impact of a rigid sphere onto an elastic membrane. *Proceedings of the Royal Society A*, 478(2266):20220340, 2022.

[5] Ching-Nam Hang, Pei-Duo Yu, Roberto Morabito, and Chee-Wei Tan. Large language models meet next-generation networking technologies: A review. *Future Internet*, 16(10):365, 2024.

[6] Sheng Wu, Ning Chen, Ailing Xiao, Haoge Jia, Chunxiao Jiang, and Peiying Zhang. Ai-enabled deployment automation for 6g space-air-ground integrated networks: Challenges, design, and outlook. *IEEE Network*, 2024.