CP-AgentNet: Autonomous and Explainable Communication Protocol Design Using Generative Agents

Dae Cheol Kwon, Xinyu Zhang
Department of Electrical and Computer Engineering, University of California San Diego, San Diego, USA
Emails:{dckwon, xyzhang}@ucsd.edu

Abstract—**Although DRL (deep reinforcement learning) has emerged as a powerful tool for making better decisions than existing hand-crafted communication protocols, it faces significant limitations: 1) Selecting the appropriate neural network architecture and setting hyperparameters are crucial for achieving desired performance levels, requiring domain expertise. 2) The decision-making process in DRL models is often opaque, commonly described as a 'black box'. 3) DRL models are data hungry. In response, we propose CP-AgentNet, the first framework to employ generative agents as autonomous decision-makers for communication protocol design. This approach addresses these challenges by creating an autonomous system for protocol design, significantly reducing human effort. As practical use cases, we developed LLMA (LLM-agents-based multiple access) and CPTCP (CP-Agent-based TCP) tailored for heterogeneous environments. Our comprehensive simulations have demonstrated the efficient coexistence of LLMA and CPTCP with nodes using different types of protocols, as well as enhanced explainability.**

I. Introduction

Conventional communication protocols are meticulously established through standardization processes. However, challenges arise from the explosive demands for data traffic and ever-increasing network heterogeneity.Typically designed for a single type of network, standard communication protocols lack the adaptability to coexist in diverse networks. Enabling operations across heterogeneous networks concurrently necessitates the creation of a new protocol for each coexistence scenario, demanding significant expertise and effort.

DRL (deep reinforcement learning) has emerged as a powerful tool for devising communication protocols, which are essentially decision-making algorithms executed by the agents, i.e., network nodes. Despite their potential, DRL methods have several drawbacks. First, choosing an appropriate neural network architecture and tuning hyperparameters are crucial yet often require tedious trial-and-error experimentation. Moreover, architectures optimized for a specific environment often generalize poorly to unseen conditions, necessitating costly redesign or retraining [1], [2], thus limiting practical adaptability. Second, the decision-making process in DRL models is often considered a "black box" [3], making it difficult to interpret or verify the rationale behind actions. Third, DRL models are data hungry, requiring not only large-scale data collection but also labor-intensive labeling and preprocessing. Data acquisition may introduce non-trivial overhead, potentially disrupting normal network operations.

Recently, generative agents—specifically those empowered by large language models (LLMs), hereafter referred to as LLM-agents—have gained attention for their ability to reason, plan, and act in interactive environments. However, their potential in communication protocol design remains largely unexplored. While recent work such as NADA [4] leverages LLMs to automate the design of DRL agents, these approaches remain fundamentally constrained by the inherent limitations of DRL itself.

In this paper, we introduce CP-AgentNet[1], the first framework designed to create communication protocols using LLM-agents as autonomous decision-makers. CP-AgentNet addresses the inherent limitations of both handcrafted and DRL-based protocols. By leveraging the in-context learning capabilities of LLMs, these LLM-agents can quickly interpret and adapt to instructions provided through context or interaction sequences, eliminating the need for explicit retraining when conditions change. Our framework requires crafting only a few demonstrations, capitalizing on the strong generalization abilities of LLM-agents to efficiently guide their training. This significantly reduces the effort involved in selecting neural network architectures, tuning hyperparameters, and gathering, labeling, and preprocessing large datasets. Additionally, since the outputs generated by LLM-agents are in natural language, they can be easily interpreted and understood by humans, greatly enhancing interpretability and facilitating straightforward interactions.

However, despite substantial advantages of using LLM-agents, their use still presents significant challenges. First and foremost, although the LLM-agent is capable of handling tasks across various disciplines, a single LLM-agent struggles to manage multiple tasks simultaneously [5]. In particular, an agent in this framework must make transmission decisions while concurrently monitoring the network environment, adding complexity to its operational tasks. To address this, we employ multi-agent role-play, an approach that has already proven effective in domains such as programming and mathematical reasoning. Harnessing multiple agents is advantageous, as each agent plays a distinct role, thereby simplifying complex tasks [6]. Additionally, this approach significantly enhances explainability by making the decision-making process traceable.

Second, establishing an autonomous design framework using LLM-agents presents unique challenges, primarily due to the absence of explicit optimization methods such as gradient descent, which are commonly used in traditional DRL to systematically improve policies. Existing LLM-agent frameworks often compensate for this limitation by relying heavily on

[1] All prompts used in CP-AgentNet are available at: https://github.com/BigIron0315/CP-AgentNet

979-8-3315-0376-5/25/$31.00 ©2025 IEEE

explicit human feedback [7], [8], thereby requiring substantial user effort. To address this fundamental challenge, we introduce *Progressive Strategy Augmentation (PSA)*, an approach specifically designed for LLM-agents. Rather than updating parameters via gradient-based optimization, PSA utilizes a structured self-reflection mechanism [9] to incrementally refine strategies. Guided by carefully crafted prompts, the LLM-agent autonomously reviews previous episodes, identifies underlying reasons for suboptimal outcomes, and progressively revises its strategies. The refined strategies obtained through multiple self-reflection episodes are subsequently aggregated, and redundant or conflicting components are systematically identified and removed. In essence, PSA serves as a symbolic counterpart to gradient descent, leveraging the generative reasoning and in-context learning capabilities inherent to LLMs. Another method we employ for autonomous design is *Autonomous Strategy Implementation (ASI)*. LLM responses can often be unpredictable, especially when prompts solicit open-ended design strategies. ASI plays a critical role in such cases by leveraging the LLM's programming capabilities to autonomously interpret and implement the generated strategies, thereby maintaining autonomy without requiring human intervention.

Lastly, to address the inherent variability of LLMs, which stems from their probabilistic token prediction, we employ an LLM ranker [10] to enhance consistency and mitigate planning errors. Inspired by The Wisdom of Crowds [11]—which suggests that large groups often outperform individual experts in problem-solving, decision-making, and prediction—we apply this principle to improve CP-AgentNet's stability. The LLM ranker evaluates multiple output candidates and selects the most reliable option as the final output. This approach significantly enhances the consistency of our framework.

As a use case of CP-AgentNet, we designed the LLMA (LLM-agents-based multiple access) protocol for heterogeneous networks. To verify the effectiveness of the LLMA protocol, we conducted comprehensive simulations across various scenarios. Specifically, we first established the ideal operation of an AWARE node—a hypothetical node equipped with complete knowledge of the environment. We then compared the performance of the LLMA node to that of the AWARE node by measuring RMSE (root mean square error). As a benchmark, we also assessed the performance of a DLMA node, which employs the DRL-based method proposed in [1]. Our results indicate that the LLMA node performs closer to the ideal than the DLMA node. Remarkably, in dynamic environments where nodes using different protocols intermittently join or leave, the LLMA node adapts more swiftly and exhibits a significantly lower RMSE value of 0.0476–just 21.0% of the RMSE of 0.227 resulting from the DLMA node.

In addition, we designed CPTCP (CP-Agent-based Transmission Control Protocol) to coexist with different types of TCP algorithms. There are mainly two types of TCP algorithms: loss-based and delay-based TCP. We show that CPTCP enables effective coexistence with both types of TCP algorithms. Moreover, this demonstrates that our framework is not limited to designing a specific protocol.

In summary, the main contributions of this paper are:

- We introduce CP-AgentNet, the first framework designed to create communication protocols using LLM-agents as autonomous decision-makers. CP-AgentNet employs multi-agent role-play to efficiently process tasks and facilitate the explainable design of communication protocols.
- We propose PSA and ASI to facilitate autonomous design. These enable LLM-agents to develop strategies and implement them to optimize network performance, without the need for human intervention.
- We have developed LLMA and CPTCP, specifically designed for a heterogeneous environment using CP-AgentNet. We demonstrate that nodes using these protocols can efficiently coexist with other nodes operating different protocols.

II. System Design

A. CP-AgentNet Architecture

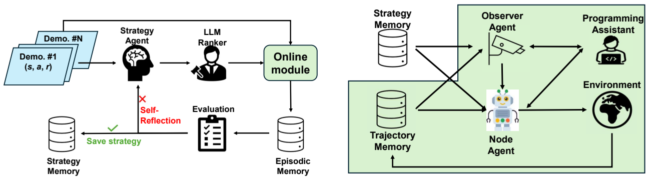
CP-AgentNet comprises a CP-agent and three distinct types of memory. The CP-agent includes multiple agents such as a strategy agent, an observer agent, and a node agent, along with a programming assistant, each assigned specific roles to perform tasks. These agents are empowered by the LLM, each using tailored prompts suited to their individual roles.

Strategy Agent. The strategy agent is responsible for developing and refining strategies. It advises the node agent on selecting appropriate actions based on trajectory data and environmental changes and suggests methods to escape from suboptimal actions.

Node Agent. The primary role of the node agent is to determine the actions. Although the node agent generally adheres to the strategy, it relies on the observer agent for assistance in making the final decisions. The node agent adapts its action every T slots, a period which we refer to as the *query period*. Once established, the action is maintained for the duration of the query period.

Observer Agent. The observer agent diligently monitors the environment, focusing on the convergence of the action, any environmental changes, and notable occurrences. Environmental changes occur when new nodes join the network or existing nodes leave the network, or when other nodes change their parameters. Although the observer agent may not identify the specific changes, it is adept at detecting when changes occur. Notable occurrences can be any anomalies or outliers that the observer agent detects. For instance, if a specific slot is consistently overused or not used at all, the observer agent identifies this irregularity. Upon identifying such environmental changes or irregularities, the observer agent assists the node agent in adjusting its action accordingly.

Programming Assistant. Given that LLMs are not adept at complex mathematical tasks, we employ a programming assistant built upon LLMs. This agent supports other agents by solving mathematical problems using advanced programming skills. It transforms tasks into programming scripts and returns



(a) Offline stage

(b) Online stage

Fig. 1: Overall workflow of CP-AgentNet

the executed results to the initiating agent. Additionally, this agent is integral to ASI, a process where the CP-Agent autonomously implements strategies.

Memory. CP-AgentNet employs three types of memories: strategy memory, episodic memory, and trajectory memory. As their names suggest, each memory type is dedicated to storing strategies, episodic data, and trajectories, respectively.

B. CP-AgentNet Workflow

The operation of consists of two stages: an offline stage and an online stage, as illustrated in Fig. 1. In the offline stage, LLM-agents develop strategies that can be effectively utilized during the online stage. In the online stage, LLM-agents determine the optimal action based on these strategies.

1) Generating Few-Shot Demonstrations: To enable CP-AgentNet to learn communication strategies efficiently, we generate few-shot demonstrations leveraging simulation-based action-reward sampling and heterogeneous protocol representations. This process ensures that the LLM-agent can generalize across different network conditions and protocols with minimal training data. While our framework designed MAC protocol and TCP as use cases, applying it to other protocols would require generating separate few-shot demonstrations tailored to their characteristics.

a) Simulation-Based Demonstration Generation: Instead of relying on large-scale datasets, we construct few-shot demonstrations through targeted simulation by leveraging in-context learning and generalization capabilities of LLM-agent. This process does not require domain expertise, but only a basic understanding of the underlying protocol is sufficient. Given an environment characterized by state space S and action space A, we sample a set of actions and derive the corresponding rewards from the simulation. Specifically, for each scenario, we:

- Sample K actions, $a_1, a_2, \ldots, a_K$, from the defined action space A. (e.g., in a slotted ALOHA setup with $N = 3$ nodes, $K = 11$ actions can be uniformly sampled from the range $[0, 1]$ with step size 0.1 representing transmission probabilities)
- Evaluate each action in the simulated environment to obtain the corresponding reward values, $r_1, r_2, \ldots, r_K$.
- Construct state-action-reward tuples (s, a, r), where $s \in S$, $a \in A$, and r is obtained from simulation feedback.
- Store these tuples as part of the few-shot demonstration set $\mathcal{D}$ to be utilized in in-context learning.

b) Heterogeneous Demonstration Generation: To ensure robustness across diverse network protocols, we generate demonstrations for both MAC and TCP protocols. Our framework constructs protocol-specific few-shot demonstrations that capture variations in communication behaviors.

For MAC protocols, we create total 4 demonstrations:

- Three protocol-specific demonstrations for CSMA, TDMA, and ALOHA.
- One dynamic demonstration that adapts to changing network conditions, ensuring adaptability to variations.

For TCP protocols, we generate total 3 demonstrations:

- Two protocol-specific demonstrations for Loss-base TCP and Delay-based TCP.
- One dynamic demonstration to account for variations in congestion and network dynamics.

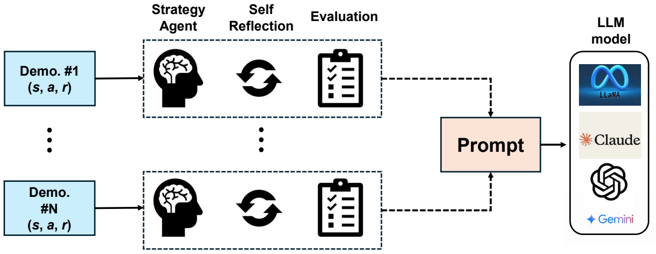
2) Strategy generation: Strategy generation focuses solely on developing effective strategies based on the provided few-shot demonstrations. Rather than learning from a vast dataset, our approach enables LLM agents to infer strategies by analyzing various heterogeneous protocols and environmental scenarios. Through this process, the agent formulates strategies that can be effectively applied during the online adaptation stage. By distilling knowledge from limited demonstrations, the strategy generation phase ensures efficient adaptation while significantly reducing training time. We define a strategy π as a function that maps the state space S to the action space A, i.e., $\pi : S \to A$. Given a set of few-shot demonstrations $\mathcal{D} = (s_i, a_i, r_i, s'_i)_{i=1}^{N}$, the optimal strategy is generated by maximizing the expected rewards over the sampled actions:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s,a,r)\sim\mathcal{D}}[R(s, a)] \tag{1}$$

where $R(s, a)$ represents the reward function. Unlike traditional reinforcement learning, which requires extensive training, CP-AgentNet enables LLM-agents to leverage few-shot demonstrations and their generalization capabilities to construct an initial strategy.

3) Refining Strategies: After generating an initial strategy, its effectiveness is evaluated by measuring the expected reward $J(\pi) = \mathbb{E}_{s\sim P, a\sim\pi(s)}[R(s, a)]$, where P represents the state distribution. If the obtained performance does not meet the predefined optimal threshold J_{opt}, then the strategy undergoes a refinement process. This is accomplished through self-reflection [9], enabling the LLM-agent to analyze its past decisions and adjust accordingly. The mathematical formulation conceptually represents a gradient-descent process, in which each candidate strategy is iteratively adjusted until it approaches its local optimum, ensuring convergence toward a good strategy. The refinement process updates the strategy iteratively: $\pi^{(t+1)} = \pi^{(t)} + \lambda \nabla J(\pi^{(t)})$, where t represents the iteration index and λ is the step size for adjusting the policy. The process continues until the strategy reaches the desired performance: $J(\pi^{(t)}) \geq J_{\text{opt}}$ or $t = N_{\max}$.

By iteratively refining the strategy, CP-AgentNet ensures that suboptimal policies are corrected, allowing for efficient



Prompt Used in PSA

The strategy agent developed for the node agent $\{n\}$ individual strategies based on separate demonstrations as listed below:

- Strategy 1: ...
- ...
- Strategy n: ...

Your task now is to create a **combined strategy** that:

- Incorporates essential elements from all provided strategies.
- Eliminates redundancy and overlap efficiently.
- Maintains the general applicability of the combined strategy to different environments, where optimal actions vary, but the action-reward relationship remains similar.

Ensure that no individual strategy is omitted from the final combined version—only remove redundant or repetitive parts. Provide your combined strategy clearly and succinctly.

Fig. 2: Structure (top) and prompt (bottom) of PSA that synthesizes multiple strategies by resolving conflicts and eliminating redundancy.

adaptation to dynamic environments. The refinement step enables the agent to incorporate feedback, adjust its decision-making process, and improve strategy performance without requiring extensive online learning. An example prompt for strategy generation.

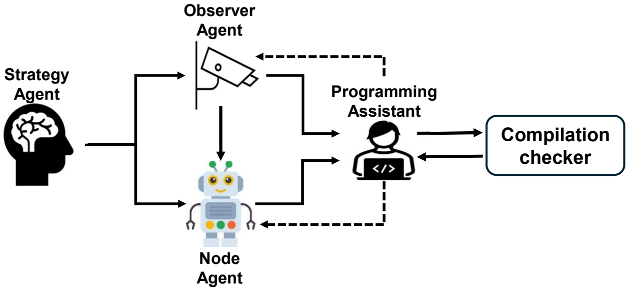
4) Online Adaptation: Once strategies are learned in the offline stage, the LLM-agent dynamically selects actions during the online stage while adapting to environmental variations. This adaptation is supported by the observer agent, which monitors the node agent's actions, evaluates environmental conditions, and assists in decision-making. During execution, the node agent follows the pre-learned strategy π^*, while the observer agent provides real-time feedback O_t to guide adaptation. Instead of directly modifying the action, the observer adjusts the decision-making process by supplying contextual insights to the node agent, ensuring robust responsiveness to dynamic conditions. Notably, in the online stage, the strategy agent and episodic memory are inactive, and the system operates solely based on the pre-learned strategies and real-time observer feedback.

C. Autonomous Design

To enable fully autonomous protocol design, CP-AgentNet must not only generate strategies but also evaluate and manage them without relying on explicit optimization techniques such as gradient descent or human feedback. After a strategy is generated and refined through self-reflection, the system must decide whether to retain, discard, or integrate it into its current strategy set. To support this, we propose Progressive Strategy Adaptation (PSA), an iterative mechanism that manages the evolution of the strategy set. As illustrated in Fig. 2, each few-shot demonstration leads to a candidate strategy, and PSA determines how to update the strategy set accordingly. This update is defined as:

$$\Pi^{(t+1)} = \Big(\Pi^{(t)} \cup \{\pi_{\text{new}}\}\Big) \setminus \{\pi_{\text{obsolete}}\} \qquad (2)$$

Here, π_{new} is a newly generated strategy, and $\pi_{obsolete}$ represents redundant or conflicting entries filtered out during self-assessment. While this process resembles batch-based learning in ML, CP-AgentNet achieves adaptation with only a few demonstrations. By iteratively applying PSA, CP-AgentNet constructs a concise, high-quality strategy set that supports scalable protocol design.



(b) Prompt Used in ASI

Strategy to escape from suboptimal action is $\{...\}$.
Based on the provided strategy, create a Python function named {`function_name`}.

- Take the initial action as input.
- Return a final action {`range`} that aligns with your strategy to escape suboptimal decisions.
- Clearly implement your strategy logic, ensuring it can be generalized to other environments.

Provide only the Python code for the function without additional explanations or comments.

Fig. 3: Structure (top) and prompt (bottom) of ASI. Implementation is delegated to the programming assistant by the node or observer agent.

In addition, since the LLM's responses can be unpredictable, CP-AgentNet requires an autonomous mechanism to implement generated strategies without human intervention. To this end, we introduce Autonomous Strategy Instantiation (ASI), in which the LLM generates executable functions that are stored and reused throughout the system's operation. This design avoids redundant queries and ensures efficient execution without manual involvement. As illustrated in Fig. 3, the strategy agent develops high-level strategies for both the node agent and the observer agent. If a strategy requires assistance from the programming assistant, the node or observer agent consults the programming assistant. Upon request, the programming assistant implements the strategy and checks for compile errors using an external tool. If an error is detected, the assistant iteratively corrects it. Once validated, the resulting function is invoked by the original requester as needed. For example, if the strategy agent suggests the ϵ-greedy algorithm for an exploration-exploitation trade-off, the node agent implements this with the assistance of a programming assistant, which translates it into a function that operates autonomously at runtime. This end-to-end process not only enables autonomous execution but also enhances robustness; by combining the compilation checker and function generation with the LLM ranker (see §II-D), ASI achieves significantly improved reliability.

D. *Stability of CP-AgentNet*

Despite their advantages, LLMs' inherently inconsistent outputs can compromise the stability of systems like CP-AgentNet. This inconsistency arises from their fundamental mechanism of predicting the next token's probability distribution, even at zero temperature. Such unpredictability can degrade CP-AgentNet's performance or lead to operational failures. To address this, we implement an LLM ranker [10], inspired by *The Wisdom of Crowds* [11]. The LLM-agent queries the LLM twice, reversing the order of inputs to mitigate the influence of input sequencing on response generation. Both responses are then evaluated by another LLM acting as a judge, which selects the most optimal strategy autonomously.

a) Querying the LLM with Different Input Orders: Let Q be the original query from the CP-agent. To reduce bias from input ordering, we construct two variations: $Q_1 = Q$ and $Q_2 = \text{ReverseOrder}(Q)$. For each query, the LLM generates a response: $R_1 = \text{LLM}(Q_1)$ and $R_2 = \text{LLM}(Q_2)$, where R_i represents the strategy proposed by the LLM.

b) Selecting the Best Strategy Using Another LLM as a Judge: Instead of manually defining evaluation criteria, we employ another LLM as a strategy judge J_{LLM}, which compares two candidate strategies and returns the preferred one as $R^* = J_{\text{LLM}}(R_1, R_2)$. The selection is entirely autonomous, relying on the judge LLM's ability to infer which response is more optimal.

c) Feasibility in Offline Processing: While the LLM ranker significantly improves decision consistency and overall reliability, it also increases processing time. However, since the LLM ranker is primarily utilized during the offline stage—where time constraints are less stringent—its impact on real-time performance is minimal. Specifically, the ranker is applied during the formulation of transmission strategies rather than during the time-sensitive online stage. The effectiveness of the LLM ranker will be demonstrated in Section IV.

III. Use case protocols with CP-AgentNet

In this section, we describe how we tailor the CP-AgentNet framework to design specific protocols. As use cases, we develop LLMA, a multiple access protocol, and CPTCP, a transmission control protocol, both designed for heterogeneous environments using CP-AgentNet.

A. *LLMA*

1) Heterogeneous networks: We consider heterogeneous networks where multiple nodes transmit data packets through a shared channel, fundamentally adhering to the assumptions outlined in DLMA [1]. We assume that the time slots are synchronized among these nodes, each using different MAC protocols. Each node initiates its data packet transmission at the beginning of a slot and completes it within the same slot. Every node consistently has a data packet to transmit. If more than one node transmits a data packet using the same channel simultaneously, a collision occurs, rendering those transmissions unsuccessful. We consider heterogeneous network scenarios where at least one "heteronode" coexists with slotted ALOHA or TDMA nodes. A slotted ALOHA node transmits a data packet with a fixed probability q in each time slot, while a TDMA node transmits data packets at X specific slots out of a frame composed of ten slots. The term "heteronode" refers to nodes such as AWARE node, DLMA node, or LLMA node. The AWARE node, considered an ideal solution, has complete knowledge of the environment, including the number of slotted ALOHA and TDMA nodes and their parameters, represented by q and X. The optimal behavior of the AWARE node can be derived as shown in [12]. Consequently, our goal is to enable the LLMA node to perform as closely as possible to the AWARE node. A DLMA node utilizes the DLMA protocol, which is developed using a DRL method, to make transmission decisions. It can listen to the channel and observe whether other nodes' transmissions are successful or if the channel is idle. Similarly, an LLMA node employs the LLMA protocol, which is developed through our CP-AgentNet framework. All assumptions for LLMA are the same as for DLMA unless specified otherwise.

Additionally, we consider more complex scenarios where the heteronode coexists not only with TDMA and ALOHA nodes but also with CSMA (carrier-sense multiple access), EB (exponential backoff window)-ALOHA, and FB (fixed window)-ALOHA nodes. The CSMA node generates a random value of w within the range [0, W-1], and senses the channel is busy or not. If it is sensed as idle, w is decreased by one, otherwise it is frozen. If this value reaches to zero, the CSMA node transmits data. Only the CSMA node has the capability to sense the channel, and the sensing time is negligible compared to the packet length. An FW-ALOHA node generates a random value of w within the range [0, W-1] after transmission, it then waits for w slots for the next transmission. An EB-ALOHA node's operation is basically the same as that of an FW-ALOHA node. However, an EB-ALOHA node doubles its window size upon a collision with other nodes, up to a maximum window size of $2^m W$. In these scenarios, only DLMA and LLMA nodes are used as heteronodes, as we do not derive the ideal behavior for the AWARE node.

2) Objective function of heteronode: We set the heteronode objective to maximize α-fairness throughput with $\alpha = 1$, rather than solely maximizing total throughput. The α-fair function is

$$g_\alpha(x) = \begin{cases} \log x, & \alpha = 1, \\ \frac{x^{1-\alpha}}{1-\alpha}, & \text{otherwise}, \end{cases} \quad (3)$$

and the overall objective is $f(\vec{x}) = \sum_{i=1}^{N} g_\alpha(x_i)$, where N is the number of nodes and x_i is the throughput of node i. Setting $\alpha = 0$ maximizes total throughput, while $\alpha = 1$ maximizes proportional fairness.

3) LLMA design: The goal of LLMA is to optimize the action of a node agent based on strategies and real-time environmental feedback. In this context, the action determines the likelihood of transmitting in each time slot, setting the transmission probability for each slot. At time step t, the action a_t is selected from continuous range $[0, 1]$. The observation space at time t, denoted as o_t, includes 'S' for 'SUCCESSFUL', 'C' for 'COLLIDED', or 'I' for 'IDLE'. It is important to note that this observation o_t is distinct from the observer feedback O_t, where O_t represents higher-level real-time feedback used by the LLM agent to adjust its action. In this setup, the observation space directly serves as the state space S. Rewards from the environment, given after an action, are defined as follows: $r_{t+1} = 1$ if o_t is 'S', and $r_{t+1} = 0$ if o_t is either 'C' or 'I'. Assuming there are N nodes in the network, the node agent receives an N dimension vector of rewards from the environment. Each element of the vector represents the transmission result of one particular node. The actions taken by the node agent and the rewards from the environment are essentially the same as those in DLMA [1].

The node agent adapts its action every T slots, referred to as the *query period*. For decisions to remain valid, this period should be shorter than the coherence time—defined here as the average duration over which the active user set remains stable. Since user arrivals and departures typically occur on the order of several seconds, a 1-second query period is a practical. During this time, the accumulated actions and rewards, along with the observations, are stored in the trajectory memory. Before the node agent adapts the action at the end of the query period, the observer agent monitors three aspects: the convergence of the plan, any environmental changes, and any notable occurrences. With assistance from the programming assistant, if the observer agent determines that the transmission has converged, the node agent ceases adaptation and implements the strategy to escape from suboptimal as described in §II-C. If the observer agent detects any environmental changes, it notifies the node agent to adjust the plan accordingly. When coexisting with a TDMA node, if the LLMA node's observer agent notices that specific slots are consistently used, it advises the node agent to avoid these slots. Through collaboration with the observer agent and the programming assistant, the node agent decides the action for each slot.

B. CPTCP

1) Heterogeneous TCPs: We consider heterogeneous TCP scenarios in which CPTCP can coexist with different TCP algorithms. There are mainly two types of TCP algorithms: loss-based and delay-based TCP. Fairness issues arise when a loss-based TCP flow coexists with a delay-based TCP flow, as the loss-based TCP reactively reduces its congestion window size, while the delay-based TCP proactively reduces it [13]. Although previous studies have attempted to overcome this problem, our goal is to mitigate it by leveraging the in-context learning capability of the LLM, ultimately enabling CPTCP to effectively coexist with both loss-based and delay-based TCP flows without the need for hand-crafted design requiring human effort.

2) Objective function of CPTCP: Since we focus on fairness when coexisting with different types of TCP flows, our objective is to maximize Jain's fairness index, which is defined:

$$f(x_1, ..., x_N) = \frac{(\sum_{i=1}^{N} x_i)^2}{N \sum_{i=1}^{N} x_i^2} \quad (4)$$

where x_i is the throughput of ith flow and N is the number of flows. In a completely fair scenario, $f(\vec{x}) = 1$. Conversely, if a single TCP flow monopolizes the link, $f(\vec{x}) = \frac{1}{N}$. Thus, the closer $f(\vec{x})$ is to 1, the greater the fairness; conversely, the closer $f(\vec{x})$ is to $\frac{1}{N}$, the lower the fairness.

3) CPTCP Design: The design procedure for CPTCP follows the same foundational principles as LLMA, with modifications tailored to its specific objectives. Accordingly, the objective of CPTCP is to optimize the action of a node agent based on strategies and real-time environmental feedback. In this context, the action determines the congestion window. At time step t, the action a_t represents the congestion window size, which is a positive integer within a predefined range. Specifically, we define the action space as $a_t \in [1, C_{\max}]$, where $C_{\max}$ is the maximum congestion window size, determined by factors such as the receiver's advertised window and network conditions. While the theoretical limit can be large in high-bandwidth networks, practical implementations impose constraints based on these factors. While the theoretical maximum congestion window can be large in high-bandwidth networks, practical implementations impose constraints based on these factors. The observation space at time t, denoted as o_t, consists of two key metrics: a, the number of received acknowledgments (ACKs), and r, the round-trip time (RTT), forming a vector representation $o_t = [a, r]$. The environment provides rewards following each action, $r_{t+1} = \log(a) - \beta r$. Here, the logarithmic function promotes fairness in throughput allocation, while β serves as a tunable parameter balancing throughput and RTT trade-offs, and is empirically set to 0.5 in our experiments. If the observer agent detects any environmental changes, it notifies the node agent, which then adapts the action accordingly. The final decision is made based on the updated conditions, ensuring an optimized response to dynamic network environments.

IV. Evaluation

A. Simulation Setup

To evaluate the effectiveness of CP-AgentNet, we conducted experiments with LLMA and CPTCP. CP-Agent is powered by GPT-4o, with the temperature value set to zero to minimize inconsistencies.

1) LLMA: We focus on heterogeneous networks incorporating ALOHA and TDMA protocols, alongside heteronodes such as AWARE nodes, DLMA nodes, and LLMA nodes. Since our MAC protocol evaluation already includes direct quantitative comparisons against both an ideal baseline (AWARE node) and a representative DRL-based method, additional comparisons against other methods are unnecessary to sufficiently validate the performance of CP-AgentNet. Specifically, within these networks, only one type of heteronode coexists with nodes operating under TDMA or ALOHA protocols. Each frame consists of ten time slots, each lasting 1ms. Specifically for TDMA nodes, we consider a scenario where only one TDMA node utilizes slots 3 and 5, denoted as $X = 2$. For ALOHA nodes, we set the transmission probability to $q = 0.2$ except for the massive nodes case. For a massive nodes scenario, q is set to 0.02. Moreover, to demonstrate the robustness of LLMA nodes within complex heterogeneous networks, we conduct additional tests involving nodes that utilize CSMA, FW-ALOHA, and EB-ALOHA. We set the contention window size W to 4 for FW-ALOHA. For EB-ALOHA, $W_{\min} = 2$ and $W_{\max} = 32$ with a maximum backoff stage m of 4. For CSMA, we follow the IEEE 802.11 standard with $W_{\min} = 16$ and $W_{\max} = 1024$. The maximum backoff stage is set to 7, but the last two stages are capped at $W_{\max}$. Following the setting in DLMA [1], we use throughput as the performance metric in this paper. However, since the objective is not to maximize total throughput but to maximize α-fairness throughput for heteronodes, we use RMSE as an additional metric to measure how closely the behavior approximates the ideal operation of the AWARE node.

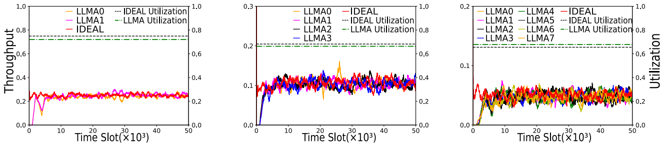
2) CPTCP: We first evaluate the throughput of homogeneous TCP configurations, followed by assessments of various combinations involving different TCP algorithms. We utilize a network capacity of 1 Mbps, with throughput and Jain's fairness index serving as the evaluation metrics. Since the primary goal of our CPTCP evaluation is to demonstrate its capability to coexist with diverse TCP variants, we benchmark CPTCP against representative loss-based (TCP CUBIC [14]), delay-based (TCP BBR [15]), and Cx-TCP [13] algorithms.

B. LLMA results

We initially define various scenarios to assess the robustness of LLMA nodes, which are detailed in Table I along with their respective RMSE results. In the following, we will review these scenarios and analyze the corresponding outcomes.

1) ***Homogeneous scenario:*** We evaluate multiple LLMA nodes in configurations of 2, 4, and 8, and compare the results with the ideal behavior of the AWARE node. After conducting separate simulations with AWARE nodes in the same configurations, we plot the throughput of one AWARE node alongside the LLMA nodes for comparison in Fig. 4. The throughput of the LLMA nodes closely aligns with the ideal behavior, demonstrating that the emerging behavior of the LLMA nodes matches the optimal behavior.

2) ***Heterogeneous scenario:*** To demonstrate that LLMA nodes can coexist with nodes using different protocols, we explore various combinations involving TDMA, ALOHA nodes, and heteronodes. The illustrations in Fig. 5, arranged from left to right, depict an AWARE node, a DLMA node, and an LLMA node. Although the LLMA node's behavior visually appears similar to that of the AWARE nodes, we used RMSE to objectively confirm this similarity. The results for all other scenarios are presented in the Table I. Except for one scenario (1H+1T), the RMSE values of the LLMA node are smaller than those of the DLMA node, indicating that the LLMA node's operation is closer to that of the AWARE node. In scenarios where multiple heteronodes coexist with different nodes, the RMSE values of LLMA nodes are approximately 49.4% and 6.8% lower than those of DLMA nodes in the 2A+2H and 1T+2A+3H scenarios, respectively. In summary, our evaluations in heterogeneous scenarios, similar to those examined in [1], demonstrate that the CP-AgentNet framework manages heterogeneous network environments more effectively than traditional DRL methods.

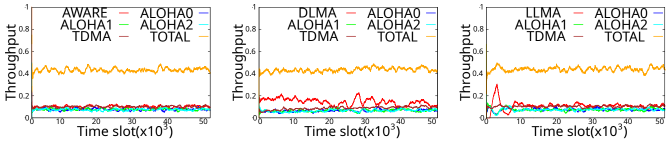


(a) 2 LLMA node (b) 4 LLMA nodes (c) 8 LLMA nodes

Fig. 4: LLMA nodes in homogeneous environment

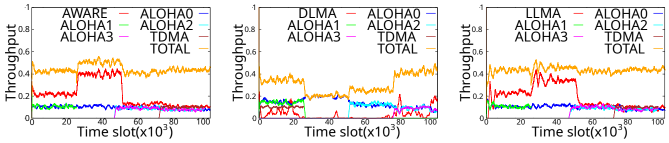
3) ***Complex Heterogeneous scenario:*** We assess whether the LLMA node can function effectively in more complex scenarios. Initially, we increase the number of ALOHA nodes to 20 to test the robustness of the LLMA node, setting the parameter $q = 0.02$ for these ALOHA nodes. In this scenario of increased complexity, the LLMA node achieves near-optimal performance, evidenced by an RMSE value of 0.044, which is lower than 0.052 observed for the DLMA node. Additionally, we introduce nodes using various other protocols, such as CSMA, EB-ALOHA, and FW-ALOHA. Since the optimal behavior of the AWARE node in coexistence with ALOHA, TDMA, CSMA, EB-ALOHA, and FW-ALOHA nodes has not been derived, we directly compare the performance of LLMA nodes against DLMA nodes by assessing the sum of α-fairness throughput. The α-fairness throughput of each node and the sum of α-fairness throughput are presented in Fig. 7. To calculate α-fairness throughput, we multiply the throughput by 100 to avoid negative values when applying the logarithm. Given our goal to maximize the sum of α-fairness throughput, these figures demonstrate that the LLMA node outperforms the DLMA node by approximately 17.7%, 29.1%, 14.1%, and 19.1%, respectively. These results indicate that the LLMA protocol, designed through the CP-AgentNet, can effectively coexist in more complex heterogeneous environments.

4) ***Dynamic scenario:*** We evaluate the adaptability of the LLMA node in a dynamic setting to assess its responsiveness to environmental changes. This scenario starts with two slotted ALOHA nodes and one heteronode (2A+1H). At frame 2500, one ALOHA node exits the network (1A+1H), followed by the entry of two new ALOHA nodes at frame 5000 (3A+1H). The



(a) AWARE+1T+3A (b) DLMA+1T+3A (c) LLMA+1T+3A

Fig. 5: Heteronode with 3 ALOHA and 1 TDMA nodes



(a) AWARE node (b) DLMA node (c) LLMA node

Fig. 6: Heteronode in a dynamic environment

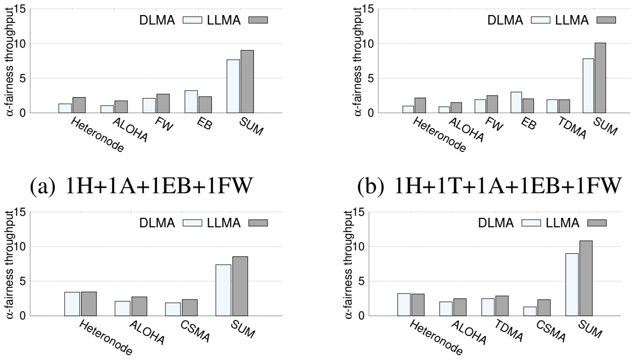
TABLE I: (N)RMSE of all scenarios. T, A, and H represent TDMA, slotted ALOHA, and heteronode, respectively.

	Combination	RMSE		NRMSE	
		DLMA	LLMA	DLMA	LLMA
Single-heteronode	1T+1H	0.1409	0.1908	0.1761	0.2385
	1A+1H	0.1175	0.0768	0.3422	0.2028
	2A+1H	0.1192	0.0491	0.5573	0.2298
	3A+1H	0.0931	0.0468	0.7323	0.368
	4A+1H	0.1044	0.0433	0.6884	0.2647
	1T+1A+1H	0.0909	0.0681	0.2822	0.2117
	1T+3A+1H	0.0544	0.0403	0.5363	0.3977
Multi-heteronode	2A+2H	0.0537	0.0272	0.4517	0.2287
	1T+2A+3H	0.019	0.0177	0.2925	0.2728
Massive	20A+1H	0.0522	0.044	0.6809	0.5873
Dynamic	Dynamic	0.227	0.0476	1.0706	0.2203

scenario progresses with the addition of one TDMA node at frame 7500 (1T+3A+1H), concluding at frame 10000. In this dynamic environment, as depicted in Fig. 6, the operation of the LLMA node closely mirrors the ideal behavior. The RMSE value for the LLMA node is 0.044, which is approximately 5 times lower than 0.227 with the DLMA node, demonstrating that the LLMA, designed using the CP-AgentNet framework, effectively adapts to environmental changes. This significant difference arises because LLMA does not depend on the model architecture for different numbers of nodes. Unlike traditional DRL methods, the LLMA does not rely on a specific model architecture, thereby enhancing its scalability.

5) *Ablation Study:*

Impact of CP-Agent. Although we have demonstrated the superior performance of the LLMA, it could be argued that this success stems from the pretrained knowledge of LLM. To demonstrate that the exceptional performance is specifically attributable to our CP-AgentNet, we conduct targeted assessments. Initially, we simulated the system using only the LLM under the same conditions as those used in CP-AgentNet. Subsequently, we evaluated the impact of each agent within the framework. Given the central role of the node agent in decision-making, we could not omit it from our simulations; therefore, we first ran simulations with the node agent paired with the strategy agent, and then with the observer agent, to isolate and assess their individual contributions. As shown in Table II, in static tests, the RMSE values for both the LLMA node and the ALOHA node are similar between “1” and “3” configurations, as well as between “2” and “4”. This suggests that the strategy agent significantly influences the performance of both the LLMA and ALOHA. Conversely, the RMSE value for the TDMA node is notably affected by the observer agent, because the node agent avoids the time slots used by the TDMA node with the assistance of the observer agent. In dynamic scenarios, the observer agent impacts not only the behavior of the TDMA node but also that of the LLMA and ALOHA nodes, as it detects environmental changes and assists the node agent in quickly adapting to new conditions. These experiments have demonstrated the impact of each agent, with the superior performance originating from the CP-AgentNet.



(a) 1H+1A+1EB+1FW (b) 1H+1T+1A+1EB+1FW

(c) 1H+1A+1C (d) 1H+1T+1A+1C

Fig. 7: α-fairness throughput in a complex heterogeneous environment. C, EB, and FW represent CSMA node, EB-ALOHA node, and FW-ALOHA node, respectively.

Effectiveness of LLM ranker. We evaluate the effectiveness of the LLM ranker. Specifically, during the offline stage, the strategy agent utilizes the LLM ranker to formulate transmission strategies. We conduct five simulations for each strategy, both with and without the LLM ranker, comparing the resultant RMSE values and their variances to assess which approach more closely approximates ideal behavior. In the online stage, we implement a transmission strategy that was formulated using the LLM ranker. Additionally, we explore scenarios where the node agent either uses or does not use the ranker to determine transmission probabilities. As shown in Table III, both the average RMSE and standard deviation are lower when the strategy agent employs the LLM ranker, indicating enhanced stability and consistency. However, the impact of the LLM ranker is less critical for the node agent during the online stage. This reduced impact is because the node agent's decisions largely depend on the predefined strategy, which is relatively simpler than strategy formulation.

6) *Explainability:*

There is no established consensus on the definition of explainability in machine learning, nor on how it should be measured [16]. Explainable AI (XAI) focuses on demystifying the decision-making processes of ML models—understanding how and why decisions are made [17]. One of the most straightforward approaches to achieving explainability is to employ decision trees [16]. In our framework,

TABLE II: The impact of each agent. 1: No Agent, 2: Strategy + Node, 3: Observer + Node, 4: Strategy + Observer + Node

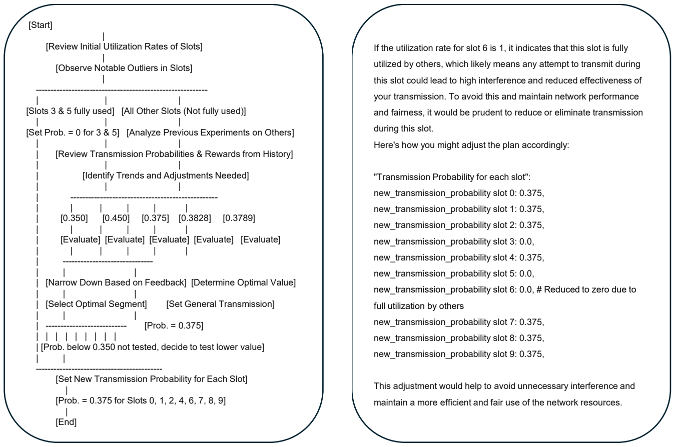
	Static			Dynamic		
	LLMA	ALOHA	TDMA	LLMA	ALOHA	TDMA
1	0.138	0.036	0.058	0.154	0.04	0.064
2	0.04	0.015	0.031	0.089	0.022	0.033
3	0.155	0.039	0.011	0.153	0.041	0.015
4	0.04	0.015	0.013	0.048	0.016	0.01

TABLE III: The impact of using an LLM ranker.

Phase	Without Ranker		With Ranker	
	RMSE	STD	RMSE	STD
Offline learning	0.0636	0.036	0.0404	0.008
Online execution	0.0362	0.007	0.0346	0.007

multi-agent interaction facilitates the generation of a decision tree, rendering the decision-making process transparent and interpretable. Fig. 8a is generated by the LLM to illustrate how the action is devised. The observer agent monitors the environment and informs the node agent that slot 3 and 5 are fully utilized. Consequently, the node agent avoids these slots and determines the action for each slot. This process is clearly depicted in the decision tree, enabling users to easily understand how the decisions are made. Moreover, the LLM facilitates the *ability to question*, enhancing comprehension of the decisions for laypeople and enabling them to explore 'what-if' scenarios— a capability not available with traditional DRL methods. In our scenario, the LLM can respond to queries like 'What if the utilization rate of slot 6 is 1?' without the need for additional training. The response is illustrated in Fig. 8b, demonstrating the capability to facilitate relevant questioning. In addition, this can help assess whether the LLM's explanations are consistent with its actual decision-making. Observing responses to controlled hypothetical inputs allows users to gauge how well the model's reasoning aligns with expected behavior.

*7) **Computational Overhead**:* We evaluate the computational overhead of LLMA nodes and find that LLMA requires 19.1 seconds to process decisions for 1,000 frames (10 seconds of simulation time). While not yet ready for real-time decision-making, this result demonstrates strong potential for near-future deployment, particularly with the continued advancement of lightweight personal LLMs. To illustrate this potential, we compare the computation time of smaller models such as GPT-4o-mini and GPT-4.1-nano. Note that while the offline phase is conducted using GPT-4o, the online phase can be replaced with smaller models. As shown in Table IV, GPT-4o-mini and GPT-4.1-nano require 11.3 and 10.1 seconds, respectively, while achieving comparable performance. Although CP-AgentNet relies on high-end LLMs during the offline phase for strategy generation, these results suggest that the online execution phase can operate efficiently with smaller models. This separation between strategy generation and execution significantly reduces the computational burden on individual agents, enabling efficient operation even in resource-constrained environments.


[Start]
[Review Initial Utilization Rates of Slots]
[Observe Notable Outliers in Slots]
[Slots 3 & 5 fully used]
[All Other Slots (Not fully used)]
[Set Prob. = 0 for 3 & 5]
[Analyze Previous Experiments on Others]
[Review Transmission Probabilities & Rewards from History]
[Identify Trends and Adjustments Needed]
[0.350]
[0.450]
[0.375]
[0.3828]
[0.3789]
[Evaluate]
[Evaluate]
[Evaluate]
[Evaluate]
[Evaluate]
[Narrow Down Based on Feedback]
[Determine Optimal Value]
[Select Optimal Segment]
[Set General Transmission]
[Prob. = 0.375]
[Prob. below 0.350 not tested, decide to test lower value]
[Set New Transmission Probability for Each Slot]
[Prob. = 0.375 for Slots 0, 1, 2, 4, 6, 7, 8, 9]
[End]

(a) Decision tree on CP-Agent

If the utilization rate for slot 6 is 1, it indicates that this slot is fully utilized by others, which likely means any attempt to transmit during this slot could lead to high interference and reduced effectiveness of your transmission. To avoid this and maintain network performance and fairness, it would be prudent to reduce or eliminate transmission during this slot.
Here's how you might adjust the plan accordingly:

```
"Transmission Probability for each slot":
new_transmission_probability slot 0: 0.375,
new_transmission_probability slot 1: 0.375,
new_transmission_probability slot 2: 0.375,
new_transmission_probability slot 3: 0.0,
new_transmission_probability slot 4: 0.375,
new_transmission_probability slot 5: 0.0,
new_transmission_probability slot 6: 0.0, # Reduced to zero due to
full utilization by others
new_transmission_probability slot 7: 0.375,
new_transmission_probability slot 8: 0.375,
new_transmission_probability slot 9: 0.375,
```

This adjustment would help to avoid unnecessary interference and maintain a more efficient and fair use of the network resources.

(b) Ability to answer questions

Fig. 8: Explainable CP-AgentNet

TABLE IV: Comparison of computation time and RMSE performance across smaller LLM variants

	Combination	LLMA (4o)	LLMA (4o-mini)	LLMA (4.1-nano)
RMSE	1T+1A+1H	0.0681	0.0592	0.0720
	1T+3A+1H	0.0403	0.0508	0.0521
	Dynamic	0.0476	0.0492	0.0517
Runtime (s)	—	19.1	11.3	10.1

C. CPTCP results

*1) **Homogeneous scenario**:* We evaluate multiple CPTCP flows in configurations of 2, 3, and 4 in a homogeneous environment, comparing the total throughput results with TCP CUBIC [14] and TCP BBR [15]. As shown in Table V, although CPTCP does not overwhelmingly outperform CUBIC and BBR, these results demonstrate that comparable TCP performance can be achieved through CP-AgentNet. Importantly, when the LLM is used without CP-Agent, performance consistently ranks as the lowest, regardless of the number of flows. These results indicate that the performance of the CPTCP does not stem solely from the pretrained LLM.

We further evaluate CPTCP against BBR, a congestion control algorithm that estimates bottleneck bandwidth and minimum RTT to maximize throughput while minimizing latency, to evaluate performance under varying RTTs as shown in Table VI. While BBR achieves slightly higher throughput, CPTCP demonstrates superior fairness, particularly in mixed RTT environments. This improvement in fairness stems from the design of the reward function, which explicitly incorporates both RTT and packet loss during training to guide the decision-making process.

*2) **Heterogeneous scenario**:* To demonstrate that CPTCP can coexist with different types of TCP algorithms, we assess its performance in a heterogeneous TCP environment. As shown in Table VII, when CUBIC and BBR coexist, a fairness issue arises, resulting in a low Jain's fairness index of 0.823.

TABLE V: Total throughput (Kbps) with homogeneous TCP

number of flows	CUBIC	BBR	CPTCP	Pure LLM
2	758.4	793.6	813.6	543.2
3	858.3	942.4	868.4	679.1
4	883.0	932.7	890.1	584.0

TABLE VI: Homogeneous TCP under different RTTs.

RTT (ms)	Algorithm	Throughput (Kbps)	Jain's fairness Index
100	CPTCP	890.1	0.999
	BBR	932.7	0.998
200	CPTCP	652.4	0.996
	BBR	724.4	0.997
Mixed	CPTCP	732.8	**0.970**
	BBR	746.0	0.884

TABLE VII: Throughput and fairness with heterogeneous TCP

TCP algorithm		Throughput (Kbps)		Jain's fairness index
flow1	flow2	flow1	flow2	
CUBIC	CUBIC	391.2	367.2	0.999
BBR	BBR	398.8	394.8	1.000
CPTCP	CPTCP	398.6	415.0	1.000
CUBIC	BBR	575.2	210.5	**0.823**
CUBIC	Cx-TCP	441.2	338.8	**0.976**
BBR	Cx-TCP	362.2	387.7	**0.999**
CUBIC	CPTCP	372.1	417.7	**0.997**
BBR	CPTCP	438.7	341.3	**0.984**

However, when CPTCP coexists with either CUBIC or BBR, it does not interfere with the other TCP algorithm, achieving Jain's fairness indices of 0.997 and 0.984. These results are comparable to those achieved by Cx-TCP [13], indicating that CPTCP can effectively coexist with both types of TCP algorithms. Furthermore, its applicability extends beyond TCP coexistence scenarios.

V. Related Works

A. MAC protocols through RL

A MAC protocol utilizing QMIX [18] has been developed to coexist with legacy systems like CSMA/CA [2]. This protocol exhibits adaptability in dynamic settings, but requires reinitialization with different weight parameters as node numbers change. In contrast, the DRL-based MAC protocols [1], [19] are designed to operate in heterogeneous networks without requiring explicit knowledge of the number of nodes or their parameters. However, their performance remains highly dependent on the underlying neural network architecture, which is often optimized for a fixed node count—limiting their robustness and adaptability in dynamic environments.

B. LLM-empowered Agents

Generative Agents [20] elicit emerging behaviors and collaboration within assemblies of agents as societal groups. Autonomous software development has improved through the collaborative use of multiple LLM agents [6], [21]–[26]. These systems utilize various LLM agents, assigning specific tasks to each to reduce complexity and facilitate collaboration in accomplishing these tasks. Notably, MetaGPT [22] and AgentVerse [21] simulate a human-like software development process, enabling efficient role-playing for each agent. Additionally, improvements in mathematical accuracy using LLM agents have been explored [27], [28]. While these LLM-agents are typically built on top of LLMs accessed via the internet, personal LLM-agents [29] can also be deployed on-device using LLM [30]. CP-AgentNet operates under the assumption that personal LLM-agents can be utilized, although in practice, we actually employ GPT-4o via the internet.

C. LLMs with Reinforcement Learning

Recent research has explored various ways of integrating reinforcement learning with LLMs [31]–[38]. When2Ask [31] method involves using RL to control the frequency of queries to the LLM, optimizing for cost-effective interaction between the agent and the LLM. LLARP [32] employs a pre-trained, frozen LLM as the core neural network within a DRL framework. This setup enhances the agent's generalization capabilities for embodied tasks through the use of LLM. ICPI [33] iteratively updates the contents of the prompt, which serves as the foundation for its policy, through trial-and-error interactions within an RL environment. This approach does not employ gradient-based methods; instead, the focal point of learning is the prompt content itself.

D. LLMs for communication network

Following the trend of leveraging LLMs for specialized disciplines, there is a growing movement to adopt LLMs in the field of communication networks [39]–[46]. The work in [41] fine-tunes LLMs with telecommunication-specific language to identify working groups within 3GPP. [39], [40], [42] utilize LLMs as QA assistants to interpret 3GPP standards. Collectively, these approaches primarily focus on classification of working groups or QA tasks, indicating that the use cases of LLMs in the communication domain remain quite limited.

VI. Conclusion

We introduced CP-AgentNet, a framework using generative agents to autonomously design communication network protocols, enhancing explainability. Using CP-AgentNet, we designed the LLMA and CPTCP for heterogeneous environments. Our simulations show that nodes using LLMA or CPTCP coexist efficiently with nodes using different protocols. This work sets the stage for broader applications of generative agents in protocol design and solving complex network problems.

Acknowledgement

The work reported in this paper is supported in part by a Samsung Fellowship, by US Army Research Office award W911NF2020267, NSF award CNS-2128588, and the UCSD Center for Wireless Communications.

References

[1] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE journal on selected areas in communications*, vol. 37, no. 6, pp. 1277–1290, 2019.

[2] Z. Guo, Z. Chen, P. Liu, J. Luo, X. Yang, and X. Sun, "Multi-agent reinforcement learning-based distributed channel access for next generation wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, pp. 1587–1599, 2022.

[3] A. Heuillet, F. Couthouis, and Natalia, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, p. 106685, 2021.

[4] Z. He, A. Gottipati, L. Qiu, X. Luo, K. Xu, Y. Yang, and F. Y. Yan, "Designing network algorithms via large language models," in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pp. 205–212, 2024.

[5] Y. Ishibashi and Y. Nishimura, "Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization," *arXiv preprint arXiv:2404.02183*, 2024.

[6] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via chatgpt," *arXiv preprint arXiv:2304.07590*, 2023.

[7] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "Autogen: Enabling next-gen llm applications via multi-agent conversation framework," *arXiv preprint arXiv:2308.08155*, 2023.

[8] R. Nakano, W. Hilton, *et al.*, "Webgpt: Browser-assisted question-answering with human feedback," *arXiv preprint arXiv:2112.09332*, 2021.

[9] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[10] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, *et al.*, "Large language models are effective text rankers with pairwise ranking prompting," *arXiv preprint arXiv:2306.17563*, 2023.

[11] J. Surowiecki, *The wisdom of crowds*. Anchor, 2005.

[12] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," Tech. Rep. Tech. Rep. 1, Chinese University of Hong Kong, Hong Kong, 2020. Available online: https://github.com/YidingYu/DLMA/blob/master/DLMA-benchmark.pdf.

[13] Ł. Budzisz, R. Stanojevic, A. Schlote, F. Baker, and R. Shorten, "On the fair coexistence of loss-and delay-based tcp," *IEEE/ACM transactions on networking*, vol. 19, no. 6, pp. 1811–1824, 2011.

[14] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[15] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[16] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[17] "Explainability and auditability in ml: Definitions, techniques, and tools." https://neptune.ai/blog/explainability-auditability-ml-definitions-techniques-tools, 2023.

[18] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[19] Y. Yu, S. C. Liew, and T. Wang, "Multi-agent deep reinforcement learning multiple access for heterogeneous wireless networks with imperfect channels," *IEEE Transactions on Mobile Computing*, vol. 21, no. 10, pp. 3718–3730, 2021.

[20] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.

[21] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C. Qian, C.-M. Chan, Y. Qin, Y. Lu, R. Xie, *et al.*, "Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents," *arXiv preprint arXiv:2308.10848*, 2023.

[22] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, *et al.*, "Metagpt: Meta programming for multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, 2023.

[23] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, and H. Ji, "Unleashing cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration," *arXiv preprint arXiv:2307.05300*, vol. 1, no. 2, p. 3, 2023.

[24] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, "Communicative agents for software development," *arXiv preprint arXiv:2307.07924*, 2023.

[25] D. Huang, Q. Bu, J. M. Zhang, M. Luck, and H. Cui, "Agentcoder: Multi-agent-based code generation with iterative testing and optimisation," *arXiv preprint arXiv:2312.13010*, 2023.

[26] G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for" mind" exploration of large language model society," *Advances in Neural Information Processing Systems*, vol. 36, pp. 51991–52008, 2023.

[27] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving factuality and reasoning in language models through multiagent debate," *arXiv preprint arXiv:2305.14325*, 2023.

[28] Z. Liu, Y. Zhang, P. Li, Y. Liu, and D. Yang, "Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization," *arXiv preprint arXiv:2310.02170*, 2023.

[29] Y. Li, H. Wen, W. Wang, X. Li, Y. Yuan, G. Liu, J. Liu, W. Xu, X. Wang, Y. Sun, *et al.*, "Personal llm agents: Insights and survey about the capability, efficiency and security," *arXiv preprint arXiv:2401.05459*, 2024.

[30] D. Xu, W. Yin, X. Jin, Y. Zhang, S. Wei, M. Xu, and X. Liu, "Llmcad: Fast and scalable on-device large language model inference," *arXiv preprint arXiv:2309.04255*, 2023.

[31] B. Hu, C. Zhao, P. Zhang, Z. Zhou, Y. Yang, Z. Xu, and B. Liu, "Enabling intelligent interactions between an agent and an llm: A reinforcement learning approach," *arXiv preprint arXiv:2306.03604*, 2023.

[32] A. Szot, M. Schwarzer, H. Agrawal, B. Mazoure, R. Metcalf, W. Talbott, N. Mackraz, R. D. Hjelm, and A. T. Toshev, "Large language models as generalizable policies for embodied tasks," in *The Twelfth International Conference on Learning Representations*, 2023.

[33] E. Brooks, L. A. Walls, R. Lewis, and S. Singh, "In-context policy iteration," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[34] H. Sun, "Offline prompt evaluation and optimization with inverse reinforcement learning," *arXiv preprint arXiv:2309.06553*, 2023.

[35] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

[36] J. Hu, L. Tao, J. Yang, and C. Zhou, "Aligning language models with offline reinforcement learning from human feedback," *arXiv preprint arXiv:2308.12050*, 2023.

[37] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma, "Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics," *arXiv preprint arXiv:2309.06687*, 2023.

[38] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," *arXiv preprint arXiv:2303.00001*, 2023.

[39] H. Holm, "Bidirectional encoder representations from transformers (bert) for question answering in the telecom domain.: Adapting a bert-like language model to the telecom domain using the electra pre-training approach," 2021.

[40] A. Karapantelakis, M. Shakur, A. Nikou, F. Moradi, C. Orlog, F. Gaim, H. Holm, D. D. Nimara, and V. Huang, "Using large language models to understand telecom standards," *arXiv preprint arXiv:2404.02929*, 2024.

[41] L. Bariah, H. Zou, Q. Zhao, B. Mouhouche, F. Bader, and M. Debbah, "Understanding telecom language through large language models," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*, pp. 6542–6547, IEEE, 2023.

[42] N. Piovesan, A. De Domenico, and F. Ayed, "Telecom language models: Must they be large?," *arXiv preprint arXiv:2403.04666*, 2024.

[43] A. Maatouk, F. Ayed, N. Piovesan, A. De Domenico, M. Debbah, and Z.-Q. Luo, "Teleqna: A benchmark dataset to assess large language models telecommunications knowledge," *arXiv preprint arXiv:2310.15051*, 2023.

[44] H. Cai and S. Wu, "Tkg: Telecom knowledge governance framework for llm application," 2023.

[45] G. Charan, M. Alrabeiah, and A. Alkhateeb, "Vision-aided 6g wireless communications: Blockage prediction and proactive handoff," *IEEE*

Transactions on Vehicular Technology, vol. 70, no. 10, pp. 10193–10208, 2021.

[46] M. Matsuura, Y. K. Jung, and S. N. Lim, "Visual-llm zero-shot classification," 2023.