SANNet: A Semantic-Aware Agentic AI Networking Framework for Multi-Agent Cross-Layer Coordination

Yong Xiao*†‡, Haoran Zhou*, Xubo Li*, Yayu Gao*, Guangming Shi†§‡, Ping Zhang¶
* School of Elect. Inform. & Commun., Huazhong Univ. of Science & Technology, China
† Peng Cheng Laboratory, Shenzhen, China
‡ Pazhou Laboratory (Huangpu), Guangzhou, China
§ School of Artificial Intelligence, Xidian University, Xi'an, China
¶ State Key Lab. of Networking & Switching Tech., Beijing Univ. of Posts & Telecom., Beijing, China

***Abstract*—Agentic AI networking (AgentNet) is a novel AI-native networking paradigm that relies on a large number of specialized AI agents to collaborate and coordinate for autonomous decision-making, dynamic environmental adaptation, and complex goal achievement. It has the potential to facilitate real-time network management alongside capabilities for self-configuration, self-optimization, and self-adaptation across diverse and complex networking environments, laying the foundation for fully autonomous networking systems in the future. Despite its promise, AgentNet is still in the early stage of development, and there still lacks an effective networking framework to support automatic goal discovery and multi-agent self-orchestration and task assignment. This paper proposes SANNet, a novel semantic-aware agentic AI networking architecture that can infer the semantic goal of the user and automatically assign agents associated with different layers of a mobile system to fulfill the inferred goal. Motivated by the fact that one of the major challenges in AgentNet is that different agents may have different and even conflicting objectives when collaborating for certain goals, we introduce a dynamic weighting-based conflict-resolving mechanism to address this issue. We prove that SANNet can provide theoretical guarantee in both conflict-resolving and model generalization performance for multi-agent collaboration in dynamic environment. We develop a hardware prototype of SANNet based on the open RAN and 5GS core platform. Our experimental results show that SANNet can significantly improve the performance of multi-agent networking systems, even when agents with conflicting objectives are selected to collaborate for the same goal.**

***Index Terms*—Agent AI network, semantic-aware.**

I. Introduction

With the fast proliferation of AI services and applications, next-generation communication networks will be dominated by a large number of diverse AI models and subsystems, coexisting, interacting, and communicating with one another [1], [2]. The existing data-transportation-focused networking architecture has faced unprecedented challenges to meet the fast-growing demand on communication and interaction needs of the future networks of AI agents. With the fast proliferation of AI services and applications, next-generation communication networks are expected to be characterized by a vast array of diverse AI models and subsystems that interact and communicate with one another. The existing data-transportation-centric networking architecture is increasingly confronted with novel challenges in accommodating the rapidly growing demands of real-time knowledge sharing, model coordination, collaborative decision-making, and adaptive resource orchestration, required by communication and networking of AI models and subsystems.

Agentic AI networking (AgentNet) has attracted significant interest recently due to its potential to fundamentally address the limitations of the existing networking architecture by embracing an interactive learning and networking paradigm involving a large number of diverse AI agents with high-level autonomous, goal-driven, and adaptable decision-making capability [3]. More specifically, instead of constructing a specific model for each individual task, agentic AI focuses on establishing and maintaining an interactive networking ecosystem where a diverse set of AI agents, each with unique knowledge, skillset and capabilities, and can autonomously communicate, interact and/or collaborate in archiving various complex goals across different environments [4].

Despite its promises, AgentNet is still in the early stage of development [5]. Most of the existing research focuses on multi-agent task planning and adaptation, ignoring the fact that the communication networking architecture plays a fundamental role in high-performance AgentNet systems. In particular, a recent study has already suggested that AgentNet is in essence a novel communication networking paradigm that supports effective communication and seamless interaction between human users and agents, as well as among multiple diverse agents within a system [6]. There is still a lack of a comprehensive framework that can automatically detect a user's semantic goal and self-orchestrate and coordinate different agents with different skillsets for fulfilling the detected goal [7].

In this paper, we investigate the agentic AI system from the communication networking perspective. More specifically, we introduce a general multi-agent cross-layer coordination framework that supports the seamless interaction and collaboration of diverse agents associated with different layers of mobile networking systems, including the application-layer, network-layer, and physical-layer. We formulate the optimization problem for goal-oriented collaboration among agents associated with different layers and identify two major challenges faced by

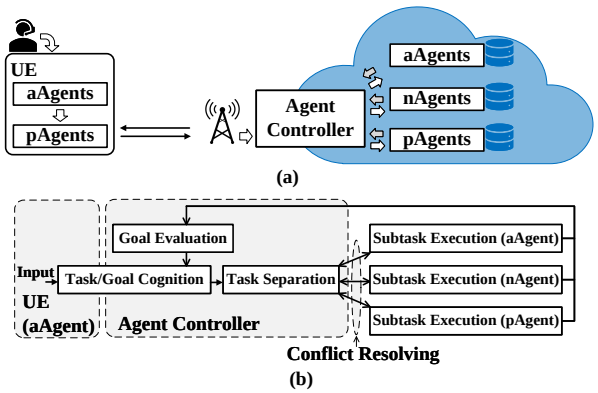


Fig. 1: (a) System model and (b) coordination procedures for a general multi-agent cross-layer mobile networking system.

the multi-agent autonomous networking systems: (multi-agent) objective conflict and model generalization. The first one is that different agents, especially agents in layers, may have conflicting objectives when collaborating for a specific task, resulting in significant performance degradation and even divergence in goal achievement. The second challenge is that each agent can only observe a limited set of local data that cannot fully capture the complex real-world scenarios, resulting in a discrepancy between the predicted agent's performance and the final outcomes over the real-world data distributions. To solve these challenges, we first introduce two metrics, referred to as the C-error and G-error, to quantify the performance degradation caused by objective conflict and model generalization, respectively, and then propose a dynamic-weighting-based conflict-resolving mechanism that can provide a theoretical performance guarantee for both errors. We propose a novel semantic-aware agentic AI networking architecture, called SANNet, to allow the user's semantic goal to be learned, autonomously recognized, and separated into different subtasks, each of which can be fulfilled by an agent with the required skillset. We include our proposed conflict-resolving mechanism into the agent controller, so agents in different layers can collaborate in achieving the user's various semantic goals with guaranteed performance. Finally, we develop a hardware prototype based on the open RAN and 5GS core platform to evaluate the performance of SANNet in real-world scenarios. Experimental results show that SANNet can significantly reduce the C-error for multi-agent networks, compared to the existing state-of-the-art solutions.

II. System Model and Problem Formulation

A. System Model

We consider a general multi-task multi-agent cross-layer optimization problem for a mobile networking system, as illustrated in Fig. 1 (a). An agent is a physical or logical entity that can perceive the relevant users' semantic goals in specific environments and can learn and act autonomously to achieve the perceived goals. Each agent is composed of a single or a set of collaborative models designed with certain objectives and/or to meet a specific task demand. We consider a finite task space and let $\mathcal{M}$ be the set of tasks that can be performed by a single agent or a set of agents. Agents deployed at different layers of the mobile system are generally designed to interact with different environments. In this paper, we focus on the cross-layer coordination and collaborative optimization involving multiple agents, and to simplify our description, we mainly focus on the following three types of agents. Our proposed solution, however, can be directly extended to more complex systems with more layers, as will be illustrated later.

(1) Application-layer agent (aAgent): This corresponds to the agent that interacts with human users in various physical or virtual environments through some application interfaces. For example, virtual assistant applications installed on users' smart devices can interact with human users and infer the semantic goals of the users from the input of the UE. More formally, we define an aAgent by a tuple $\mathcal{G}^a = \langle \mathcal{A}^a, \mathcal{S}^a, \mathcal{L}^a, \mathcal{D}^a \rangle$ where $\mathcal{A}^a$ is the set of acts that can be taken by aAgent, $\mathcal{S}^a$ is the set of environmental state for an aAgent to sense and decide its act, $\mathcal{L}^a$ is the set of task-related loss functions for aAgent to optimize, and $\mathcal{D}^a$ is the set of data samples for training the aAgent's model. In this paper, we consider the AI model-based agentic AI networking system in which each agent focuses on training an AI model that can output a specific act based on the sensed state. Let ω^a be the model parameters associated with the aAgent. Generally speaking, aAgent cannot control the decision-making processes of other agents. Its performance, however, can be influenced by other agents' actions, especially those in the physical layer and network layer. We can write the loss function associated with task m of aAgent as $l_m^a(\mathbf{\Omega}_m, \mathcal{D}^a)$ where $\mathbf{\Omega}_m = \langle \boldsymbol{\omega}_m^a, \boldsymbol{\omega}_m^p, \boldsymbol{\omega}_m^n \rangle$ for ω^p and ω^n are model parameters associated with the agents in physical-layer and network-layer, respectively, as will be illustrated later. $\alpha_m^a \in \mathcal{A}^a$, $s_m^a \in \mathcal{S}^a$, and $l_m^a \in \mathcal{L}^a$.

(2) Physical-layer agent (pAgent): This corresponds to the agent that can interact and adapt to various physical-layer environments. For example, a pAgent deployed at the gNB or UE in a mobile network can estimate the spectrum availability and, in some cases, keep track of the channel state information (CSIs) connecting UE and gNB and make recommendations in channel selections or transmitter and receiver parameter choices. Similarly, we can also define a pAgent by a tuple $\mathcal{G}^p = \langle \mathcal{A}^p, \mathcal{S}^p, \mathcal{L}^p, \mathcal{D}^p \rangle$ where $\mathcal{A}^p$ is the act space, $\mathcal{S}^a$ is the environmental state space for the pAgent to estimate and decide its act, and $\mathcal{L}^p$ is the set of task-related loss functions, and $\mathcal{D}^p$ is the set of data samples for training the relevant models. Similar to the aAgent, the decision-making process of the pAgent can also be performed by a model parameterized by ω_m^p that can output act decision based on the sensed physical-layer state information. We can write the loss function associated with task m of pAgent as $l_m^p(\mathbf{\Omega}_m, \mathcal{D}^p)$ for $\alpha_m^p \in \mathcal{A}^p$, $s_m^p \in \mathcal{S}^p$, and $l_m^p \in \mathcal{L}^p$.

(3) Network-layer agent (nAgent): This corresponds to the agent that interacts with the network-layer environments. For example,

an nAgent deployed in the transportation layer or the core network of a mobile networking system can keep track of the routing and bandwidth resources between any network entities connecting the UE and its intended destination and adjust the routing and the amount of bandwidth resource accordingly. We define a nAgent by a tuple $\mathcal{G}^n = \langle \mathcal{A}^n, \mathcal{S}^n, \mathcal{L}^n, \mathcal{D}^n \rangle$ where $\mathcal{A}^n$ is the set of acts that can be taken by nAgent, $\mathcal{S}^n$ is the set of network-layer environmental states for an nAgent to estimate and decide its act, and $\mathcal{L}^n$ is the set of task-related loss functions, and $\mathcal{D}^n$ is the set of data samples for constructing the nAgent's model. Let ω_m^n be the model parameters associated with the nAgent. We can also write the loss function of pAgent as $l_m^n(\mathbf{\Omega}_m, \mathcal{D}^n)$ for $\alpha_m^n \in \mathcal{A}^n$, $s_m^n \in \mathcal{S}^n$, and $l_m^n \in \mathcal{L}^n$.

Each agent will not expose its local state or action to others. It can however report its capability via an agent-specific information tag, e.g., an agent card, to an agent controller and will only be activated when it has been called for a certain task.

B. Problem Formulation

Let us now describe the multi-agent cross-layer coordination procedure when a task request has been detected from the user's input as illustrated in Fig. 1 (b): (1) *(Semantic) goal detection:* A special aAgent can be deployed at the UE to keep track of the users' demand and recognize the user's semantics, e.g., a virtual assistant app can understand the user's semantics based on some key prompts using an LLM-based model; (2) *task separation:* The detected semantics will then be reported to an agent controller to decide the associated task as well as the corresponding subtasks that need to be performed in order to fulfill the user's semantic goal. Each subtask is assigned to a specific agent that has been previously exposed to its capability to the agent controller, e.g., if the user's demand involves initiating some data-heavy and high-bandwidth-demanding applications, pAgent and nAgent will be called to scan physical-layer resources such as spectrum availability and channel conditions, as well as network-layer routing and bandwidth resources; (3) *subtask execution:* Each agent, once received the assigned subtask, will make independent decisions, act accordingly, and report the result to the agent controller. (4) *goal (fulfillment) evaluation:* The user's goal will be fulfilled once all the agents assigned by the agent controller finish their duties. The aAgent on the UE side will continue to keep track of the user's input and repeat the above procedures when new demands are detected.

Since each agent is an independent decision maker, the agent controller cannot directly control its decision-making process. It can choose a specific set of agents when a certain task demand is detected. In this case, the local decision-making processes of the selected agents need to be carefully coordinated to make sure all the selected agents can achieve their objectives. We can define the multi-agent cross-layer optimization problem as follows:

$$\textbf{P1:} \min_{\mathbf{\Omega}_m} L_m(\mathbf{\Omega}_m) = \langle l_m^a(\mathbf{\Omega}_m, \mathcal{D}^a), l_m^p(\mathbf{\Omega}_m, \mathcal{D}^p), l_m^n(\mathbf{\Omega}_m, \mathcal{D}^n) \rangle.$$

Note that the optimization objective of problem **P1** involves a vector of objectives from different agents associated with different layers of the mobile system. These agents often have different or even conflicting objectives, and therefore it is generally impossible to find a single global optimal solution that can minimize the loss functions of all the agents. Another major challenge faced by the multi-task multi-agent mobile systems is that the models developed by each agent need to be pre-trained to ensure responsive decision making, resulting in inaccurate and biased results when being called to act in new, unseen environment.

III. SANNet Architecture

We propose SANNet, a semantic-aware agentic AI networking framework that supports multi-agent cross-layer optimization on a new functional entity, the *agent controller*. The agent controller keeps track of the semantics of the user based on the input of the UE, e.g., language or visual expression of the user recorded by a UE aAgent. Each detected semantic demand can be associated with a set of different requirements associated with different layers of the system. A set of agents will then be selected to address various requirements raised from the users' demand. The agent controller will also evaluate the progress of task execution of various agents and mediate conflicts arising from divergent or conflicting objectives among them.

Let us first discuss the operational details of the agent controller as follows:

(1) Semantic cognition: The semantics as well as the task/goal of the users can be recognized by the agent controller and linked to a specific task m for $m \in \mathcal{M}$.

(2) Task separation and semantic translation: The requirements associated with each task m recognized by the agent controller will be translated into a set of subtasks, each is associated with an individual requirement raised by the users' demand. In this paper, we consider the cross-layer task separation, as illustrated in Fig. 1. In this case, the requirements of each task m recognized from the user's semantics will be translated into requirements associated with different layers, including application-layer, physical-layer, and network-layer.

(3) Agent selection: Each agent will maintain a *agent card*, a meta-file consisting of the description of the agent's function, implementable environmental state, action space, associated layer, functional objective, and loss functions. Each agent will submit its agent card to the agent controller before it can be called. The agent controller will then map the requirements of each task m in the application-layer, physical-layer, and network-layer to a set of agents associated with different layers. Each agent, once called, will sense its local environment and output the corresponding act under the sensed environment state.

(4) Task evaluation and adjustment: The agent controller will keep track of the subtask executions of different agents. Since different agents at different layers generally have different learning and optimization objectives, e.g., loss functions, when

divergent objectives among some agents are detected, the agent controller will need to mediate and resolve the conflicts. In this paper, we propose a dynamic weighting-based conflict resolution solution that allows different agents to resolve conflicts and obtain a direction that optimizes all agents' objective functions jointly.

The above operations will be repeated as long as new semantic demands of the users are detected. We assume that these operations cannot be interrupted once initiated, even if new demands are identified during their execution.

Let us now describe how the agent controller can resolve conflicting objectives when agents associated with different layers have different goals, represented by different loss functions. In the rest of this paper, we focus on the cross-layer collaboration among agents of different layers with conflicting objectives, i.e., the loss functions of different agents selected for performing specific tasks are different from each other. It is known that in this case, there does not generally exist a single global optimal solution that minimizes all the agents' loss functions. Therefore, in this paper, we focus on the Pareto optimal solution where no agents can further improve their performance without making other agents worse off. More formally, we define the Pareto optimal solution for the multi-agent networking system as follows:

Definition 1: A solution profile $\mathbf{\Omega}$ is called *Pareto stationary* if there exists a set of non-negative weights γ_m^a, γ_m^p, and γ_m^n summing to 1 such that $\sum_{i\in\{a,p,n\}} \gamma_m^i \nabla l_m^i(\mathbf{\Omega}_m^i, \mathcal{D}^i)$. A solution profile $\mathbf{\Omega}^*$ is *Pareto optimal* if no other Pareto stationary solution $\mathbf{\Omega}$ for $\mathbf{\Omega} \neq \mathbf{\Omega}^*$ such that $l_m^i(\mathbf{\Omega}) \leq l_m^i(\mathbf{\Omega}^*, \mathcal{D}^i)$ for all $i \in \{a,p,n\}$ and $l_m^i(\mathbf{\Omega}) < l_m^i(\mathbf{\Omega}^*, \mathcal{D}^i)$ for at least one $i \in \{a,p,n\}$.

We further introduce the following metric to quantify the error caused by the conflicting objectives among agents, and we will then propose solutions to minimize this error.

(Muti-agent) conflicting error (C-error): This corresponds to the error caused by the conflicting gradients among different agents, defined as follows:

$$\mathcal{E}_C := \| \sum_{i\in\{a,p,n\}} (\gamma_m^i - \gamma_m^{i*}) \nabla l_m^i(\mathbf{\Omega}_m, \mathcal{D}^i) \|. \tag{1}$$

Previous results [8] have already proved that the optimal solution achieves the Pareto stationary solution minimizes the convex combination of all agents' losses, i.e., we can therefore rewrite the optimization problem **P1** into the following problem:

$$\textbf{P2:} \quad \min_{\mathbf{\Omega}_m} \| \sum_{i\in\{a,p,n\}} \gamma_m^i \nabla l_m^i(\mathbf{\Omega}_m, \mathcal{D}^i) \|, \tag{2}$$

where γ_m^i satisfies $\sum_{i\in\{a,p,n\}} \gamma_m^i = 1$.

In this paper, we propose a dynamic weighting-based conflict resolving solution to minimize the C-error. In the rest of this section, we first introduce the dynamic weighting algorithm for the agent controller to dynamically control the weights among agents to influence the convergence directions. We will prove that our proposed algorithm can minimize the C-error and accelerate the convergence of the multi-agent decision-making process for cross-layer optimization.

Algorithm 1: Conflict Resolving Mechanism

Input: Training datasets $\mathcal{D}^a, \mathcal{D}^p, \mathcal{D}^n$; initial model $\mathbf{\Omega}_0$; initial weight $\gamma^a, \gamma^p, \gamma^n$.

Output: Task-specific agents $\mathbf{\Omega}_m = [\boldsymbol{\omega}_m^a, \boldsymbol{\omega}_m^p, \boldsymbol{\omega}_m^n]$.

while *not converged* **do**
  for $i \in \{a,p,n\}$ **parallel do**
    Compute dynamic weight γ_{t+1} by (4);
    Update agents' parameters by (4);
return Trained agents $\mathbf{\Omega}_m$ for specific task

Let us first introduce the dynamic weighting algorithm as follows: Generally speaking, the agent controller should always estimate the gradient descent directions that improve the losses of all the agents. In practice, however, calculating the full-batch gradients can be costly and therefore we adopt a stochastic learning solution in which the gradients of agents have been replaced by the stochastic approximated versions sampled at a single data sample d_m^i for each agent i for $i \in \{a,p,n\}$. The gradient directions as well as the model parameters of all three layers of agents are updated iteratively with independently sampled data. More specifically, at each iteration t, we draw three independent data samples $\{d_{m,t(j)}^i\}_{j\in\{1,2,3\}}$ from $\mathcal{D}^i$ in parallel for different agents and perform the following updates on weighting factors of gradient directions and agents' model parameters, respectively, as follows:

$$\gamma_{m,t+1}^i = \gamma_{m,t}^i - \eta_t \nabla l_m^i(\mathbf{\Omega}_{m,t+1}, d_{m,t(1)}^i)^\top \nabla l_m^i(\mathbf{\Omega}_{m,t+1}, d_{m,t(2)}^i), \tag{3}$$

$$\mathbf{\Omega}_{m,t+1} = \mathbf{\Omega}_{m,t} - \beta_t \nabla l_m^i(\mathbf{\Omega}_{m,t}, \gamma_{m,t+1}^i, d_{m,t(3)}^i), \tag{4}$$

where η_t, β_t are step sizes of the gradient direction and agents' model parameter update. The detailed procedures are illustrated in Algorithm 1.

We can prove the upper bound on the C-error of Algorithm 1.

Theorem 1: Suppose the following assumptions holds: *(i)* $\nabla l_m^i(\mathbf{\Omega}, \gamma)$ is ℓ_f'-Lipschitz continuous for any data sample; *(ii)* $l_m^i(\mathbf{\Omega}, \gamma)$ is ℓ_f-Lipschitz continuous for any data sample. Then, the upper bound on the C-error holds for Algorithm 1:

$$\mathcal{E}_C \leq \frac{4}{\eta T} + 6\sqrt{3\ell_f'\ell_f^2 \frac{\beta}{\eta}} + 3\eta\ell_f^4. \tag{5}$$

Proof: Due to the limit of space, we provide a sketch of proof of the above theorem as follows. According to Lemma 18 in [9], there always exists a positive constant ρ such that the following holds

$$\mathcal{E}_C \leq \rho + \frac{4}{\eta T}(1 + \rho^{-1}\beta T C_1) + \eta C_2, \tag{6}$$

where $C_1 = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla L_m(\mathbf{\Omega}_{m,t+1}) + \nabla L_m(\mathbf{\Omega}_{m,t})\| \cdot \|\sum_{i\in\{a,p,n\}}\nabla l^i(\mathbf{\Omega}_{m,t}, \gamma^i_{m,t+1}, d^i_{m,t(3)})\|$ and $C_2 = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\sum_{i\in\{a,p,n\}}\nabla l^i_m(\mathbf{\Omega}_{m,t}, \gamma^i_{m,t}, d^i_{m,t(1)})^\top \cdot \nabla l^i_m(\mathbf{\Omega}_{m,t}, \gamma^{m,t}_t, d^i_{m,t(2)})\|^2$.

According to the Lipschitz property, C_1 and C_2 are bounded by $6\ell'_f\ell_f$ and $3\ell_f^2$, respectively. By setting $\rho = 2\sqrt{3\beta\ell'_f\ell_f^2/\eta}$, we can obtain the result in (5). ■

We can observe that, by setting $\beta = \Theta(T^{-3/4})$ and $\eta = \Theta(T^{-1/4})$ in Theorem 1, the C-error converges to the conflict-resolving direction at rate $\mathcal{O}(T^{-1/4})$.

In addition to the C-error, let us also evaluate the agent's generalization capability when being deployed in the open dynamic environment that cannot be fully captured by its local model training dataset. Specifically, we adopt a commonly adopted metric to quantify the generalization capability of each agent, defined as follows:

(Model) generalization error (G-error): We define the generalization error as the discrepancy between the agent deployment performance based on the real data distribution and the training performance obtained on the training dataset. More formally, let $\tilde{l}^i_m(\omega^i_m)$ be the population loss of the agent i's model for $i \in \{a, n, p\}$, evaluated over the real data distribution of task m. We can then define G-error of agent i's model $\mathcal{E}^i_G$ as the difference of the gradients between the population loss $\tilde{l}^i_m(\omega^i_m)$ calculated based on the real distribution of agent deployment and the empirical loss $l^i_m(\omega^i_m, \mathcal{D}^i)$ obtained based on the training dataset $\mathcal{D}^i$, i.e., the G-error of agent i $\mathcal{E}^i_G$ can be written as:

$$\mathcal{E}^i_G = \|\nabla l^i_m(\omega^i_m, \mathcal{D}^i) - \nabla\tilde{l}^i_m(\omega^i_m)\|, \tag{7}$$

We can also define the overall generalization error of all the agents selected to solve task m as follows:

$$\mathcal{E}_G = \|\sum_{i\in\{a,p,n\}}\gamma^i_m(\nabla l^i_{m,t}(\mathbf{\Omega}_{m,t}, \mathcal{D}^i) - \nabla\tilde{l}^i_{m,t}(\mathbf{\Omega}_{m,t}))\| \tag{8}$$

We can then prove that the conflict-resolving mechanism can have the following bound on the G-error.

Theorem 2: Suppose the Frobenius norm of the summation of gradients of all agents is upper bounded by constant, i.e, $\mathbb{E}[\|\sum_{i\in\{a,p,n\}}\gamma^i_m\nabla l^i_m(\mathbf{\Omega}_m, \mathcal{D}^i)\|^2_{\mathrm{F}}] \leq U^2$ for U is a constant. Then, the G-error of Algorithm 1 is upper bounded by $\mathcal{E}_G \leq \mathcal{O}(T^{\frac{1}{2}}D^{-\frac{1}{2}})$.

Proof: We provide a sketch of proof of the above theorem as follows. We can first prove that, if a single training data sample is modified or removed, e.g., the training dataset $\mathcal{D}^i$ of agent i is changed to $\tilde{\mathcal{D}}^i$ where $\tilde{\mathcal{D}}^i$ is identical to $\tilde{\mathcal{D}}^i$ except that a single data sample is different, the gradient of the loss function in our proposed Algorithm 1 is upper bounded by a constant $\sup\mathbb{E}[\|\sum_{i\in\{a,p,n\}}\gamma^i_m(\nabla l^i_m(\mathbf{\Omega}_m, \mathcal{D}^i) - \nabla l^i_m(\mathbf{\Omega}_m, \tilde{\mathcal{D}}^i))\|^2_{\mathrm{F}}] \leq \epsilon^2$.

We can then follow the same line as [9] and prove the G-error has the following upper bound $\mathcal{E}_G \leq 4\epsilon + \sqrt{\frac{V}{D}}$, where V is the variance of gradients and D is the size of the training dataset. Substituting the upper bounds of gradients assumed in Theorem 2, we can also prove that ϵ is bounded by a constant, given by $\epsilon^2 \leq \frac{4U^2T}{D}$. Finally, by substituting this bound of ϵ into the upper bound of $\mathcal{E}_G$, we can obtain the result. ■

IV. Prototype and Experimental Results

A. Prototype

We develop an open RAN and open 5GS-based prototype for evaluating the practical performance of SANNet, as shown in Fig. 2. The hardware and software platforms implemented in our prototype are described as follows:

Hardware platform: The hardware of our SANNet prototype is composed of a UE, a gNB, and a 5G core network. The UE is an NI 2944R USRP connected to a desktop computer installed with an Intel(R) Core(TM) i7 CPU@2.4 GHz with 32 GB memory and 1TB SSD. The gNB is an NI 2944R USRP connected to a workstation computer installed with an Intel(R) Core(TM) i9-13900K CPU@5.8GHz, 128.0GB RAM@4000.0MHz, 1 TB SSD, 4 TB HDD, and 1 NVIDIA GeForce RTX 4090 GPU.

Software platform: The software of our prototype consists of an open source 5G open RAN compatible software radio platform, srsRAN, installed on both UE and gNB. An open 5GS-based 5G core network is also installed on the gNB.

B. Experimental Setup

We consider immersive communication driven by the user's QoE as a case study to demonstrate how SANNet can infer the implicit semantics of the user and autonomously initiate cross-layer joint optimization between aAgent, nAgent, and pAgent to meet the inferred QoE demand of the user. In our case study, an aAgent, e.g., a virtual assistant application, detects a user's unsatisfactory with the quality (resolutions) of an online streamed video by detecting one or multiple prompts from the user's language. For example, if the aAgent detects prompts such as "increase video resolution" and "make video clearer" from the user's language, it will understand that the semantic goal of the user corresponds to "increasing the resolution of an online streamed video without sacrificing other QoE-related performance such as smoothness and latency of the video streaming services". In this case, it will send this detected goal to the agent controller. The agent controller will infer the semantic goal of the user and separate the task goal into three subtasks to be assigned to three different agents. Then, the immersive communication aAgent corresponds to a Unity-based 3D video rendering application that can adjust video resolutions among 360p, 480p, 640p, 720p, and 1080p; a pAgent which corresponds to a multi-channel sensing and assigning agent that can sense the CSIs of multiple channels and estimate the data rates that can be supported by each channel; and nAgent which corresponds to a network bandwidth tracking and prediction agent that can keep track of the network bandwidth, max data traffics, between the IP addresses of the UE and 5G core.

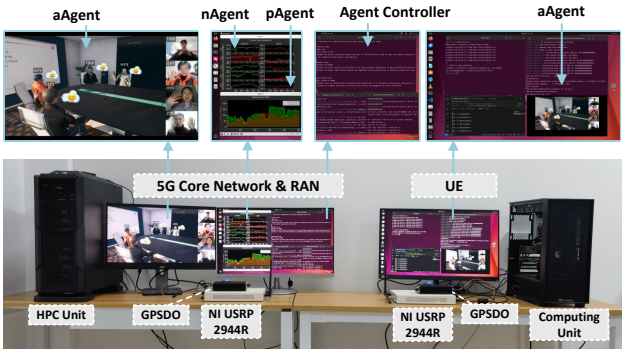
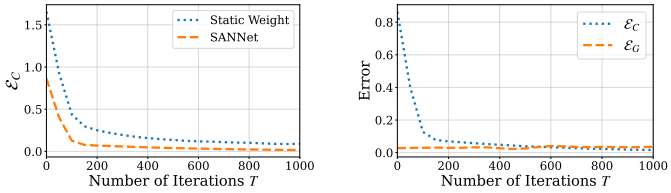


Fig. 2: A SANNet prototype.



(a) (b)

Fig. 3: (a) C-error of our proposed dynamic weight-based conflict resolving mechanism, compared to the static weight solution, and (b) a tradeoff between C-error and G-error.

We describe our dataset, configurations of different agents' models, and that of the agent controller as follows:

Dataset: We build a multi-channel dataset consisting of data samples collected by aAgent, pAgent, and nAgent in application-layer, physical-layer and network-layer. More specifically, for the dataset of aAgent, we assume the user can only request changes of video quality every 5 seconds and randomly generate a time sequence of the user's requests. For the dataset of pAgent, we record CSI data samples in five 5G NR frequency bands, including n1, n2, n3, n5, n7 bands. For the dataset of nAgent, we record the maximum achievable bandwidth between IP addresses of the UE and 5GS core measured by the iPerf3 tool.

Agents: We train three time series prediction models based on the transformer network with three different loss functions l_1 loss, mean square error (MSE) loss, and log-Cosh loss for the sensing module of aAgent, pAgent, and nAgent, respectively. Each agent will feedback its prediction result to the agent controller and coordinate with each other following the aforementioned steps.

Agent controller: We adopt an open-source AI agent platform, OpenManus, as the agent controller and install an open-source transformer-based LLM model, Qwen-7B, to infer the user's QoE via language input.

C. Experimental Results

We first evaluate the conflict resolving performance of our proposed dynamic-weighting mechanism for the agent controller. As mentioned earlier, three different agents adopt three different loss functions, representing three different objectives, resulting in degraded performance for model training. In Fig. 3(a), we present the C-error of the models trained under different numbers of iterations, compared to the traditional multi-agent collaboration model training solution with static weight. We can observe that our proposed dynamic weight-based solution can reduce up to 63% of C-error, compared to the static weight approach.

We also compare the G-error and C-error under different numbers of iterations in Fig. 3(b). We observe that, compared to the C-error, which decreases dramatically, the G-error increases slightly when the number of iterations becomes large. This is because when the number of iterations becomes large, the model tends to fit too closely to the training dataset, resulting in decreased performance when the ground truth environment is different.

V. Conclusion

This paper proposes SANNet, a novel semantic-aware agentic AI networking architecture that autonomously identifies the user's semantic goal and autonomously divides the identified goal into different subtasks for different agents. We introduce a novel functional entity, the agent controller, employed with a novel dynamic weighting-based conflict-resolving mechanism. We develop a hardware prototype and our experiment result suggests that SANNet significantly improves the performance of multi-agent networking systems.

Acknowledgment

The work of Y. Xiao was supported in part by the National Natural Science Foundation of China (NSFC) under grant 62525109. The work of G. Shi was supported in part by the National Natural Science Foundation of China (NSFC) under grant 62293483. The work of P. Zhang was supported in part by the NSFC under grants 62293480 and 62293481. The work of Y. Xiao, G. Shi, and P. Zhang was supported in part by the Mobile Information Network National Science and Technology Key Project under grant 2024ZD1300700.

References

[1] Y. Yang *et al.*, "6G network AI architecture for everyone-centric customized services," *IEEE Network*, vol. 37, no. 5, Sep. 2023.

[2] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, "Toward Self-Learning Edge Intelligence in 6G," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 34–40, Dec. 2020.

[3] M. R. Morris *et al.*, "Position: Levels of AGI for operationalizing progress on the path to agi," in *ICML*, 2024.

[4] Z. Durante *et al.*, "Agent AI: Surveying the horizons of multimodal interaction," *arXiv preprint arXiv:2401.03568*, 2024.

[5] Y. Shavit, S. Agarwal, M. Brundage, S. Adler, C. O'Keefe, R. Campbell, T. Lee, P. Mishkin, T. Eloundou, A. Hickey *et al.*, "Practices for governing agentic ai systems," *Research Paper, OpenAI*, 2023.

[6] Y. Xiao, G. Shi, and P. Zhang, "Towards agentic AI networking in 6G: A generative foundation model-as-agent approach," *IEEE Communications Magazine*, vol. 63, no. 9, Sep. 2025.

[7] G. Shi and Y. Xiao, "An introduction to semantic communication and semantic-aware networking standardization for 6G," *GetMobile: Mobile Comp. and Comm.*, vol. 28, no. 3, p. 14–19, Oct. 2024.

[8] J.-A. Désidéri, "Multiple-gradient descent algorithm (mgda) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5-6, pp. 313–318, 2012.

[9] L. Chen *et al.*, "Three-way trade-off in multi-objective learning: Optimization, generalization and conflict-avoidance," *NIPS*, vol. 36, pp. 70 045–70 093, 2023.