SCIENCE CHINA
Information Sciences

• Supplementary File •

Service agent for user-centric QoE-aware communication service provision in mobile AR

Conghao Zhou[1], Lulu Sun[1], Peng Yang[2*], Feng Lyu[3], Nan Cheng[1*] & Xuemin (Sherman) Shen[4]

[1]*School of Telecommunications Engineering, Xidian University, Xi'an 710071, China*
[2]*School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China*
[3]*School of Computer Science and Engineering, Central South University, Changsha 410083, China*
[4]*Department of Electrical and Computer Engineering, University of Waterloo, Waterloo N2L 3G1, Canada*

Appendix A Details of MAR-SA workflow and tool encapsulation

Appendix A.1 Prompt structure

In the proposed agent-driven framework, each UDT interacts with the MAR-SA on behalf of its corresponding MAR user to obtain the inferred QoE value under a given bandwidth allocation decision. Each such interaction is referred to as a *UDT request*. In a UDT request, the UDT sends a structured prompt that encapsulates user-related contextual information, such as QoE requirement, MAR device configuration, and the characteristics of the immersive scene being viewed, along with a target future timestamp $f + W$. Upon receiving this prompt, the LLM core within the MAR-SA invokes the relevant tools from the SFT to estimate the user's QoE at the specified future moment.

To ensure accurate tool selection by the LLM core, we design a structured prompt tailored to the MAR-SA. The core of this prompt is to establish a clear role, task, and output format for the LLM core, thereby constraining its general language capabilities within our specific task framework. As illustrated in Fig. A1, we first define the role of the LLM core as the assistant of the network controller. We then specify its core task: to analyze the user's request, select the most appropriate tool, and, crucially, to respond with a strictly formatted JSON object for the tool call. This mandatory format is vital as it transforms the LLM core non-deterministic semantic understanding into a machine-readable instruction that our backend can deterministically parse and execute. This achieves a reliable mapping from a natural language request to a programmatic action.

Appendix A.2 Designed service function tools

The LLM core itself cannot execute external code or directly call external APIs, nor does it possess inherent awareness of the SFT. To enable the LLM core to access and utilize these external tools, it must be provided with a detailed, well-structured description of the available tools in a format it can interpret. To this end, we design service function tools based on MCPs, through which all callable tools within the SFT are abstracted into a standardized list referred to as "`TOOLS`" list. This design allows the LLM core to reason over the defined tool set and invoke the appropriate functions during QoE inference. Each tool entry in the "`TOOLS`" list is defined using three key fields:

- "`Name`": A string corresponding to the executable tool name within the SFT.
- "`Description`": A natural language text describing the tool's functionality, applicable scenarios, and expected outcomes, serving as the primary basis for the semantic understanding of the tool's purpose.
- "`Parameters`": An object following the JSON schema specification that precisely defines all input arguments required for tool invocation, including parameter names, data types, and required attributes.

Before processing UDT requests, the MAR-SA serializes the entire "`TOOLS`" list into a JSON-formatted string and dynamically embeds it into the structured prompts, as shown in Fig. A1. This process informs the LLM core of the available tools and their usage specifications, thereby enabling it to reason over and invoke the appropriate tools when performing QoE inference.

To enable QoE-aware service provisioning, our framework designs and encapsulates two core service function tools within the SFT. First, the `QoE_Prediction_Tool` encapsulates the deterministic computation of user QoE based on the algorithms or mechanisms employed in the MAR application within the service domain. This application-specific tool is the key factor enabling MAR-SA to outperform conventional QoE modeling-based approaches. Second, the `User_Context_Query_Tool` supports the retrieval and processing of historical user data, such as querying a user's past bandwidth allocations and actually perceived QoE values stored in the UCR.

As shown in Fig. A2, each tool is defined with a clear name, description, and parameter structure. This standardized definition results in two advantages. First, the LLM core can leverage its powerful natural language understanding capabilities to precisely map the intent in the UDT request to a specific tool invocation. Second, by abstracting the complex mechanisms and algorithms employed in MAR applications into independent, callable tools, the MAR-SA does not require any training or fine-tuning to understand the internal algorithmic details implemented by the MAR service provider, thereby enhancing the flexibility of the agent-driven framework.

* Corresponding author (email: yangpeng@hust.edu.cn, dr.nan.cheng@ieee.org)

Prompt for the MAR service agent

```
# 1. Role and Task Definition
You are an assistant of the network controller, with access to the foll-
owing tools. Your task is to analyze UDT requests and select the most
appropriate tool to match the user's intent based on the functionalities
of the tools in the TOOLS list.

# 2. Tool Selection Guide
Your decision should be based on the following guidelines:
- If the user's intent involves QOE-AWARE RESOURCE MANAGEMENT (e.g.,
  contains keywords like "optimize bandwidth", "allocate resources",
  "predict future performance"), you must use the "QoE_Prediction_Tool".
- If the user's intent involves QUERYING HISTORICAL DATA (e.g.,
  contains keywords like "get user history", "query past data", "check
  user behavior"), you must use the "User_Context_Query_Tool".

# 3. Output Specification
Your response MUST ONLY be a JSON object for tool calling. For example:
{
    "tool_name": "Name_Of_The_Tool_To_Call",
    "parameters": {
        "argument_name": "extracted_value"}}
```

Figure A1 The exemplary structure of the prompt.

Appendix B System model and QoE-aware service provisioning algorithm

Appendix B.1 System model and problem formulation

Appendix B.1.1 *Considered scenario*

We investigate a multi-user edge-assisted MAR scenario, where users equipped with MAR devices move within the coverage area of a base station (BS), as shown in Fig. B1. To support MAR applications, each device must perform annotation rendering, i.e., merging the camera-captured scene of the physical environment with the virtual content to be displayed. Specifically, each device periodically captures camera frames and overlays spatially aligned virtual content onto them, thereby presenting an augmented view in the user's field of view (FOV) in real time [1]. Annotation rendering necessitates accurate device pose tracking, not only for the current pose but also for predicting future poses, to ensure that virtual content is precisely rendered and seamlessly aligned with the physical scene in forthcoming camera frames. However, due to the limited computational capability of current MAR devices, a promising paradigm is to offload compute-intensive tasks to an edge server, where the MAR service provider conducts device pose tracking and prediction, and then transmits the pose information back to each MAR device for rendering. Let $\mathcal{U}$ denote the set of MAR devices, and let $\mathcal{F}_u$ denote the set of camera frames captured by MAR device $u \in \mathcal{U}$. All MAR devices are assumed to operate under a synchronized time reference. For notational simplicity, we use f to represent both the index of a camera frame and the timestamp at which it is captured.

The workflow of edge-assisted annotation rendering for MAR is summarized as follows:

1. **Key Frame Uploading:** Each MAR device periodically uploads selected camera frames, referred to as key frames, to an edge server co-located with the BS. These key frames serve as the basis for device pose tracking and prediction;

2. **Device Pose Tracking:** When a camera frame $f \in \mathcal{F}_u$ is captured, the MAR service provider estimates the 6-DoF pose of device u at timestamp f on the edge server. For device u, the pose corresponding to frame f is represented by the vector

$$\mathbf{q}_{u,f} = [t^x_{u,f}, t^y_{u,f}, t^z_{u,f}, \theta^x_{u,f}, \theta^y_{u,f}, \theta^z_{u,f}]^\top, \tag{B1}$$

   which specifies the device's 3D position and orientation in space [2].

3. **Device Pose Prediction:** The MAR service provider subsequently predicts the future 6-DoF pose of device u at timestamp $f + W$, where W denotes the look-ahead window length. This prediction is based on a historical sequence of poses, denoted by $\mathcal{Q}_{u,f}$. Based on the assumption that the uploaded camera frames are uniformly sampled in temporal order from the complete sequence of captured frames, the set $\mathcal{Q}_{u,f}$ is as follows:

$$\mathcal{Q}_{u,f} = \left\{ \mathbf{q}_{u,k} \mid f - H < \frac{k}{\lambda_u} \leqslant f \right\}, \tag{B2}$$

Example Tool Definitions for the MAR Service Agent

```
TOOLS:[
  {
    "name": "QoE_Prediction_Tool",
    "description": "Used for resource management tasks. This tool predicts a
       user's future Quality of Experience (QoE) under a given bandwidth
       resource allocation decision.",
    "parameters": {
        "type": "object",
        "properties": {
            "user_identifier": {
                "type": "string",
                "description": "The unique user identifier corresponding
                                to the prediction target."
            }
        },
      "required": ["user_identifier"]
    }}]
```

Figure A2 The structured definition of the "QoE_Prediction_Tool".

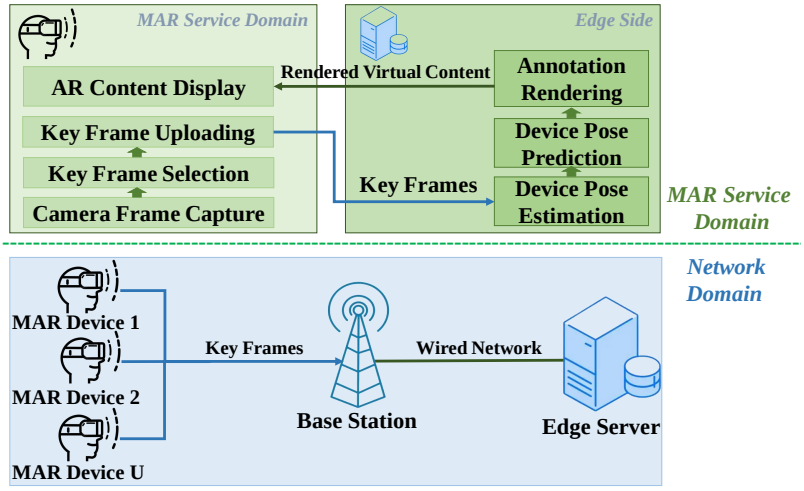

Figure B1 The illustration of the considered scenario.

where H is the history window length, and λ_u denotes the uplink sampling frequency (in Hz), i.e., the rate at which device u selects camera frames for transmission from the continuous capture stream. The predicted pose is obtained as

$$\hat{\mathbf{q}}_{u,f+W} = G(\mathcal{Q}_{u,f}), \tag{B3}$$

where $G(\cdot)$ denotes the pose prediction function proposed and implemented by the MAR service provider. Importantly, the internal design and parameters of G are typically inaccessible to the network controller, posing challenges for cross-layer optimization in networking and communication fields.

4. **Virtual Content Delivery:** Using the predicted pose $\hat{\mathbf{q}}_{u,f+W}$, the edge server proactively renders the virtual content expected to appear in the user's FOV, thereby enabling low-latency and efficient content delivery to the corresponding MAR device. Leveraging the pose information and rendered virtual content provided by the edge server, each MAR device then performs annotation rendering locally.

Appendix B.1.2 *Key frame uploading*

The accuracy of device pose prediction, and thus the immersion level during MAR rendering, strongly depends on the uplink transmission of camera frames. Therefore, the network controller must allocate sufficient bandwidth resources to each MAR device to ensure timely frame uploading. For device u, the uplink data rate at the time when camera frame $f \in \mathcal{F}_u$ is captured is expressed as:

$$r_{u,f} = b_u \log(1 + \gamma_{u,f}), \tag{B4}$$

where b_u represents the spectrum bandwidth (in Hz) allocated to device u, and $\gamma_{u,f}$ is the corresponding signal-to-noise ratio.

Given the limited availability of uplink communication resources, only a subset of the captured camera frames can be selected for transmission [2]. Each frame is assumed to have an identical data size of α bits. To guarantee timely uploading, the average uplink latency of MAR device u, denoted by τ_u, measured from the moment a frame is selected to the completion of its transmission, must not exceed a predefined maximum tolerable delay T. According to the D/G/1 queuing model, the frame uploading process is thus subject to the following constraint:

$$\tau_u \approx \mathbb{E}[S_u] + \frac{\lambda_u \mathbb{E}[S_u^2]}{2(1-\lambda_u \mathbb{E}[S_u])} \leqslant T, \tag{B5}$$

where $\mathbb{E}[S_u]$ denotes the average transmission time for uploading a camera frame from device u, as follows:

$$\mathbb{E}[S_u] = \mathbb{E}\left[\frac{\alpha}{r_{u,f}}\right]. \tag{B6}$$

Given the delay requirement for key frame uploading, i.e., τ_u, allocating more bandwidth to user u increases the value of λ_u, thereby enlarging the set $\mathcal{Q}_{u,f}$ used for device pose prediction.

Appendix B.1.3 *QoE model for MAR rendering*

The immersion level during MAR rendering is determined by the alignment accuracy between the rendered virtual object and the user's actual real-time viewpoint, especially regarding the object's spatial consistency within the FOV. Accordingly, the QoE for MAR device u at time instant f is quantified using the virtual content hit rate (VCHR), defined as $h_{u,f}$. This metric reflects the degree of spatial correspondence between the virtual object rendered from the predicted device pose and the one that should have been rendered based on the user's true pose at that future time, formulated as [3]:

$$h_{u,f} = \frac{|\mathcal{C}(\mathbf{q}_{u,f+W}) \cap \mathcal{C}(\hat{\mathbf{q}}_{u,f+W})|}{|\mathcal{C}(\mathbf{q}_{u,f+W}) \cup \mathcal{C}(\hat{\mathbf{q}}_{u,f+W})|}, \tag{B7}$$

where $\mathcal{C}(\mathbf{q})$ represents the set of virtual object cells that fall within the user's view frustum at a given device pose $\mathbf{q}$. The symbols $\cap$ and $\cup$ denote the intersection and union of the two sets, respectively, and $|\cdot|$ indicates the cardinality of a set. A VCHR value close to 1 signifies an almost perfect spatial alignment between the rendered virtual content and the user's actual FOV, corresponding to a highly immersive experience. Conversely, a lower VCHR value implies noticeable misalignment, leading to a degradation in perceived immersion. The QoE of user u is therefore defined as follows:

$$h_u = \frac{\sum_{f \in \mathcal{F}_u} h_{u,f}}{|\mathcal{F}_u|}, \ \forall u \in \mathcal{U}. \tag{B8}$$

Since the VCHR is intrinsically determined by the accuracy of device pose prediction, it can be modeled as a function of the allocated uplink frequency spectrum resources, expressed as

$$h_u = \Phi_u\left(\left\{G(\mathcal{Q}_{u,f}) | \forall f \in \mathcal{F}_u\right\}, b_u\right), \ \forall u \in \mathcal{U}, \tag{B9}$$

where the functional form of $\Phi_u(\cdot)$ is *user-specific*, as the VCHR value strongly depends on the movement trajectory of the corresponding device pose.

Appendix B.1.4 *Problem formulation*

Our objective is to reserve sufficient uplink spectrum resources to improve overall user QoE while minimizing spectrum consumption. Let the minimum QoE requirement of MAR user u be denoted by $h_u^{\min}$. Since the available spectrum resource may not always guarantee that every user's QoE requirement is satisfied, we define a slack variable δ_u to represent the gap between the actual QoE perceived by user u and its required level, given by:

$$\delta_u(b_u) = \max\left(0, h_u^{\min} - h_u(b_u)\right), \tag{B10}$$

where $\delta_u(b_u)$ indicates the QoE deficiency of user u under bandwidth allocation b_u, which becomes active when the user's QoE falls below the minimum acceptable threshold $h_u^{\min}$. The resource reservation decision for all MAR devices is represented by the set $\mathbf{b} = \{b_u\}_{u \in \mathcal{U}}$. The proactive QoE-aware MAR service provisioning problem is formulated as follows:

$$\text{P1:} \ \max_{\mathbf{b}} \sum_{u \in \mathcal{U}} h_u - w \sum_{u \in \mathcal{U}} b_u - v \sum_{u \in \mathcal{U}} \delta_u \tag{B11a}$$

$$\text{s.t.} \ h_u \geqslant h_u^{\min} - \delta_u, \quad \forall u \in \mathcal{U}, \tag{B11b}$$

$$\sum_{u \in \mathcal{U}} b_u \leqslant B^{\text{total}}, \tag{B11c}$$

where $\Delta = \{\delta_u\}_{u \in \mathcal{U}}$, B^{total} denotes the total available bandwidth, and w and v are weighting parameters that balance user QoE, spectrum consumption, and the QoE shortfall, i.e., the discrepancy between the actual QoE perceived by a user and its required level.

Solving Problem P1 is challenging due to two primary factors. First, the functional relationship $\Phi_u(\cdot)$ that maps the allocated bandwidth resource b_u to the resulting user QoE h_u is unknown, as it is inherently non-stationary and user-specific. This arises from the continuous evolution of user mobility patterns and environmental contexts, which differ considerably among users and thus compromise

the accuracy of QoE estimation when using conventional predictive approaches. Second, the functional decoupling between network control and the MAR service provider limits cross-domain coordination. For example, the decision-making process for allocating b_u lacks access to the employed pose-prediction mechanism G_u, which resides within the OTT MAR service domain rather than the network domain. Consequently, achieving QoE-aware service provisioning for MAR calls for a novel network management framework that can bridge the data and functional isolation between the MAR service and network domains, thereby enabling cross-layer QoE optimization in the 6G era.

Appendix B.2 Semi-distributed communication service provisioning

In this section, we develop a QoE-aware communication service provisioning approach to solve Problem P1 based on the proposed MAR-SA-driven framework.

While the proposed MAR-SA effectively addresses the challenge of the unknown mapping between h_u and b_u in Problem P1, this mapping remains *implicit*, i.e., the closed-form expression of the function $\Phi_u(\cdot)$ is still unavailable. Moreover, since the functional form of $\Phi_u(\cdot)$ is user-specific, it poses additional challenges for conventional centralized optimization-based resource allocation approaches. Considering the overhead of prompt processing and tool orchestration at the LLM core, we propose a semi-distributed approach that leverages the interactions between UDTs and the MAR-SA.

The key idea of the proposed semi-distributed approach is that each UDT sends a request on behalf of MAR user u to the MAR-SA, and reports its desired bandwidth b_u based on its own utility function. The MAR-SA returns the inferred QoE value of user u under a given bandwidth allocation decision b_u. Subsequently, the network controller determines the final bandwidth allocation by considering the overall QoE performance across all users represented by their respective UDTs.

Although the network controller, the MAR-SA, and UDTs operate as distinct network management entities with different execution logics, they can be physically co-located, allowing multi-round interactions with negligible latency, unlike conventional distributed approaches, where inter-entity coordination typically incurs high interaction delays. Meanwhile, the semi-distributed design alleviates the stringent requirement of conventional centralized schemes that rely on prior knowledge of all users' closed-form QoE models for global optimization. Instead, the LLM core triggers the underlying tools for on-demand QoE computation only when triggered by a UDT request, thereby reducing computational overhead while enabling user-specific QoE inference.

Appendix B.2.1 *Marginal utility*

Owing to the implicit form of $\Phi_u(\cdot)$, the bandwidth resource is discretized for service provisioning. The total bandwidth B^{total} is uniformly divided into Z units, each with bandwidth $\Delta b = B^{\text{total}}/Z$. Based on the objective function of Problem P1, the overall system utility is defined as:

$$\Pi(\mathbf{b}) = \sum_{u\in\mathcal{U}} h_u - w \sum_{u\in\mathcal{U}} b_u - v \sum_{u\in\mathcal{U}} \delta_u. \tag{B12}$$

The marginal utility of the system when allocating an additional bandwidth unit Δb_u to user u is defined as the difference between the total utilities before and after the allocation:

$$\Delta\Pi_u^{k'} = \Pi(\mathbf{b}^k + \Delta b_u) - \Pi(\mathbf{b}^k). \tag{B13}$$

where k and k' denote the system states before and after the allocation of Δb to user u. Substituting $\Pi(\mathbf{b})$ and noting that the allocation only affects user u, the marginal utility can be expressed as:

$$\begin{aligned}\Delta\Pi_u^{k'} = &\underbrace{\left(h_u(b_u^k + \Delta b) - h_u(b_u^k)\right)}_{\text{Gain from QoE improvement}} \\ &\underbrace{-w \cdot \Delta b_u}_{\text{Bandwidth consumption}} \\ &\underbrace{-v \cdot \left(\max(0, h_u^{\min} - h_u(b_u^k + \Delta b)) - \max(0, h_u^{\min} - h_u(b_u^k))\right)}_{\text{Penalty for not guaranteed QoE}}\end{aligned} \tag{B14}$$

Appendix B.2.2 *Semi-distributed service provisioning algorithm*

Based on the marginal utility function, we develop a semi-distributed service provisioning algorithm that operates through the interaction between the UDTs and the MAR-SA. The service provisioning decisions are made periodically with an interval of T. The procedure for one decision cycle is summarized in Algorithm 1.

In Line 2, the initial bandwidth allocation decisions are set, and the "`TOOLS`" list in the SFT is invoked to support subsequent tool usage by the LLM core. The term X^k denotes the total amount of bandwidth allocated up to the k-th iteration, and $\mathcal{G}$ represents the set of potential marginal gains for all users if an additional bandwidth unit Δb_u is assigned. From Lines 3 to 8, each UDT, on behalf of its corresponding MAR user u, sends a request to the MAR-SA to get the user's QoE when an additional Δb is allocated. The UDT request follows the structured prompt format defined in Section **??**. Upon receiving the prompt, the LLM core leverages the UCR and the pre-defined tools in the SFT via the MCP to prediction the QoE value. Notably, the raw user data remain within the MAR service domain, and only the QoE values are returned to the UDT and exposed to the network controller. From Lines 9 to 13, the marginal utility $\Delta\Pi_u^k$ is calculated for each user u, capturing the QoE gain, bandwidth cost, and QoE satisfaction penalty. Subsequently, from Lines 14 to 24, the network controller aggregates the marginal utilities of all UDTs and determines the service provisioning decisions by balancing bandwidth allocation across users. Finally, in Line 25, once the actual perceived QoE values of all users are recorded, the corresponding data are stored in the UCR by the MAR application service provider, in parallel with the network controller.

Algorithm B1 Semi-distributed service provisioning algorithm

Input: $\mathcal{U}$, B^{total};
Initialization: $k=0$, $\mathbf{b} \leftarrow \mathbf{0}_{N\times 1}$, $X^k \leftarrow 0$, $\mathbf{h}^k$, "TOOLS" List;
while $X^k + \Delta b_u \leqslant B^{\text{total}}$ **do**
  $\mathcal{G} = \emptyset$;
  for $u \in \mathcal{U}$ **do**
    $\hat{b}^{k+1} \leftarrow b_u + \Delta b$
    The UDT on behalf of user u generates prompt following Section 4;
    $\hat{h}_u^{k+1} \leftarrow$ The MAR-SA returns the inferred QoE value for user u given $\hat{b}^{k+1}$;
    $\Delta h_u^{k+1} = \hat{h}_u^{k+1} - h_u^k$;
    $\delta_u^k = \max(0, h_u^{\min} - h_u^k)$;
    $\hat{\delta}_u^{k+1} = \max(0, h_u^{\min} - \hat{h}_u^{k+1})$;
    Calculate the marginal utility $\Delta\Pi_u^{k+1}$ for user u;
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{\Delta\Pi_u^{k+1}\}$;
  end for
  $u^* \leftarrow \arg\max_{u\in\mathcal{U}} \mathcal{G}$;
  $\Delta\Pi_u^{k+1,*} \leftarrow \Delta\Pi_{u^*}^{k+1}$;
  if $\Delta\Pi_u^{k+1,*} \leqslant 0$ **then**
    break;
  end if
  $b_{u^*} \leftarrow b_{u^*} + \Delta b$;
  $h_{u^*} \leftarrow \hat{h}_{u^*}^{k+1}$;
  $X^{k+1} \leftarrow X^k + \Delta b$;
  $k \leftarrow k+1$;
end while
The actually perceived QoE values of all users are asynchronously recorded in the UCR;
Output: $\mathbf{b}^*, \mathbf{h}^*$;

Appendix C Detailed simulation results and discussions

Appendix C.1 Performance evaluation

In this section, we conduct trace-driven simulations to validate the performance of our proposed agent-driven communication service provisioning approach.

Appendix C.1.1 *Simulation settings*

We conduct simulations involving at most 40 MAR users to evaluate the performance of the proposed framework. To realistically capture user mobility patterns, we utilize a real-world 6DoF pose dataset. The virtual scene content is sourced from the 8i Voxelized Full Bodies (8iVFBv2) dataset, which provides high-quality dynamic point cloud sequences containing four human motion scenes: longdress, loot, redandblack, and soldier. Each sequence is rendered at 30 frames per second over a 10-second duration. For each camera frame, the virtual content is represented as a point cloud, which is spatially divided into a $4 \times 4 \times 2$ grid of cells [4]. Following the approach in [3], the visible cell set is determined based on both the ground-truth and predicted user poses, defined as the non-occluded cells located within the user's viewing frustum. The look-ahead window for pose prediction is set to $W \in \{1, 2, 3, 4, 5\}$.

The proposed MAR-SA-driven framework is designed for personalized QoE modeling and inference, which can be formulated as a time-series forecasting problem in conventional studies. Accordingly, we adopt six representative state-of-the-art AI-based approaches from the time-series modeling domain as baselines for comparison: Transformer [5], TimesNet [6], PatchTST [7], FreTS [8], iTransformer [9], and TSMixer [8]. These methods represent the mainstream paradigms for time-series prediction and serve as baselines for evaluating the performance of the proposed framework. To enable a fair and comprehensive performance assessment, we employ well-defined evaluation metrics. For each user, the mean squared error (MSE) is computed to quantify the accuracy of QoE prediction. In addition, the QoE inference task is reformulated as a ten-class classification problem by discretizing the continuous QoE range $[0, 1]$ into ten equal intervals. A prediction is regarded as correct if the inferred QoE value falls within the same interval as its ground truth. The final MSE and classification accuracy results are obtained by averaging the corresponding values across all users.

Appendix C.1.2 *Performance of QoE inference*

To evaluate the robustness of the model, we conducted tests on the *redandblack* dataset. This dataset is characterized by user movement trajectories that are fast-changing and involve frequent rotations, making future poses highly unpredictable. As shown in Table C1, the proposed MAR-SA framework exhibits a significant performance gap over the baseline AI-based approaches on this dataset.

It is important to emphasize that this comparison is not intended to claim algorithmic superiority in traditional time-series forecasting. Instead, it illustrates the fundamental limitations of relying purely on historical data regression under cross-domain information isolation.

Table C1 Performance comparison between the MAR-SA and baselines (a factor of 10^4 has scaled MSE values).

Datasets → / Methods ↓	redandblack									
	step1		step2		step3		step4		step5	
	MSE	Acc.	MSE	Acc.	MSE	Acc.	MSE	Acc.	MSE	Acc.
Transformer	125	0.782	128	0.791	123	0.797	123	0.795	107	0.805
PatchTST	98	0.832	100	0.823	97	0.832	102	0.822	90	0.835
TimesNet	289	0.807	279	0.807	273	0.818	261	0.820	271	0.817
FreTS	361	0.799	327	0.786	321	0.787	312	0.783	315	0.787
iTransformer	156	0.802	177	0.783	136	0.768	162	0.805	129	0.810
TSMixer	91	0.841	104	0.828	108	0.834	106	0.833	100	0.834
MAR-SA	**18**	**0.961**	**27**	**0.949**	**34**	**0.935**	**42**	**0.923**	**51**	**0.909**

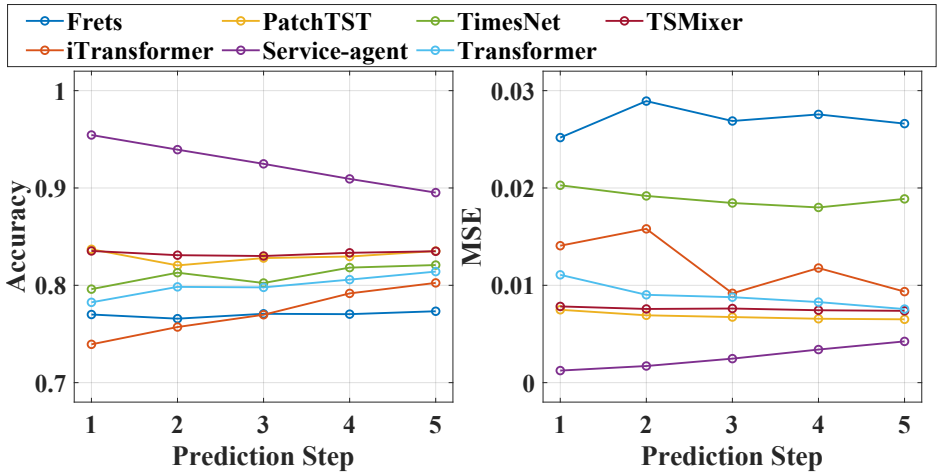

Figure C1 QoE modeling performance versus look-ahead window length.

The baseline models, constrained by this isolation, attempt to learn and reproduce complex user movement patterns solely through black-box data-driven fitting. In contrast, MAR-SA achieves exceptionally high accuracy due to its unique architectural privilege: it utilizes the LLM's "Tool-use" mechanism to break the cross-layer black box, invoking the actual deterministic logic (i.e., the application's ground truth algorithm) residing in the MAR service domain for QoE computation.

Therefore, the final evaluated QoE value is derived from a precise tool invocation rather than from a conventional model trained to learn latent data relationships. The overwhelming performance advantage of MAR-SA inherently stems from an architectural paradigm shift toward cross-domain computational synergy, rather than a mere algorithmic improvement in data-dimensional fitting capability.

Crucially, the LLM itself does not directly predict QoE; rather, it exclusively performs intent recognition and tool selection. By invoking the actual deterministic logic, MAR-SA leverages its architectural privilege to break the traditional cross-layer "black box", selectively extracting only the information necessary for QoE-aware service provisioning from data residing within the MAR service domain, thereby achieving dynamic cross-domain interaction and synergy.

In Fig. C1, we present the multi-step prediction performance of each model on the Loot dataset, which features user movement trajectories that are relatively smooth and more predictable. As illustrated, the MSE of the baseline models is not only higher in value but also exhibits significant unstable fluctuations as the number of prediction steps increases. While the error of our framework also increases steadily with the horizon, its absolute value consistently remains at a low level. This monotonically increasing error is a consequence of the recursive prediction mechanism in our framework's design. Because the prediction is conditioned on previously predicted states, an accumulation of prediction errors over time is inevitable. Our framework's design incorporates an effective self-correction mechanism. This mechanism can gradually offset the negative impacts of accumulated error during the prediction process, ultimately breaking the paradigm of perpetual error growth.

In Fig. C2, we provide a statistical overview of each model's performance across all test users via box plots. As illustrated, our model's box plot is narrower and has fewer outliers. In contrast to conventional AI-based approaches that depend on data fitting, our framework encapsulates the core QoE evaluation process into a deterministic algorithmic API, it can effectively handle novel user behavior patterns not present in the training data, as the designed SFT enables the LLM core to learn each individual user's behavioral pattern from just a few data samples. This shows the potential of tool invocation in personalized QoE modeling compared to conventional big data-driven regression.

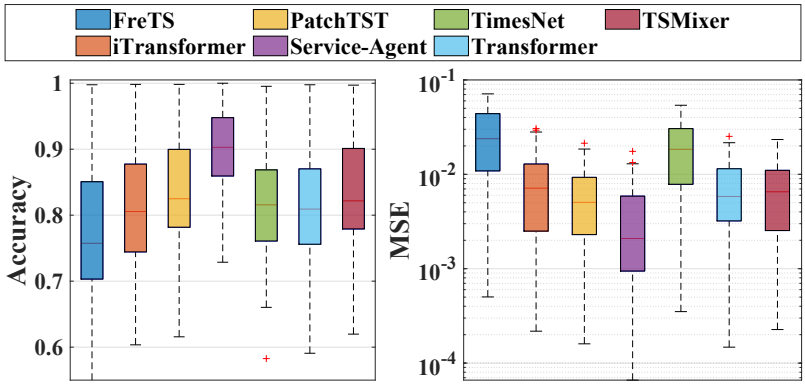

Figure C2 Distribution of QoE modeling performance.

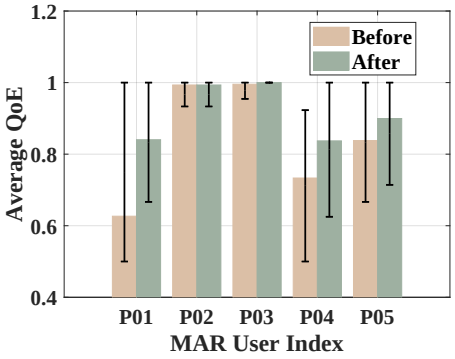

Figure C3 The change of user QoE after service provisioning.

Appendix C.1.3 *Performance of QoE-aware service provisioning*

To evaluate the effectiveness of the proposed semi-distributed service provisioning mechanism, we examine a test scenario involving five MAR users. As shown in Fig. C3, we compare the average QoE achieved by each user under two different resource allocation strategies. The dark brown and dark green bars represent the average QoE results obtained before and after the service provisioning adjustment, respectively. Through its tool-use capability, the MAR-SA detects that Users P01, P04, and P05 are prone to considerable QoE degradation when bandwidth is distributed uniformly. To mitigate this issue, MAR-SA instructs the network controller to reallocate part of the redundant bandwidth from Users P02 and P03, whose QoE levels are already sufficient, to those experiencing degradation. As a result, the enhanced performance, illustrated by the dark green bars, demonstrates a marked QoE improvement for Users P01, P04, and P05, while Users P02 and P03 maintain consistently high satisfaction levels.

Appendix C.2 Distributed collaborative mechanism

Appendix C.2.1 *Multi-user coordination mechanism*

As the number of MAR users grows, the LLM core faces a surge in concurrently processing UDT requests, which may result in deviations or even errors in the calculated user QoE. Furthermore, the dynamically changing user population poses additional challenges to the adaptability of the LLM core in assisting the network controller with accurate and consistent decision-making. To enable scalable and consistent decision-making under massive concurrent UDT requests, we develop an event-driven asynchronous approach that decouples the concurrent UDT request handling from the QoE prediction performed by the MAR-SA.

Specifically, for each incoming UDT request, the MAR-SA spawns a dedicated processing thread and assigns a unique session ID that encapsulates the user ID, timestamp, and related metadata. After completing the computation of QoE, the MAR-SA stores the results in a distributed in-memory database (e.g., Redis) for temporary caching. When the network controller initiates a service provisioning decision, the MAR-SA retrieves the corresponding data of all users associated with that decision cycle to provide the predicted QoE values required for multi-user coordination. Once the decision-making process is completed, the network controller sends a signal to the MAR-SA to release the cached data and archive the relevant results into the UCR for long-term storage and future reference.

Appendix C.2.2 *Computational complexity and scalability analysis*

The computational complexity of the MAR-SA framework is primarily determined by the resource allocation loop of the network controller. In the allocation algorithm, the system divides the total available bandwidth B^{total} into Z discrete allocation units, where $Z \approx B^{total}/\Delta b$. Resource provisioning is performed by incrementally allocating bandwidth units. In the worst-case scenario, the entire allocation loop requires Z iterations. Within each iteration, the network controller iterates over all U currently active MAR users to calculate and compare the marginal utility brought by allocating an additional bandwidth unit, subsequently selecting the user with the

maximum utility gain to execute the allocation. Therefore, the time complexity of this centralized decision-making process is strictly bounded by $\mathcal{O}(Z \cdot U)$.

The space complexity of the system mainly depends on the caching requirements for user states and QoE results. In the MAR-SA architecture, the system introduces a distributed in-memory database (e.g., Redis) as a temporary cache, utilizing unique Session IDs to isolate and store the calculated QoE results and states of concurrent users. Since the storage requirement scales linearly with the number of currently active users, the space complexity of the system is $\mathcal{O}(U)$.

Regarding system scalability, MAR-SA demonstrates exceptional horizontal scalability in application scenarios facing surging MAR user densities. By achieving an architectural decoupling between the processing of concurrent UDT (User Digital Twin) requests and the decision-making process of the core controller, MAR-SA can dynamically allocate independent processing threads for massive concurrent requests. While performing the lightweight $\mathcal{O}(Z \cdot U)$ scheduling, the network controller, benefiting from the underlying asynchronous concurrent architecture, does not need to synchronously wait for the inference responses from individual Agents. This mechanism effectively prevents system latency caused by single-point blocking, fundamentally ensuring the smooth scaling and efficient operation of the framework as the user scale expands.

Appendix C.3 Mitigating latency bottlenecks via decoupled reasoning and execution

To address a core contradiction in highly dynamic MAR scenarios—namely, the massive gap between the inherent inference latency of LLMs (typically on the order of seconds) and the coherence time of user mobility in MAR applications (milliseconds)—and to thoroughly overcome this fatal bottleneck in practical deployment, the MAR-SA framework introduces a decoupled reasoning and execution mechanism at the system design level. This mechanism aims to completely separate the high-latency semantic understanding process from the low-latency real-time control loop:

Asynchronous Semantic Reasoning and Pipeline Orchestration: The LLM core absolutely does not participate in the frame-by-frame real-time QoE computation. Instead, the LLM is awakened only under the following low-frequency triggering conditions: system initialization, the first-time access of a new user, or a fundamental change in network management intent. During this stage, its sole responsibility is to perform Intent Recognition, precisely match the corresponding toolset in the SFT, and ultimately output a strictly formatted, machine-readable instruction specification (JSON Schema). Although this semantic reasoning and planning process may take several seconds, it is executed asynchronously and does not block current real-time operations.

High-Frequency Tool Execution and Real-Time Closed Loop: Once the LLM determines the tool invocation logic for the target task, this execution path is hardened by the system into an Automated Workflow. In subsequent high-frequency real-time resource allocation loops, the system directly triggers and executes the underlying deterministic algorithmic tools encapsulated within the SFT. Through this decoupled design, the extremely time-consuming overhead of LLM semantic generation is completely stripped away from the real-time control loop. The LLM is only responsible for generating the invocation scripts, leaving the deterministic underlying APIs to handle the millisecond-level physical world changes, thereby perfectly ensuring the absolute real-time nature of service provisioning.

Appendix C.4 Performance under multi-user scenarios

While we illustrated a 5-user scenario in Appendix C.1.3 as a micro-level case study to clearly visualize the dynamic bandwidth reallocation process among specific individuals, it is imperative to evaluate the macro-level scalability of the proposed framework. To this end, we have conducted extensive trace-driven simulations with expanded user scales of 10, 20, and up to 40 MAR users.

In our semi-distributed architecture, the effectiveness of resource provisioning intrinsically relies on the precision of MAR-SA in predicting user QoE. During the expanded simulations, the network controller iteratively allocates these discrete, uniform bandwidth units. Because MAR-SA accurately predicts the marginal QoE utility (i.e., the specific QoE variance before and after allocating a specific bandwidth unit), the controller can consistently and precisely distribute resources to the users who need them most.

To quantitatively demonstrate this scalability, we evaluated the accuracy of the MAR-SA architecture as the concurrent user scale increases. As summarized in Table C2, the average QoE prediction accuracy of the system maintains a highly stable performance even when the user population multiplies.

Table C2 Average prediction accuracy of MAR-SA under different user scales

Number of Users	**5 Users**	**10 Users**	**20 Users**	**40 Users**
Average Accuracy	86.38%	87.92%	87.72%	89.52%

This robust prediction accuracy fundamentally proves that the proposed agent-driven approach is highly scalable. Even when the user population scales up by a factor of 8 and the bandwidth allocation decision space expands exponentially, MAR-SA effectively avoids performance degradation. By maintaining highly stable inference capabilities, the framework successfully guarantees optimal and personalized resource scheduling in densely populated 6G MAR networks.

Appendix C.5 Detailed comparison with the conventional QoE-driven resource allocation method

To comprehensively and meticulously evaluate the superiority of the proposed framework in practical network resource allocation tasks, we conduct a multi-dimensional and in-depth comparison between the MAR-SA dynamic bandwidth allocation strategy and the conventional QoE-driven bandwidth allocation strategy in this section. Since the resource demands of MAR applications are reflected not only in the system-average throughput but also in the fairness of individual experiences and instantaneous temporal continuity, we carry out detailed quantitative analyses from four aspects: micro-level dynamic bandwidth reallocation, macro-level system scalability, individual compliance rate, and micro-level temporal stability.

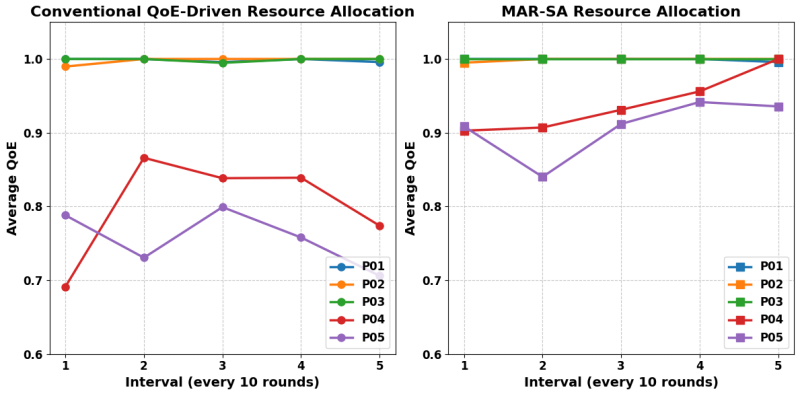

Figure C4 Dynamic user QoE evolution under different resource allocation strategies (adjustment interval = 10 steps).

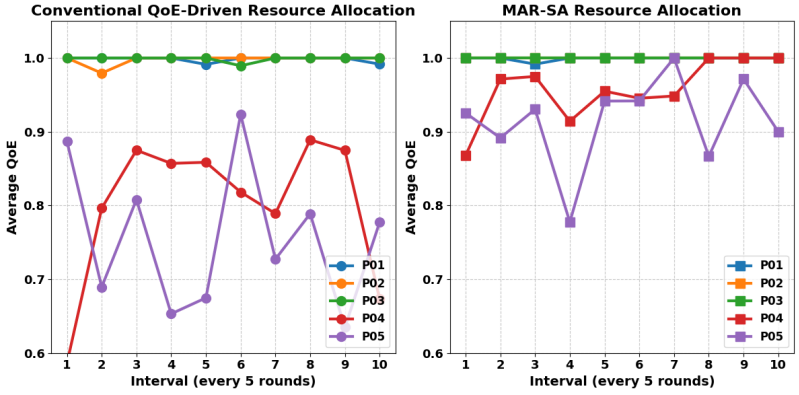

Figure C5 Dynamic user QoE evolution under different resource allocation strategies (adjustment interval = 5 steps).

Appendix C.5.1 *Micro-level dynamic bandwidth reallocation process analysis*

To intuitively demonstrate the specific scheduling behaviors of the two strategies when facing the heterogeneous dynamic demands of users, we extract 5 concurrent users (P01-P05) to form a micro-level test scenario under a constrained total system bandwidth. We set two decision frequencies: executing bandwidth reallocation every 10 timestamps and every 5 timestamps, respectively.

As shown in Fig. C4 and Fig. C5, under the conventional QoE-driven bandwidth allocation strategy, the system lacks application-side feedback and assumes that users with identical network conditions share the same QoE demands, thereby executing static or quasi-static equal bandwidth allocation. This blindness leads to severe resource mismatch: users with highly dynamic movement characteristics (e.g., P04 and P05) experience a continuous decline in their virtual content hit rate (VCHR) because the allocated bandwidth cannot meet their real-time keyframe uploading requirements. Specifically, in Fig. C5, the average QoE of P04 even plummets to around 0.6 at one point, and P05 fluctuates between 0.65 and 0.70 multiple times, resulting in a severe loss of immersion. Meanwhile, the QoE of relatively stationary users (e.g., P02 and P03) constantly remains close to 1.0, indicating that they occupy substantial redundant bandwidth without generating additional experience gains.

Conversely, the MAR-SA dynamic bandwidth allocation strategy accurately captures the marginal QoE utility of each user through the Tool-use mechanism. Within each adjustment cycle, MAR-SA astutely identifies the experience degradation of P04 and P05, and proactively releases a portion of the redundant bandwidth from P02 and P03 to transfer it to the former. From the trajectory changes, it is evident that although the QoE of P02 and P03 experiences a slight drop (while still maintaining an extremely high level above 0.95), the QoE of P04 and P05 is robustly elevated to above 0.85. Through such fine-grained application-side orchestration for dynamic supply-demand matching, MAR-SA successfully raises the lower bound of experience for disadvantaged users and greatly enhances the marginal utility conversion rate of overall resources.

Appendix C.5.2 *Macro-level system scalability*

To further verify the robustness of this mechanism when coping with user scale expansion, we extend the simulation scenario to compare the overall average QoE performance when the number of concurrent users in the system is $N \in \{2, 5, 10, 20, 40\}$.

As shown in Fig. C6, the system-average QoE under the conventional QoE-driven bandwidth allocation strategy consistently hovers between 0.910 and 0.931, failing to achieve any gain as the number of concurrent users increases. This is because its rigid allocation logic cannot exploit the spatiotemporal diversity of movement states among different users. In contrast, the MAR-SA dynamic bandwidth allocation strategy demonstrates significant multi-user diversity gains. As the number of concurrent users increases, the combinatorial space of heterogeneous demands within the system expands, enabling MAR-SA to more effortlessly schedule and match resources among a massive number of users. Consequently, its system-average QoE consistently stabilizes at an extremely high level. These data fully

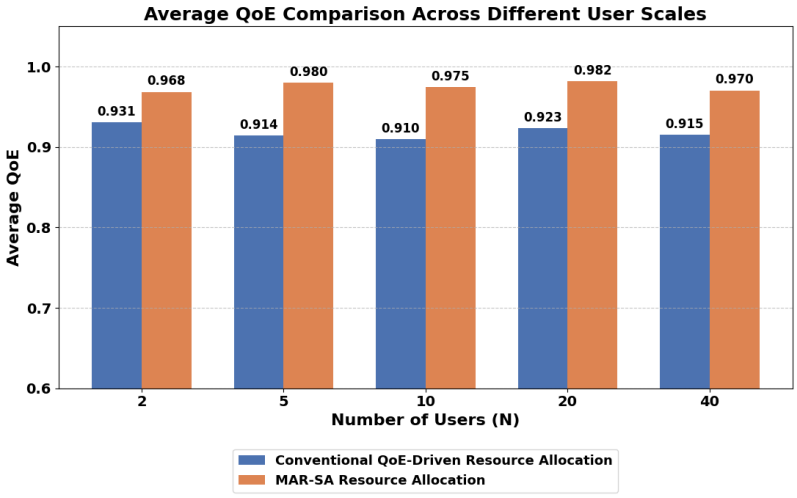

Figure C6 Comparison of average system QoE across different user scales.

confirm the large-scale scalability of MAR-SA in future 6G high-density concurrent scenarios.

Table C3 Number of non-compliant users under different global average QoE thresholds across various user scales

Threshold	Strategy	Number of Users (N)				
		2	**5**	**10**	**20**	**40**
0.85	Conventional	0	1	2	3	6
	MAR-SA	0	0	0	0	2
0.90	Conventional	0	1	3	6	12
	MAR-SA	0	0	0	0	2
0.95	Conventional	2	4	7	12	20
	MAR-SA	0	0	1	1	3

Appendix C.5.3 *Cross-scale individual compliance rate, mitigating the long-tail effect*

In resource-constrained networks, a mere system-average QoE often causes a performance masking phenomenon, where a minority of extremely degraded long-tail users are offset by other high-experience users. As shown in Table C3, to comprehensively measure the fairness of resource allocation, we establish three stringent thresholds for individual global average QoE compliance (0.95, 0.90, 0.85) and count the number of non-compliant users under different user scales.

The statistical results reveal severe flaws in the conventional method: when the user scale reaches $N = 40$, under the conventional QoE-driven bandwidth allocation strategy, up to 20 users fail to achieve an overall average QoE of 0.95. More severely, 12 users cannot even reach the basic threshold of 0.90, and 6 users fall below 0.85. This indicates that the equal allocation strategy sacrifices the fundamental experience of a large number of highly dynamic users. In comparison, the MAR-SA dynamic bandwidth allocation strategy achieves near-perfect bottom-line guarantees relying on precise resource tilting: in the $N = 40$ scenario, the number of severely disadvantaged users falling below the 0.90 threshold is drastically reduced from 12 to only 2, and even the number of non-compliant users under the highest standard (0.95) is forcefully suppressed to 3. MAR-SA fundamentally narrows the experience variance among users, effectively mitigating the system's long-tail effect.

Appendix C.5.4 *Micro-Level temporal continuity guarantee: ensuring frame-level experience stability*

Another core challenge of MAR applications lies in their extreme sensitivity to short-term fluctuations. Even if a user's global average QoE complies with the standard, if instantaneous rendering deviations (i.e., sudden QoE drops at certain timestamps) occur frequently during their experience, their subjective immersion will still be severely disrupted, potentially even triggering cyber-sickness. To this end, as shown in Table C4, twe further compile the cumulative frequency distribution of single-step QoE falling below 0.85 within 50 continuous micro-level decision steps.

The data demonstrate that the conventional QoE-driven bandwidth allocation strategy is extremely fragile when handling sudden posture changes. In the highly loaded $N = 40$ scenario, 19 users experience instantaneous degradation for more than 5 steps, 15 users suffer degradation for more than 10 steps, and 11 users even encounter extreme freezing lasting for more than 15 steps. Relying on the high-frequency closed-loop feedback of LLM Tool-use, the MAR-SA dynamic bandwidth allocation strategy can achieve rapid response and compensation: under the same scale, the number of users suffering degradation for more than 10 steps plummets by 80% (leaving

Table C4 Number of users experiencing frequent severe QoE degradation (QoE < 0.85) within 50 Decision Steps

Degradation Frequency	**Strategy**	**Number of Users (N)**				
		2	**5**	**10**	**20**	**40**
> 5 steps	Conventional	2	4	8	11	19
	MAR-SA	0	0	1	2	3
> 10 steps	Conventional	0	2	5	8	15
	MAR-SA	0	0	1	1	3
> 15 steps	Conventional	0	1	3	6	11
	MAR-SA	0	0	0	0	2

only 3 users), and extreme cases of degradation for more than 15 steps are reduced to just 2 instances. This irrefutably proves that MAR-SA not only optimizes resource allocation at the macro dimension but also endows each user with a highly coherent immersive experience at the micro-level temporal scale of frames by deeply coupling with the actual application-side logic.

References

1 Campos C, Elvira R, Rodríguez J J G, et al. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. IEEE Trans Robot, 2021, 37: 1874–1890
2 Chen Y, Inaltekin H, Gorlatova M. AdaptSLAM: Edge-assisted adaptive SLAM with resource constraints via uncertainty minimization. In: Proceedings of Proc. IEEE INFOCOM, 2023, New York, NY, USA
3 Han B, Liu Y, Qian F. Vivo: Visibility-aware mobile volumetric video streaming. In: Proceedings of Proc. ACM MobiCom, 2020, New York, NY, USA
4 Liu Z, Li Q, Chen X, et al. Point cloud video streaming: Challenges and solutions. IEEE Network, 2021, 35: 202–209
5 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in neural information processing systems, 2017, 30
6 Wu H, Hu T, Liu Y, et al. Timesnet: Temporal 2d-variation modeling for general time series analysis. In: Proceedings of ICLR, 2022
7 Nie Y, Nguyen N H, Sinthong P, et al. A time series is worth 64 words: Long-term forecasting with transformers. In: Proceedings of ICLR, 2023
8 Yi K, Zhang Q, Fan W, et al. Frequency-domain mlps are more effective learners in time series forecasting. In: Proceedings of NeurIPS, 2024
9 Liu Y, Hu T, Zhang H, et al. itransformer: Inverted transformers are effective for time series forecasting. In: Proceedings of ICLR, 2024