LONGBUSBENCH: BENCHMARKING LLM AGENTS FOR LONG-HORIZON BUS CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents are increasingly capable of long-horizon control, yet their ability to operate continuously in real-world dynamic systems remains largely unexplored. Public transit provides a challenging setting in which control actions have delayed and interacting consequences: for example, bus holding changes vehicle headways and passenger queues, thereby altering the conditions encountered by subsequent vehicles and the future validity of past experience. We introduce LONGBUSBENCH, a data-grounded benchmark for evaluating LLM agents in continual and asynchronous bus control under evolving operating conditions. We further propose MEMBUS, a memory-augmented LLM agent that structures experience into episodic and procedural memories and selectively retrieves, consolidates, revises, and forgets information towards efficient transit control. Experiments across various operating protocols compare MEMBUS with memory-free LLM agents, alternative memory designs, and classical control baselines. Results show that operational memory can improve public transit efficiency, while diagnostic analyses reveal that its effectiveness depends critically on the relevance and validity of accumulated experience. These findings establish long-horizon public-transit control as a practical and promising testbed for studying continual LLM agents in persistent real-world operational systems.

1 INTRODUCTION

Recent years have witnessed remarkable advances in artificial intelligence (AI), driven by the emergence of large foundation models that enable increasingly general-purpose capabilities across diverse domains (Bommasani et al., 2021; Wei et al., 2022; OpenAI, 2023). Among these developments, large language model (LLM) agents have emerged as a promising paradigm by augmenting LLMs with capabilities for reasoning, memory, and interaction with external environments (Yao et al., 2023; Wang et al., 2023; Liu et al., 2024). While LLM agents have demonstrated impressive performance in digital environments such as web navigation, software engineering, and knowledge-intensive tasks (Yao et al., 2023; Zhou et al., 2024; Liu et al., 2024), their capability in real-world operational systems that directly affect large populations remains largely unexplored.

Public transportation provides an important yet challenging testbed for evaluating the above capabilities. Public-transit services operate continuously under changing passenger demand, traffic conditions and operational issues (Wang & Sun, 2020; 2021). Unlike conventional episodic decision-making problems with clearly defined objectives and environment resets (Sutton & Barto, 2018), bus operations form a persistent socio-technical system where the consequences of an intervention may propagate over extended periods. Bus bunching exemplifies this challenge: a small delay can increase passenger accumulation and dwell time, further delaying the affected bus while allowing following buses to approach with fewer passengers. Such local disturbances may gradually evolve into fleet-level instability. Moreover, control actions reshape future operating conditions by modifying vehicle headways, passenger distributions, and subsequent decision contexts encountered by other buses. Therefore, effective bus-control agents must reason over delayed consequences, coordinate across multiple interacting entities, and continuously adapt to an evolving operational environment. As a result, these characteristics fundamentally distinguish public-transit control from existing long-horizon agent tasks. Particularly, current LLM-agent benchmarks mainly evaluate interactive tasks in predefined environments (Liu et al., 2024), or assess long-term memory over conversational histories (Wu et al., 2025). However, public-transit agents face a different form of long-term intelligence:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

they must maintain operational experience generated through their own previous interventions, identify when historical knowledge remains applicable under changing conditions, and preserve rare but critical experiences without allowing outdated information to degrade routine decisions.

Despite the importance of this capability, existing bus-control studies primarily optimize specialized reinforcement learning policies under predefined simulation episodes and relatively stationary operating distributions (Wang & Sun, 2020; 2021; Tang et al., 2024). Meanwhile, existing LLM-agent benchmarks do not evaluate downstream impacts in continuous physical systems, such as passenger waiting time, service regularity, and fleet stability. Consequently, it remains unclear whether current LLM agents can effectively acquire, manage, and utilize long-term operational experience to control dynamic public transportation systems. We address this gap by providing a benchmark for evaluating LLM agents in long-horizon bus control under continuously evolving service conditions.

To address these gaps, we first introduce a benchmark for long-horizon LLM-agent control of public-transit services, with bus-bunching mitigation as its central task. The benchmark is designed to evaluate not only service-level control performance, but also whether an agent can retain useful operational experience, adapt to changing operating regimes, recover knowledge when earlier regimes recur, and avoid harmful reuse of stale or superficially similar memories. We further introduce MEMBUS, a memory-augmented LLM-agent framework that organizes operational experience into structured episodic and procedural memories and is designed to support selective retrieval, consolidation, revision, and forgetting according to estimated relevance, regime-dependent validity, and downstream control utility. Our main contributions are summarized as follows:

- **A new real-world operational benchmark for continual LLM agent control.** We formulate long-term public transit management as a continual and asynchronous agentic control problem, where decisions have delayed consequences, affect future operating states, and must be made under evolving service regimes.
- **A data-grounded testbed for long-horizon transit agents.** We introduce LONGBUSBENCH, a testbed built upon real-world transit data that evaluates LLM agents in a continuously evolving operational environment.
- **MEMBUS: a memory-augmented bus control agent.** We propose MEMBus, a memory framework tailored for real-world operational control. Unlike generic retrieval-based agent memory, MEMBus explicitly models memory relevance, regime-dependent validity, and intervention utility to support continual operational decision-making.
- **Comprehensive empirical analysis.** We benchmark representative LLM agents, memory strategies, and classical control baselines under diverse operating conditions. Beyond performance comparison, we conduct diagnostic studies to reveal when historical experience improves decision-making and when outdated memories become detrimental.

2 RELATED WORK

LLM Agents for Traffic Control Recent studies have explored LLM agents for transportation decision-making. TrafficGPT connects natural-language queries with traffic-analysis tools for decision support (Zhang et al., 2023). LLMLight uses traffic observations to guide signal-phase selection with an LLM (Lai et al., 2025). CoLLMLight further investigates cooperative reasoning among multiple intersections with asynchronous LLM-based controllers (Yuan et al., 2026). Specifically, public transit provides a challenging yet underexplored domain for LLM-based control agents. Recent studies have explored LLMs for transportation control, including traffic signal control with LLM-based decision making (Lai et al., 2025) and cooperative multi-intersection control with multiple LLM agents (Yuan et al., 2026). For public transit, LLMs are investigated as auxiliary components for improving bus control, such as designing reward functions for reinforcement learning controllers (Yu et al., 2025). However, existing approaches either focus on short-horizon control or use LLMs as supporting modules rather than autonomous agents. LongBusBench addresses this gap by evaluating whether LLM agents can perform long-horizon bus control through continuous interaction, experience reuse, and adaptation to evolving service conditions.

Long-Horizon LLM Agents Long-horizon LLM agents increasingly rely on externalized memory to maintain information beyond a single interaction. ReAct interleaves reasoning, actions, and

environmental observations to enable iterative decision making (Yao et al., 2023), while Reflexion stores textual reflections from previous attempts to improve future behavior (Shinn et al., 2023). Generative Agents extract useful memory from accumulated experiences (Park et al., 2023), and MemGPT manages information across memory tiers under limited context windows (Packer et al., 2023). Voyager further demonstrates lifelong skill acquisition by storing reusable skills during embodied exploration (Wang et al., 2023). Recent long-horizon LLM agent benchmarks evaluate whether agents can effectively utilize such long-term information. AgentBench assesses LLM agents across diverse interactive environments (Liu et al., 2024), while LongMemEval evaluates memory retrieval, temporal reasoning, and knowledge updates over extended conversation histories (Wu et al., 2025). LifelongAgentBench studies learning across interdependent tasks (Zheng et al., 2025), and GAIA2 considers asynchronous environments with temporal constraints (Froger et al., 2026). However, these benchmarks primarily focus on digital tasks or knowledge-based interactions, where the consequences of agent actions are often abstracted from physical system dynamics. LongBusBench extends long-horizon agent evaluation to a new real-world operational problem.

3 PRELIMINARY

Consider a bus fleet $\mathcal{B}$ operating over a service horizon $[0, T]$. The state s_t includes vehicle positions and passenger loads, stop queues, traffic conditions, operating resources, and the execution status of pending interventions. Controller i receives observations authorized by the protocol and issues action proposals at decision times $\mathcal{T}_i$. These times need not coincide across controllers. Let r_j^i denote an observation, received message, prior action, or execution-feedback record that becomes available to controller i at time $t_j^{i,\mathrm{avail}}$. Its available history is $\mathcal{H}_t^i = \left(r_j^i\right)_{j:\, t_j^{i,\mathrm{avail}} \le t}$, with records ordered by availability time. Delayed feedback enters this history when it becomes available. The history $\mathcal{H}_t^i$ defines the information available for decision-making; it does not require the controller to include every record in each model input. The retained memory $\mathcal{M}_t^i$ contains the experience maintained under the protocol's initialization and update rules. Given its latest available observation o_t^i, history $\mathcal{H}_t^i$, and retained memory $\mathcal{M}_t^i$, controller i proposes

$$a_t^{i,\mathrm{prop}} = \pi_i\left(o_t^i, \mathcal{H}_t^i, \mathcal{M}_t^i\right), \qquad t \in \mathcal{T}_i. \tag{1}$$

Here π_i includes the controller's selection and use of available information. Controllers may differ in their observation access, retained experience, and decision schedules. A proposed holding command may be delayed, clipped, modified, or rejected by operational constraints before execution. We therefore distinguish proposed actions from the joint fleet interventions that physically take effect, $\mathbf{a}_t^{\mathrm{exec}}$. The transition equation indexes consecutive simulation steps, whereas controller i only acts at times in $\mathcal{T}_i$. Under operating regime z_t and exogenous variation ξ_t, the state evolves as:

$$s_{t+1} \sim P(\cdot \mid s_t, \mathbf{a}_t^{\mathrm{exec}}, z_t, \xi_t). \tag{2}$$

The regime z_t captures demand, traffic, and capacity conditions, and is observable only when explicitly exposed by the protocol. Holding alters departure times, subsequent headways, passenger accumulation, and therefore the future observations experienced by the fleet. As a result, later service states and operational experience depend partly on earlier holding decisions. A regime change within a service trajectory does not reset either the service state or the memory retained under the protocol. The objective is to use holding interventions to maintain service regularity and passenger service under operational constraints. We report passenger waiting time, headway variability, and served fraction separately to expose the resulting trade-offs.

4 LONGBUSBENCH

LongBusBench combines real-route assets, simulated bus service, and a common agent interface. It supports evaluation of operational history, changing service conditions, and recurring disruptions.

4.1 DATA LANDSCAPE

The benchmark includes 95 bus routes from four major cities in China: Beijing, Shanghai, Shenzhen, and Chengdu (e.g., Four-city environments in Fig. 1). We use the trajectories and stop-level

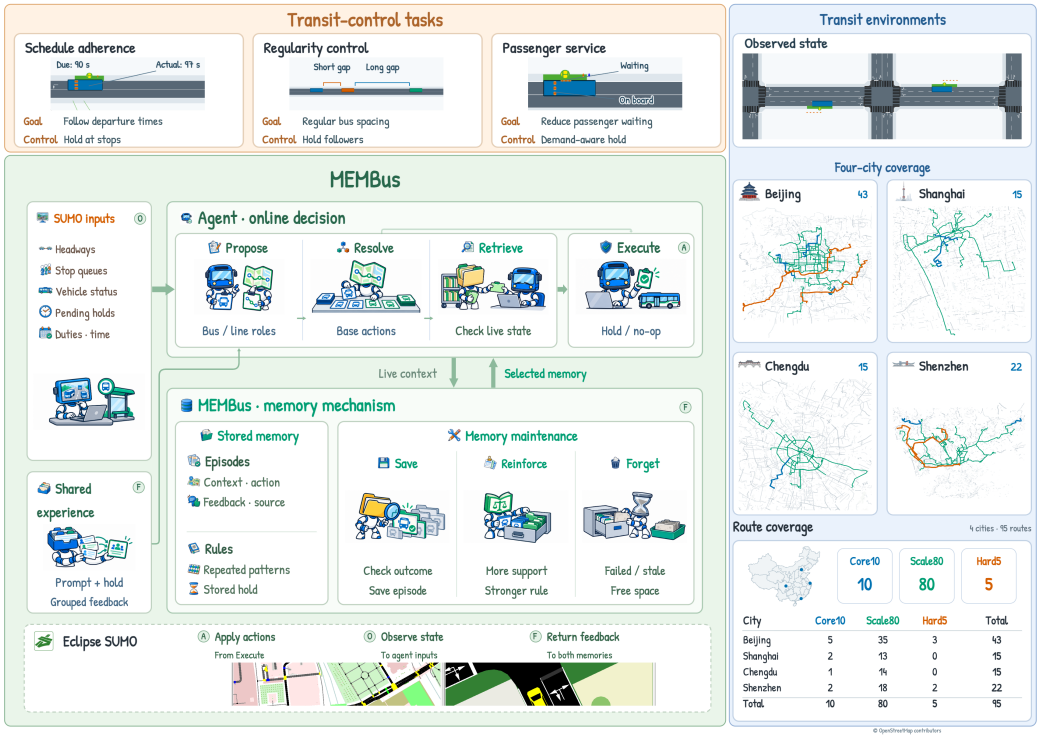


Figure 1: Overview of LongBusBench and the proposed MEMBus framework.

records from CPTOND (Wang et al., 2026) to construct stop sequences and inter-stop distances. To enable reliable simulation, we ground each route on real road networks, resulting in 92 strictly validated network bindings and three reconstructed bindings. Detailed binding criteria and provenance are provided in Appendix A. We derive stop-level demand weights and conditional destination distributions from TransitLM route-planning records (Guo et al., 2026) to construct passenger-demand scenarios. Other operational parameters, including arrival rates, temporal demand profiles, fleet sizes, target headways, and traffic conditions, are specified separately in route and scenario configurations. Table 1 summarizes the route suites.

Table 1: Route suites and planning-demand evidence

Suite	Size	Horizon	Stops	Mapped planning legs per stop
Core10	10 routes	16 h	31	42.44
Scale80	80 routes	16 h	34	61.40
Hard5	5 routes	16 h	67	41.79

4.2 Environment Composition

Long-horizon bus control requires an interactive and reliable environment that captures both traffic evolution and passenger dynamics. The route collection defines the set of benchmark environments. We use SUMO (Krajzewicz et al., 2012) to simulate traffic dynamics on OSM-derived road networks and extend it with a transit service layer that models passenger dynamics and vehicle states.

Service dynamics. Passenger demand is generated from the declared base rate, stop weights, and temporal profile. Depending on the service model, arrivals are accumulated as fractional mass or sampled as seeded Poisson counts. Passengers who cannot board remain in the queue. Boarding and alighting affect dwell time, while holding changes later departures, headways, and queues.

Destination-based configurations use the declared destination distribution; other configurations use the specified fractional alighting rule. Detailed transition rules are provided in Appendix B.1.

Agent interface. Agents interact with the environment through a unified API that provides structured observations and receives control actions. Observations include the current decision context, controllable vehicles, and available operational information, while actions correspond to feasible holding decisions. We support both event-driven control based on vehicle stop arrivals and periodic fleet-level control based on system snapshots, enabling evaluation under different decision granularities. Execution details are provided in Appendix B.2.

Benchmark scenarios and evaluation. Each benchmark instance specifies a route, fleet configuration, demand process, disruption schedule, observation setting, and random seed. Additional history and recurrence protocols evaluate agents' ability to utilize prior experience under evolving conditions. Performance is measured using service-level metrics, including passenger waiting time, headway regularity, and served demand, under standardized evaluation windows.

5 MEMBUS

MEMBus uses historical interventions and corresponding outcomes to guide LLM-based bus control, as illustrated in Fig. 1. It compiles related records into procedural cards and allows controllers to inspect the original evidence during decisions. A centralized LLM controller uses this memory to generate holding plans for the fleet.

5.1 MEMORY REPRESENTATION

An episodic record is represented by $e_k = (c_k, a_k^{\text{exec}}, [t_k, t_k + \Delta_k], \mathbf{y}_k, p_k)$. Here c_k is the pre-intervention context, a_k^{exec} the executed action, and $\mathbf{y}_k$ the service measurements over the feedback interval. The provenance p_k identifies the source, vehicles, stops, measurement protocol, and feedback availability. Paired records additionally specify the reference action and continuation policy. Then the procedural cards are compiled deterministically from records grouped by source conditions and comparison protocol. They summarize alternative actions and their outcomes, represent actions by vehicle roles, and retain links to supporting and adverse evidence. The original records remain in an evidence archive indexed by a source catalog. Compilation requires no additional LLM call.

5.2 RETRIEVAL AND EXECUTION

For each controller, the query c_t contains the current observation, recent history and execution feedback. Retrieval first removes records that are not yet available or whose cards are inactive. It then ranks the remaining cards by comparison-protocol match, agreement between the recorded and current operating conditions, and record recency. When a non-exact match is retained, the differing and unknown conditions are shown to the model. The LLM may request additional source records before generating a plan:

$$\begin{aligned} \mathcal{C}_t &= \text{Retrieve}(\mathcal{M}_t, c_t), \\ \mathbf{a}_t^{\text{prop}} &= f_\theta(c_t, \mathcal{C}_t, \mathcal{F}_t), \end{aligned} \tag{3}$$

where $\mathcal{C}_t$ contains the selected cards and $\mathcal{F}_t$ the additional source records.

Each controller assigns KEEP, SET, or CANCEL to its vehicles. These operations preserve, replace, or clear the pending holding duration, respectively. A controller may revise its own pending command before consumption, while ownership prevents another controller from overwriting it. The executor checks vehicle eligibility and holding bounds and tracks accepted commands through consumption and dwell. Changing a pending command does not undo dwell already served.

5.3 UTILITY AND MEMORY UPDATES

Candidate holding action a and reference a^0 are compared from the same traffic state under a common feedback horizon Δ, continuation policy π^+, and measurement protocol $\mathcal{P}$:

$$\Delta\mathbf{y}_{\mathcal{P}}(a, a^0 \mid c; \Delta, \pi^+) = \left(W_a - W_{a^0},\, C_a - C_{a^0},\, S_a - S_{a^0}\right)^\top. \tag{4}$$

Here, W, C, and S denote passenger waiting time, headway CV, and served fraction, respectively. These outcomes are kept separate so that service trade-offs remain visible, and each comparison retains the operating conditions under which it was observed. When memory updates are enabled, paired comparisons update the status of the corresponding actions in the order that feedback becomes available, while evidence already counted during initialization or from the same pre-intervention trajectory is not counted again. Comparisons are classified as supporting, adverse, or neutral according to fixed rules, whereas execution receipts and unpaired observations are retained as contextual feedback rather than intervention evidence. If a procedural card no longer has sufficient support, it may be suspended from active retrieval while its underlying records remain available for inspection; suspension does not invalidate the corresponding control action, and the card can be reactivated once new qualifying evidence is observed. Finally, deleting a procedural card removes only the compiled representation, while the raw records remain in the archive for possible reconstruction.

6 EXPERIMENTS

We use two evaluation horizons. The 1.5h-window experiments provide the paired memory comparison and the incremental ablations, while the extended experiments examine service over an 8h-window. Results from both horizons are presented together, with outcomes computed separately under their respective decision schedules and scoring windows. Detailed protocol records, source assignments, and execution audits are provided in Appendix C.

6.1 EXPERIMENTAL SETUP

Our MEMBus uses a centralized fleet controller based on `gpt-5.6-luna` with reasoning effort `none`. At each decision, the controller first selects source records that may be useful for the current decision and then produces a complete fleet action plan. The plan uses the KEEP, SET, and CANCEL operations defined by the execution interface. Requested holding durations and total physical dwell are bounded by 90 s; commands are also checked against vehicle eligibility and the pending-command execution rules. The control action uses the same model, observations, action permissions, and two-call decision procedure as MEMBus. Our framework exposes current vehicle states, pending commands, legal actions, and execution information,while in the current-observation control we remove the historical observations, procedural cards, source records, and queryable memory. This construction makes the comparison a test of access to the MEMBus memory configuration rather than a comparison between different LLMs or different action spaces.

1.5h-window protocol. The 1.5h-window comparison uses 50 matched route–seed cases. For each simulation, the control-entry time t_0 is determined before model evaluation, once all expected vehicles are active and every scheduled stop has recorded at least two arrivals. The LLM controls the fleet for 600 s (10 minutes), making decisions at 15 s intervals. A common numerical policy then continues service for 1200 s (20 minutes), respecting outstanding LLM command ownership. The scoring window is $[t_0, t_0+1800)$, covering 30 minutes. Including warmup, total simulated durations range from approximately 60 to 93.5 minutes.

8h-window protocol. The 8h-window comparison uses the same ten routes and five seeds per route.Control begins at the case-specific time t_0 after warmup. The LLM controls the fleet for 28,800 s (8 hours), making 96 decisions at 300 s (5-minute) intervals. Simulation continues between decisions without additional numerical control and ends 300 s after the final decision. The scoring window is $[t_0, t_0 + 28800)$, covering the full 8h-window control period. Including warmup, total simulated duration is $8 + t_0/3600$ hours. The 180 s recent-history window and memory lifecycle thresholds remain unchanged from the 1.5h-window configuration.

Metrics. For both horizons, our primary metric is fixed-cohort restricted mean waiting time. We additionally report the boarded fraction, the queue remaining at the horizon, and headway regularity measured by the coefficient of variation, $\mathrm{CV}_{\mathrm{headway}} = s_g/\bar{g}$, where s_g and $\bar{g}$ are the sample standard deviation and mean of pooled within-stop arrival intervals, respectively. Lower CV indicates more regular arrivals. Arrival intervals are computed between consecutive observed arrivals at the same stop and pooled over the scoring window.We also report token usage, command submissions,

and bunching-event counts, aggregated using the same route-weighted procedure.Command counts measure submissions rather than executed holding interventions.

Baselines. We compare MEMBus with fifteen baselines spanning three categories: LLM-based, learning-based, and traditional control methods. **LLM-based baselines** include *DiLu*, which retrieves prior examples without reflection (Wen et al., 2024); *ReAct*, which interleaves reasoning and tool use over the current bus state (Yao et al., 2023); *Recent-history*, which retains 180 s of observations, actions, and execution feedback but excludes external experience and procedural memory;*Self-Refine*, which performs one draft–feedback–refinement cycle per decision (Madaan et al., 2023); and a decision-level *Reflexion* adaptation that maintains a bounded buffer of verbal reflections (Shinn et al., 2023). **Learning-based baselines** include *DQN* (Mnih et al., 2015), *DDPG* (Lillicrap et al., 2016), *SAC* (Haarnoja et al., 2018), *MADDPG* (Lowe et al., 2017), and *CAAC*, an asynchronous multi-agent controller designed for bus operations (Wang & Sun, 2021). **Rule-based baselines** include *TwoWay*, *ForwardHeadway*, *BackwardHeadway*, and *Headway feedback*, which implement representative headway-control strategies from the bus-control literature (Daganzo, 2009; Daganzo & Pilachowski, 2011), together with *Zero additional holding*, which removes control-induced holding while preserving passenger-service dwell. All learned policies are frozen during evaluation; DDPG and SAC are trained on a fixed development transition dataset and transferred to the evaluation cases. All methods share the same execution interface while retaining their specified action constraints and decision schedules. Additional implementation details are provided in Appendix E.

6.2 Comparison with Baseline Controllers

We compare MEMBus with fifteen baselines under both protocols: five LLM-based, five rule-based, and five learning-based controllers, detailed in Appendix E.

Table 2: Controller comparisons. Each cell reports 1.5h-window / 8h-window results.

Controller	Wait (s)↓	CV↓	Served (%)↑	Tokens (M)↓	Commands	Bunching↓
MEMBus	609.53 / 2672.54	0.72706 / 0.89431	63.355 / 84.351	1.420 / 1.821	129.06 / 455.02	2.66 / 176.98
LLM-based controllers						
DiLu	607.67 / 2679.40	0.69723 / 0.85902	63.737 / 84.426	1.355 / 3.015	166.34 / 467.10	2.70 / 201.04
ReAct	609.97 / 2658.68	0.69767 / 0.92696	63.819 / 84.393	1.500 / 6.269	189.30 / 490.18	2.86 / 199.06
Recent-history	636.11 / 2769.28	0.84678 / 1.12626	61.638 / 83.821	1.230 / 1.507	227.32 / 536.86	5.08 / 270.12
Self-Refine	611.77 / 2676.39	0.75074 / 1.05526	63.581 / 85.096	1.978 / 4.167	224.98 / 536.96	2.74 / 242.02
Reflexion	613.59 / 2706.09	0.78339 / 1.09737	63.156 / 84.289	1.950 / 4.149	225.66 / 537.18	3.86 / 244.47
Rule-based controllers						
TwoWay	632.21 / 2818.84	0.53792 / 0.44621	62.363 / 83.276	—	227.40 / 534.22	0.84 / 67.63
ForwardHeadway	653.90 / 3080.71	0.54696 / 0.41490	61.334 / 81.534	—	227.76 / 534.63	1.76 / 71.06
BackwardHeadway	632.82 / 3037.33	0.60322 / 0.71627	62.279 / 81.835	—	227.62 / 535.88	1.14 / 134.27
Headway feedback	674.68 / 3345.36	0.48846 / 0.37687	60.961 / 79.553	—	228.22 / 535.96	0.70 / 38.35
Zero additional holding	599.20 / 2539.86	0.72903 / 1.00151	64.023 / 85.274	—	224.64 / 536.22	1.92 / 241.14
Learning-based controllers						
DQN	646.73 / 2938.67	0.65576 / 0.63762	61.183 / 82.858	—	227.50 / 536.69	3.22 / 151.53
DDPG	718.39 / 3918.69	0.71730 / 0.75632	57.508 / 75.936	—	229.58 / 543.33	0.68 / 92.20
SAC	718.13 / 3950.08	0.72567 / 0.81013	57.328 / 75.753	—	229.58 / 543.53	1.56 / 108.53
MADDPG	679.74 / 3484.13	0.69462 / 0.71238	59.377 / 78.870	—	228.42 / 539.00	1.82 / 118.20
CAAC	621.18 / 2802.01	0.63524 / 0.72797	62.561 / 83.477	—	226.54 / 536.55	1.58 / 154.98

Table 2 shows fewer bunching events for MEMBus than for the other LLM controllers. Compared with the five learned baselines and four headway rules in the table, MEMBus improves waiting and served fraction but has higher CV.Zero additional holding has the lowest waiting and highest served fraction among the listed controllers.

Weighted evaluation. To emphasize waiting, bunching, and sustained operation, we rank the six LLM controllers using

$$R_m = \sum_h \alpha_h \sum_k w_k r_{m,h,k}, \tag{5}$$

where $r_{m,h,k}$ is the within-window metric rank. Lower values receive better ranks except for served fraction, and ties receive average ranks. The metric and horizon weights are

$$(w_W, w_B, w_C, w_S, w_T) = (0.30, 0.30, 0.15, 0.15, 0.10),$$
$$(\alpha_{\text{short}}, \alpha_{8\text{h}}) = (0.30, 0.70), \tag{6}$$

where W, B, C, S, and T denote waiting, bunching, CV, served fraction, and token usage. These weights prioritize passenger delay and bunching mitigation during sustained service while retaining regularity, service coverage, and computational cost. Commands are excluded. Using the main-table values, MEMBus ranks first with a score of 2.075, followed by DiLu (2.460) and ReAct (2.565); lower scores are better.

6.3 Effect of Operational Memory

Full MEMBus has access to the fixed paired source records, procedural summaries, and experience acquired during its own trajectory. It can use these records when selecting actions and update the memory according to the lifecycle rules declared for the experiment. The current-observation control has none of these sources, while retaining the same model, current-state inputs, action permissions, and execution interface. Table 3 reports the paired comparison at both horizons.

Table 3: MEMBus and current-observation control under the two evaluation horizons

Controller / contrast	Wait (s)↓	CV↓	Served (%)↑
1.5h-window: 10 routes × 5 seeds			
Current-observation LLM	639.94	0.81323	61.333
MEMBus	609.53	0.72706	63.355
Paired difference	−30.41	−0.08617	+2.022
8h-window: 10 routes × 5 seeds			
Current-observation LLM	2744.97	1.09826	84.060
MEMBus	2672.54	0.89431	84.351
Paired difference	−72.43	−0.20395	+0.291

MEMBus improves mean waiting, headway regularity, and served fraction over the current-observation LLM in both windows. In the matched case shown in Figure 2, memory-enabled control produces more evenly spaced vehicle trajectories and less pronounced bunching, particularly over eight hours.

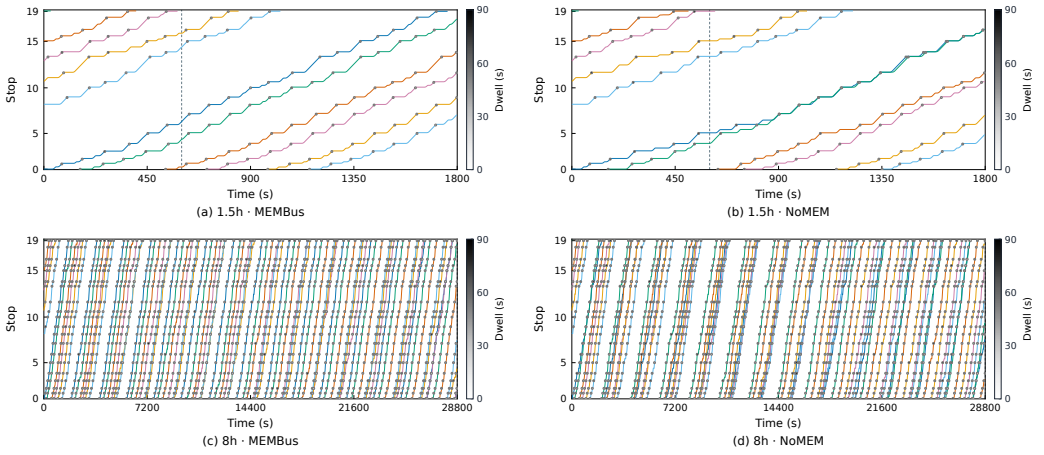


Figure 2: Bus trajectories on Shanghai Route 96 (seed 2026120181): MEMBus (left) and NoMEM (right), over 1,800 s (top) and 8h (bottom). Colors identify vehicles; grayscale indicates total dwell. Dashed lines mark the end of 1.5h-window LLM control.

6.4 INCREMENTAL MEMORY ABLATIONS

Ablation design We compare three nested memory configurations under the 1.5h-window protocol. All three conditions use the same initial procedures, paired evidence, action space, and continuation policy. Failures and KEEP fallbacks remain part of the evaluated trajectories. Online observational procedures and their archival and re-entry mechanisms remain enabled in all three configurations.

Fixed source procedures (B1) B1 initializes external-source procedures from the initial paired evidence and keeps them fixed.Newly released external records remain queryable but do not update these procedures.The initial source conditions remain in active presentation.

Source updates (B12) B12 adds external-source procedure updates from newly released paired evidence.The initial source conditions remain in active presentation.

Source-context exit (B123) B123 additionally removes the initial source conditions from active presentation after 300 s of control.The underlying records remain queryable in the archive.

Table 4: Nested MEMBus configurations over 50 matched route–seed cases.

Configuration	Wait (s)↓	CV↓	Served (%)↑
Fixed source procedures (B1)	610.91	0.73942	63.152
Source updates (B12)	610.59	0.71727	63.186
Source-context exit (B123)	609.53	0.72706	63.355

Increment definitions. The update contrast measures external-source procedure updates from newly released paired evidence.The context-exit contrast measures the additional effect of removing the initial source conditions from active presentation.

For an endpoint Y, we define

$$\Delta_{\text{update}} = Y_{\text{B12}} - Y_{\text{B1}}, \qquad \Delta_{\text{exit}} = Y_{\text{B123}} - Y_{\text{B12}}. \tag{7}$$

Ablation results. Source updates improve mean waiting, headway regularity, and served fraction relative to fixed source procedures. Source-context exit further lowers mean waiting and increases served fraction, with a small increase in headway CV. Full MEMBus therefore achieves the lowest mean waiting and highest served fraction, while B12 has the lowest CV.Relative to B1, the full configuration improves all three mean service metrics.

7 CONCLUSION AND LIMITATIONS

We formulate long-horizon bus control as a continual LLM-agent problem, where past actions reshape future operating conditions and the value of experience changes over time. To study this setting, we introduce LONGBUSBENCH, a data-grounded benchmark for continual and asynchronous bus control, together with MEMBUS, a task-specific memory framework that organizes operational experience by relevance, regime validity, and intervention utility. Experiments show that MEMBus is competitive in long-horizon control tasks with LLM agent. Broader routes, models, and real-world validation remain for future work.

AI USE STATEMENT

Generative AI tools were used to provide feedback on experimental design and methodology, assist with method and software implementation, inspect and summarize experimental outputs, and support result interpretation. They were also used to edit code, draft and revise manuscript text, organize related-work notes, format tables, and prepare figures. The authors selected the research questions and methods, executed the experiments, tested AI-assisted code, checked reported numerical values against retained experiment logs and summary files, and reviewed all AI-assisted manuscript content. Generative AI was not used to generate experimental data or replace experimental execution. The authors take responsibility for the final text, claims, code, and artifacts in this work.

REPRODUCIBILITY STATEMENT

Section 5 describes the MEMBus memory representation, retrieval process, action interface, and update rules. Section 6 specifies the experimental comparisons, control and measurement windows, and statistical analysis. Appendices F and B provide additional details on comparison controls and execution records. Detailed evaluation protocols, memory ablations, and baseline implementations are provided in Appendices C, D, and E.

Reproduction requires the corresponding code and asset versions, controller configuration, initial memory, random seeds, and metric definitions. Archived model responses support replay and inspection of individual decisions; new online model calls may produce different responses.

REFERENCES

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Stefanie von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. URL https://arxiv.org/abs/2108.07258.

Carlos F. Daganzo. A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons. *Transportation Research Part B: Methodological*, 43(10):913–921, 2009. doi: 10.1016/j.trb.2009.04.002.

Carlos F. Daganzo and Joshua Pilachowski. Reducing bunching with bus-to-bus cooperation. *Transportation Research Part B: Methodological*, 45(1):267–277, 2011. doi: 10.1016/j.trb.2010.06.005.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Moreno-Torres Bertran, Ulyana Piterbarg, Mikhail Plekhanov, Mathieu Rita, Andrey Rusakov, Vladislav Vorotilov, Mengjue Wang, Ian Yu, Amine Benhalloum, Grégoire Mialon, and Thomas Scialom. Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments. *arXiv preprint arXiv:2602.11964*, 2026. URL https://arxiv.org/abs/2602.11964.

Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/a8166da05c5a094f7dc03724b41886e5-Abstract.html.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

Hanyu Guo, Jiedong Yang, Chao Chen, Longfei Xu, Kaikui Liu, and Xiangxiang Chu. TransitLM: A large-scale dataset and benchmark for map-free transit route generation. *arXiv preprint arXiv:2605.22355*, 2026. URL https://arxiv.org/abs/2605.22355.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.06169.

Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO—simulation of urban mobility. *International Journal on Advances in Systems and Measurements*, 5(3–4):128–138, 2012. URL https://elib.dlr.de/80483/.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c2073ffa77b5357a498057413bb09d3a-Abstract.html.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

Siqi Lai, Zhao Xu, Weijia Zhang, Hao Liu, and Hui Xiong. LLMLight: Large language models as traffic signal control agents. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025. doi: 10.1145/3690624.3709379.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1509.02971.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.03688.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.02275.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.17651.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. doi: 10.1038/nature14236.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. URL https://arxiv.org/abs/2006.09359.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael 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. doi: 10.1145/3586183.3606763.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Yuhan Tang, Ao Qu, Xuan Jiang, Baichuan Mo, Shangqing Cao, Joseph Rodriguez, Haris Koutsopoulos, Cathy Wu, and Jinhua Zhao. Robust reinforcement learning strategies with evolving curriculum for efficient bus operations in smart cities. *Expert Systems with Applications*, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Jiawei Wang and Lijun Sun. Dynamic holding control to avoid bus bunching: A multi-agent deep reinforcement learning framework. *Transportation Research Part C: Emerging Technologies*, 116:102661, 2020. doi: 10.1016/j.trc.2020.102661.

Jiawei Wang and Lijun Sun. Reducing bus bunching with asynchronous multi-agent reinforcement learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 426–433, 2021. doi: 10.24963/ijcai.2021/60. URL https://www.ijcai.org/proceedings/2021/60.

Liang Wang, He Wei, Yu Guan, Libin Ouyang, DanDan Xu, Xuehua Han, Min Zhang, Meng Chen, Daosheng Sun, Daqing Gong, Zhenji Zhang, Xinghua Zhang, and Xiaodong Zhang. China public transport operation network dataset (CPTOND-2025): National-scale bus-metro vector dataset. *Scientific Data*, 13:188, 2026. doi: 10.1038/s41597-025-06505-4. URL https://www.nature.com/articles/s41597-025-06505-4.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S. Merel, Jost Tobias Springenberg, Scott E. Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/588cb956d6bbe67078f29f8de420a13d-Abstract.html.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. DiLu: A knowledge-driven approach to autonomous driving with large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16292.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10813.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

X. Yu et al. Large language model-enhanced reinforcement learning for generic bus holding control strategies. *Transportation Research Part E: Logistics and Transportation Review*, 2025.

Zirui Yuan, Siqi Lai, and Hao Liu. CoLLMLight: Cooperative large language model agents for network-wide traffic signal control. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KeJqoEVOeY.

Siyao Zhang, Daocheng Fu, Zhao Zhang, Bin Yu, and Pinlong Cai. TrafficGPT: Viewing, processing and interacting with traffic foundation models. *arXiv preprint arXiv:2309.06719*, 2023. URL https://arxiv.org/abs/2309.06719.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. LifelongAgentBench: Evaluating LLM agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025. URL https://arxiv.org/abs/2505.11942.

Shuyan Zhou, Frank Xu, Hao Zhu, et al. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

A Route and Demand Assets

Route suites. LongBusBench contains 95 routes organized into three disjoint suites: Core10, Scale80, and Hard5, containing 10, 80, and 5 routes, respectively. CPTOND provides the route and stop records used to construct stop sequences and inter-stop distances (Wang et al., 2026). The mean numbers of stops per route are 30.6, 33.9375, and 67.0; Table 1 reports these values rounded to integers. The listed 16-hour horizons describe the full-service benchmark protocols. The control and scoring windows of the MEMBus experiments are specified separately in Appendix C.

Road-binding categories. The road-network catalog contains 92 strictly validated executable bindings and three reconstructed bindings. Strict validation checks stop accessibility on bus-permitted lanes, stop-to-road snapping distances of at most 60 m, route-length deviation of at most 15%, successful SUMO traversal, and zero teleports. Distance-constrained map matching additionally checks that the 90th-percentile segment-relative-distance error does not exceed 50%. The reconstructed category permits snapping distances of up to 150 m and route-length deviation of up to 20%, subject to the corresponding matching and execution checks.

The three reconstructed routes are Shanghai 71 interval (Fenghong Road–Huangpi North Road), Chengdu K12 (Zhonghai International Station–Chengdu East Railway Station), and Shenzhen M359 (Guanlan Dashuitian Terminal–Xinghe Shengjing Bus Terminal). All belong to Scale80. Core10 and Hard5 contain only strict bindings, while Scale80 contains 77 strict and three reconstructed bindings. Reconstructed bindings are available for coverage and sensitivity analyses and are excluded from the strict primary road-network evaluation.

These bindings define executable paths on OSM-derived topology constrained by the source route records. They do not establish agency-published road-edge sequences or measured vehicle GPS trajectories.

Planning-demand evidence. TransitLM route-planning records provide mapped public-transit legs for constructing relative stop-demand weights and conditional destination distributions (Guo et al., 2026). Each mapped leg identifies a boarding origin and a downstream destination on a selected route. These records represent planning demand rather than observed passenger boardings.

Let $n_{r,i}$ denote the number of mapped forward planning legs originating at stop i on route r. The stored stop weight is

$$w_{r,i} = n_{r,i} + 1.$$

The unit smoothing term assigns positive weight to stops without mapped origins. Weights are normalized to have a within-route mean of one:

$$\widetilde{w}_{r,i} = \frac{w_{r,i}}{N_r^{-1} \sum_{j=1}^{N_r} w_{r,j}},$$

where N_r is the number of stops on route r. These weights specify relative demand across stops; absolute arrival rates and temporal profiles are configured separately.

Table 5 lists the mapped-leg counts before smoothing. The final column, also reported in Table 1, is the total number of mapped legs divided by the number of route–stop entries in each suite. A physical stop served by multiple routes contributes a separate entry for each route.

Conditional destination distributions are constructed from mapped downstream destinations for each origin, without OD smoothing. Origins without an available downstream distribution use the synthetic fallback specified by the selected service configuration.

Relation to the evaluated environments. The benchmark distinguishes OSM-derived road-network bindings from generated route environments. Generated environments preserve route stop order and inter-stop distances in a service path with a return connection. The road-binding validation results above apply to the OSM-derived catalog.

The research configurations specify their own route geometry, demand weights, and passenger-service settings. The demand construction described above characterizes the benchmark catalog and does not imply that every research configuration uses the same weights or destination distributions. Experiment-specific settings are reported in Appendices B.1 and C.

B Simulation and Control Interface

SUMO simulates vehicle movement (Krajzewicz et al., 2012), while a separate service layer maintains passenger queues, boarding, alighting, onboard loads, and execution feedback. Each experiment specifies its network construction, arrival process, passenger-service model, and control interface. The benchmark supports multiple backend configurations; their observation and execution semantics are distinguished below.

B.1 Passenger Service and Cross-Day Operation

Passenger arrivals. The demand model combines a base rate, normalized stop weights, a temporal profile, and disruption multipliers:

$$\lambda_{r,i}(t) = \lambda_{0,r}\, \widetilde{w}_{r,i}\, m_r(t)\, d_{r,i}(t).$$

Here $\lambda_{0,r}$ is measured in passengers per stop per second, $\widetilde{w}_{r,i}$ is the normalized stop weight defined in Appendix A, $m_r(t)$ is the temporal multiplier, and $d_{r,i}(t)$ is the disruption multiplier. An absent temporal profile or demand disruption contributes a multiplier of one.

The fractional-arrival model accumulates $\lambda_{r,i}(t)\Delta t$ passenger mass at one-second intervals and retains fractional remainders. The public fleet backend instead uses seeded Poisson integer arrivals with the corresponding expected rate. The selected arrival process is part of the experiment configuration; the two processes are not treated as interchangeable.

Configuration used in the MEMBus experiments. The reported MEMBus configuration uses the fractional-arrival model with a nominal base rate of 0.05 passengers per stop per second. Available stop weights are normalized within each route; reconstructed research instances use uniform weights. Alighting follows the configured fractional rule. These settings are distinct from the public fleet backend's Poisson arrival and directional-destination configuration.

The planning-demand construction in Appendix A describes the benchmark catalog. The research configurations use their assigned route weights and do not necessarily load the catalog's destination distributions. Route assignments, source-memory conditions, and evaluation windows are specified in Appendix C.

Table 5: Mapped planning-demand evidence before smoothing. Counts represent planning legs, not observed boardings.

Suite	Route–stop entries	Mapped legs	Legs per stop
Core10	306	12,988	42.44
Scale80	2,715	166,713	61.40
Hard5	335	14,001	41.79

Boarding, alighting, and dwell. Passengers alight before boarding. Under finite capacity, passengers who cannot board remain in the stop queue. FIFO arrival records track waiting times and boarded passenger mass. Alighting follows either destination assignments or the configured fractional rule. The directional destination model restricts destinations to downstream stops and uses its specified fallback when destination evidence is unavailable.

Natural dwell depends on boarding and alighting volumes, service times, crowding penalties, and dwell perturbations. At a normally served stop, ordinary pending-hold execution adds the control hold and any event-induced hold to natural dwell. The configured total-dwell limit then caps the resulting duration. Consequently, a requested hold need not produce an equal increment in realized dwell.

Boarding and alighting are updated at the handled arrival event. Passengers arriving later during the holding interval remain in the queue until a subsequent boarding event; the model does not assume continuous boarding throughout the hold.

Cross-day extension. The separate seven-day corridor protocol contains seven 16-hour service windows and seven 8-hour depot windows, spanning 604,800 s. Physical vehicle identity, block identity, cumulative trip counts, and battery state are carried across service days.

In the terminal-clearing service configuration, trips and passenger accounting are closed before the next dispatch. Road position, onboard passengers, and stop queues are not restored as the next day's initial road state. Daily simulator instances may restart while the specified operational state is carried forward.

The agent runtime supports retaining the same agent instance across days and delivering day-start and day-end events without an automatic agent reset. This cross-day capability is separate from the reported 8h-window MEMBus evaluation, which covers sustained service within a single run.

B.2 Agent Interface and Execution

Observations. The API exchanges structured observations and action batches rather than simulator-owned objects. Observations identify the episode, decision, simulation time, visible vehicles, eligible vehicles, and available action capabilities. Backend-specific fields provide vehicle positions, loads, queues, pending commands, dwell status, and execution feedback. Unavailable common fields are represented as `null`.

The road interface supplies decisions at vehicle stop events, whereas the fleet interface supplies periodic snapshots. Road headways are arrival-time gaps at shared stops; fleet control headways are derived from spatial separation. Evaluation headway CV is computed separately from recorded arrival intervals. Controllers receive only the fields authorized by their observation configuration.

Action validation. Let $\mathcal{B}_t^{\mathrm{cmd}}$ denote the vehicles receiving submitted holding commands and $\mathcal{B}_t^{\mathrm{elig}}$ the eligible set. For requested duration $h_{b,t}$ and permitted upper bound $\bar{h}_{b,t}$, valid holding commands satisfy

$$\begin{aligned} \mathcal{B}_t^{\mathrm{cmd}} &\subseteq \mathcal{B}_t^{\mathrm{elig}}, \\ 0 \le h_{b,t} \le \bar{h}_{b,t}, &\qquad b \in \mathcal{B}_t^{\mathrm{cmd}}. \end{aligned} \tag{8}$$

Public API batches must match the current decision identifier and API version and contain only permitted fields with finite, in-range values. Invalid batches are rejected before backend execution.

The public API and research controllers use their respective action contracts. The public API permits holding bounds of up to 30 s, whereas the reported MEMBus research configuration uses a 90 s command limit and a 90 s total physical dwell limit. Vehicle-specific eligibility and holding bounds still apply within these limits.

Pending commands. In the MEMBus plan interface, KEEP preserves a pending command, SET replaces its duration, and CANCEL clears the pending duration. Where exact-zero normalization is enabled, a SET command with duration zero is interpreted as CANCEL; otherwise, it is handled by the configured plan validator.

Under the research controller's ownership rules, cancellation retains ownership of the zero-duration command until consumption or replacement. The issuing controller may revise its pending command before consumption. Waiting for execution feedback does not by itself disable this permission.

For ordinary fleet holding, a pending command is consumed at the next handled stop. Omitting a vehicle from an accepted backend action batch preserves its pending command, whereas an explicit zero clears the pending duration. A controller's response schema may nevertheless require an explicit operation for every listed vehicle.

Road holding is scheduled at the current stop event. Separately enabled current-stop extensions and dwell reductions follow their own execution rules. No operation reverses dwell already served.

Let $\mathbf{q}_t$ denote accepted commands awaiting execution. The execution interface maps proposals to physical interventions:

$$(\mathbf{a}_t^{\text{exec}}, \mathbf{q}_{t+1}) = \mathcal{E}_t\left(s_t, \mathbf{q}_t, \mathbf{a}_t^{\text{prop}}\right). \tag{9}$$

Current-stop interventions need not pass through a next-stop command queue.

Execution records and timing. Execution records distinguish proposal validation, command acceptance, stop-level consumption, and observed departure. Command identifiers link subsequent feedback to the issuing decision. Acceptance alone does not establish additional physical dwell or a passenger-service benefit.

The evaluated simulation loop waits for model inference before advancing simulated time. Wall-clock inference latency is therefore accounted for as computational cost rather than simulated operating delay. Vehicle-event timing and controller-internal concurrency are distinct from advancing the simulation during inference.

C MEMBus Evaluation Protocols

This section documents the 1.5h-window and 8h-window memory comparisons reported in Section 6. Both use centralized fleet control over the same ten routes and five seeds per route. MEMBus and the current-observation control each contribute 50 completed episodes per protocol, yielding 50 matched pairs in each window. Case records associate each route and seed with its control-entry time, scoring boundaries, and result version.

C.1 Route and Seed Assignments

Table 6 lists the route–seed assignments shared by both protocols. Table 7 gives the corresponding control-entry times. The short-window endpoint is $t_0 + 1800$, and the 8h-window endpoint is $t_0 + 28800$.

Table 6: Route–seed assignments shared by both protocols

Route	Seeds
Beijing 16	2026120100–2026120104
Shanghai 17	2026120110–2026120114
Beijing 135	2026120120–2026120124
Shanghai 152	2026120130–2026120134
Shanghai 167	2026120140–2026120144
Chengdu 1	2026120150–2026120154
Shenzhen M401	2026120160–2026120164
Beijing Special 183	2026120170–2026120174
Shanghai 96	2026120180–2026120184
Shenzhen M425	2026120190–2026120194

Table 7: Control-entry times t_0 in seconds for both protocols. Columns follow the five seeds in ascending order within each range in Table 6.

Route	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5
Beijing 16	2220	2220	2220	2220	2220
Shanghai 17	1830	1830	1830	1830	1830
Beijing 135	3180	3180	3360	3180	3180
Shanghai 152	1800	1800	1800	1800	1800
Shanghai 167	2490	2490	2490	2490	2490
Chengdu 1	2010	2040	1800	2010	1800
Shenzhen M401	3810	3810	3810	3810	3810
Beijing Special 183	1950	1950	1950	1950	1950
Shanghai 96	1830	1830	1830	1830	1920
Shenzhen M425	1800	1920	1830	1920	1920

C.2 Control and Scoring Windows

Warmup and control entry. Before model evaluation, a numerical warmup determines the earliest permitted 30-second-aligned control-entry time t_0. The scan begins no earlier than 1800 s and the route-specific minimum warmup. All six expected vehicles must be active, and every scheduled stop must have at least two arrivals recorded before the control boundary. This requirement concerns arrivals per stop, not per vehicle. Coverage is evaluated without inspecting candidate service outcomes.

The resulting t_0 values range from 1800 to 3810 s. Matched runs use the same route, seed, control-entry time, and numerical warmup. Each simulation is initialized independently rather than loaded from a shared SUMO snapshot.

1.5h-window protocol. The controllers act at $t_0, t_0+15, \ldots, t_0+585$ s, giving 40 decisions over a 600-second control period. From t_0+600 to t_0+1800, the common numerical policy continues service while respecting outstanding LLM command ownership. The scoring interval is $[t_0, t_0+1800)$ and covers both control and numerical continuation.Including warmup, total simulated durations range from 60 to 93.5 minutes.

8h-window protocol. Control and scoring both cover $[t_0, t_0+28800)$. MEMBus and the current-observation control have 96 decision opportunities at 300-second intervals, with the final decision at $t_0+28500$. Simulation continues for the remaining 300 s and then ends. No numerical control is inserted between these decisions, and no additional continuation phase follows scoring.

Both protocols use the same route–seed set. They differ in control duration, decision frequency, and numerical continuation, so differences between their outcomes cannot be attributed to duration alone.

C.3 Environment and Action Settings

Route and passenger configuration. The experiments use generated SUMO service networks constructed from real-route stop sequences and inter-stop distances, rather than directly executing the complete OSM road-network bindings.

Passenger demand accumulates as fractional mass at one-second intervals. The nominal base rate is 0.05 passengers per stop per second, multiplied by the normalized stop weights. The research binding supplies no additional temporal demand profile or demand-surge multiplier.

The normal passenger capacity is 70, reduced to 60 during absolute simulation-time intervals 720–1440 s and 2160–2880 s. These intervals are measured from simulation start, not from control entry. Their overlap with the scoring window therefore depends on t_0. At served stops, 25% of onboard passenger mass alights before capacity-constrained boarding.

Action execution. The main LLM interface permits holding durations from 0 to 90 s per vehicle. SET replaces the pending duration for the next handled stop, and an exactly zero SET is normalized to CANCEL. KEEP preserves the pending command and its ownership. CANCEL clears its duration while retaining ownership until consumption or replacement. Total physical dwell is capped at 90 s.

The short-window numerical continuation resumes the specified two-way holding policy on its 60-second schedule. It respects unconsumed LLM-owned commands, including controlled zero-duration commands. Further execution semantics are given in Appendix B.2.

C.4 Current Observations and Memory Access

Decision procedure. MEMBus uses an evidence-selection stage followed by action generation. The first stage may select up to two available evidence identifiers, including the option to select none. The second stage produces a complete fleet plan using the returned evidence. Evidence selection does not itself execute an action.

The experiments use CLI and API invocation configurations. Prompts, response schemas, timeout settings, retries, and fallback outcomes are retained with the run records. Model and invocation versions are documented in Appendix G.

A failed evidence query is recorded and supplies no fabricated retrieved content. An action-generation failure produces the all-KEEP fallback. Such failures remain part of the evaluated trajectories.

Current-observation control. The current-observation control receives current vehicle states, queues, pending commands, legal actions, and their execution semantics. Historical observations, past decisions, execution-history records, procedural cards, source evidence, and queryable runtime memory are excluded from its input.

History and retrieval limits. MEMBus additionally receives authorized recent history, external development evidence, and available experience from its own trajectory. The recent-history window is 180 s, with observations sampled at 5-second intervals. Action records, execution feedback, and completed observational windows are included when available.

The two-record query limit applies jointly to external and online evidence. The online view displays at most four procedures, with one case per procedure. External physical-source presentation is limited to eight rows. Memory context and returned archive records share an 80,000-token budget measured with a funtion. Both conditions have the same current-state observation fields and physical command permissions. Their observed values can differ after their trajectories diverge. Unexecuted numerical action recommendations are excluded from the memory-comparison inputs.

C.5 Source Evidence and Online Procedures

External paired evidence. The memory configuration uses six paired development records from Route 152, seed 2026113020. They comprise three action comparisons from each of two source states at simulation times 3000 and 3120 s.

Each candidate changes one target vehicle's pending hold to zero, one-half, or three-quarters of its existing duration, while all other vehicles KEEP. The reference plan assigns KEEP to the entire fleet. Candidate and reference outcomes are compared over 1800 s under the recorded numerical continuation.

Each record retains the candidate and reference plans, source conditions, feedback interval, outcomes, and provenance. These are completed development comparisons rather than counterfactual labels generated online from the evaluation trajectory. The six records are branches from two states of one development run, not six independent runs.

Release and source-context exit. Three records are available at control entry, and three additional records are scheduled for release after 150 s of control. After 300 s, the initial source conditions leave active procedural presentation. Their underlying records remain queryable in the archive.

At each decision, the memory update first releases newly available records and then applies the scheduled source-context exit. The updated view is constructed before evidence selection and action generation. Removing a source condition from active presentation does not establish that its associated action has become harmful.

Online feedback. Online acquisition links submitted plans to command consumption or non-consumption, concurrent interventions, and completed observation windows. Feedback uses predefined, non-overlapping 120-second windows beginning at $t_0 - 120$. Plans are assigned to windows by submission time; a separate window is not started after each action.

Execution feedback is linked through `command_id`. Records distinguish consumption within the window, later consumption, superseded commands, and commands whose consumption has not been observed. A record enters memory after its window closes and the corresponding feedback becomes available. Late receipts may produce a new version of the same plan–window record.

Online procedural lifecycle. Procedural records group experiences by their joint plan, route, reference plan, feedback horizon, continuation, measurement protocol, cohort, units, and evidence type. Source cases and record versions are retained.

At a memory review, a procedure is archived after 180 s without an update. New matching evidence available after archival can trigger re-entry into the active view. This restores retrieval eligibility without certifying causal utility.

The displayed procedures are ordered by update recency, with procedure identifier used to resolve ties. At most four procedures and one case per procedure are displayed.

Timing in the 8h-window protocol. The 180-second recent-history and archival thresholds remain unchanged when decision spacing increases to 300 s.The previous decision's observation is outside the recent-history window at the next decision, although newer observations and newly available feedback may still be included.Online feedback windows remain 120 s. External-evidence release and source-context exit are processed at decisions. Consequently, the 150-second release threshold is first handled at $t_0 + 300$ in the 8h-window protocol.

C.6 Case Identity and Result Reuse

Pairing uses route, seed, control-entry time, scoring boundaries, and the common numerical warmup. Passenger arrivals follow the deterministic demand schedule. Dwell perturbations are coupled by vehicle identity, stop identity, and visit number, using noise seed from absolute simulation time 600 s onward. This matches perturbations by corresponding visits rather than arrival time and does not imply identical realizations of every stochastic mechanism after trajectories diverge.

The recorded endpoint times and decision schedules match the specified protocols for all 200 episodes. When the same MEMBus endpoints appear in both the memory comparison and a baseline table, they remain reused observations rather than additional independent runs. Subset comparisons use the same cases for both methods. Recorded-response replay requires matching requests, prompts, and schemas and does not increase the independent sample count.

D INCREMENTAL MEMORY ABLATIONS

We compare three nested configurations over 50 matched 1.5h-window route–seed cases. All configurations share the initial external-source procedures, paired evidence, action space, and numerical continuation policy. Online observational procedures and their archival and re-entry mechanisms remain enabled throughout. Table 8 summarizes the differences.

Table 8: Mechanisms enabled in the incremental memory ablation.

Mechanism	B1	B12	B123
Initial source procedures and paired evidence	Yes	Yes	Yes
Queryable newly released external evidence	Yes	Yes	Yes
External-source procedure updates	No	Yes	Yes
Initial source-context exit at 300 s	No	No	Yes
Online observational procedure formation/revision	Yes	Yes	Yes
Online procedure archival and re-entry	Yes	Yes	Yes

Configuration differences. B1 keeps the initial external-source procedures fixed. The three additional external records, released after 150 s of control, enter the queryable archive but do not update these procedures. B12 additionally uses the newly released paired evidence to update the external-source procedures. B123 further removes the initial source conditions from active presentation after 300 s of control. Their underlying records remain available for querying.

All three configurations retain runtime observations and can form or revise online observational procedures. These procedures are archived after 180 s without an update and can re-enter the active view when new matching evidence becomes available.

Contrast definitions. For endpoint Y, the two incremental contrasts are

$$\Delta_{\text{update}} = Y_{\text{B12}} - Y_{\text{B1}}, \qquad \Delta_{\text{exit}} = Y_{\text{B123}} - Y_{\text{B12}}.$$

The first measures external-source procedure updates from newly released paired evidence. The second measures the additional effect of removing the initial source conditions from active presentation. The B123-minus-B1 contrast measures their combined effect. These contrasts do not isolate online procedural formation, archival, or re-entry, which remain enabled in all conditions.

Paired results. Table 9 reports the paired contrasts computed from unrounded endpoints. Differences are computed from unrounded endpoints; displayed increments may therefore differ slightly from the displayed combined contrast due to rounding.

Table 9: Incremental ablation contrasts over 50 matched cases. Differences are the first configuration minus the second; served-fraction differences are in percentage points.

Contrast	ΔW (s)	Δ CV	ΔS (pp)
B12−B1	−0.32	−0.02215	+0.034
B123−B12	−1.06	+0.00979	+0.169
B123−B1	−1.39	−0.01236	+0.203

External-source updates improve all three mean service metrics relative to B1. Source-context exit further reduces mean waiting and increases served fraction, with a small increase in CV. B123 achieves the lowest mean waiting and highest served fraction, whereas B12 achieves the lowest CV.

Failures and KEEP fallbacks remain part of the evaluated trajectories. The contrasts include differences in realized execution reliability as well as memory behavior. Repeated runs of an existing route–seed case do not increase the independent sample count.

E BASELINE IMPLEMENTATIONS AND COMPARISONS

The main comparison evaluates MEMBus against five LLM-based, five rule-based, and five learning-based baselines. Both protocols use ten routes with five seeds per route. This section describes the evaluated implementations, provides extended aggregate comparisons, and specifies usage accounting and supplementary ranking.

Methods share an execution interface while retaining their specified observation access, action restrictions, and model-call schedules. Let H denote the target headway, f_b and b_b the forward and backward headways of vehicle b, and $u_{b,t}$ its current legal holding limit. These inputs are derived from authorized fleet observations. Evaluation CV instead uses recorded arrival intervals.

E.1 SHARED EVALUATION PROTOCOLS

All comparisons use the route–seed assignments and scoring boundaries specified in Appendix C. The short-window protocol provides 40 execution opportunities over 600 s, followed by 1200 s of numerical continuation. The 8h-window protocol provides 96 execution opportunities at 300-second intervals without numerical continuation.

Execution opportunities are distinct from fresh model calls. Retrieval queries, intermediate reasoning, refinement, reflection, and plan reuse are accounted for separately. Service outcomes use 50 cases per method and protocol. Execution comparisons use 50 short-window cases. Token and call coverage is stated separately.

E.2 LLM-BASED BASELINES

DiLu. The adaptation uses experience retrieval inspired by DiLu (Wen et al., 2024). Current-state text is encoded with all-MiniLM-L6-v2 into a 384-dimensional vector. Cosine similarity selects three available development examples, with ties resolved by record identifier. The model receives the current state, the previous actual plan, and the retrieved examples. Reflection is disabled.

Where a plan is reused, commands are aligned by vehicle identity and checked against current execution constraints. Departed vehicles are omitted and newly appearing vehicles receive KEEP. Plan reuse does not invoke the model. The retained call records contain 20 new-plan calls per short-window case and 48 per long-window case.

ReAct. The ReAct adaptation combines reasoning and tool use over the authorized current bus state (Yao et al., 2023). The final holding plan is submitted through the common execution interface. Tool responses inform the plan rather than directly executing physical interventions. The call records contain one recorded model stage per scheduled decision.

Recent-history. Recent-history receives the current observation and 180 s of recent observations, actions, and execution feedback from its own trajectory. External development examples, procedural cards, and long-term runtime memory are excluded. It does not update procedural memory or manage its lifecycle. The comparison evaluates the broader MEMBus memory configuration beyond access to recent operational context.

Self-Refine. Self-Refine performs one draft–feedback–refinement cycle per decision (Madaan et al., 2023). The model drafts a holding plan, critiques that draft, and generates a refined plan. Only the final plan is submitted for execution. The intermediate feedback is model-generated critique rather than newly observed traffic feedback. All three stages enter the call and token records.

Reflexion. The decision-level Reflexion adaptation maintains a bounded buffer of verbal reflections on available execution feedback (Shinn et al., 2023). These reflections inform later decisions without restarting the simulation. Action generation and reflection are recorded as separate model stages. The retained records contain 40 action and 39 reflection calls per short-window case, and 96 action and 95 reflection calls per long-window case.

E.3 RULE-BASED BASELINES

The following rules are implemented headway-control references informed by the bus-control literature (Daganzo, 2009; Daganzo & Pilachowski, 2011). Their formulas and adapter settings define the evaluated configurations.

TwoWay. TwoWay combines forward and backward headway information:

$$h_b^{\text{raw}} = \max(0,\ 0.6(H - f_b) - 0.4\max(0, H - b_b))\,.$$

The baseline adapter bounds the duration by the legal holding limit. Unavailable forward-headway information produces KEEP.

ForwardHeadway. The forward-headway rule computes

$$h_b^{\text{raw}} = \max(0,\ 0.5(H - f_b))\,, \qquad h_b = \min(h_b^{\text{raw}}, u_{b,t}).$$

A zero-duration command clears pending holding duration.

BackwardHeadway. The implementation recorded as `BackwardHeadway45` computes

$$h_b = \min(\max(0, 0.15b_b),\ \max(0, \min(45, u_{b,t})))\,.$$

It uses the headway field supplied by its adapter and may command multiple eligible vehicles.

Headway feedback. The implementation recorded as `Daganzo` applies a slack-offset correction:

$$h_b = \text{clip}\,(15 - 0.6(f_b - H),\ 0,\ \min(90, u_{b,t}))\,.$$

This is a parameterized reference related to classical headway-based holding (Daganzo, 2009).

Zero additional holding. This baseline clears pending control-hold duration where permitted while preserving passenger-service dwell. It does not reverse dwell already served. Submitting zero is distinct from KEEP, which preserves the pending command.

NoControl. NoControl provides a separate reference without control interventions. Normal vehicle movement and passenger-service dwell remain active. Unlike Zero additional holding, it does not issue commands to clear pending holding durations.

E.4 LEARNING-BASED CONTROLLERS

All learned policies are frozen during evaluation. Their native action spaces may differ even though they use the common physical execution interface. The descriptions below summarize the algorithmic basis of each method and the available details of its bus-control adaptation. Training mechanisms are distinguished from online control: policy parameters are not updated during the reported evaluation episodes.

DQN. DQN selects holding durations using a frozen action-value network (Mnih et al., 2015). Its action set is $\{0, 5, 10, 15, 20, 25, 30\}$ s. Inputs contain normalized headway deviations, three zero-filled coordinates, vehicle load fraction, sine and cosine clock features, and a constant. The highest-value action is selected. The evaluated adapter returns KEEP if that action exceeds the legal holding limit.

DDPG and SAC. DDPG uses a deterministic actor for continuous control (Lillicrap et al., 2016), whereas SAC uses entropy-regularized actor–critic learning (Haarnoja et al., 2018). The evaluated implementations are trained on a fixed development transition dataset and transferred to the evaluation cases. Their actors remain frozen, and proposals are processed by their holding adapters.

TD3. TD3 extends deterministic actor–critic learning with twin critics, delayed policy updates, and target-policy smoothing (Fujimoto et al., 2018). Using the smaller target value from the two critics reduces overestimation during training. The trained actor provides continuous action proposals during evaluation.

MADDPG. MADDPG uses centralized critics during multi-agent training (Lowe et al., 2017). The evaluated actors generate holding proposals from their assigned observation inputs. Policy parameters remain fixed during evaluation.

CAAC. CAAC addresses credit assignment for asynchronous bus-control events (Wang & Sun, 2021). Its training framework accounts for intervening actions by other vehicles when attributing outcomes to a decision. The frozen policy is evaluated through the decision schedule and execution interface of each protocol.

MPPO. The transferred MPPO implementation maps normalized forward and backward headways and a scaled target-stop queue through a frozen actor to a holding duration bounded by the legal action interface. It serves as a transferred policy-based bus-control reference.

RLHold. RLHold uses a frozen matrix to score the discrete actions $\{0, 7.5, 15, 22.5, 30\}$ s. Its features comprise normalized forward and backward headway deviations, their product, and a constant. The selected action is processed by the holding adapter. Only the largest positive holding candidate is retained at an execution opportunity, with vehicle identity resolving ties.

AWAC. AWAC combines off-policy value learning with advantage-weighted policy regression (Nair et al., 2020). Actions in the training data receive larger policy-update weights when their estimated advantages are higher. Although the original method supports subsequent online fine-tuning, the evaluated policy remains frozen during the bus-control episodes.

CRR. Critic Regularized Regression uses critic estimates to filter or weight actions in the training data during policy regression (Wang et al., 2020). This favors recorded actions with higher estimated value while anchoring policy learning to available data. The resulting policy is used without online parameter updates during evaluation.

BCQ. Batch-Constrained Q-learning restricts action selection to candidates supported by the training data (Fujimoto et al., 2019). In its continuous-action formulation, a learned generative model proposes candidate actions, a bounded perturbation model adjusts them, and the critic scores the resulting candidates. This reduces reliance on value estimates for actions far outside the data distribution.

BEAR. BEAR reduces bootstrapping error by constraining the learned policy toward the support of the behavior data (Kumar et al., 2019). Its training objective uses a maximum mean discrepancy constraint between policy actions and actions sampled from a learned behavior model. The constraint limits unsupported action choices during policy optimization.

CQL. Conservative Q-learning augments value learning with a regularizer that discourages overestimated values for actions insufficiently supported by the training data (Kumar et al., 2020). It provides an offline-learning reference based on conservative value estimation. The learned controller remains frozen during evaluation.

IQL. Implicit Q-learning estimates state values through expectile regression and extracts a policy using advantage-weighted behavioral cloning (Kostrikov et al., 2022). Its value-learning procedure avoids explicitly maximizing over out-of-distribution actions. The extracted policy supplies actions without further training during the evaluation episodes.

TD3+BC. TD3+BC adds a behavior-cloning term to the TD3 actor objective (Fujimoto & Gu, 2021). The resulting update balances estimated action value with agreement with actions in the training dataset. This provides a simple offline-learning reference based on a regularized deterministic actor.

E.5 Extended Controller Comparisons

Table 10 supplements the main comparison with additional learning-based controllers and a fully uncontrolled reference. Each method is evaluated on ten routes with five seeds per route under both

protocols. Outcomes are averaged across seeds within each route and then equally across routes. The two protocols retain their respective control and scoring windows.

NoControl applies no control intervention, whereas Zero additional holding clears pending control holds while preserving passenger-service dwell. The table reports descriptive mean outcomes.

The extended comparison shows that controller performance depends on the evaluation setting. CQL achieves lower mean waiting and CV and a higher served fraction than MEMBus in both windows. RLHold improves all three mean metrics over MEMBus in the 8h window. Compared with NoControl, MEMBus improves all three metrics in the short window; over eight hours, it achieves lower CV but higher waiting and a lower served fraction.

E.6 Model Usage and Execution Accounting

Model calls and tokens. Model usage includes the stages implemented by each controller, including evidence selection, action generation, refinement, and reflection where applicable. Logical model calls are distinguished from physical request attempts and traffic decision opportunities. Reusing an existing plan does not invoke the model.

Token values used in the supplementary ranking are taken directly from Table 2, so the ranking uses the same waiting, service, regularity, and usage values as the main comparison. Rule-based and learning-based controllers do not invoke an LLM during evaluation.

Commands and physical execution. Command submissions are distinguished from KEEP choices and subsequent physical execution. KEEP preserves a pending command, whereas SET and CANCEL replace or clear its duration. An accepted command may be replaced before consumption. Execution statistics cover the controller's intervention window, excluding warmup and the common short-window numerical continuation.Service outcomes use 50 cases per protocol.

E.7 Supplementary Weighted Ranking

The supplementary ranking compares MEMBus with the five LLM-based baselines using the aggregate values reported in Table 2. For controller m, window h, and metric k, $r_{m,h,k}$ denotes its rank among these six controllers. Waiting, CV, token usage, and bunching are ranked in ascending order, while served fraction is ranked in descending order. Ties receive average ranks.

To emphasize passenger waiting and bunching mitigation, the within-window score is

$$R_{m,h} = 0.30\, r_{m,h,W} + 0.15\, r_{m,h,C} + 0.15\, r_{m,h,S} + 0.10\, r_{m,h,T} + 0.30\, r_{m,h,B},$$

where W, C, S, T, and B denote waiting, headway CV, served fraction, token usage, and bunching events, respectively. Command counts are excluded because fewer submissions do not necessarily indicate better control.

The combined score assigns greater weight to sustained eight-hour operation:

$$R_m = 0.30\, R_{m,\mathrm{short}} + 0.70\, R_{m,8\mathrm{h}}.$$

Under this weighting of the main-table values, MEMBus ranks first with a score of 2.075, followed by DiLu (2.460), ReAct (2.565), Self-Refine (3.470), Reflexion (4.930), and Recent-history (5.500). Lower scores indicate better weighted ranks. The weights were selected for a supplementary analysis after inspection of the results. This descriptive ranking does not replace the primary waiting-time endpoint or establish statistical significance.

Result identity and reuse. Each comparison retains its route–seed identities, controller configuration, endpoint version, and measurement definitions. Service outcomes and execution statistics use their respective case sets and accounting units. An endpoint appearing in multiple comparisons remains one observation. Repeated runs and recorded-response replay do not increase the independent sample count.

Table 10: Extended controller comparisons over 50 cases per method and protocol. Each cell reports 1.5h-window / 8h-window means. Served fractions are percentages.

Controller	Wait (s)↓	CV↓	Served (%)↑
MEMBus	609.53 / 2672.54	0.72706 / 0.89431	63.355 / 84.351
Rule-based controllers			
TwoWay	632.21 / 2818.84	0.53792 / 0.44621	62.363 / 83.276
ForwardHeadway	653.90 / 3080.71	0.54696 / 0.41490	61.334 / 81.534
BackwardHeadway	632.82 / 3037.33	0.60322 / 0.71627	62.279 / 81.835
Headway feedback	674.68 / 3345.36	0.48846 / 0.37687	60.961 / 79.553
Zero additional holding	599.20 / 2539.86	0.72903 / 1.00151	64.023 / 85.274
Learning-based controllers			
MPPO	632.77 / 2745.50	0.46510 / 0.27897	62.905 / 83.647
DQN	646.73 / 2938.67	0.65576 / 0.63762	61.183 / 82.858
RLHold	617.39 / 2607.22	0.64040 / 0.75233	62.936 / 84.726
CAAC	621.18 / 2802.01	0.63524 / 0.72797	62.561 / 83.477
MADDPG	679.74 / 3484.13	0.69462 / 0.71238	59.377 / 78.870
DDPG	718.39 / 3918.69	0.71730 / 0.75632	57.508 / 75.936
TD3	717.12 / 3885.71	0.70418 / 0.72088	57.585 / 76.177
SAC	718.13 / 3950.08	0.72567 / 0.81013	57.328 / 75.753
AWAC	638.47 / 3097.53	0.76223 / 1.23509	61.975 / 81.649
CRR	709.41 / 3833.33	0.74415 / 0.84872	57.799 / 76.603
BCQ	602.23 / 2666.22	0.73760 / 1.06339	64.146 / 84.289
BEAR	646.01 / 3132.95	0.72207 / 0.86743	62.763 / 80.933
CQL	600.32 / 2499.71	0.58321 / 0.52438	65.401 / 85.346
IQL	640.62 / 2979.62	0.65705 / 0.94440	62.947 / 82.143
TD3+BC	634.08 / 2928.76	0.77364 / 1.26676	61.922 / 82.794
Uncontrolled reference			
NoControl	616.56 / 2582.24	1.14582 / 1.23594	61.933 / 85.125

F Metrics and Statistical Analysis

Scoring cohort. For scoring interval $[t_0, t_1)$, the cohort contains passengers waiting immediately before events at t_0 and passengers arriving during the interval. Passenger mass can be fractional. FIFO records track arrival times and boarding events. When a record is only partly boarded, boarded and unserved portions are accounted for separately.

Let a_p denote arrival time, b_p boarding time, and m_p the mass of a passenger record or portion. Set $b_p = \infty$ for mass remaining unserved. For a nonempty cohort, restricted mean waiting is

$$W = \frac{\sum_{p\in\mathcal{P}} m_p \left[\min(b_p, t_1) - \max(a_p, t_0)\right]}{\sum_{p\in\mathcal{P}} m_p}. \tag{10}$$

Waiting before t_0 is excluded. Unserved mass contributes waiting through t_1 and remains in the denominator. The endpoint measures queue waiting rather than total journey time.

Served fraction and endpoint queue. Served fraction uses the same cohort:

$$S = \frac{\sum_{p\in\mathcal{P}} m_p \, \mathbf{1}[b_p < t_1]}{\sum_{p\in\mathcal{P}} m_p}. \tag{11}$$

Endpoint queue is the remaining waiting mass at the scoring boundary, before arrivals at t_1 enter the next interval. Tables express served fraction as a percentage and between-controller differences in percentage points.

Headway CV. For each stop, consecutive arrivals within the scoring window define arrival intervals. Both arrivals defining an interval must lie within $[t_0, t_1)$. The intervals from all stops are

pooled as $g_1, \ldots, g_n$. We compute

$$\bar{g} = \frac{1}{n}\sum_{j=1}^{n} g_j, \qquad s_g = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(g_j - \bar{g})^2}, \qquad C = \frac{s_g}{\bar{g}}.$$

The metric requires at least two intervals and a positive mean. It is not the average of separately computed stop-level CVs. The arrival intervals used for evaluation are distinct from the spatial headways supplied to controllers.

Bunching events. A bunching event is counted when the interval between two consecutive arrivals at the same stop is strictly less than 25% of the target headway. Both arrivals must lie within the scoring window: 1,800 s for the 1.5h-window protocol and 28,800 s for the 8h-window protocol. Using the pooled within-stop arrival intervals $g_1, \ldots, g_n$, the case-level count is

$$B = \sum_{j=1}^{n} \mathbf{1}[g_j < 0.25H],$$

where H is the route's target headway.

Counts are computed from stop-arrival records rather than periodic state scans. Each qualifying interval contributes one event. Repeated bunching by the same vehicle pair at different stops is counted separately, rather than merged into one continuous episode. Tables report mean case-level counts, averaged within routes and then equally across routes.

Aggregation and pairing. Service outcomes are evaluated on ten routes with five seeds per route under each protocol. For route r with seed set $\mathcal{S}_r$, the controller mean is

$$\bar{Y}_r = \frac{1}{|\mathcal{S}_r|}\sum_{s\in\mathcal{S}_r} Y_{r,s}.$$

The overall mean gives equal weight to the R routes:

$$\bar{Y} = \frac{1}{R}\sum_{r=1}^{R} \bar{Y}_r.$$

With five seeds on each of ten routes, this equals the mean over the 50 case-level endpoints. Waiting and served fraction are first computed within each case; passengers from different cases are not pooled into a single aggregate ratio.

Pairwise contrasts use matched route–seed cases within the same protocol. For baseline k, the case-level difference is

$$d_{r,s}^{(k)} = Y_{r,s}^{\mathrm{MEMBus}} - Y_{r,s}^{(k)}.$$

These differences are averaged within routes and then equally across routes. Negative waiting, CV, and bunching differences favor MEMBus; positive served-fraction differences favor MEMBus. Reported contrasts use unrounded endpoints.

Comparisons based on a subset identify the included cases and use the same subset for both controllers. The complete 50-case sample size is not assigned to statistics computed from fewer cases.

Usage and execution statistics. Service endpoints, model usage, and command execution have distinct accounting units. Service comparisons use 50 cases per protocol. Execution comparisons use 50 short-window cases.

Logical calls, physical request attempts, per-vehicle action choices, submitted commands, and consumed commands are distinguished. A command submission does not by itself establish additional physical dwell. Token usage and execution accounting are described in Appendix E.

Uncertainty intervals. The reported short-window MEMBus versus current-observation comparison uses descriptive 95% percentile intervals from 10,000 bootstrap resamples. Paired seeds are resampled within each fixed route, and the route-level means are then equally averaged. Routes themselves are not resampled. The intervals describe seed variation conditional on the evaluated routes rather than uncertainty over unseen-route performance.

Comparisons across protocols. Both protocols use the same ten routes and five seeds per route. Their results are presented together, but service outcomes are computed separately under their respective control schedules and scoring windows. The protocols differ in control duration, decision spacing, and numerical continuation.

Bunching counts cover scoring windows of different lengths and are not event rates. They are compared between controllers within each protocol rather than interpreted directly as changes in bunching frequency across protocols.

The supplementary weighted ranking combines within-window ranks rather than pooling raw service endpoints across horizons. Its metric and window weights are specified in Appendix E.7. This descriptive ranking remains separate from the primary waiting-time comparison.

Failures and dependent observations. Recorded query failures, invalid action responses, and KEEP fallbacks remain part of completed evaluated trajectories. Query failures and action fallbacks are counted separately because a failed intermediate query need not cause an action fallback.

Decision counts within a run are not independent traffic replications. Reusing endpoints across tables, replaying recorded responses, or repeating a route–seed case does not increase the independent sample count. Repair results retain their configuration and result versions. Incomplete cases and missing execution records are identified rather than assigned zero outcomes.

G Compute, Software, and LLM Usage Disclosure

Model usage. The 1.5h-window protocol provides 40 decision opportunities per run, and the 8h-window protocol provides 96. MEMBus uses evidence selection followed by action generation. Other LLM controllers follow their configured retrieval, reasoning, refinement, or reflection procedures.

The short-window current-observation control retains the evidence-selection stage with an empty catalog. The long-window implementation includes versions that skip this empty query. Decision opportunities therefore do not imply identical call counts across all configurations.

Usage accounting distinguishes logical model calls, physical request attempts, successful responses, and responses used for control. Input, cached-input, and output tokens, reference-token estimates, retries, and wall-clock latency are recorded separately where available. Missing usage records are not assigned zero. Rule-based and learning-based controllers make no LLM calls during evaluation.

Failure accounting. In the original 50-case short-window memory comparison, MEMBus records 98 action fallbacks among 2000 decisions, while the current-observation control records the same. The two conditions together have 8000 stage-level transport records, including 351 failures. These counts apply to this comparison rather than the full set of baseline experiments.

Query-stage failures and action fallbacks are separate events. A failed evidence query need not prevent action generation, whereas an action failure invokes the specified KEEP fallback. Fallback decisions remain in the trajectories used to compute service outcomes. Per-vehicle fallback KEEP choices are distinguished from failed fleet-level decisions.

Software and replay. Reproduction requires the simulator and asset versions, controller configuration, initial memory, random seeds, and metric definitions associated with each result. Stored requests, responses, actions, and execution records support inspection and replay.

Exact reproduction through fresh OpenAI API calls is challenging because the hosted model is accessed as a black box. The experiment code does not control the provider's serving infrastructure or updates to the model and API behavior. The same requested model identifier, prompt, and invocation settings may therefore produce different responses at different times. Differences can also arise from generation nondeterminism, even without a documented model update. In closed-loop control, an early response difference can change subsequent actions, traffic states, and later model inputs.

Archived responses support replay of the recorded controller decisions. Response reuse requires the request, prompt, and schema to match the saved originals; simulation-state and execution checks are

also required. Replay is distinguished from a fresh online evaluation and does not add independent experimental replications.

The simulation waits for model inference before advancing simulated time. Wall-clock latency is therefore recorded as computational cost rather than added to simulated passenger waiting.

H LLM ASSISTANCE IN MANUSCRIPT PREPARATION

Codex assisted with manuscript organization, language editing, comparison of method descriptions with implementation code, and preparation of LaTeX sources. This assistance is separate from the LLM controllers evaluated in the experiments. The authors reviewed the resulting text and checked experimental claims and numerical values against the corresponding configurations and recorded outputs.