# Ping-Ponder: Concurrent Dual-Agent Spoken Dialogue via Shared Belief State

**Akiko Masaki-Kato** **Haris Gulzar**
NTT, Inc.
Musashino R&D Center, 3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
akiko.masaki@ntt.com

## Abstract

Maintaining responsiveness while performing complex task reasoning remains a fundamental challenge in spoken dialogue systems. Existing dual-agent approaches separate conversation from reasoning but typically run the two sequentially, beginning reasoning only after the necessary user information is collected, causing substantial planning latency. We propose Ping-Ponder, a dual-agent framework in which two agents operate concurrently through a shared belief state. The conversational agent (Ping) handles real-time spoken interaction while incrementally extracting intent from user utterances and updating the shared belief state. The reasoning agent (Ponder) begins planning and knowledge retrieval as soon as partial intent information becomes available, without waiting for complete slot filling, and incrementally updates the belief state with intermediate results. This allows Ping to continue eliciting missing information while promptly presenting candidate plans generated by Ponder. In a paired single-operator case study, the proposed approach reduces average planning-phase latency by 93% in English (from 7.1 s to 0.5 s) and 87% in Japanese (from 7.3 s to 1.0 s) compared with a sequential dual-agent baseline. In a larger replay study on two spoken task-oriented corpora, the same ordering holds under controlled conditions: Ping-Ponder maintains time-to-first-response parity with the blocking baseline while reducing planning-phase latency on planning-triggered turns by 31–32%, with no evidence of degraded reference-based response quality under an LLM judge; human-perceived conversational quality remains to be tested in a user study. These results suggest that concurrent processing through a shared belief state is a promising direction for reconciling responsiveness and reasoning capability in spoken dialogue systems.

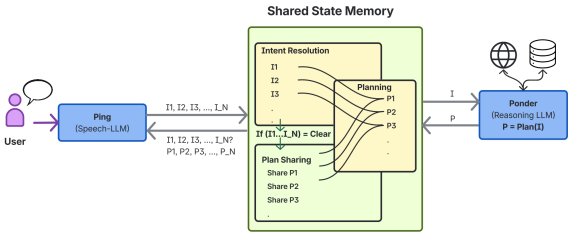


Figure 1: Ping-Ponder architecture. Ping writes intent slots into Shared State Memory; Ponder reads available intents, produces plans $P = \mathrm{Plan}(I)$, and writes them back. Stable fragments move into the Plan Sharing buffer and are verbalised by Ping.

## 1 Introduction

Spoken dialogue systems operate under a strict responsiveness budget. Silences longer than roughly one second begin to feel unnatural and erode perceived fluency (Skantze, 2017). At the same time, useful task-oriented assistants must perform non-trivial reasoning. They need to disambiguate underspecified requests, retrieve external information, and compose multi-step plans, and when these operations are implemented with modern large language models (LLMs) and tool use their latency routinely reaches several seconds per turn. This tension between *responsiveness* and *deliberation* is the central architectural problem we address.

A natural response is to split the system into a fast agent and a slow one, echoing Kahneman's System 1/System 2 distinction (Kahneman, 2011). Recent LLM-based instantiations include the *Talker-Reasoner* architecture (Christakopoulou et al., 2024) and the *chat-supervisor* pattern popularised by industrial real-time voice demonstrations. In practice, however, these systems are arranged *sequentially*. The conversational agent finishes eliciting a request, hands control to the reasoning agent, and then waits, producing an awkward multi-second silence precisely when planning is most visible. Spoken input compounds the prob-



*Proceedings of the 27th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 188–204
August 2–5, 2026. 

lem: speech is noisy and indirect, making intent extraction harder than in written dialogue (Gulzar et al., 2025), while incremental dialogue processing (Schlangen and Skantze, 2011; Skantze and Hjalmarsson, 2013) has long argued that components should emit and consume partial hypotheses rather than wait for turn boundaries. A strictly sequential dual-agent design is at odds with both constraints.

We propose *Ping-Ponder*, a dual-agent architecture in which the two agents run concurrently and communicate through an explicit, structured shared belief state. The conversational agent, *Ping*, drives ASR and TTS, manages turn-taking, and writes partial intent information to the belief state as soon as evidence becomes available. The reasoning agent, *Ponder*, subscribes to the belief state and begins planning as soon as a viable fragment of intent is posted, writing intermediate results back for Ping to verbalise while elicitation continues. The belief state thus serves as a *bidirectional coordination medium*. This revives a long-standing idea of shared, incrementally updated state across cooperating components (Erman et al., 1980; Schlangen and Skantze, 2011), applying it as an online coordination channel between two concurrent LLM agents rather than as the passive per-turn representation typical of dialogue state tracking (Williams et al., 2013; Mrkšić et al., 2017).

We implement Ping-Ponder on top of a real-time speech interface and evaluate it in English and Japanese. In a paired live case study against a synchronous dual-agent baseline, concurrent execution via the shared belief state reduces average planning-phase latency by 93% in English (7.1 s to 0.5 s) and 87% in Japanese (7.3 s to 1.0 s), and a subsequent large-scale replay study reproduces the ordering with statistical confidence and shows no evidence of degraded reference-based response quality under an LLM judge. The long silences characteristic of current dual-agent spoken systems need not be intrinsic under synchronous coordination schemes; the evidence here suggests they can be substantially reduced by restructuring coordination rather than by replacing the underlying models.

**Contributions.** (i) We introduce Ping-Ponder, a concurrent dual-agent spoken dialogue architecture in which two LLM-based agents operate in parallel rather than in sequence. (ii) Building on the long tradition of shared-state and incremental dialogue processing, we instantiate the belief state as an explicit, structured coordination medium through which two agents exchange partial intent and intermediate plans concurrently rather than per turn. (iii) We provide a controlled empirical evaluation in English and Japanese showing that this design removes the bulk of planning-phase silence with no evidence of degraded reference-based response quality under an LLM judge, with ablations that separate speculative prefetch from the dual-agent decomposition.

The remainder of the paper reviews related work (§2), describes the architecture (§3), reports the experimental setup and results (§4–5), and discusses implications and limits (§6).

## 2 Related Work

**Dual-agent architectures.** The System 1/System 2 distinction (Kahneman, 2011) has been adopted as an organising principle for compound AI systems, and the closest conceptual antecedent of Ping-Ponder is the *Talker-Reasoner* architecture of Christakopoulou et al. (2024), in which a Talker LLM handles turn-level conversation while a Reasoner LLM performs hierarchical planning and updates a belief state consumed by the Talker on the subsequent turn; the framework permits the Talker to proceed independently or optionally wait for the Reasoner. Ping-Ponder adopts this division of labour, but differs in three concrete respects that matter for spoken interaction: (i) we realise the non-blocking promise of the framework via *explicit prefetch scheduling*, dispatching the reasoning agent against the next turn before the user speaks rather than sequentially after; (ii) we adopt a *structured, bidirectional* belief state in the form of a `{slots, proposals}` record that both agents read and write, rather than a Reasoner-only natural-language scratchpad; (iii) we provide a multi-domain empirical evaluation on spoken task-oriented dialogue with paired ablations that isolate prefetch from the dual-agent decomposition, whereas Talker-Reasoner is evaluated only qualitatively on text-based sleep-coaching dialogues. On the industrial side, the open-source *Chat-Supervisor* pattern (Agentic Pattern 1 in OpenAI's Realtime Agents (OpenAI, 2024)) synchronously invokes a heavier text supervisor on reasoning-heavy turns and is functionally equivalent to the *Talker-waits-for-Reasoner* variant of Christakopoulou et al. (2024); we adopt it as the

synchronous dual-agent baseline against which we measure Ping-Ponder (§4.3).

**Shared state, blackboard architectures, and incremental dialogue.** The idea of decomposing language processing into asynchronous components that cooperate through a shared, structured representation long predates LLM-based agents. The classic *blackboard* model, exemplified by the Hearsay-II speech-understanding system (Erman et al., 1980) and later systematised by Nii (1986a,b), lets independent knowledge sources post and revise hypotheses on a shared data structure to incrementally resolve uncertainty. In dialogue, the TRAINS project (Allen et al., 1995) separated conversational management from plan reasoning in a mixed-initiative planning assistant, and the incremental-processing tradition, exemplified by the IU framework of Schlangen and Skantze (2011) and work such as DeVault et al. (2009); Skantze and Hjalmarsson (2013), argued that dialogue components should consume and emit *partial* hypotheses across module boundaries rather than wait for turn completion. Ping-Ponder is squarely in this lineage, and we do not claim that separating dialogue management from planning, coordinating through shared state, or processing intent incrementally is new in itself. What differs is the setting and the coordination discipline. Classic blackboard systems rely on a centralised *control* component that opportunistically schedules and arbitrates which knowledge source fires next (Nii, 1986a); Ping-Ponder has no such controller, using instead a *partitioned single-writer* discipline in which Ping owns the intent slots and Ponder owns planning status and plan drafts, so that the two agents read and write concurrently without a central arbiter. The cooperating parties are also asymmetric in a way the classic systems were not: a real-time streaming conversational agent runs alongside a higher-latency deliberative LLM, and we hide the cost of the slow side through *speculative prefetch* that dispatches reasoning against the anticipated next turn. Finally, where the earlier systems targeted speech recognition or symbolic planning, we instantiate these ideas in an LLM-era real-time spoken task-oriented dialogue setting and report a controlled latency evaluation across two languages.

**Spoken task-oriented dialogue and real-time latency.** Task-oriented dialogue has traditionally been organised around *dialogue state tracking* (DST) (Williams et al., 2013), whose accuracy degrades sharply in spoken settings where disfluencies, indirect phrasings, and ASR errors are pervasive (Gulzar et al., 2025). Ping-Ponder is complementary to that literature: rather than improving an offline DST model, it uses an online, LLM-maintained belief state as the coordination substrate between two concurrent agents at inference time. On the latency side, a line of full-duplex, end-to-end spoken dialogue models targets sub-second responsiveness and natural overlap: dGSLM generates two-channel conversational audio with naturalistic turn-taking and overlap directly from raw speech (Nguyen et al., 2023), and Moshi (Défossez et al., 2024) pushes perceived latency below one second via full-duplex audio-to-audio generation. These models achieve responsiveness and interruption handling by tightly coupling speech and language modelling into a single network. We take the opposite path: we keep conversation and reasoning as separate agents so that the reasoning side can use general-purpose LLMs and tool use, while still achieving sub-second planning latency through concurrent execution. We do not claim that separating dialogue management from planning, sharing state between cooperating components, or incremental processing is new. Our contribution is to instantiate these ideas in an LLM-era real-time spoken task-oriented dialogue system in which a shared, incrementally updated belief state coordinates two concurrent agents under speculative prefetch scheduling, and to report a controlled planning-latency evaluation across English and Japanese.

## 3 Ping-Ponder Architecture

### 3.1 Overview

Figure 1 shows Ping-Ponder's three first-class components: the *Ping* agent, a low-latency, bounded-cost System 1 component; the *Ponder* agent, a higher-cost, deliberative System 2 component; and the *shared belief state* $\mathcal{B}$ that mediates between them. User audio streams into Ping via ASR; Ping emits responses via TTS and writes extracted intent to $\mathcal{B}$. Ponder subscribes to $\mathcal{B}$, begins planning as soon as a viable fragment of intent is posted, and writes intermediate plans back. The two agents run as independent processes; neither blocks on the other, and coordination is entirely mediated through explicit reads and writes on $\mathcal{B}$. Whereas recent LLM-based dual-process dialogue architectures typically alternate between the two

agents (Christakopoulou et al., 2024), here the two agents run *simultaneously*, coordinating only through reads and writes on $\mathcal{B}$.

### 3.2 Shared Belief State

$\mathcal{B}$ is a structured record with three field groups. The *intent slots* $S = \{(k_i, v_i, c_i)\}_i$ hold typed key-value-confidence triples that Ping writes as it incrementally extracts information from partial ASR hypotheses. A *planning status* field in $\{\texttt{idle}, \texttt{planning}, \texttt{partial}, \texttt{ready}\}$ is controlled by Ponder. A *plan draft* section holds the current best candidate plan together with a *plan-sharing* queue of stable fragments that Ping has been cleared to verbalise. Read/write access is mediated by a lightweight in-process key-value store with per-field version counters; both agents subscribe to a change-notification stream, so $\mathcal{B}$ is an active publish/subscribe channel between two concurrently running LLM agents rather than a passive per-turn snapshot as in classical DST. A worked example is given in Appendix B.

### 3.3 Ping and Ponder

Ping runs a tight loop: consume partial ASR hypotheses, incrementally update $\mathcal{B}$'s intent slots, consult the plan-draft and planning-status, and emit speech. At each decision point Ping produces one of three outputs: an *acknowledgement*, a short filler emitted while Ponder is still planning; a *clarification question*, used for slots below a confidence threshold; or a *final response*, issued once $\mathcal{B}$'s planning status reaches `ready`. Ping never waits for Ponder. In our implementation Ping is driven by the OpenAI Realtime API, which provides streaming ASR, dialogue management, and TTS at sub-second latency.

Ponder is event-driven. Its trigger condition is deliberately permissive: as soon as a domain-specified minimally sufficient subset of intent slots has been posted, it enters the `planning` state. A planning episode has three phases. It begins with *retrieval* against external knowledge, then *plan drafting* by a reasoning-capable LLM, and finally *commit* back into $\mathcal{B}$. These phases are not atomic. Ponder writes intermediate results (`partial`) as soon as they become available so that Ping can begin verbalising candidate answers before the full plan has stabilised. If Ping writes new or corrected slot values while Ponder is mid-plan, Ponder detects the version mismatch on its next tick, discards the stale draft, and restarts. Ponder is driven by a GPT-4.1-family model, specifically `gpt-4.1` in our interactive system and `gpt-4.1-nano` in the large-scale replay study, with external services wired in as tool calls.

### 3.4 Concurrency

Because Ping and Ponder both mutate $\mathcal{B}$, we guard against *stale reads* and *lost updates* with partitioned write authority and per-field version counters. Intent slots are write-owned by Ping; planning status and plan draft are write-owned by Ponder. When Ponder commits a plan, it records the versions of the slots it was based on; if any is subsequently overwritten, the next tick detects the mismatch and restarts the relevant phase. Because writes within each group are single-writer, last-writer-wins semantics are safe without a distributed consensus protocol. In our implementation both agents run as threads of a single process communicating via shared memory; the protocol generalises to multi-process deployment with standard compare-and-swap primitives.

## 4 Experimental Setup

We evaluate Ping-Ponder under two regimes. A small interactive study establishes the headline latency claim in a live voice-in/voice-out setting, and a large-scale replay study re-evaluates four architectural conditions against two spoken task-oriented dialogue corpora, scoring candidate responses against human-played agent turns with a reference-based LLM judge.

### 4.1 Corpora

Our primary corpus is an internally collected Japanese spoken task-oriented dialogue dataset covering 6 domains: accommodation consultation, restaurant and dining reservation, taxi and private-transport booking, rail and long-distance train booking, medical appointment booking, and food and culinary-tour booking. The medical domain is limited to appointment scheduling and previsit logistics and contains no clinical advice, diagnosis, or real patient information; the full composition, sub-category breakdown, collection procedure, and consent scope are given in Appendix C. Reference dialogues are two-channel audio with a user (CH1) and agent (CH2), fully transcribed with spoken features preserved (fillers, self-repairs, backchannels). We sample 20 dialogues per domain, giving $6 \times 20 = 120$ dialogues per condition and 8,765 judged turns aggregated across the four

conditions. For cross-lingual robustness we additionally evaluate on single-domain subsets of SpokenWOZ (Si et al., 2023), restricted to dialogues whose gold annotations contain exactly one active domain (hotel, restaurant, taxi, train, attraction, hospital). This yields 106 dialogues per condition across six domains and 5,067 judged turns. SpokenWOZ dialogues are short (seven to eight turns on average) and operationally narrow.

### 4.2 Replay Protocol

To isolate *architectural* cost from *conversational* variability, the system does not drive its own simulated user. For every reference dialogue we replay the logged user side turn by turn. The agent receives the $t$-th user utterance, is allowed to produce an internal trace, including planning-phase events, and a spoken-style response, and is only then advanced to user turn $t{+}1$. Because the simulated user does not advance until the candidate turn is emitted, wall-clock response speed and response quality are decoupled. We record, per turn, *time to first response* (TTFR), measured from the end of the user turn to the first agent audio/text chunk, and *planning-phase latency*, the wall-clock duration from a heavy-model call's initiation to the availability of its output, regardless of whether that output is a structured $\{\text{slots}, \text{proposals}\}$ update (Ping-Ponder and No-prefetch) or a natural-language reply that the chat agent reads verbatim (Baseline). The measurement is identical across conditions; only the downstream consumption differs, namely background state injection versus verbatim chat-agent read-out; Appendix B.2 gives a side-by-side comparison and the correspondence to the Abstract values. Planning latency is reported on the *fired subset*, i.e. the turns on which the heavy-model call is invoked. Fire rates are *per turn* unless a table states otherwise. The candidate response is passed to the judge of §4.4.

### 4.3 Baseline and Ablations

We compare four conditions that share the same LLM backbone, STT, TTS, and domain prompts. **Baseline** is a classic blocking Chat-Supervisor system (OpenAI, 2024), functionally equivalent to the *Talker-waits-for-Reasoner* variant of Christakopoulou et al. (2024), retaining the reference implementation's natural-language supervisor output and decoding settings. **No-prefetch** is identical to Ping-Ponder except that the Ponder call is dispatched after the user utterance rather than prefetched, isolating speculative prefetch. **Single-agent** removes the supervisor entirely and is a lower-bound control on chat-only latency. Because Baseline also differs from Ping-Ponder in output format and decoding (see §4.5), we treat No-prefetch, not Baseline, as the clean architectural control for prefetch scheduling. Full per-condition configurations are deferred to Appendix B.1 (Table A5).

| Condition | Lang | Intent-res. (s) | Planning (s) |
|---|---|---|---|
| Baseline | EN | **0.42** | 7.11 |
| Ping-Ponder | EN | 0.45 | **0.49** |
| Baseline | JA | **0.70** | 7.31 |
| Ping-Ponder | JA | 0.75 | **0.96** |

Table 1: Interactive planning and intent-resolution latency in a paired single-operator case study (lower is better). Planning latency is measured from heavy-model call initiation to the availability of its output, under the unified definition of Table 3 and Appendix B.2; intent-resolution is end-of-user to first user-visible token. Baseline is the blocking Chat-Supervisor condition.

### 4.4 Reference-Based Judge

We score candidate responses against the gold agent turn at the same dialogue position (CH2 in JA; system side in SpokenWOZ) with an LLM judge (`gpt-4.1`, temperature 0, JSON mode). The judge returns integer scores on a 0–3 scale for *Appropriateness*, the topic fit of the response with the user turn; *Faithfulness*, its information-content agreement with the gold, scored to discount fillers and repairs under the v2 disfluency-aware rubric; and *Fluency*, its target-language fluency, which penalises awkward repetition and machine-translationese. Scores are aggregated per domain and per condition; see Appendix A (Tables A1 and A2).

### 4.5 Implementation Details

The Ping chat agent runs on a real-time speech interface with streaming ASR, TTS, and turn-taking. All four conditions share the same underlying LLM family and the same language-matched prompts, but differ in decoding hyper-parameters and output format because each condition is designed to instantiate a different canonical architecture, not to sit on the same Pareto frontier. Ping-Ponder and No-prefetch share an identical Ponder reasoning call (`fastSupervisor` in the configuration tables of Appendix B.1; 500 tokens, temperature 0.3, JSON

mode for {slots, proposals}) on top of a 120-token chat agent at temperature 0.3; Baseline uses the Chat-Supervisor reference setting of 100-token blocking chat triggered by a `CALL_SUPERVISOR` marker and a 220-token natural-language supervisor at temperature 0.4; Single-agent is chat-only. Full per-condition configurations are listed in Appendix B.1 (Table A5). Note that Ping-Ponder versus No-prefetch is a single-factor ablation on speculative prefetch, with all other hyper-parameters held identical, whereas Ping-Ponder versus Baseline is an end-to-end comparison against a representative blocking dual-agent architecture and therefore confounds coordination, output format, and token budget; we report both comparisons because they answer different questions. Latencies are measured from wall-clock timestamps on the event stream. We release replay JSONLs, judge prompts, and per-turn numeric scores with the camera-ready version; raw judge rationales are not redistributed (see Ethics Statement).

## 5 Results

We report two complementary evaluations that together establish Ping-Ponder's latency/quality profile: a live interactive study (§5.1) that produces the headline planning figures quoted in the abstract, and a large-scale replay study (§5.2) that verifies the effect generalises under controlled conditions across two languages, two corpus styles, and six domains. Response quality is evaluated by a disfluency-aware reference-based LLM judge (§5.3, full per-language tables in Appendix A).

### 5.1 Headline: Planning-Phase Latency in Interactive Sessions

We first report the *planning-phase* latency measured on a paired interactive session in which a single operator performed the same task-oriented dialogue through Ping-Ponder and through the Chat-Supervisor baseline of §4.3, under identical prompts and LLM versions in both English and Japanese. A turn-by-turn trace of the travel-planning session, showing how Ping's incremental slot writes and Ponder's concurrent knowledge-base queries interleave in the Shared Belief State, is given in Appendix B.3 (Figure A1). Planning latency, the quantity quoted as the headline in the abstract, is the elapsed time from the initiation of a heavy-model reasoning call to the point at which its output becomes available to the user-facing agent. This call is a blocking supervisor call in the Baseline and a Ponder call in Ping-Ponder, and its output is correspondingly a natural-language supervisor reply or a structured belief-state update. It is distinct from intent-resolution latency, the time to the first user-visible token, which we report alongside planning latency to confirm that the planning speedup does not come at the cost of perceived responsiveness.

Table 1 summarises. Ping-Ponder reduces average planning latency from 7.11 s to 0.49 s in English (a 14.4× speedup, or 93% reduction) and from 7.31 s to 0.96 s in Japanese (7.6×, 87%), while preserving intent-resolution responsiveness (0.42 → 0.45 s EN, 0.70 → 0.75 s JA). Planning latency in this regime is brought comfortably inside the one-second conversational budget (Skantze, 2017); the large-scale replay study of §5.2 verifies that this ordering holds under controlled conditions with statistical confidence.

The improvement is consistent with the architectural scheduling change. Ping-Ponder's Ponder begins planning from the *partial* slot state available at each turn, whereas the baseline blocks on a supervisor call for turns that the chat agent classifies as requiring reasoning. The large-scale replay study of §5.2 establishes that this ordering holds across two corpora and six domains with statistical confidence, and further isolates the contribution of speculative prefetch.

### 5.2 Large-Scale Replay Study

To complement §5.1 we reproduce the latency measurements on a replay harness (§4.2) that holds user turns fixed and injects them into each architectural condition. This fixes the user-side trajectory and reduces conversational variability, and yields a sample roughly an order of magnitude larger than the interactive study. We run 480 Japanese dialogue-condition runs (120 unique dialogues × four conditions) from our in-house spoken TOD evaluation set across six domains, and 424 English dialogue-condition runs (106 × 4) from SpokenWOZ in its single-domain setting. The four ablation conditions of Table A5 are Baseline (Chat-Supervisor), No-prefetch (Ping-Ponder with speculative prefetch disabled), Ping-Ponder (full), and Single-agent (no supervisor).

#### 5.2.1 Time to First Response (TTFR)

TTFR captures the user-perceived delay to the first audible token. All four conditions share STT, TTS,

| Lang | Baseline | No-prefetch | Ping-Ponder | Single-agent |
|---|---|---|---|---|
| JA | 1576 ±99 | 2803 ±121 | **1526 ±67** | 1542 ±70 |
| EN | **886 ±50** | 1547 ±81 | 891 ±43 | 999 ±68 |

Table 2: Time to first response across four ablations (mean ± 95 % CI, ms; end-of-user to first agent chunk). $N$=120 runs per JA cell, $N$=106 per EN cell.

| Lang | Condition | Planning (ms) | N | Fire rate (%) |
|---|---|---|---|---|
| JA | Baseline | 1376 ±71 | 384 | 16.4 |
| JA | **Ping-Ponder** | **950 ±33** | 526 | 75.8 |
| EN | Baseline | 1029 ±30 | 255 | 19.6 |
| EN | **Ping-Ponder** | **703 ±38** | 25 | 13.2 |

Table 3: Planning latency on the fired subset (mean ± 95 % CI, ms; $N$ = fired turns; Single-agent omitted). Baseline's fired event is the blocking `CALL_SUPERVISOR` invocation; its fire rate is the fraction of *turns* on which that invocation occurred. Ping-Ponder's fire rate is the fraction of *dialogues* with $\geq 1$ planner fire. The latency column is comparable across rows; the fire-rate column is not.

and the underlying LLM, and only the agent architecture differs. Table 2 reports the result. Ping-Ponder is at TTFR parity with Baseline in both languages ($\Delta = -50$ ms JA, $+5$ ms EN), while No-prefetch incurs a substantial TTFR overhead of $+1227$ ms in JA and $+661$ ms in EN. The TTFR cost of the Ponder call is therefore almost entirely absorbed by Ping-Ponder's speculative prefetching. Single-agent, which has no supervisor path, is essentially at chat-only latency in both languages ($1542 \pm 70$ ms JA, $999 \pm 68$ ms EN); we discuss its interpretation as a lower-bound control in §6.

#### 5.2.2 Planning Latency on the Fired Subset

TTFR measures how fast the *first* token arrives, while planning latency measures how fast task-level reasoning completes. We compute planning latency on the subset of turns on which the architecture emits a structured $\{\text{slots}, \text{proposals}\}$ update ($\text{proposals.length} > 0$). Baseline (Chat-Supervisor) returns natural-language content on such turns rather than structured proposals, so its fire event is instead the blocking supervisor invocation triggered by the `CALL_SUPERVISOR` marker; the fire-rate definitions therefore differ across conditions, but the latency column of Table 3 is directly comparable because both entries measure elapsed time until heavy-model output becomes available; only the Baseline blocks the user-facing agent while this occurs. Single-agent has no separate planner stage and is omitted. No-prefetch's planning latency is nearly identical to Ping-Ponder's, since the Ponder stage is unchanged between them, so we focus Table 3 on the headline Baseline-vs-Ping-Ponder comparison. We read the 31–32% reduction in this table as an *end-to-end* comparison against a representative blocking baseline, which also differs in output format and token budget (§4.5); it is not a clean isolation of scheduling alone. The clean isolation is the No-prefetch contrast. Because the Ponder stage is identical in that contrast, its effect does not appear in this planning-latency column but instead in the time-to-first-response parity of §5.2.1 and in the off-turn absorption of reasoning cost on non-planning turns (§5.2.3).

Table 3 shows that Ping-Ponder's planning arrives substantially faster than Baseline's blocking supervisor call in both tested languages. In JA, latency falls from 1376 ms to 950 ms ($-31\%$), and in EN from 1029 ms to 703 ms ($-32\%$). The two languages agree to within a single percentage point, which suggests that the scheduling gain is not a quirk of either language but a property of the architecture itself; whether it generalises to languages beyond the two we test is left to future work.

#### 5.2.3 Mechanism: Turn-Level Latency Conditional on Planning

Figure 2 provides visual corroboration. In JA (panel A), Baseline and Ping-Ponder both hover around 1–2 s on ordinary turns, but Baseline spikes to 6.5 s at turn 12 where planning is needed, while Ping-Ponder stays near 1 s at the same turn; the aggregate panel (B) confirms that the planning-turn gap survives averaging across dialogues. EN (panel C) shows the same shape with a clear Baseline spike to 4.9 s at turn 10 and Ping-Ponder consistently around 1 s; the aggregate panel (D) shows that the gap holds in the mean even though SpokenWOZ is a short-dialogue corpus in which Ping-Ponder's planner fires on only 13.2% of dialogues versus 75.8% on JA (Table 3). On non-planning turns, Ping-Ponder's heavy-model cost is absorbed off-turn by speculative prefetching and is therefore

| Lang | Fire status | Baseline (ms) | Ping-Ponder (ms) | Δ (PP vs. Baseline, ms) | Δ (PP vs. Baseline, %) |
|---|---|---|---|---|---|
| JA | Fired (planning needed) | 3257 ±103 | **2122 ±150** | -1135 | -35% |
| JA | Not fired | 2106 ±27 | **2001 ±33** | -105 | -5% |
| EN | Fired (planning needed) | 2342 ±83 | **1176 ±206** | -1166 | -50% |
| EN | Not fired | 1208 ±63 | **1166 ±26** | -42 | -3% |

Table 4: Turn-level latency partitioned (per turn) by whether Ping-Ponder's planning path fired (mean ± 95 % CI, ms). Δ columns are Ping-Ponder's reduction vs. Baseline.

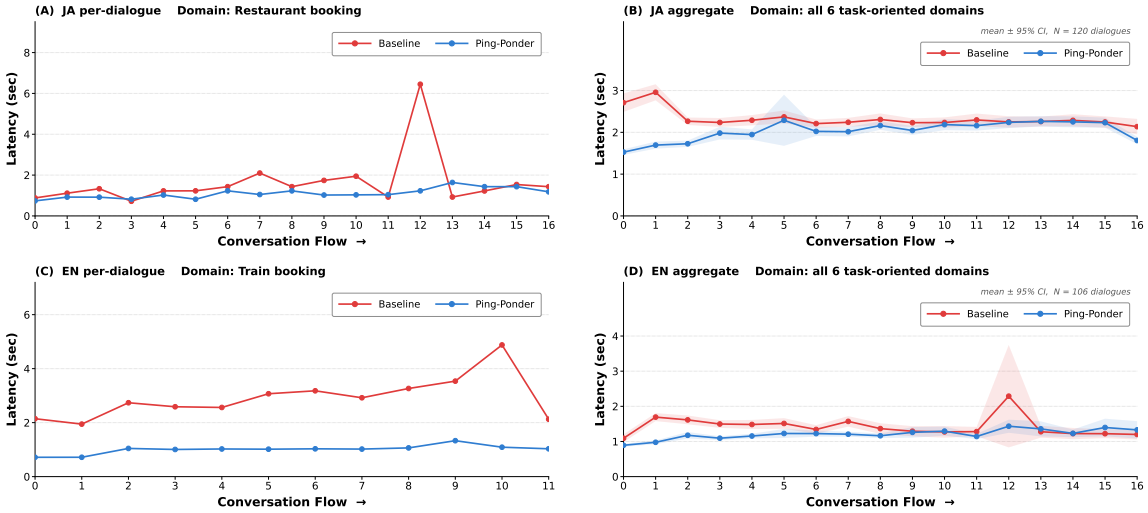


Figure 2: Turn-by-turn response latency over a replayed dialogue. Top row: JA (left: a representative restaurant-booking dialogue; right: mean ± 95% CI across all 120 JA dialogues). Bottom row: EN (left: a representative train-booking dialogue; right: mean ± 95% CI across 106 EN dialogues). Baseline is the blocking Chat-Supervisor condition; Ping-Ponder is the full architecture with speculative prefetch. Turn-level phase-conditional summaries are reported numerically in Table 4.

not visible on the response-time axis, which is exactly the architectural property the design targets.

Table 2 is an average over all turns, many of which do not require planning. Our mechanism claim is that speculative prefetch absorbs supervisor cost only when planning is needed, so the gain should concentrate on planning-required turns and leave non-planning turns essentially unchanged. We partition all turns by whether Ping-Ponder's internal planning fire-decision was triggered on that turn, and compute turn-level latency for both Baseline and Ping-Ponder on the same turn indices. Table 4 shows that the fired-turn reduction is $\approx 7\times$ the non-fired reduction in JA and $\approx 17\times$ in EN. The architecture does not make every turn faster, but rather makes planning-required turns dramatically faster while leaving simple turns essentially unchanged. This effect transfers across the two tested languages despite substantially different fire rates (JA 75.8%, EN 13.2%), suggesting that the scheduling gain is not a quirk of either language; whether it generalises beyond the two languages we test is left to future work. The fire-rate gap itself reflects corpus structure rather than architecture: SpokenWOZ consists of short, largely linear task-oriented dialogues in which few user turns require replanning, whereas our JA set contains longer, more ambiguous dialogues that more frequently invoke the supervisor.

### 5.3 No Degradation Under a Reference-Based LLM Judge

Latency reductions would be uninteresting if they came at the cost of response quality. Using the disfluency-aware reference-based LLM-judge evaluation of §4.4 (`gpt-4.1`, temperature 0, JSON mode; v2 (disfluency-aware) rubric; 8,765 judged turns JA, 5,067 turns EN), we find no evidence of degraded quality relative to the blocking Baseline on any of the three scored axes (Appropriateness, Faithfulness, Fluency) in either language. In JA the gap is directionally favourable to Ping-Ponder (+0.21 Appr., +0.15 Faith.); in EN the three structured variants cluster within judge noise on every

axis. These are judge means without confidence intervals, and we therefore frame the result as *absence of a speed-quality trade-off*, not as a confirmed quality win. A blind human validation on a stratified sample (Appendix A.1) finds moderate human-judge agreement on faithfulness (Spearman $\rho = 0.66$, within-one agreement 93–95%) and shows that the judge under-penalises mid-dialogue re-greetings on appropriateness; the two automatic judges agree with each other more than with the human, indicating shared blind spots. Full per-language tables and discussion are given in Appendices A and A.1.

## 6 Discussion

The latency reductions in §5.1 and the replay-scale TTFR gap of §5.2 reflect a structural change in *when* reasoning can begin rather than a faster model. Synchronous dual-agent systems treat a turn as atomic, so all reasoning is scheduled after conversation management has completed, whereas a shared, incrementally written belief state couples Ponder's schedule to *information availability*. As soon as a slot is written, Ponder can use it, and as soon as a plan fragment stabilises, Ping can verbalise it. This lens also sharpens the relationship to recent fast-agent/slow-agent proposals (Christakopoulou et al., 2024), which differ less in their top-level decomposition than in how the two sides exchange information. Acting on partial slots does expose two residual failure modes. One is systematic ASR errors that are re-asserted with growing apparent evidence, and the other is races in which Ponder publishes an over-confident fragment before re-reading Ping's latest write. Per-field version counters mitigate the second, while the first is an ASR-level problem beyond our scope. Finally, the narrow Fluency margin held by Single-agent (Appendix A, $+0.03$–$0.04$ points in both languages) is consistent with the absence of a second agent whose injections can perturb surface continuity. A monolithic model produces slightly smoother surface output but loses the structured $\{\text{slots}, \text{proposals}\}$ planning state that Ping-Ponder needs to drive tool calls and external APIs. We read this gap as the expected cost of coordinating a planner, rather than as a structural disadvantage. More broadly, Single-agent reaches quality parity with the planner-equipped conditions, with a marginal edge on English, which reflects a limitation of the metric rather than the system: the reference-based judge rewards fluent, on-topic text but does not measure task grounding, knowledge retrieval, or tool use, which is precisely what Single-agent omits and Ponder provides. We thus read this parity as evidence that the planner does not degrade surface quality, not that it is unnecessary; task-success and groundedness evaluation is left to future work.

## 7 Conclusion

We presented Ping-Ponder, a concurrent dual-agent architecture for spoken dialogue in which a real-time conversational agent and a deliberative reasoning agent run as independent processes and coordinate through a bidirectional, incrementally written shared belief state. Across a paired live case study and a large-scale replay on two spoken task-oriented corpora in English and Japanese, Ping-Ponder reduces planning-phase latency by roughly an order of magnitude in the single-operator case study and by 31–32% on planning-required turns in the large-scale replay study, compared to a synchronous Chat-Supervisor baseline, while maintaining time-to-first-response parity in both tested languages and showing no evidence of degraded response quality on a reference-based LLM judge. Our main contribution is not any single component but the observation that the bottleneck in LLM-based spoken assistants is not model speed but scheduling, and that a small amount of structure, namely a shared per-field versioned belief state with partitioned write authority, is enough to decouple the two.

Several directions follow naturally. First, edge deployment: nothing in the protocol requires either agent to be large, and distilled speech and reasoning models should slot into the same shared-state contract, making on-device operation plausible. Second, persistent personalisation: extending the belief state from a per-dialogue scratchpad to a long-lived user model would let Ponder amortise background reasoning across sessions. Third, multi-agent extensions: the shared-state primitive is not specific to two writers, and scaling to, e.g., a third *critic* process that monitors belief-state coherence is a straightforward next step. We release the replay harness outputs (per-turn JSONLs, judge prompts, and per-turn scores) to help make concurrent-dialogue evaluation easier to reproduce; the underlying Japanese corpus is slated for public release after annotation (see Ethics Statement).

## Limitations

Our study has several limitations that temper how broadly its conclusions should be read.

**Task breadth.** We evaluate on two spoken task-oriented corpora: an internally collected Japanese corpus spanning 6 domains (accommodation consultation, restaurant and dining reservation, taxi and private-transport booking, rail and long-distance train booking, medical appointment booking, and food and culinary-tour booking), and a single-domain subset of SpokenWOZ covering hotel, restaurant, taxi, train, attraction, and hospital. All 12 domains share a structure in which the bottleneck is external knowledge-base reasoning after several intent slots have been elicited, which is the regime Ping-Ponder is designed to help. We do not evaluate on open-domain chat, emotional support, or scenarios where the slow component's work is negligible; in the latter case we expect Ping-Ponder to match, not beat, a single-agent baseline.

**User source and LLM-based judging.** Our large-scale replay (§4.2) injects the *logged* user side of each reference dialogue turn by turn, so the "user" is a recorded human speaker rather than a simulated one. This avoids LLM-user-LLM-agent feedback loops but inherits the distributional properties of the source corpora: spontaneity and code-switching in the JA corpus, shorter and more operationally narrow turns in SpokenWOZ. Because each turn replays the logged user utterance and does not condition on the system's own responses from previous turns, the protocol isolates per-turn architectural latency under matched inputs but does not capture the multi-turn interaction effects (e.g. error recovery and divergent dialogue trajectories) that a fully interactive evaluation would surface; we therefore read the replay results as evidence about per-turn scheduling rather than about end-to-end multi-turn task success. Response quality is scored by an LLM judge against the gold agent turn (CH2 in JA; system side in SpokenWOZ), which may systematically favour plans that are stylistically similar to its own outputs. A human-subjects study of perceived responsiveness and plan quality is needed before strong usability claims can be made.

**Languages and voices.** We test only English and Japanese, using the voices supplied by the underlying Realtime API. Spoken-dialogue latency profiles depend on the TTS system's first-audio-chunk latency, which can vary substantially across voices and languages; our absolute numbers should therefore be read as characteristic rather than universal.

**Dependency on proprietary components.** Ping and Ponder are instantiated with the OpenAI Realtime API and GPT-4.1-family models, respectively. Both are closed-source and subject to provider-side changes in latency, availability, and output distribution; indeed, the beta Realtime interface used during our evaluation was subsequently retired in favour of a generally-available successor. Because coordination is mediated only through the shared belief state, the protocol is in principle agnostic to these choices: the fast side could instead be served by a vendor-neutral orchestration stack (e.g. Pipecat or LiveKit Agents) or an open full-duplex model such as Moshi (Défossez et al., 2024), and the slow side by any tool-capable LLM. Our claim that *any* pair of fast and slow models can be coordinated through the shared-state protocol should therefore be reproducible with open components, but we have not yet empirically verified this.

**Error propagation from ASR.** As discussed in Section 6, the shared belief state is vulnerable to systematically mistranscribed slot values because each individual write is internally consistent. A more robust design would attach ASR confidence scores and a disagreement-resolution policy to the belief state, which we leave to future work.

## Ethics Statement

**Dataset collection and consent.** The Japanese spoken task-oriented dialogue corpus used in this work was assembled through a commissioned professional recording for research and development purposes. All speakers provided informed consent at the time of collection to the use of the recordings for research, including publication of aggregate statistics and short anonymised examples. No personally identifying information (real names, addresses, phone numbers, patient records, or real reservation identifiers) appears in the dialogues; all user profiles and booking targets are fictional role-play scenarios built from scripted scenario briefs. Full composition and collection details are given in Appendix C.

**Health-related content.** The "medical" subdomain in our Japanese corpus is restricted to appointment-booking and pre-visit logistics (e.g.

scheduling a test or a clinic visit) and contains no clinical diagnosis, treatment advice, drug dosing, or real patient history. We do not evaluate or deploy the system on tasks that offer medical advice to end users.

**Public corpora.** SpokenWOZ (Si et al., 2023) is used under its existing public release terms.

**Data release scope.** We intend to release the Japanese corpus publicly. Its raw assets are complete, comprising the two-channel audio, the scenario design scripts, and the time-aligned manual transcripts; what is not yet finished is the ontology layer (domain, dialogue-act, and slot annotations), which currently covers only a subset of dialogues (Appendix C). The corpus is therefore *not* withheld for sensitivity or licensing reasons; all scenarios are fictional role-plays with no personally identifying content. Its full release is instead deferred until this annotation is complete across the corpus, so that we ship a consistently annotated dataset rather than a partial one. Representative user-side excerpts are included in Appendix C (Figure A2) to document the corpus structure and annotation conventions; these excerpts are not themselves a release of the underlying corpus. Independently of the corpus timeline, we release alongside this paper the replay-harness output (per-turn JSONLs, judge prompts, and per-turn scores for the replay study of §5.2), which is sufficient to reproduce the numeric claims of this paper without access to the source audio or transcripts.

**Model and compute.** Ping and Ponder are instantiated with closed-source commercial APIs, and the reference-based LLM judge is likewise a closed-source model. Latency measurements therefore reflect the state of those services during our evaluation window and may change with provider-side updates. We release the per-turn numeric scores that drive Tables A1 and A2, but *not* the raw judge rationales or full judge response payloads, in line with provider terms for redistribution of model outputs.

**Risk of misuse.** Ping-Ponder is a general scheduling pattern for dual-agent dialogue systems and does not introduce new capabilities in speech synthesis, persuasion, or content generation beyond those of its component models. We therefore identify no dual-use risks that are specific to this work rather than to the underlying foundation models.

## References


James F. Allen, Lenhart K. Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel G. Martin, Bradford W. Miller, Massimo Poesio, and David R. Traum. 1995. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):7–48.

Konstantina Christakopoulou, Shibl Mourad, and Maja Matarić. 2024. Agents thinking fast and slow: A Talker-Reasoner architecture. *arXiv preprint arXiv:2410.08328*.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. 2024. Moshi: A speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*.

David DeVault, Kenji Sagae, and David Traum. 2009. Can I finish? learning when to respond to incremental interpretation results in interactive dialogue. In *Proceedings of the SIGDIAL 2009 Conference*, pages 11–20.

Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. 1980. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys*, 12(2):213–253.

Haris Gulzar, Monikka Roslianna Busto, Akiko Masaki, Takeharu Eda, and Ryo Masumura. 2025. Leveraging LLMs for written to spoken style data transformation to enhance spoken dialog state tracking. In *Proceedings of Interspeech 2025*, pages 1743–1747.

Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788.

Tu Anh Nguyen, Eugene Kharitonov, Jade Copet, Yossi Adi, Wei-Ning Hsu, Ali Elkahky, Paden Tomasello, Robin Algayres, Benoît Sagot, Abdelrahman Mohamed, and Emmanuel Dupoux. 2023. Generative spoken dialogue language modeling. *Transactions of the Association for Computational Linguistics*, 11:250–266.

H. Penny Nii. 1986a. Blackboard systems, part one: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, 7(2):38–53.

H. Penny Nii. 1986b. Blackboard systems, part two: Blackboard application systems, blackboard systems from a knowledge engineering perspective. *AI Magazine*, 7(3):82–106.

OpenAI. 2024. OpenAI Realtime Agents. Agentic Pattern 1: Chat-Supervisor.

David Schlangen and Gabriel Skantze. 2011. A general, abstract model of incremental dialogue processing. *Dialogue & Discourse*, 2(1):83–111.

Shuzheng Si, Wentao Ma, Haoyu Gao, Yuchuan Wu, Ting-En Lin, Yinpei Dai, Hangyu Li, Rui Yan, Fei Huang, and Yongbin Li. 2023. SpokenWOZ: A large-scale speech-text benchmark for spoken task-oriented dialogue agents. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Gabriel Skantze. 2017. Towards a general, continuous model of turn-taking in spoken dialogue using LSTM recurrent neural networks. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 220–230.

Gabriel Skantze and Anna Hjalmarsson. 2013. Towards incremental speech generation in conversational systems. *Computer Speech & Language*, 27(1):243–262.

Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

# A Response Quality: Reference-Based LLM-Judge

Our reference-based judge uses the gold human-authored reply available for each dialogue turn. The JA reference is our in-house human reply, and the EN reference is the CH2 human-recorded agent track of SpokenWOZ. Each system response is scored on three 0–3 axes using the disfluency-aware v2 (disfluency-aware) rubric. *Appropriateness* (Appr.) asks whether the reply addresses the user's last utterance given the dialogue history. *Faithfulness* (Faith.) asks whether the reply is consistent with the gold reference's informational content. *Fluency* (Flu.) asks whether the reply is natural as spoken output.

The judge is `gpt-4.1` at temperature 0 in JSON mode. Per-turn scores are averaged across domains and conditions and reported in Tables A1 and A2. The captions discuss the main patterns, and here we restate the headline. In JA, Ping-Ponder has the highest mean on all three axes against Baseline, has the highest Appropriateness mean on 6/6 domains and the highest Faithfulness mean on 5/6, and lies within 0.03 points of the highest-mean ablation on Fluency. In EN, the three structured variants cluster within a narrow range (Appr. 2.40–2.45, Faith. 1.75–1.79) and each has a higher mean than Baseline by a comparable margin. We report judge means rather than win/loss verdicts and do not compute confidence intervals on per-condition quality; we therefore treat these patterns as descriptive, and interpret them as showing no evidence of degraded response quality rather than as statistically certified improvements. Single-agent's small Fluency lead in both languages is discussed in §6.

| Condition | Appr. (0–3) | Faith. (0–3) | Flu. (0–3) |
|---|---|---|---|
| Baseline | 2.52 | 1.97 | 2.89 |
| No-prefetch | 2.64 | 2.09 | 2.89 |
| Ping-Ponder | **2.73** | **2.12** | 2.96 |
| Single-agent | 2.65 | 2.09 | **2.99** |

Table A1: Japanese reference-based judge scores (mean over six domains, `gpt-4.1` at $T$=0, JSON mode, v2 (disfluency-aware) rubric). Total judged turns 8,765. Faithfulness was recomputed after correcting a reference-alignment issue (§5.3); appropriateness and fluency are reference-independent and unchanged.

| Condition | Appr. (0–3) | Faith. (0–3) | Flu. (0–3) |
|---|---|---|---|
| Baseline | 2.21 | 1.60 | 2.96 |
| No-prefetch | 2.44 | **1.79** | 2.94 |
| Ping-Ponder | 2.40 | 1.75 | 2.96 |
| Single-agent | **2.45** | 1.78 | **3.00** |

Table A2: English reference-based judge scores, computed against the SpokenWOZ CH2 gold reply (`gpt-4.1`, $T$=0, JSON mode, v2 (disfluency-aware) rubric). Total judged turns 5,067.

**Japanese reference alignment.** An earlier version of the judge harness aligned each candidate turn to the agent reference by position. Because the Japanese dialogues open with an agent greeting, this shifted every Japanese gold reference by one turn; we corrected the alignment so that each user turn is matched to the agent turn that follows it, and recomputed Japanese faithfulness. Appropriateness and fluency are reference-independent and unchanged, and English is unaffected, since its dialogues open with the user. The corrected faithfulness preserves the original ordering across conditions.

## A.1 Human Validation of the Reference-Based Judge

**Protocol.** To assess whether the automatic judge tracks human judgement, one author re-annotated a stratified random sample of 120 replayed turns (60 Japanese, 60 English; balanced across the four con-

ditions, 15 turns per language each) on the same 0–3 axes (appropriateness, faithfulness, fluency) under the disfluency-aware rubric of §4.4. **Annotation was blind to condition**: the system identity (Baseline / No-prefetch / Ping-Ponder / Single-agent) was hidden during scoring, so scores could not be biased toward any architecture. Appropriateness and fluency are reference-free and were scored on all turns; faithfulness was scored only on turns with a usable human reference. We additionally re-scored the same turns with a different-vendor judge (Claude Opus 4.8) under an identical rubric.

**Results.** Table A3 reports Krippendorff's $\alpha$, human-judge Spearman $\rho$ (with within-one-point agreement), and human-Claude $\rho$. On *faithfulness*, which underlies our quality claim, human and judge agree moderately well ($\rho = 0.66$ overall, 0.75 on English; within-one agreement 93–95%), supporting a cautious reading of the no-degradation result. *Appropriateness* agreement is lower, and lowest in Japanese ($\rho = 0.24$), where the disagreement concentrates on *mid-dialogue re-greetings*, in which the system re-opens with a greeting in the middle of a conversation. The human annotator penalises these but the automatic judge does not. *Fluency* is near-ceiling, with almost all responses scoring 3, so rank correlation is uninformative, although within-one agreement is 93%.

**Considerations.** Notably, the two automatic judges (`gpt-4.1` and Claude) agree with *each other* more strongly (faithfulness $\rho = 0.82$) than either agrees with the human annotator. We read this as evidence that LLM judges share systematic blind spots, most visibly the under-penalisation of conversational-flow violations such as mid-dialogue re-greetings, and that human validation, rather than a second automatic judge alone, is what substantiates quality claims. We therefore present these results as evidence that the latency gains do not obviously harm reference-based response quality, *not* as a confirmed human-perceived quality win. Limitations of this validation include a single blind annotator, the fluency ceiling, and a reduced sample for Japanese faithfulness against the realignment-corrected judge, for which only $N$=16 turns were available because the full re-judge was bounded by API quota; we therefore rely instead on the human-Claude comparison ($N$=60, within-one agreement 97%).

| Axis | Lang | $\alpha$ | Human-Judge $\rho$ | Human-Claude $\rho$ |
|---|---|---|---|---|
| Appr. | EN | 0.70 | 0.70 (w1 97%) | 0.76 |
| Appr. | JA | 0.40 | 0.24 (w1 80%) | 0.63 |
| Faith. | EN | 0.69 | 0.75 (w1 95%) | 0.58 |
| Faith. | JA | 0.58 | 0.34$^{\dagger}$ | 0.58 (w1 97%) |
| Flu. | EN+JA | n/a | near-ceiling (w1 93%) | |

Table A3: Human validation of the reference-based judge on a blind, stratified sample (120 turns). $\alpha$ = Krippendorff's $\alpha$; $\rho$ = Spearman; w1 = within-one-point agreement. Faithfulness is over usable-reference turns. $^{\dagger}$JA faithfulness vs. the realignment-corrected `gpt-4.1` is over $N$=16 re-judged turns; we rely on the human-Claude comparison ($N$=60) for JA faithfulness.

| | Human | | | `gpt-4.1` | | |
|---|---|---|---|---|---|---|
| Condition | Appr | Faith | Flu | Appr | Faith | Flu |
| Baseline | 1.20 | 1.32 | 1.80 | 2.13 | 1.57 | 2.93 |
| No-prefetch | 1.83 | 2.32 | 2.53 | 2.53 | 1.69 | 2.97 |
| Ping-Ponder | 2.57 | 2.32 | 2.83 | 2.57 | 1.75 | 3.00 |
| Single-agent | 2.70 | 2.48 | 2.97 | 2.57 | 2.29 | 3.00 |

Table A4: Per-condition mean scores on the blind human-validation sample (120 turns; appropriateness/fluency over all turns, faithfulness over usable-reference turns), with human and automatic (`gpt-4.1`) means side by side. The human ranking matches the automatic one (Baseline lowest; Ping-Ponder on par with Single-agent), with sharper separation and a much harsher human rating of the blocking Baseline.

**Per-condition human ratings.** Because annotation was per response, the blind scores also let us compare the *conditions* under human judgement (Table A4). The human ranking reproduces the automatic one, with the blocking Baseline rated lowest and the structured conditions higher and Ping-Ponder on par with Single-agent. The separation, however, is sharper, with the human annotator penalising the Baseline far more than the judge does (e.g. appropriateness 1.20 vs. 2.13 on this sample), reflecting the same under-penalisation of stalls and mid-dialogue re-greetings noted above. This corroborates the no-degradation reading with human judgement and, if anything, suggests the automatic judge *understates* the quality cost of the blocking baseline. These per-condition means are over a small blind sample (30 turns per condition; faithfulness over usable-reference turns) and are indicative rather than definitive.

| | Ping-Ponder (ours) | No-prefetch | Baseline | Single-agent |
|---|---|---|---|---|
| Chat agent | Fast chat (120 tok, $T$=0.3) | Fast chat (same) | Blocking chat (100 tok, $T$=0.3) + `CALL_SUPERVISOR` marker | Chat only (1–3 sentences) |
| Supervisor | fastSupervisor (500 tok, $T$=0.3, JSON mode) | fastSupervisor (same) | Blocking supervisor (220 tok, $T$=0.4) | n/a |
| Coordination | Background prefetch | Serial, before chat | Serial, after chat (only on marker) | n/a |
| Supervisor output | Structured {slots, proposals} | Structured (same) | Natural language | n/a |

Table A5: The four architectural conditions. All share the same Ping chat agent family and domain prompts.

# B Additional Details

## B.1 Per-Condition Configuration

Table A5 lists the token budgets, temperatures, coordination policies, and supervisor output formats for each condition. Three points of design intent are worth noting. First, the fast-path supervisor used by Ping-Ponder and No-prefetch is given a larger token budget (500) than the Baseline supervisor (220), because it must return structured $\{\text{slots}, \text{proposals}\}$ JSON rather than a single natural-language utterance. This makes the comparison between Ping-Ponder and Baseline conservative on planning latency. Second, Baseline's supervisor emits a natural-language reply that the chat agent reads verbatim, whereas Ping-Ponder's fast-path supervisor emits a structured $\{\text{slots}, \text{proposals}\}$ update that is consumed by the next chat turn. Both paths produce the agent's next-turn output; they differ only in representation and downstream consumption, and are measured under the single planning-phase latency definition of §4.2 (see also Appendix B.2). Third, Single-agent has no supervisor stage at all and therefore produces no entries in the planning-latency or fire-rate columns. It is included as a lower bound on latency and an upper bound on surface fluency, not as a direct architectural competitor.

## B.2 Unified Planning-Phase Latency Definition

This appendix supplements the *planning-phase latency* definition in §4.2 by making the correspondence between Baseline and Ping-Ponder explicit, and by tying the replay definition back to the interactive values reported in the Abstract and Table 1.

Table A6 contrasts the two architectures at the level of the heavy-model call. The inputs are the same (conversation history plus the relevant context from the latest user turn). The downstream consumption differs, but the *measurement* is identical: in both cases we measure wall-clock time from heavy-model call initiation to output availability.

**Unified definition.** From Table A6 it is clear that, despite the difference in output representation, the planning phase is defined identically in both architectures:

> *Planning-phase latency* is the wall-clock duration from the initiation of a heavy-model call to the availability of its output, irrespective of whether the output is natural language or structured JSON.

Under this single definition, the Abstract values decompose cleanly:

- Baseline 7.1 s (EN) and 7.3 s (JA) = time from `CALL_SUPERVISOR` escalation to availability of the supervisor's natural-language reply;
- Ping-Ponder 0.5 s (EN) and 1.0 s (JA) = time from fastSupervisor kickoff to availability of the structured $\{\text{slots}, \text{proposals}\}$ update.

The large-scale replay study in §5.2 applies the same definition to every heavy-model call that fires in each condition. Table 3 reports planning-phase latency on the fired subset, and Figure 2 reports the turn-level response-time trajectory under this definition for a representative dialogue and across the aggregate; phase-conditional summaries are reported numerically in Table 4.

## B.3 Dialogue Example for the Interactive Session

Figure A1 shows a representative trace of the paired interactive session reported in §5.1 for the travel-

| | **Baseline (Chat-Supervisor)** | **Ping-Ponder (ours)** |
|---|---|---|
| Heavy-model call | Blocking supervisor, triggered by `CALL_SUPERVISOR` marker emitted by the chat agent | Background fastSupervisor, prefetched concurrently with the chat agent |
| Input | Conversation history + `CALL_SUPERVISOR` context + latest user turn | Conversation history + current belief state + latest user turn |
| Output representation | Natural-language reply (2–4 sentences) | Structured JSON $\{\text{slots}, \text{proposals}\}$ |
| Downstream consumption | Read *verbatim* by the chat agent as the next-turn reply | Injected into the chat agent's system prompt as updated belief state for the next turn |
| Blocks chat agent | Yes, since the chat agent must wait for the reply before speaking | No, since the chat agent speaks from the belief state already in hand |
| Planning-phase latency (measured) | Call initiation → natural-language reply available | Call initiation → structured JSON parsed and merged into state |
| Reported value (interactive, EN/JA) | 7.1 s / 7.3 s (Table 1) | 0.5 s / 1.0 s (Table 1) |

Table A6: Heavy-model call in Baseline vs. Ping-Ponder. The two architectures differ in how the call is triggered (marker-driven vs. prefetched), in the output representation (natural language vs. structured JSON), and in how the output is consumed downstream, but they are measured under a *single* planning-phase latency definition: wall-clock time from heavy-model call initiation to output availability.

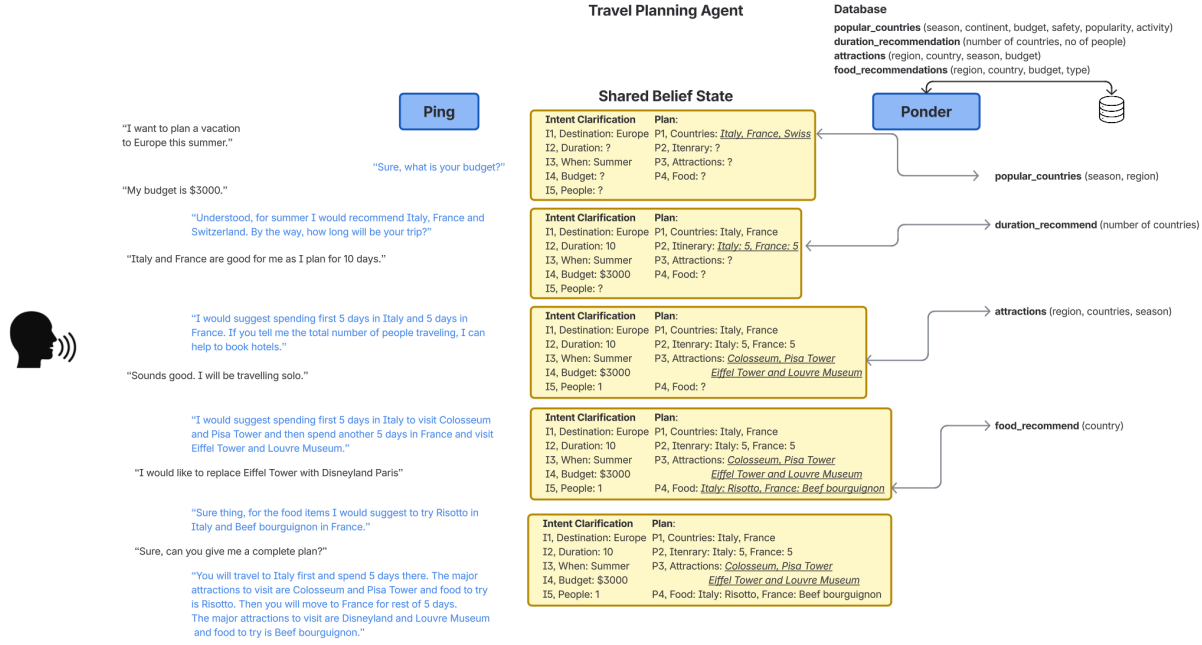


Figure A1: Representative trace of the interactive travel-planning session (§5.1). Each row is one user turn; Ping (left) and Ponder (right) run concurrently.

planning domain. The left column is the user's spoken turn, the middle column is the Shared Belief State ($\mathcal{B}$) after each turn as seen by both agents, and the right column is the set of external knowledge sources that Ponder queries to populate the Plan fields.

Three behaviours visible in the trace are worth highlighting. First, intent clarification and planning overlap. As soon as destination=`Europe`, season=`Summer`, and num_people=1 are in $\mathcal{B}$, Ponder already issues a `popular_countries` query and writes a candidate country list back, so that when the user next volunteers a budget Ping can verbalise the shortlist without blocking on a fresh supervisor call. Second, the belief state grows monotonically but plan fields are versioned. When the user later narrows the trip to `Italy/France` and supplies a duration (`5 days each`), Ponder discards the earlier country-level draft and commits a per-country day allocation, which Ping picks up on its next turn. Third, the hand-off is structured rather than natural-language. Every plan fragment Ping verbalises (*"I'd suggest spending first 5 days in Italy and 5 days in France..."*) is sourced from

the Plan column of $\mathcal{B}$, not from free-form supervisor output, which is what makes the fast-path supervisor of §4.3 a drop-in replacement for the blocking Chat-Supervisor of the Baseline. The complete dialogue spans eight user turns; a single final turn produces a consolidated itinerary covering destinations, duration, attractions (Colosseum, Pisa Tower, Eiffel Tower, Louvre Museum) and food (*Pizza*, *Risotto*, *Beef bourguignon*), illustrating how the structured $\{\text{slots}, \text{proposals}\}$ contract accumulates into a coherent plan without ever forcing the user to wait on a blocking reasoning call.

### B.4 Worked Belief-State Example

For the travel-planning domain used in our main experiments, the intent-slot section $S$ is instantiated with the slot names {destination, duration, season, num_people, budget, attractions, food}, each with a typed value domain and a confidence score in $[0, 1]$. An early-turn snapshot of $\mathcal{B}$ in a representative dialogue looks as follows:

| Slot | Value | Conf. |
|---|---|---|
| destination | `"Europe"` | 0.95 |
| season | `"Summer"` | 0.92 |
| num_people | 1 | 0.87 |
| budget | `"$3000"` | 0.81 |
| duration | ? | n/a |

Planning status is `partial`, and the plan-sharing queue already holds a candidate country set {`Italy`, `France`, `Switzerland`} that Ping has been cleared to verbalise while Ponder continues to populate the attractions and food subsections of the plan draft.

## C Japanese Corpus: Collection and Composition

The Japanese spoken task-oriented dialogue corpus described in §4.1 was assembled through a commissioned professional recording, carried out to a written specification prepared for this project. This appendix gives the corpus's composition and the provenance of the information used for replay.

**Scope and scale.** The full corpus comprises 1,200 dialogues organised as 6 domains × 5 subcategories × 40 dialogues per sub-category. The 6 domains, summarised in Table A7, are (A) accommodation consultation, (B) restaurant and dining reservation, (C) taxi and private-transport booking, (D) rail and long-distance train booking, (E) medical appointment booking, and (F) food and culinary-tour booking. The medical domain (E) is restricted to appointment scheduling and previsit logistics; no sub-category contains clinical advice, diagnosis, or real patient history. Subcategories within a domain vary the speaker profile, booking target, or occasion while keeping the task type fixed; for instance, sub-categories under (A) range from corporate group trips to individual leisure stays, those under (B) cover anniversary dinners and event-venue consultation, those under (D) cover sleeper-train and observation-train itineraries, and those under (F) cover regional gourmet tours, seasonal-ingredient tours, and food-walking itineraries. For the present study we sample one sub-category per domain and 20 dialogues per sub-category, yielding $6 \times 20 = 120$ evaluation dialogues per condition (§4.1).

| ID | Domain |
|---|---|
| A | Accommodation consultation |
| B | Restaurant and dining reservation |
| C | Taxi and private-transport booking |
| D | Rail and long-distance train booking |
| E | Medical appointment booking |
| F | Food and culinary-tour booking |

Table A7: The 6 task domains of the internal Japanese spoken TOD corpus. Each domain contains 5 subcategories × 40 dialogues. Domain-level labels are intentionally coarser than individual sub-category labels; see the prose of this section for representative sub-category examples.

**Recording and assets.** Each dialogue is provided as a two-channel 16 kHz PCM-16 waveform, with the user on one channel (CH1) and the agent on the other (CH2). Dialogues are produced in two stages: a scenario-level *design intent* script specifies the user's goals, constraints, and planned turn structure; a pool of 39 professional voice actors then performs the dialogues, with speakers rotated across domains and sub-categories to diversify voice profile, and the recording is manually transcribed with time-aligned turns. For each dialogue we therefore have (a) the designed script, (b) the actual transcript produced by time-aligned manual transcription of the recording, and (c) ontology annotations on a subset, identifying the domain, dialogue acts, and slot fillers used for downstream analysis. Replay experiments consume the actual transcript stream: CH1 user-side turns are played into the agent under evaluation, and CH2 gold agent turns are used by the reference-based LLM judge (§4.4).

**Privacy and content.** Speaker identities, geographical references, reservation identifiers, personal names, and contact details in the dialogues are fictional or have been anonymised at script-design time. The medical sub-domain intentionally avoids any content that would constitute clinical advice, diagnosis, or real patient history; it contains only scheduling and logistics-oriented conversation (date/time selection, location, reason-for-visit category). Consent for use of the recordings in research, including publication of aggregate statistics and short anonymised examples, was obtained at the time of collection.

**Language and style.** All dialogues are in Japanese. The transcribed audio preserves spoken-language phenomena including fillers, self-repairs, back-channels, and mild code-switching, which is the source of the spoken-style difficulty discussed in §2 and referenced by our replay analysis of fired-turn latency. Figure A2 shows representative user-side excerpts, one per domain, with fillers and self-repairs highlighted.

**A: Accommodation consultation (A_01_010)**
はいえーとー、11月12日の木曜日から、1泊の予定です。
*yes, uhh, from Thursday Nov. 12, for one night.*
あとあの、ネットの口コミもー、できれば確認してから決めたいです。
*and, um, I'd also like to check the online reviews…before deciding, if possible.*

**B: Restaurant and dining reservation (B_01_010)**
あ、あの実は付き合って5周年でしてきちんとしたお店を予約したいです。
*oh, um actually it's our 5th anniversary, so I want to book a proper restaurant.*
あ、でも、あの、あんまりかたすぎず、それでもちゃんと特別感があると嬉しいです。
*oh, but, um, not too formal, but I'd still like it to feel special.*

**C: Taxi and private-transport booking (C_01_020)**
国内線で到着予定ですが、山口県内の天候にゃ、天候によるー、フライトの遅延にも柔軟に対応できると助かります。
*we arrive on a domestic flight; it'd help if you could flexibly handle delays due to the weath-, weather…in Yamaguchi.*
しゃすー、あっ、あのー、車種?については、こちらにふさわしいものを提案いただけますか?
*[false start], ah, um, as for the vehicle type, could you suggest something suitable?*

**D: Rail and long-distance train booking (D_01_010)**
あの、一度でいいから乗ってみたいって前から話してたんで、今回実現したくて。
*um, we'd long said we'd love to ride it just once, so this time I want to make it happen….*
実はあの、夫婦ともに足が弱くて、長距離の移動が少し不安なんです。
*actually um, we both have weak legs, so long-distance travel worries us a little.*

**E: Medical appointment booking (E_01_002)**
もしもし、あー、人間ドックについて相談したくて電話しました。正直、これまでほとんど検査を受けたことがなくて。
*hello, uhh, I'm calling to ask about a full medical check-up. honestly I've barely had any exams before.*
あー、自営業なのでできればコスパの良いところがいいですね。
*uhh, I'm self-employed, so somewhere good value would be best.*

**F: Food and culinary-tour booking (F_01_010)**
行き先は特に決めていなくて、あの、詳しい人にお勧めしてもらえたらなあって思ってます。
*um, we haven't fixed a destination; I'm hoping someone knowledgeable could recommend places….*
あ、ちなみに、ツアーで少し心配なのが歩きすぎて疲れないかなあって。
*oh, by the way, one worry about the tour is whether we'd get tired from walking too much….*

Figure A2: User-side transcript excerpts from the JA corpus, one per domain (A–F; see Table A7), each selected as a disfluency-dense turn pair from the evaluated subcategory. Turns are verbatim from the time-aligned manual transcription (CH1, user side); fillers, self-repairs, and trailing-off tokens are highlighted in red, with an English gloss below each turn. Dialogue IDs are given in parentheses. Speaker names and contact details appearing elsewhere in the dialogues are omitted; all scenarios in this corpus are fictional role-plays.