RelayS2S: Dual-Path Speculative Generation for Real-Time Dialogue

Anonymous ACL submission

Abstract

Real-time spoken dialogue systems face a fundamental tension between latency and response quality. End-to-end speech-to-speech (S2S) models respond immediately and naturally handle turn-taking, backchanneling, and interruption, but produce semantically weaker outputs. Cascaded pipelines (ASR → LLM) deliver stronger responses at the cost of latency that grows with model size. We present RelayS2S, a hybrid architecture that runs two paths in parallel upon turn detection. The fast path — a duplex S2S model — speculatively drafts a short response prefix that is streamed immediately to TTS for low-latency response onset, while continuing to monitor live audio events. The slow path — a cascaded ASR → LLM pipeline — generates a higher-quality continuation conditioned on the committed prefix, producing a seamless utterance. A lightweight learned verifier gates the handoff, committing the prefix when appropriate or falling back gracefully to the cascaded pipeline. With GPT-4.1 as the back-end, RelayS2S achieves P90 first-chunk latency of 81 ms on synthetic voice dialogues—down from 1,006 ms for the cascaded baseline—and reduces average first-chunk latency by roughly $2.8\times$ on real voice dialogues (269 ms vs. 748 ms), while retaining 99% of the cascaded pipeline's textual quality on average. Benefits widen as the slow-path model scales. Because the prefix handoff requires no architectural modification to either component, RelayS2S serves as a lightweight, drop-in addition to existing cascaded pipelines. Our code is publicly available at https://anonymous.4open.science/r/anonymous_relays2s-8338/

1 Introduction

Real-time spoken dialogue is among the most demanding settings for conversational AI: a system must listen, reason, and respond with the speed and fluidity of human conversation. Existing approaches face a central trade-off. End-to-end full-duplex speech-to-speech models (Zhang et al., 2023; Rubenstein et al., 2023; Défossez et al., 2024; Hu et al., 2025; Roy et al., 2026) can begin speaking almost immediately and natively support other turn-taking behaviors, but produce semantically weaker outputs. Cascaded pipelines (ASR → LLM) (Chen et al., 2025; Liu et al., 2025; Radford et al., 2023; Likhomanenko et al., 2025) deliver stronger responses, but their onset latency often exceeds the roughly 200 ms response gap typical of human conversations (Levinson and Torreira, 2015).

We build on a simple empirical observation: although S2S responses are weaker overall, their first few words are frequently usable. In our analysis, 82.5%–95% of five-word S2S prefixes are judged contextually appropriate—high enough to make speculative prefixing viable when paired with a learned verifier. Motivated by this asymmetry, we propose **RelayS2S**, a speculative hybrid architecture that runs two paths in parallel upon turn detection. The *fast path*—a duplex S2S model—drafts a short response prefix streamed immediately to TTS. The *slow path*—a cascaded ASR–LLM pipeline—generates a higher-quality continuation conditioned on the committed prefix. A lightweight learned verifier gates the handoff, committing the prefix when appropriate or falling back to the slow path alone.

A key design challenge is that the fast-path model must generate a prefix and continue monitoring live audio for interruptions. We resolve this with *forked speculative generation*: upon deciding to speak, the model forks into a main stream that tracks live audio for barge-in detection and a speculative stream that drafts the prefix at maximum speed. Unlike concurrent approaches that emit only short discourse connectives (Liu et al., 2026), rely on text-only pipelines that delay audible response despite early reasoning (Zou et al., 2026), or inject oracle LLM tokens into S2S transformers via joint training (Kuroki et al., 2025), RelayS2S generates substantive spoken prefix content while natively

supporting full-duplex conversational behaviors, and the prefix handoff requires no architectural modification to either component.

We train the fast-path model on 194K synthetic and real voice conversations (2,437 hours) and evaluate across four back-end configurations. With GPT-4.1 (OpenAI, 2025) as the back-end, RelayS2S achieves a P90 first-chunk latency of 81 ms on synthetic voice dialogues—down from 1,006 ms for the cascaded baseline—while preserving 99% of the cascaded system's average quality score. On the real voice dialogues, RelayS2S cuts average first-chunk latency by roughly $2.8\times$ (269 ms vs. 748 ms) at comparable quality retention. Our main contributions are:

- **Forked speculative generation**, which decouples fast response drafting from real-time audio monitoring within a single duplex S2S model, enabling immediate prefix generation. The resulting buffered multi-token prefix serves two downstream roles in the architecture: it provides the prefix verifier with a meaningful sequence of hidden states and calibration signals over which to score the draft, and it lets streaming TTS render from longer text segments, which generally improves synthesis quality (Bai et al., 2025).

- **Selective prefix handoff**, in which a lightweight learned verifier gates whether a speculative prefix is safe to commit. The verifier reuses already-computed decoder hidden states and calibration signals, adding negligible overhead while achieving 95% good-prefix throughput at an 8% fallback rate on synthetic dialogues.

- **RelayS2S**, a hybrid pipeline that combines a duplex S2S model with a cascaded back-end and scales monotonically with progress in either subsystem, cutting response latency substantially while preserving nearly all of the slow path's textual quality across four LLMs.

2 RelayS2S

2.1 Overview

Figure 1 illustrates the inference-time design. Upon detecting the end of the user's turn, RelayS2S launches two parallel paths. The fast path (§2.2) uses an S2S model to draft the opening few words of the response. A verifier (§2.4) decides whether to commit the draft to streaming TTS. If committed, the slow path (§2.5)—a cascaded ASR–LLM pipeline—generates the remainder conditioned on the committed prefix; otherwise it generates the full response as the system behaves like a conventional cascade. Both paths write to a shared response buffer that the streaming TTS module consumes continuously. Because a 5-word prefix yields roughly 2 seconds of speech, it creates a temporal buffer (the *relay margin*; §2.6) that lets the slow path produce its first continuation chunk before the prefix audio is exhausted, yielding a single uninterrupted utterance. While the prefix is playing, the fast-path model continues to monitor user speech so that any interruption stops playback immediately.

2.2 Fast Path: Duplex S2S Model

The fast path is a duplex speech-conditioned response model that maps streaming user speech directly to agent-side control symbols and text response tokens, which are then rendered by streaming TTS. It serves two roles in RelayS2S: enabling low-latency conversational behavior (early response initiation, backchanneling, interruption handling) and drafting an initial response prefix for handoff to the slow path. It comprises a streaming speech encoder, an LLM adapter, and an autoregressive LLM backbone (Figure 2).

Streaming speech encoder. We adopt the encoder of Wang et al. (2024): a conformer with chunked attention and limited left context that consumes audio causally. Log-Mel filterbank features are processed into higher-level acoustic representations, with internal subsampling producing a compact frame rate. Full hyperparameters are given in Appendix A.5.

LLM adapter. A two-stage causal 1D convolutional stack (convolution, normalization, GELU per stage) further compresses the temporal resolution and projects encoder states into the LLM embedding space. The combined encoder–adapter emits one representation every 160 ms, defining the online inference update interval.

LLM backbone. An autoregressive LLM consumes the adapted representations. Its vocabulary is extended with duplex control symbols: `[SIL]` (continue listening), `[BOC]` (begin backchannel), `[BOS]` (begin substantive response), `[STP]` (terminate ongoing response, e.g., on interruption), and

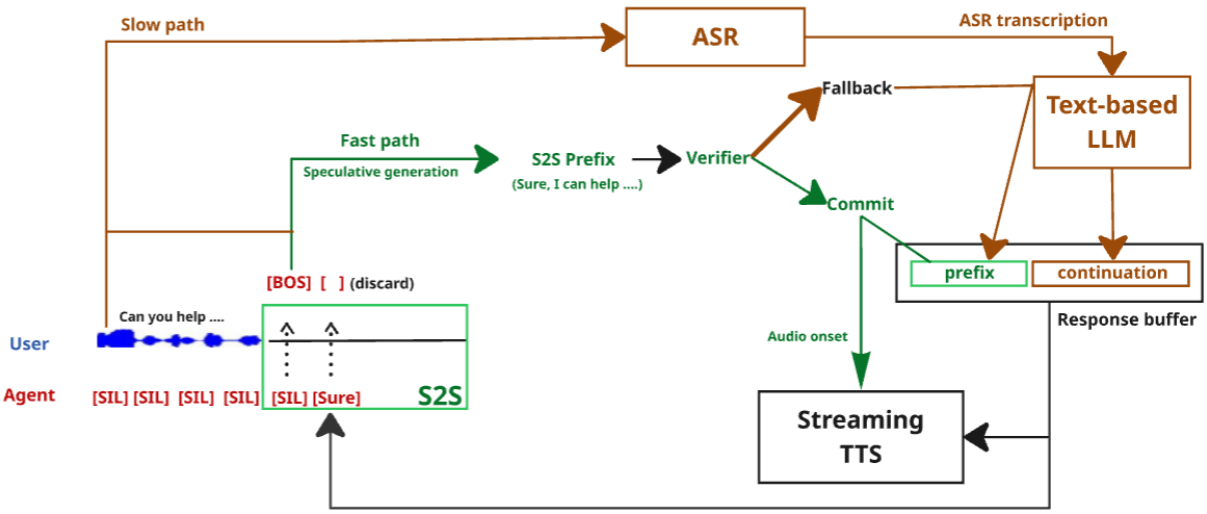

Figure 1: Inference-time architecture of RelayS2S, including the fast path (green) and the slow path (brown).

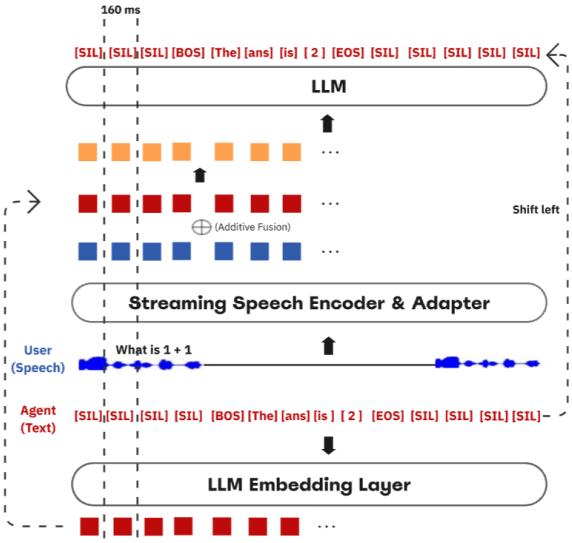

Figure 2: Fast-path duplex S2S model

[EOS] (complete response). Ordinary text tokens carry response content, letting a single decoder unify dialogue control and lexical generation. At each step, the user speech representation $\mathbf{e}_t^{\text{speech}}$ is fused with the aligned agent token embedding $\mathbf{e}_t^{\text{text}}$ by element-wise addition and passed through the LLM to predict the next agent token.

2.3 Forked Speculative Generation

When the fast-path model decides to initiate a response by emitting [BOS] or [BOC], it must generate a text prefix quickly enough for streaming TTS to begin synthesis with minimal delay. A naive approach—continuing to emit one token per 160 ms update tick and forwarding tokens incrementally to TTS—is too slow for natural interaction. High-quality streaming TTS typically requires a short buffered chunk of text before it can begin rendering (Bai et al., 2025). If the minimum chunk size is C words, the earliest audio onset under tick-synchronous generation would be delayed by at least $C \times 160$ ms. For realistic chunk sizes of 5–8 words, this corresponds to roughly 0.8–1.3 s, which exceeds the threshold for natural conversational flow. To avoid this bottleneck, we *fork* the fast path after a response initiation event into two branches: a **main online stream** and a **speculative generation stream**.

Main online stream. This branch remains the authoritative execution path. It continues processing live user speech every 160 ms and maintains the primary KV-cache, preserving the model's ability to detect barge-in and other real-time interaction events. The main stream does not generate the response tokens itself but consumes the tokens produced by the speculative stream as agent-side history, keeping its autoregressive state aligned with the response that the listener is already hearing.

Speculative stream. This branch is initialized from the same decoder state at timestep t_{init} but stops observing future user speech inputs, which in our implementation is realized by setting

$$\mathbf{e}_t^{\text{speech}} = \mathbf{0}, \qquad \forall t \geq t_{\text{init}} \tag{1}$$

Freed from the 160 ms speech-tick constraint, the speculative branch switches to free-running autoregressive decoding and generates a short text prefix at the model's maximum decoding speed. If approved by the prefix verifier (§2.4), these drafted

tokens are buffered and forwarded to the streaming TTS system as soon as the minimum synthesis chunk is reached, substantially reducing the time to first audio.

Interruption handling. Because the speculative stream no longer observes live user speech, it cannot detect barge-in. The main online stream retains this responsibility. If the main stream predicts `[STP]`, playback is halted immediately and the remaining buffered draft is discarded. This design decouples fast response generation from real-time interaction monitoring, achieving low response-onset latency without sacrificing interruption handling.

2.4 Selective Prefix Handoff

Once the speculative stream drafts a prefix, a verifier decides whether to commit it for immediate streaming to TTS or suppress it and defer entirely to the cascaded pipeline. When committed, the slow path continues generation conditioned on the prefix. Because the verifier reuses already-computed decoder states, it adds negligible overhead.

Verifier input representation. At each drafted token position $t \in \{1, \ldots, n\}$, we extract two signals: (i) the final-layer decoder hidden state $\mathbf{h}_t \in \mathbb{R}^d$, capturing contextual semantics, and (ii) a scalar feature vector $\mathbf{s}_t \in \mathbb{R}^3$ comprising the output distribution entropy $H(P_t)$, the selected-token log-probability $\log P_t(y_t)$, and the top-two logit margin $z_t^{(1)} - z_t^{(2)}$, where $z_t^{(1)}$ and $z_t^{(2)}$ are the largest and second-largest pre-softmax logits at step t. Under greedy decoding this margin coincides with the log-probability gap between the top two candidates. Together, these signals provide both semantic and calibration information per token.

Verifier architecture. The verifier projects each hidden state to a lower-dimensional space, modulates it with the scalar features via a multiplicative gate plus an additive bypass, adds learned positional embeddings, and applies a token-level feed-forward block with a residual connection. The resulting per-position representations are aggregated by attention pooling with a single learned query and passed through a sigmoid head to produce a confidence score $c \in [0, 1]$. Full architectural equations and dimensions are provided in Appendix A.6.

Training. The verifier is trained as a binary classifier over drafted prefixes labeled as good or bad. We generate speculative prefixes by running the S2S model on its own training split, then produce a prefix-conditioned continuation with GPT-5.4 (OpenAI, 2026) and submit the (context, prefix, continuation) triple to an ensemble of three LLMs whose majority vote sets the final label (see Appendix A.11 for prompts and LLM details). A prefix is labeled good if it is judged contextually sensible given the dialogue context, and bad otherwise. The validation and test sets for the prefix verifier are obtained from the corresponding validation and test splits of the S2S model. To ensure high-fidelity evaluation signals, the ground-truth labels for both the validation and test sets are annotated by human annotators.

2.5 Slow Path: Cascaded Pipeline

The slow path is a conventional ASR–LLM pipeline whose role is to produce a higher-quality response in text space once the fast path has secured low-latency response onset.

Triggering and ASR. Rather than relying on a separate end-of-utterance detector, the slow path reuses the fast path's `[BOS]` decision as its trigger. Because this decision already implies that the user has yielded the floor, the full utterance is typically available at trigger time. The buffered speech segment is passed to an offline ASR model, and the resulting transcript is combined with the dialogue history to form the LLM input context.

Prefix-conditioned generation and self-correction. If the verifier commits the fast-path prefix, the committed tokens are supplied to the slow-path LLM as a hard constraint—so that the model's output is conditioned on the exact token sequence the listener is already hearing. For local LLMs, we prefill the assistant message with the committed prefix and continue decoding from that state. For API LLMs, the prefix is passed as already-spoken assistant context with instructions to emit only the continuation.

The slow-path LLM is also prompted to handle prefix failures dynamically. When the prefix is contextually sound, the model continues it smoothly. When the prefix contains an early factual error, hallucination, or contextual inconsistency, the slow path executes a conversational repair within the continuation stream (*e.g.*, *"...wait, I mean..."* or *"...actually, let me correct that..."*) rather than propagating the flaw. The full prompt template appears in Appendix A.12.

2.6 Streaming TTS

The streaming TTS module (ElevenLabs, 2026) synthesizes chunk-incrementally, beginning as soon as a minimum text chunk is available. For seamless audio, the slow-path continuation must arrive before the TTS exhausts the prefix; we call this gap the *relay margin*, with positive values yielding an uninterrupted utterance and negative ones producing a perceptible pause. In practice, a 5-word prefix yields roughly 2 seconds of speech, comfortably exceeding the slow-path generation time for most LLM back-ends.

3 Training Data

We train on a 2,437-hour, 194K-conversation corpus combining (a) 104K fully synthetic dialogues (2,133 h) synthesized with CosyVoice2 (Du et al., 2024) and noise-augmented from TAU (Heittola et al., 2024), and (b) 90K real-voice two-turn conversations (304 h) built by pairing filtered human audio from GigaSpeech (Chen et al., 2021), Common Voice (Ardila et al., 2020), and Switchboard (Godfrey et al., 1992) with GPT-5.4-generated assistant responses. We additionally inject backchannels, interruptions, and mid-utterance pauses to train the duplex control tokens described in §2.2. Full data construction details are provided in Appendix A.1.

4 Experiments

4.1 Implementation Details

The fast-path S2S model uses a streaming conformer encoder, a two-stage causal CNN adapter producing one speech representation every 160 ms, and a Qwen2.5-0.5B (Qwen Team et al., 2025) LLM backbone. Training proceeds in three stages: (1) encoder–adapter pretraining on speech transcription; (2) full-duplex S2S training end-to-end on the data described in §3; and (3) prefix verifier training as described in §2.4. More training details are provided in Appendix A.5.

4.2 Baselines and Concurrent Works

We compare RelayS2S against two primary baseline families:

S2S only. This baseline uses the same fast-path model with forked free-running generation. It is functionally similar to native end-to-end full-duplex speech architectures such as SALM-Duplex (Hu et al., 2025), but without a slow-path LLM or prefix verifier. This setup isolates the performance of the fast path without any cascaded guidance.

Cascaded ASR–LLM. We pair an offline ASR module with text-based LLMs of varying capacity. For ASR, we use Whisper (Radford et al., 2023) (`medium.en`) via the CTranslate2 implementation. The model transcribes a user turn in approximately 200 ms on average using NVIDIA L40S. Although production deployments may use streaming ASR, such systems still incur a typical 150–300 ms finalization step at turn end, placing our offline ASR latency within the same range as a comparable streaming setup. For the text-based LLM, we evaluate four configurations spanning both local (Qwen2.5-0.5B, Qwen2.5-7B) and API-based inference (Grok-4.1-fast (xAI, 2025) and GPT-4.1.

We also compare with two recent concurrent works. As public code, models, and training data were unavailable at the time of our experiments, we instead isolate the architectural axis each system explores by evaluating the corresponding RelayS2S configuration.

DDTSR. Liu et al. (2026) propose a dual-track architecture in which the fast path emits only short discourse connectives (1–3 tokens) while the back-end LLM reasons in parallel. We approximate this regime with RelayS2S at `prefix_length=3`, isolating the effect of a short speculative prefix in a conventional streaming pipeline (Tables 2 and 3).

KAME. Kuroki et al. (2025) use a tandem architecture in which an S2S front-end speaks first and is progressively corrected by an asynchronous LLM "oracle" stream, with no gating between drafted and committed tokens. We approximate this ungated commit behavior by disabling the prefix verifier in RelayS2S, so every drafted prefix is committed unconditionally (Table 3).

4.3 Evaluation Metrics

Textual response quality. We prompt Gemini-3-Flash (Google, 2025) to rate each generated response on a 1–5 scale given the dialogue context. Because absolute scores can be subjective and exhibit non-trivial variance across runs, we report both the average score and the *low-quality rate*: the fraction of responses scored ≤ 3, where a score of 3 or below indicates noticeable issues under our rubric (Appendix A.14). We validate the Gemini-3-Flash judge against a 3-annotator human panel (Cohen's $\kappa = 0.70$ vs. majority vote) and verify run-to-run stability ($\sigma = 0.47$ pp on low-quality rate,

Fleiss' $\kappa = 0.81$); full details in Appendix A.7. We adopt the low-quality rate as our primary headline metric because it offers superior interpretability (the percentage of noticeably problematic responses) and is robust to minor calibration shifts in the 4–5 range that do not affect the practical acceptability boundary. We report 95% percentile bootstrap confidence intervals ($B = 5{,}000$ resamples, seed $= 42$) for all quality metrics.

First-chunk latency. To isolate the core algorithmic efficiency of our architecture, we report first-chunk latency: the interval from turn detection (the moment the agent decides to speak) to the availability of the agent's first N response words for the streaming TTS module. We adopt $N = 5$ as our default operating point, following Bai et al. (2025), who identify a text window length of 5 as optimal for high-quality streaming speech synthesis. Also, a 5-word chunk yields roughly 2 seconds of speech, providing a sufficient temporal buffer for the slow path to deliver its first continuation chunk. We explicitly exclude infrastructure-dependent variables—specifically raw turn-detection lag and TTS audio rendering time—as these components are shared identically across all baselines and do not affect relative evaluation. Under this definition, first-chunk latency is computed as follows:
S2S only: time from turn detection to the S2S decoding of the N-th response word: L_{S2S}.
Cascaded only: ASR transcription time plus LLM generation of the first N response words: $L_{\text{cascaded}} = T_{\text{ASR}} + T_{\text{LLM}}$.
RelayS2S (COMMIT): S2S decoding time plus the verifier decision: $L_{\text{commit}} = L_{\text{S2S}} + T_{\text{verifier}}$.
RelayS2S (FALLBACK): falls back to the cascaded path: $L_{\text{fallback}} = L_{\text{cascaded}}$.
Latency is measured on a single NVIDIA L40S GPU for local models, and client-side (dispatch to first chunk) for API back-ends.

4.4 Results

We evaluate our system on both synthetic and real dialogue test sets. The synthetic subset comprises 1,000 held-out conversations (2,175 context–response pairs), while the real subset contains 2,000 held-out conversations (2,000 context–response pairs). For each configuration, we report the average and P90 first-chunk latency, alongside the textual response quality. Table 1 summarizes the performance across these subsets. Note that the fast-path component remains identical across all evaluated RelayS2S configurations, utilizing the same 0.5B S2S model and prefix verifier; performance variations are driven solely by the choice of the slow-path LLMs.

Latency. On the synthetic subset, RelayS2S achieves a P90 first-chunk latency of 81 ms regardless of the slow-path LLM, closely matching the 72 ms of the pure S2S model—the verifier adds only $\sim$10 ms of overhead. The cascaded baselines, by contrast, scale with model size, ranging from 418 ms with Qwen2.5-0.5B up to 1,006 ms with GPT-4.1. The picture is more nuanced on real voice dialogues. The verifier commits 70.7% of prefixes here (vs. 91.9% on synthetic), so the remaining 29.3% of turns fall back to the cascade and pay its full latency. P90 therefore tracks this fallback tail (655 ms with GPT-4.1, down from 823 ms), but the bulk of committed turns run at S2S latency, pulling the average down sharply: 748 $\rightarrow$ 269 ms with GPT-4.1, with comparable 57–64% reductions across the other back-ends.

Response quality. RelayS2S retains nearly all of the cascaded pipeline's quality while substantially outperforming the standalone S2S baseline. On the synthetic subset, the larger back-ends match the cascade essentially exactly (e.g., 4.84 vs. 4.84 with GPT-4.1), well above the S2S-only score of 3.29. On real voice dialogues, the low-quality rate rises by about a point (10.9% vs. 9.8% with GPT-4.1, within the bootstrap CI band)—still far below the S2S-only baseline's 56.0%. With the 0.5B back-end, RelayS2S is statistically indistinguishable from the cascade on both subsets, likely because the S2S prefix occasionally compensates for ASR-induced errors in the cascade's opening.

Quality–latency trade-off. Taken together, RelayS2S effectively decouples the quality–latency trade-off of conventional pipelines. This pattern reveals its role as a composition operator rather than a competitor to either component: on synthetic dialogues, the latency gap it closes widens from 227 ms with Qwen2.5-0.5B to 734 ms with GPT-4.1, while committed-turn quality tracks the cascade ceiling within $\sim$99%. Symmetrically, improvements to the fast path would reduce the fallback rate and tighten the latency tail. RelayS2S thus inherits, rather than competes with, advances on either axis, making it attractive for deployments that rely on large or API-served LLMs where cascaded latency would otherwise be prohibitive.

System	Slow-path LLM	Avg Lat. (ms)	P90 Lat. (ms)	Avg Quality	Low-quality (%)
Synthetic voice dialogues					
S2S only	—	59	72	$3.29_{\pm 0.06}$	$52.7_{\pm 2.1}$
Cascaded	Qwen2.5-0.5B	304	418	$3.45_{\pm 0.06}$	$48.7_{\pm 2.1}$
RelayS2S	Qwen2.5-0.5B	77	81	$3.49_{\pm 0.06}$	$47.6_{\pm 2.1}$
Cascaded	Qwen2.5-7B	392	514	$4.21_{\pm 0.05}$	$27.5_{\pm 1.9}$
RelayS2S	Qwen2.5-7B	85	81	$4.21_{\pm 0.05}$	$26.9_{\pm 1.9}$
Cascaded	Grok-4.1-fast	726	876	$4.73_{\pm 0.03}$	$8.4_{\pm 1.2}$
RelayS2S	Grok-4.1-fast	112	81	$4.73_{\pm 0.03}$	$9.4_{\pm 1.2}$
Cascaded	GPT-4.1	853	1006	$4.84_{\pm 0.03}$	$5.6_{\pm 0.9}$
RelayS2S	GPT-4.1	119	81	$4.84_{\pm 0.03}$	$5.3_{\pm 0.9}$
Real voice dialogues					
S2S only	—	59	76	$3.17_{\pm 0.07}$	$56.0_{\pm 2.2}$
Cascaded	Qwen2.5-0.5B	205	239	$3.82_{\pm 0.06}$	$37.3_{\pm 2.1}$
RelayS2S	Qwen2.5-0.5B	101	214	$3.81_{\pm 0.06}$	$37.7_{\pm 2.1}$
Cascaded	Qwen2.5-7B	297	354	$4.06_{\pm 0.06}$	$30.6_{\pm 2.0}$
RelayS2S	Qwen2.5-7B	129	309	$4.03_{\pm 0.06}$	$32.2_{\pm 2.1}$
Cascaded	Grok-4.1-fast	634	735	$4.54_{\pm 0.05}$	$14.3_{\pm 1.6}$
RelayS2S	Grok-4.1-fast	232	665	$4.52_{\pm 0.05}$	$15.3_{\pm 1.6}$
Cascaded	GPT-4.1	748	823	$4.70_{\pm 0.04}$	$9.8_{\pm 1.3}$
RelayS2S	GPT-4.1	269	655	$4.67_{\pm 0.04}$	$10.9_{\pm 1.4}$

Table 1: First-chunk latency (Lat.) and response quality across system configurations on synthetic and real dialogues. RelayS2S uses a 5-word prefix throughout; the verifier commits 91.9% and 70.7% of prefixes on the synthetic and real-voice subsets respectively.

Back-end	Prefix len	P10 (s)	Med (s)	Stall (%)
Qwen2.5-7B	3 words	0.23	0.57	0.5
	5 words	0.84	1.45	0.0
	7 words	1.46	2.15	0.0
Grok-4.1-fast	3 words	-0.16	0.25	22.7
	5 words	0.44	1.07	0.5
	7 words	1.09	1.76	0.3

Table 2: Relay margin statistics on committed-prefix turns. P10 and median are computed over the per-turn relay-margin distribution.

Relay margin analysis. While Table 1 establishes that RelayS2S preserves textual response quality, the prefix-to-continuation handoff introduces an audio-level failure mode: a relay-margin stall, in which the slow path fails to deliver its first chunk before the prefix audio is exhausted. We define the relay margin as the duration of the prefix audio minus the slow-path time to first chunk. Table 2 reports the distribution across different prefix lengths and LLM back-ends on committed turns.

Two effects stand out. The local 7B back-end is robust across all prefix lengths: even a 3-word prefix yields a positive median relay margin (0.57 s) and a negligible stall rate (0.5%). Conversely, API-hosted back-ends are highly sensitive to prefix length. At 3 words on Grok-4.1-fast, the heavier latency tail pushes the lower quantile negative (P10 $= -0.16$ s), producing audible stalls on 22.7% of committed turns. This contextualizes why the 3-word scale used by DDTSR (Liu et al., 2026) is fragile in conventional deployments: without specialized streaming mechanisms, a short prefix provides insufficient audio runway to absorb API latency tails. Extending the prefix to 5 words reduces this stall rate to 0.5%, and 7 words eliminates stalls almost entirely.

Prefix verifier performance. The verifier is trained on 302K prefixes (5.9% labeled as bad) and evaluated on a synthetic test set of 2,175 examples (4.78% bad) and a real-speech test set of 2,000 examples (17.5% bad), achieving AUROC of 0.94 and 0.88 respectively. At our default operating threshold of 0.50, the verifier commits 95.0%/81.1% of good prefixes on synthetic/real test sets and rejects 71.2%/78.3% of bad prefixes. This results in fallback rates of 8.1%/29.3%. The full threshold sweep is provided in Appendix A.8.

Prefix	Verifier	Commit	Quality	Low-Q (%)
3 words	✗	100%	$4.67_{\pm 0.04}$	$10.9_{\pm 1.4}$
3 words	✓	87.2%	$4.70_{\pm 0.04}$	$9.8_{\pm 1.3}$
5 words	✗	100%	$4.52_{\pm 0.04}$	$14.3_{\pm 1.5}$
5 words	✓	70.7%	$4.67_{\pm 0.04}$	$10.9_{\pm 1.4}$
7 words	✗	100%	$4.44_{\pm 0.05}$	$16.6_{\pm 1.6}$
7 words	✓	60.8%	$4.67_{\pm 0.04}$	$10.5_{\pm 1.4}$

Table 3: Ablation on prefix length and verifier gating on real voice subset with the GPT-4.1 back-end.

		Avg Quality		Low-Q (%)	
Subset	n	Casc.	RelayS2S	Casc.	RelayS2S
Committed	1,414	4.83	4.79	5.5	7.1
Fallback	586	4.36	4.38	20.3	20.1
All	2,000	4.70	4.67	9.8	10.9

Table 4: Fallback quality decomposition with GPT-4.1 as the back-end on real speech (n=2,000).

Prefix length influence. Table 3 ablates the effect of prefix length and verifier gating.

Two trends are clear. First, longer prefixes commit more S2S content before the stronger LLM can intervene, so quality degrades monotonically without gating: the low-quality rate rises from 10.9% at 3 words to 16.6% at 7 words. Tandem architectures like KAME (Kuroki et al., 2025) address the same trade-off without a verifier, instead training the front-end S2S to follow a progressively refined LLM "oracle" stream and prioritizing the most recent oracle at inference. This enables mid-utterance self-correction once better guidance arrives, but the authors note it only mitigates premature commitment rather than eliminating it. The residual early-commitment problem accounts for most of KAME's remaining quality gap to cascaded systems. Our ablation shows the cost of making that early commitment with no safety mechanism.

With the verifier enabled, that cost disappears. The low-quality rate stays within a tight band (9.8%–10.9%) across all prefix lengths, and the protection widens as prefixes grow: the verifier reduces low-quality responses by 1.1 points at 3 words (within CI), 3.4 points at 5 words, and 6.1 points at 7 words. However, gating's protection has a latency-coverage cost: commit rates drop from 87.2% at 3 words to 60.8% at 7 words, motivating a moderate prefix length of 5 as the operating point in our main experiments.

Fallback decomposition. Table 4 confirms that the verifier correctly segregates turns by macro-level difficulty: the cascaded pipeline scores significantly lower on the subset of turns triggered for fallback than on committed turns (4.36 vs. 4.83) and yields a nearly $4\times$ higher low-quality rate (20.3% vs. 5.5%). This explains our latency profile: RelayS2S captures S2S-latency speed on the 70.7% of easier, committed turns while gracefully absorbing full cascaded latency on the harder 29.3% of fallbacks. To isolate whether the residual 1.1% aggregate quality gap stems from prefix propagation or baseline difficulty, we cross-tabulate human prefix-sensibility labels against final response quality on committed turns (Appendix A.9). Only 5.4% of committed turns contain a non-sensible prefix; crucially, on these difficult turns, the slow path actively recovers to generate an acceptable response in 63.2% of cases. This demonstrates that prefix-conditioned generation actively repairs, rather than passively propagates, flawed prefixes. The remaining 1.98% of committed turns where both the prefix and response fail represents the irreducible cost of selective handoff, accounting for the bulk of RelayS2S's committed-turn delta.

5 Conclusion

We presented RelayS2S, a speculative hybrid architecture that pairs a duplex S2S model with a cascaded ASR–LLM pipeline to break the latency–quality trade-off in real-time spoken dialogue. The fast path drafts a short response prefix via forked speculative generation, decoupling rapid drafting from live audio monitoring. A lightweight learned verifier then gates the handoff, committing the prefix when it is contextually safe and falling back to the cascade otherwise.

Across four slow-path back-ends, RelayS2S approaches the latency profile of a pure S2S model while retaining $\sim$99% of the cascaded pipeline's textual quality. With GPT-4.1, P90 first-chunk latency drops from 1,006 ms to 81 ms on synthetic dialogues, and average first-chunk latency drops by roughly $2.8\times$ on real voice dialogues. More fundamentally, this pattern reveals RelayS2S's role as a composition operator over its components: advances on either axis propagate directly into the system, narrowing the latency–quality gap between S2S and cascaded endpoints without architectural change.

Limitations

RelayS2S has several limitations worth noting. First, although the average latency drops sharply, the latency tail remains dominated by fallback events. On real voice dialogues, 29.3% of turns route through the full cascade pipeline. Consequently, the P90 latency when utilizing a GPT-4.1 back-end rises to 655 ms, contrasting with the 81 ms achieved on committed turns. This behavior is primarily driven by poor prefix drafting from the fast-path S2S model, which triggers a verifier fallback. Improving the underlying S2S model by training it on more diverse acoustic and textual data would likely enhance initial prefix quality, thereby reducing the fallback rate and tightening the latency tail. Second, the system is vulnerable to relay-margin stalls, where the slow path fails to produce its first continuational chunk before the prefix audio ends. This is worst with API-hosted LLMs, whose latency tails are heavier: a 3-word prefix with Grok-4.1-fast stalls on 22.7% of committed turns, and the prefix must be extended to 5 words to mitigate this. Possible fixes include using API providers with tighter tail latency or adding stall-handling mechanisms such as filler speech or adaptive prefix length.

Ethical Considerations

Data compliance and asset use. All speech corpora used in this work—GigaSpeech, Common Voice, Switchboard, and TAU Urban Acoustic Scenes—are publicly released for research use, and we use them strictly within the terms of their respective licenses. We do not redistribute raw audio data.

Deployment risks and mitigation. By design, RelayS2S initiates speech generation before the slow path has reasoned over the complete user utterance. Although the verifier successfully rejects most invalid prefixes and the slow path dynamically repairs committed errors (Appendix A.9), a small residual fraction of turns may still produce an audible error before a correction can be executed. Consequently, we caution against deploying this system in safety-sensitive domains (e.g., medical, legal, or mental health applications) without additional safeguards, such as domain-specific verifiers, more conservative operating thresholds, or restricting speculative generation to non-substantive conversational openers.

Scientific use of AI assistants. Large language models were utilized during the development of this project to assist with code optimization, conceptual brainstorming, and stylistic polishing of the manuscript. The core research directions, experimental designs, and data interpretations were conceived and directed entirely by the human authors, who maintain full accountability for the scientific integrity and accuracy of the work.

References

Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber. 2020. Common voice: A massively-multilingual speech corpus. In *Proceedings of the twelfth language resources and evaluation conference*, pages 4218–4222.

Richard He Bai, Zijin Gu, Tatiana Likhomanenko, and Navdeep Jaitly. 2025. SpeakStream: Streaming text-to-speech with interleaved data. *arXiv preprint arXiv:2505.19206*.

Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, and 1 others. 2021. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio. *arXiv preprint arXiv:2106.06909*.

Junjie Chen, Yao Hu, Junjie Li, Kangyue Li, Kun Liu, Wenpeng Li, Xu Li, Ziyuan Li, Feiyu Shen, Xu Tang, and 1 others. 2025. Fireredchat: A pluggable, full-duplex voice interaction system with cascaded and semi-cascaded implementations. *arXiv preprint arXiv:2509.06502*.

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*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, and 1 others. 2019. The second conversational intelligence challenge (convai2). In *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*, pages 187–208. Springer.

Zhihao Du, Yuxuan Wang, Qian Chen, Xian Shi, Xiang Lv, Tianyu Zhao, Zhifu Gao, Yexin Yang, Changfeng Gao, Hui Wang, and 1 others. 2024. Cosyvoice 2: Scalable streaming speech synthesis with large language models. *arXiv preprint arXiv:2412.10117*.

ElevenLabs. 2026. Models: Flash v2.5. https://elevenlabs.io/docs/overview/models#flash-v25.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520. IEEE.

Google. 2025. Gemini 3 Flash: Frontier intelligence built for speed. https://blog.google/products/gemini/gemini-3-flash/.

Google. 2026. Gemini 3.1 Pro. https://deepmind.google/models/gemini/pro/.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2023. Topical-chat: Towards knowledge-grounded open-domain conversations. *arXiv preprint arXiv:2308.11995*.

Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. 2024. Tau urban acoustic scenes 2024 mobile, evaluation dataset. https://doi.org/10.5281/zenodo.11366913.

Ke Hu, Ehsan Hosseini-Asl, Chen Chen, Edresson Casanova, Subhankar Ghosh, Piotr Żelasko, Zhehuai Chen, Jason Li, Jagadeesh Balam, and Boris Ginsburg. 2025. Salm-duplex: Efficient and direct duplex modeling for speech-to-speech language model. *arXiv preprint arXiv:2505.15670*.

So Kuroki, Yotaro Kubo, Takuya Akiba, and Yujin Tang. 2025. Kame: Tandem architecture for enhancing knowledge in real-time speech-to-speech conversational ai. *arXiv preprint arXiv:2510.02327*.

Stephen C Levinson and Francisco Torreira. 2015. Timing in turn-taking and its implications for processing models of language. *Frontiers in psychology*, 6:136034.

Tatiana Likhomanenko, Luke Carlson, Richard He Bai, Zijin Gu, Han Tran, Zakaria Aldeneh, Yizhe Zhang, Ruixiang Zhang, Huangjie Zheng, and Navdeep Jaitly. 2025. Chipchat: Low-latency cascaded conversational agent in mlx. *arXiv preprint arXiv:2509.00078*.

Siyuan Liu, Jiahui Xu, Feng Jiang, Kuang Wang, Zefeng Zhao, Chu-Ren Huang, Jinghang Gu, Changqing Yin, and Haizhou Li. 2026. Discourse-aware dual-track streaming response for low-latency spoken dialogue systems. *arXiv preprint arXiv:2602.23266*.

Zhanxun Liu, Yifan Duan, Mengmeng Wang, Pengchao Feng, Haotian Zhang, Xiaoyu Xing, Yijia Shan, Haina Zhu, Yuhang Dai, Chaochao Lu, and 1 others. 2025. X-talk: On the underestimated potential of modular speech-to-speech dialogue system. *arXiv preprint arXiv:2512.18706*.

OpenAI. 2025. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/.

OpenAI. 2026. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

Rajarshi Roy, Jonathan Raiman, Sang-gil Lee, Teodor-Dumitru Ene, Robert Kirby, Sungwon Kim, Jaehyeon Kim, and Bryan Catanzaro. 2026. Personaplex: Voice and role control for full duplex conversational speech models. *arXiv preprint arXiv:2602.06053*.

Paul K Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, and 1 others. 2023. Audiopalm: A large language model that can speak and listen (2023). *arXiv preprint arXiv:2306.12925*.

Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y-Lan Boureau. 2020. Can you put it all together: Evaluating conversational agents' ability to blend skills. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2021–2030.

Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Xiangyang Liu, Hang Yan, Yunfan Shao, Qiong Tang, Shiduo Zhang, and 1 others. 2024. Moss: An open conversational large language model. *Machine Intelligence Research*, 21(5):888–905.

Xiaoyu Wang, Yukun Li, Changfeng Fu, Yong Shen, Liang Xie, Kai Li, Xiulin Sun, and Long Ma. 2024. Freeze-Omni: A smart and low latency speech-to-speech dialogue model with frozen LLM. *arXiv preprint arXiv:2411.00774*.

xAI. 2025. Grok 4.1. https://x.ai/news/grok-4-1.

Zhifei Xie and Changqiao Wu. 2024. Mini-omni: Language models can hear, talk while thinking in streaming. *arXiv preprint arXiv:2408.16725*.

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. 2023. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15757–15773.

Wenhao Zou, Yuwei Miao, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Jingwen Xu. 2026. Lts-voiceagent: A listen-think-speak framework for efficient streaming voice interaction via semantic triggering and incremental reasoning. *arXiv preprint arXiv:2601.19952*.

A Appendix

A.1 Training Data Details

We construct a hybrid training corpus from two sources: (1) fully synthetic dialogues rendered entirely with text-to-speech (TTS) systems, and (2) real voice dialogues combining authentic human speech recordings with synthetic assistant responses.

A.2 Synthetic Voice Dialogues

We source foundational text dialogues from two channels. First, we collect single- and multi-turn user–assistant conversations from VoiceAssistant (Xie and Wu, 2024) and OpenMOSS (Sun et al., 2024). Second, we prompt GPT-5.4 to generate additional conversations using topic and plot outlines derived from human–human dialogue corpora, including TopicalChat (Gopalakrishnan et al., 2023), ConvAI (Dinan et al., 2019), and BlendedSkillTalk (Smith et al., 2020). All dialogues are synthesized into 16 kHz speech using CosyVoice2 (Du et al., 2024) with a default inter-turn silence of 200 ms.

To improve robustness to real-world acoustic conditions, we mix in non-overlapping noise segments sampled from the TAU Urban Acoustic Scenes dataset (Heittola et al., 2024). Noise clips span 5–10 s, cover 50–100% of each conversation, and are applied at signal-to-noise ratios uniformly sampled between 0 and 20 dB.

The resulting synthetic corpus contains 104K conversations totaling 2,133 hours of audio, with conversation lengths ranging from 3 s to 3 min.

A.3 Real Voice Dialogues

To complement the synthetic data with authentic vocal acoustics, speaking styles, and recording conditions, we incorporate real human speech from three large-scale automatic speech recognition (ASR) corpora: GigaSpeech (Chen et al., 2021), Common Voice (Ardila et al., 2020), and Switchboard (Godfrey et al., 1992).

Because raw ASR transcripts often contain incomplete conversational fragments, narrative speech, or context-dependent utterances unsuitable for assistant interactions, we apply an automated filtering pipeline. Specifically, we use Gemini-3-Flash to classify each transcript for whether it is both context-independent and plausibly directed at an AI assistant (see Appendix A.13 for prompts and filtering criteria).

This process yields approximately 90,000 curated human audio prompts. We then prompt GPT-5.4 to generate instruction-following assistant responses conditioned on these transcripts, producing 90K two-turn real voice conversations totaling 304 hours of audio.

Figure 3: Training examples for duplex control tokens. Top: the agent emits [STP] when interrupted by the user. Bottom: the agent emits [BOC] to produce a backchannel during user speech.

A.4 Duplex Phenomena Injection

To train the duplex control tokens described in §2.2, we augment the synthetic dialogues with three categories of overlapping-speech events.

Backchannels. We prompt Gemini-3.1-Pro (Google, 2026) to identify appropriate insertion points and generate backchannel utterances (e.g., *uh-huh*, *right*) during user speech.

Interruptions. We randomly truncate assistant utterances between 20% and 60% of their duration, then overlap the following user turn by 320 ms to simulate barge-in behavior. Figure 3 shows examples of interruption and backchannel events.

Mid-utterance pauses. We insert [PAUSE] tokens into user utterances to simulate extended hesitations, exposing the model to natural pause patterns and discouraging premature responses.

A.5 Implementation Details

Fast-path S2S model. The speech encoder is a 24-block conformer with 1024-dimensional hid-

den states, 16 attention heads, and relative position encodings. It consumes 80-dimensional log-Mel features extracted at 16 kHz with a 10 ms frame shift and applies $4\times$ subsampling internally. The encoder operates with a chunk size of 4 subsampled frames and 16 left-context chunks. The CNN adapter consists of two causal 1D convolutional stages (kernel size 5, stride 2, left-padded). The first stage preserves the 1024-dimensional encoder output; the second projects down to 896 dimensions to match the LLM hidden size. Together with the encoder's internal $4\times$ subsampling, the two stride-2 adapter stages yield an overall $16\times$ temporal downsampling from $10\,\mathrm{ms}$ input frames, producing one speech representation every $160\,\mathrm{ms}$. The LLM backbone is Qwen2.5-0.5B (Qwen Team et al., 2025).

Training procedure. Training proceeds in three stages: (1) *Encoder–adapter pretraining*: the LLM backbone is frozen and only the speech encoder and CNN adapter are updated, using speech transcription as the learning objective so that the adapted representations align with the LLM's text embedding space. (2) *Full-duplex S2S training*: all parameters are unfrozen and the model is trained end-to-end on the synthetic and real voice conversations data described in §3. (3) *Prefix verifier training*, following the procedure described in §2.4. All three stages are run on two NVIDIA L40S GPUs (48 GB VRAM each), with each stage converging within 48 hours.

A.6 Verifier Architecture

This section provides the full architectural specification of the prefix verifier described in §2.4. Given the per-position decoder hidden state $\mathbf{h}_t \in \mathbb{R}^d$ and scalar feature vector $\mathbf{s}_t \in \mathbb{R}^3$, the hidden state at each position is first projected to a lower-dimensional space:

$$\hat{\mathbf{h}}_t = \mathrm{GELU}(\mathrm{LayerNorm}(W_p\,\mathbf{h}_t + b_p)) \tag{2}$$

where $W_p \in \mathbb{R}^{d'\times d}$ with $d' \ll d$. The scalar features then modulate this representation through a *scalar gate*:

$$\tilde{\mathbf{z}}_t = \hat{\mathbf{h}}_t \odot \sigma(W_g\,\mathbf{s}_t + b_g) \;+\; W_a\,\mathbf{s}_t + b_a \tag{3}$$

where $W_g, W_a \in \mathbb{R}^{d'\times 3}$. The multiplicative gate controls how much each hidden dimension is trusted given the calibration signals, while the additive bypass ensures scalar information flows even when the gate saturates. Learned positional embeddings are added, followed by a token-level feedforward block with a residual connection:

$$\mathbf{z}_t = \tilde{\mathbf{z}}_t + \mathrm{pos}_t + \mathrm{FF}(\tilde{\mathbf{z}}_t + \mathrm{pos}_t) \tag{4}$$

The sequence $\mathbf{z}_{1:n}$ is aggregated via attention pooling with a single learned query $\mathbf{q}$ and learned key/-value projections $W_k, W_v \in \mathbb{R}^{d'\times d'}$:

$$\mathbf{p} = \mathrm{LayerNorm}\big(\mathrm{Attn}(\mathbf{q},\, W_k\,\mathbf{z}_{1:n},\, W_v\,\mathbf{z}_{1:n})\big) \tag{5}$$

and passed through a linear head followed by a sigmoid to produce a confidence score $c \in [0, 1]$.

A.7 Judge Validation Details

This section details the validation of the Gemini-3-Flash judge used to score textual response quality (§4.3). The judge is given the reference dialogue transcript, not the system ASR transcript, so ASR-induced misunderstandings are reflected in the response-quality score.

Run-to-run stability. We analyzed the judge's variance across three independent runs on a 1,000-sample evaluation set. The aggregate low-quality rate varied by only 0.47 percentage points (standard deviation) across runs, yielding high inter-run consistency (Fleiss' $\kappa = 0.81$ for the binary label), which is comparable to the run-to-run variation of the average quality score ($\sigma = 0.016$). While 13.9% of individual items received inconsistent binary classifications across evaluations (mean per-item $\sigma = 0.21$), these label flips were symmetric and neutralized at the population level.

Validation against human annotators. To validate the Gemini-3-Flash judge against human standards, we compared its binarized low-quality decisions on the same 1,000-sample set against the majority vote of a panel consisting of three human annotators. The automated judge achieved a strong Cohen's $\kappa = 0.70$ against the human consensus, with a precision of 0.96 and a recall of 0.83 for the low-quality class. Within the panel, individual annotators agreed with the majority at $\kappa \geq 0.78$, confirming high internal alignment. The panel was comprised of three expert human annotators with backgrounds in computer science, based in Ireland, who participated on a voluntary basis. The evaluation protocol did not involve the collection or storage of any personal data; annotators were exclusively tasked with providing objective classifications of model output quality, ensuring

Subset	Thr.	Bad reject	Good commit	Fallback
Synthetic	0.25	24.0%	99.7%	1.4%
	0.50	71.2%	95.0%	8.1%
	0.75	98.1%	70.1%	33.2%
Real	0.25	30.9%	98.8%	6.4%
	0.50	78.3%	81.1%	29.3%
	0.75	98.6%	27.6%	76.9%

Table 5: Prefix verifier operating points on synthetic and real-speech test sets.

no personally identifiable information or sensitive participant data was captured.

A.8 Prefix Verifier Threshold Sweep

Table 5 shows the tradeoff between rejecting bad prefixes and preserving fast-path coverage across operating thresholds. Lower thresholds maximize latency savings by committing nearly all prefixes, while higher thresholds reduce bad commits at the cost of more frequent fallback to the slower cascaded pipeline. We select a threshold of 0.50 as the default operating point. On synthetic data, this threshold rejects 71% of bad prefixes while committing 95% of good prefixes, with an overall fallback rate of 8%. On real speech, it rejects 78% of bad prefixes while committing 81% of good prefixes, with a 29% fallback rate. Increasing the threshold to 0.75 nearly eliminates bad commits (1–2%) but substantially reduces fast-path coverage, especially on real speech.

A.9 Prefix-to-Response Contingency Analysis

To isolate whether RelayS2S's residual 1.1% aggregate gap on committed turns (10.9% vs. 9.8% low-quality, Table 4) stems from prefix propagation errors or baseline turn difficulty, we cross-tabulate human prefix-sensibility labels against final response quality within the committed subset. Table 6 reports the resulting contingency matrix on real speech with the GPT-4.1 back-end.

	Response Quality			
Prefix	Good	Bad	Total	% Bad
Good	1,266	72	1,338	5.4
Bad	48	28	76	36.8
Total	1,314	100	1,414	7.1

Table 6: Contingency matrix of prefix sensibility against RelayS2S response quality on real speech (GPT-4.1 back-end). Rows partition the 1,414 committed turns by whether the human annotator judged the prefix sensible.

Event	Precision	Recall	F1
Start speaking (`[BOS]`)	78.9	96.1	86.7
Stay silent (`[SIL]`)	99.9	99.6	99.7
Stop speaking (`[STP]`)	89.2	97.5	93.2
Backchannel (`[BOC]`)	36.3	72.7	48.4

Table 7: Turn-taking event detection performance of the fast-path S2S model, evaluated with a ± 1 frame tolerance window.

The verifier filters the incoming stream effectively: only 5.4% (76/1,414) of committed turns contain a non-sensible prefix. On the remaining 94.6% of turns where the prefix is sensible, RelayS2S essentially matches the cascaded baseline (5.4% vs. 4.9% low-quality rate). On the 76 bad-prefix turns, the slow path successfully recovers to generate an acceptable response in 63.2% (48/76) of cases, showing that prefix-conditioned generation actively repairs rather than passively propagates acoustic or semantic flaws. The remaining 1.98% of committed turns (28/1,414, where both prefix and response are bad) represents the irreducible cost of selective handoff.

A.10 Turn-Taking Performance

Because the fast-path S2S model is responsible for all real-time dialogue control in RelayS2S, we separately evaluate its turn-taking accuracy on the held-out set. For each control token we compute precision, recall, and F1 with a tolerance window of one frame (± 160 ms). Table 7 reports the results.

The model achieves strong performance on the three core turn-management events: stay-silent F1 of 99.7% indicates that the model rarely speaks out of turn, start-speaking recall of 96.1% shows it reliably detects response opportunities, and stop-speaking F1 of 93.2% confirms robust interruption handling. Backchannel prediction is harder (F1 48.4%), with high recall (72.7%) but lower precision (36.3%), reflecting the inherent subjectivity of when a backchannel is appropriate; we therefore treat it as auxiliary behavior rather than a primary evaluation target.

A.11 Prefix verification Ensemble Details

As detailed in Section 2.4, the prefix verification training dataset labels are constructed using a model ensemble strategy. The ensemble comprises three distinct large language models spanning varied parameter scales and architectures: Gemma-4-

31B[1], Qwen3.6-27B[2], and GPT-OSS-120B[3].

Each ensemble worker executes up to 3 retries under a strict JSON enforcement configuration using the specific rubric prompt detailed below. The final ground-truth training label is determined via a majority vote.

Verification Ensemble Prompt

```
You are evaluating whether a response
    PREFIX (the opening of an assistant
    reply) is a sensible way to begin
    replying to the user's latest
    message.

You will see:
- The conversation context.
- The PREFIX under evaluation. This may
    be as short as one word or as long
    as a couple of sentences.
- A continuation that a strong model
    wrote starting from that prefix,
    doing its best to produce a sensible
     reply.

The continuation is EVIDENCE about
    whether the prefix forced the model
    into a bad direction. You are NOT
    grading the continuation. A
    continuation can hallucinate, drift,
     or fail on its own -- that does not
     make the prefix bad unless the
    prefix itself caused or committed to
     the failure.

--- WHAT MAKES A PREFIX NOT SENSIBLE ---

A prefix is NOT SENSIBLE only if it
    CONCRETELY commits the response to a
     bad direction. Specifically:

1. Ungrounded stance. The prefix affirms
    , denies, or agrees with something
    the user did not actually say or
    establish.
   - Affirmative: "Yes" / "Yeah" / "That
    's right" / "Correct" / "Exactly"
    with no clear proposition to agree
    with.
   - Negative: "No" / "That's wrong"
    when the user did not make a claim
    to reject.
   - Refusal: "I can't help with that" /
     "I'd rather not" when the request
    is benign.
   - Sycophantic agreement with a user
    claim that is factually wrong.

2. Stated incorrect content. The prefix
    itself asserts a wrong fact, a
    fabricated source, an invented
    entity, or a misattribution. The
    wrong content must be visible in the
     prefix's OWN WORDS -- do not infer
    it from the continuation. "The
    Eiffel Tower, built in" has not
    committed to a date; "The Eiffel
    Tower, built in 1923," has.

3. Misread of the user. The prefix locks
     in a topic, intent, or framing that
     does not match the user's latest
    message. A reframe ("The underlying
    question here...", "Before answering
    ...") is fine if it can plausibly
    land on the user's actual concern.

4. Constraint or persona violation. The
    user set an explicit rule (one-
    syllable words only, JSON only,
    calculator persona, specific format)
     and the prefix already breaks it.

5. Wrong refusal direction. The prefix
    refuses a benign request, OR begins
    to comply with a request that should
     be refused on safety grounds. A
    prefix that asks for clarification
    on a genuinely ambiguous or unsafe
    request is sensible.

--- WHAT DOES NOT MAKE A PREFIX NOT
    SENSIBLE ---

- The prefix is short and uncommitted ("
    Sure, " / "Here's " / "Let me " / "
    Great, " / "Okay, so "). If nothing
    specific has been committed to yet,
    default to SENSIBLE.
- The continuation hallucinates a fact,
    cites a fake source, or wanders off
    topic, but nothing in the prefix
    forced that. Attribute the failure
    to the continuation, not the prefix.
- The opener is stylistically bland,
    hedgy, or could be better written.
    Sensibility is about not foreclosing
     a good reply, not about being the
    best possible opener.
- The prefix is grammatically incomplete
    . A good continuation can finish the
     sentence.

--- HOW TO DECIDE ---

Ask: could a careful, well-informed
    continuation have followed this
    prefix and produced a correct, on-
    topic, appropriately-toned reply?
- If yes -> SENSIBLE, even if the
    continuation you actually see failed
    .
- If no, because the prefix itself
    states something wrong, agrees with
    nothing, refuses in the wrong
    direction, or violates a constraint
    -> NOT SENSIBLE.

When uncertain, default to SENSIBLE.
    Reserve NOT SENSIBLE for cases where
     you can point to specific text in
    the prefix that does the committing.

-----
```

[1]https://huggingface.co/google/gemma-4-31B-it
[2]https://huggingface.co/Qwen/Qwen3.6-27B
[3]https://huggingface.co/openai/gpt-oss-120b

```
Conversation:
{{ context }}
-----
PREFIX (under evaluation): "{{ prefix
    }}"
-----
Continuation that followed: {{
    continuation }}
-----

Reply with valid JSON only, in this
    exact shape:
{"sensible": true|false}
```

A.12 Slow-Path Prompt Templates

A.12.1 Prefix-Conditioned Generation and Repair

When the prefix verifier commits a fast-path draft, the downstream slow-path LLM is invoked to complete the response. The generation is forced to decode directly from the committed token history and is explicitly instructed to self-correct any flaws that leaked through the fast path.

Prefix-Conditioned Generation Prompt

```
You are continuing a conversation in a
    real-time dialogue system.

### CONVERSATION HISTORY:
{{ history_str }}

### TASK:
The Assistant has already started
    speaking their next response. The
    initial part of the response (the
    prefix) has already been played
    aloud to the user and cannot be
    taken back. Your task is to continue
     the response starting from the
    exact point where the prefix ends.

### CORRECTION AND CONTINUATION
    GUIDELINES:
1. Assume the prefix has already been
    heard by the user exactly as it is
    written.
2. If the prefix is contextually
    appropriate and correct, continue it
     smoothly to complete the response
    naturally.
3. If the prefix contains a factual
    error, hallucination, repetition, or
     is contextually inappropriate, you
    must perform a graceful self-
    correction or conversational repair
    within the continuation (e.g., by
    saying "... wait, I mean...", "...
    actually, let me correct that...",
    or pivoting to the right answer
    naturally) without breaking the flow
    .

### OUTPUT FORMAT:
Output ONLY the continuation of the
    response. Do not repeat the prefix.
    Do not include labels, markdown
    conversational fillers, or any meta-
    text.

### PREFIX TO CONTINUE:
{{ forced_prefix }}
```

A.12.2 Standard Fallback Generation

On verifier fallback, the system relies on unconstrained response generation from the back-end cascade pipeline utilizing the following prompt template:

Standard Fallback Generation Prompt

```
You are generating a response in a real-
    time dialogue system.

### CONVERSATION HISTORY:
{{ history_str }}

### TASK:
Generate the Assistant's next response
    based on the conversation history
    above. The response will be played
    aloud to the user in real time, so
    it should sound natural when spoken.

### RESPONSE GUIDELINES:
1. Respond directly and naturally to the
     user's most recent message.
2. Ensure the response is factually
    accurate, contextually appropriate,
    and consistent with the tone and
    style established in the
    conversation history.
3. Avoid hallucinations, unnecessary
    repetition, or filler content.
4. Keep the response conversational and
    suitable for being spoken aloud.

### OUTPUT FORMAT:
Output ONLY the Assistant's response. Do
     not include labels, markdown,
    conversational fillers, or any meta-
    text.
```

A.13 Real Voice Filtering Criteria

To filter uncooperative, incomplete, or purely narrative text from raw ASR corpora, we employ Gemini-3-Flash. The configuration leverages explicit criteria to retain only self-contained prompts suitable for an AI assistant:

Real Voice Filtering Prompt

```
Keep ASR utterances that are BOTH (1)
    fully context-independent and
(2) plausibly something a user would say
     to an AI assistant.

CORE TEST - reject if any is no:
(a) Are all WHO/WHAT/WHEN/WHICH clear
    from the sentence alone?
```

```
(b) Does it read like a real utterance (
    question, request, opinion,
    standalone fact) - not a mid-article
     fragment?
(c) If addressed to "you," could it
    sensibly target an AI (which has
    no location, family, age, job,
    hometown, hobbies, feelings, or
    personal history)?

REJECT - personal probes at the listener
Questions/remarks about the listener's
    life, identity, relationships,
or experiences. Nonsensical when aimed
    at an AI.
  - Origin/status: "Where are you from
    ?", "Are you married?"
  - Greetings-as-questions: "How are you
     doing?"
  - Personal plans/reactions: "Where are
     you going for spring break?",
    "Drums are awesome."
  - Tone probes: "Are you being
    sarcastic?"
  - Opinion solicitations as turn-taking
    : "...do you?"
(Generic AI-answerable questions like "
    What is the capital of France?"
are fine.)

REJECT -- context-dependence
  - Unresolved pronouns (he/she/it/they/
    his/her) unless antecedent is
    a globally famous entity named in-
    sentence.
    Reject: "Chris Hattingh filled his
    seat."
  - Bare names (first or last alone, or
    full names of obscure people)
    as subjects/possessives. Single-name
     refs are NEVER globally
    identifiable.
    Reject: "Navarro is fluent in
    Spanish, French and English."
    Accept: "Einstein's theory of
    general relativity..."
  - Unresolved demonstratives (this/that
    /these/those/there).
    Reject: "In this sense, Jesus
    becomes a God-Man."
  - Discourse connectives implying prior
     context: then, also, instead,
    too, however, moreover, another,
    finally, for the first time.
    Reject: "PowerShell uses backtick
    instead."
  - Definite "the X" where X isn't
    specified in-sentence. Allow if
    fully specified ("the capital of
    France") or unambiguous generic
    ("the immune system").
    Reject: "The park is featured in an
    episode of K-On!."
  - Fragments, headlines, titles,
    isolated quotes, maxims, captions.
    Reject: "Delegata potestas non
    potest delegari."

REJECT -- encyclopedia narration
Even if referents resolve, reject
    Wikipedia-style prose:
  - Dry biographical reporting:
    "John Smith was born in 1842 in a
    small Yorkshire village."
  - Infobox stat-dumps, transit/schedule
    /operator facts:
    "The 1998 Honda Civic has a 1.6-
    liter engine and 106 horsepower."
  - Plot/episode summaries, credits-
    style sentences.
  - "X is a [type] in/at [location]"
    landmark/institution openers:
    "Breed's Hill is a glacial drumlin
    in Charlestown, Boston."

ACCEPT -- self-contained AND prompt-like
  - AI-answerable questions: "What is
    the boiling point of mercury?"
  - Requests: "Write a haiku about
    autumn."
  - First-person opinions: "I think
    Mondays are the worst day."
  - Generic claims: "Photosynthesis
    converts light into chemical energy
    ."
  - Facts about globally identifiable
    entities:
    "The Eiffel Tower was completed in
    1889."

When in doubt, REJECT.

Output one line of JSON only:
{"is_standalone_prompt": true|false}

Transcription:
"""
{{ transcription }}
"""
```

A.14 Evaluation Rubric for Textual Response Quality

Evaluation Rubric Prompt

```
You are an expert AI quality auditor.
    Score the assistant's LAST RESPONSE
    based on the provided CONTEXT.

SCOPE:
- Grade ONLY the assistant's last
    response. Earlier turns are context,
     not subjects.
- If the user's request is ambiguous or
    unanswerable, a graceful
    clarification or
  honest refusal can score up to 5,
    provided it is accurate and well-
    phrased.

DIMENSIONS (consider all four; do not
    output them separately):
1. Relevance -- directly addresses the
    request without drifting off-topic.
2. Helpfulness -- provides useful,
    complete, actionable value for the
    request.
```

```
3. Accuracy -- free of hallucinations,
    contradictions, invented details,
    overstatements.
4. Clarity -- well-organized and easy to
     follow in the stated modality.

HARMFUL CONTENT OVERRIDE:
If the response contains content that is
     harmful, dangerous, or unsafe (and
    the
request did not justify it), score = 1
    regardless of other dimensions.

ACCURACY CAP RULES (apply after the harm
     check):
- CLEAN: no factual mistakes anywhere.
    No cap; can reach 5.
- SELF-CORRECTED: a mistake appears
    earlier in the response but is
    explicitly
  repaired later, leaving the listener
    with the right answer. Cap = 4.
  Earlier, smoother repairs sit near 4;
    late or clumsy repairs sit lower.
- UNCORRECTED: a factual error,
    hallucination, or misleading claim
    remains
  standing or is glossed over without
    acknowledgement. Cap = 3.

AGGREGATION INTO A SINGLE SCORE (1  5
    integer):
Start from the accuracy cap, then adjust
     DOWN for weaknesses in relevance,
helpfulness, or clarity. Do not adjust
    UP past the cap.

5 = Excellent. Fully addresses the
    request; accurate; clear; no
    mistakes
    that required repair. Relevance,
    helpfulness, and clarity all strong.
4 = Good. Mostly complete, relevant, and
     accurate, with small gaps or minor
    clarity issues. OR: the ceiling for
    a smooth, early self-correction.
3 = Fair. Surface-level answer with
    minor uncorrected issues, OR a
    clumsy
    self-correction that left the
    listener confused even though the
    final
    answer was right, OR notable gaps in
     helpfulness/relevance.
2 = Poor. Partially relevant but missing
     key content, unclear, or contains
    notable uncorrected errors that
    affect the takeaway.
1 = Very poor. Fails the request,
    materially inaccurate, confusing, or
     harmful.

OUTPUT FORMAT:
Return ONLY this JSON, with no
    surrounding prose, no code fences,
    no commentary:
{"score": <integer 1-5>}

--------------------
CONTEXT:
{{ context }}
--------------------
RESPONSE (to be graded):
Assistant: {{ response }}
```