
MuDuSi: A Multi-Agent Acoustic Simulator for Full-Duplex Speech Language Models

Kyudan Jung Jaegul Choo
KAIST AI

Technical Report MUDUSI-TR-2026-01

October 13, 2026 · Version 1.2

<https://kyudan1.github.io/MuDuSi/paper/>

Abstract

MuDuSi (Multi-agent Duplex Simulator) is a virtual acoustic testbed for independently running full-duplex speech models. It couples simultaneous listening and speaking through a 24 kHz PCM engine with spatial attenuation, propagation delay, occlusion, reflection, and microphone noise. Aligned source, receiver, and model-input recordings support frozen-model evaluation and acoustic trajectory collection. We describe the interaction and timing contracts and report NVIDIA B200 experiments with PersonaPlex, Moshi, and Raon. Paired conversations completed at real-time input pacing. A larger experiment ran all nine ordered model pairs concurrently as eighteen isolated workers on four GPUs, recording 60 audio seconds per room over approximately 240 wall-clock seconds. All eighteen agents generated audio; routing checks passed, with no sustained underruns, input overflows, output drops, or clipping. This is a reduced-pacing collection result, not a real-time throughput claim. Offline ASR review nevertheless exposed substantial speech-quality problems in both Moshi–Raon pairings. We distinguish operational validity from conversational competence and acoustic recordings from token-policy rollouts. Prospective applications include HiPLEX-style hierarchical policy evaluation, overlap-aware benchmarks, and reinforcement learning; full speech-model policy training is not implemented.

1 Introduction

Full-duplex speech models can listen while speaking, producing overlaps, backchannels, and interruptions that alternating-turn text harnesses cannot represent directly. In a shared room, each participant also hears a different signal: distance, occlusion, and noise transform the sound emitted by other participants. Studying these interactions with physical devices can require a separate microphone/loudspeaker setup for each agent and repeated room configuration.

MuDuSi provides a software testbed before such deployment. Independent model sessions exchange audio through a common acoustic engine, while a 3D observer exposes source activity, local microphone mixtures, and timing. Its two workflows are frozen-model inference/evaluation and acoustic trajectory collection for reinforcement-learning (RL) research. Recorded experiments can be replayed without keeping GPUs allocated. This reduces dependence on physical devices for software experiments, but does not replace hardware or perceptual validation.

The contribution is an executable interaction substrate: simultaneous PCM actions, explicit receiver-to-input alignment, bounded asynchronous transport, and inspectable audiovisual records. We report engineering measurements from real model runs rather than a semantic benchmark. Full-weight speech-language model (SLM) RL is not implemented; the included trainable example is a

small stochastic gate. The reference implementation and operational instructions are maintained at <https://github.com/KyuDan1/MuDuSi>.

Our design separates three questions often conflated in a demonstration: whether a model emitted samples, whether those samples reached another model through the intended acoustic path, and whether the resulting interaction was useful. Each question requires different evidence. A visually active scene cannot answer the second without aligned recordings, and zero buffer errors cannot answer the third without perceptual and task evaluation. The experiments deliberately retain unsuccessful cases alongside selected presentation clips. Appendix A shows four recorded demonstration views, complementing the quantitative results.

2 Related work

Speech models. Moshi jointly models incoming and outgoing speech streams, with time-aligned text generation [Défossez et al., 2024]. PersonaPlex adds role and voice conditioning to a Moshi-derived full-duplex model [Roy et al., 2026]. MuDuSi connects independent instances of such models through spatial microphone mixtures; it does not propose a new speech architecture. Whisper [Radford et al., 2023] is used as an external recognizer, with its output kept separate from native model text.

Simulation and multi-agent interfaces. MuJoCo supplies a physics engine designed for model-based control [Todorov et al., 2012]. MuDuSi shares the motivation of testing interacting policies in simulation, but models acoustic exchange rather than rigid-body/contact dynamics and does not use MuJoCo as its backend. PettingZoo standardizes multi-agent environment interaction [Terry et al., 2020]; MuDuSi’s parallel dictionaries follow a similar interface convention, without claiming PettingZoo registration or compliance.

Acoustic environments. Pyroomacoustics provides room impulse-response simulation and audio-array processing tools [Scheibler et al., 2017]. SoundSpaces supplies geometrical acoustic renderings for audio-visual navigation in 3D environments [Chen et al., 2020]. MuDuSi’s present acoustics are deliberately simpler. Its focus is the closed interaction loop between independently stateful speech generators, live buffer diagnostics, and aligned source/receiver/input recordings. Neither richer room simulation nor spatial visualization alone establishes a validated speech-agent benchmark.

Dialogue evaluation and acoustic control. Full-Duplex-Bench evaluates turn-taking capabilities, while its v1.5 extension emphasizes overlap handling [Lin et al., 2025a,b]. These motivate event-level evaluation beyond our current energy detector: an overlap may be a useful backchannel, an intrusive interruption, or simultaneous unrelated speech. OmniACBench distinguishes contextual acoustic control from textual correctness [Kim et al., 2026]. MuDuSi currently reports neither those benchmark scores nor validated acoustic-style judgments. Its trajectories provide candidate inputs to such evaluators, with the emitted and received versions kept distinct.

Hierarchical policies and speech pipelines. HiPLEX factors full-duplex policy decisions into control and conditional content components [Jung et al., 2026c]. This offers a concrete future integration target because acoustic timing and linguistic content can receive different credit. Sommelier addresses scalable multi-turn audio preprocessing [Jung et al., 2026a]; SLICE studies speech enhancement through layer-wise conditioning [Moon et al., 2026]; and speculative execution can hide tool latency in cascaded voice agents [Jung et al., 2026d]. These are related research directions, not components already integrated or evaluated in MuDuSi. Section 7 specifies the additional contracts needed for each.

3 Method: the simulation and recording contract

3.1 Closed-loop partially observed interaction

Let $\mathcal{I}$ denote the active agents in one room. At decision index t , agent i observes $o_t^i = (u_t^i, p_t^i, t\Delta)$, where $u_t^i \in [-1, 1]^B$ is its local microphone block, p_t^i is its own position, and $\Delta = B/f_s$. A conceptual recurrent policy produces a PCM action $x_t^i \in \mathbb{R}^B$ from its local history $h_t^i = (o_{0:t}^i, x_{0:t-1}^i)$.

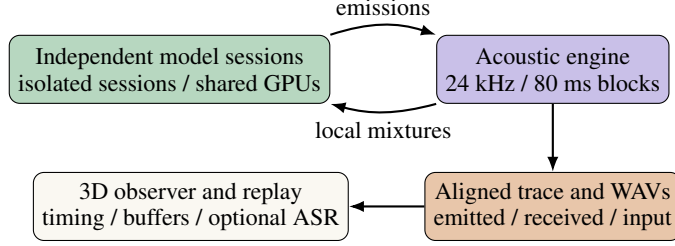


Figure 1: All sessions may emit and receive concurrently. The observer does not drive simulation time. Clean source audio is available for analysis, not substituted for a peer’s microphone input.

In a native speech worker, text and acoustic tokens are internal decisions that are not yet exposed as environment actions. The acoustic transition is

$$(s_{t+1}, y_t) = F(s_t, \{x_t^i\}_{i \in \mathcal{I}}, \xi_t), \quad u_{t+1}^i = \text{clip}(y_t^i, -1, 1), \quad u_0^i = 0. \quad (1)$$

The simulator state s_t contains scene geometry, in-flight acoustic queues, noise-generator state, and engine counters; ξ_t represents microphone noise. A Markov description of the entire deployed system would additionally include each model’s recurrent state and transport queues. Those hidden states are not exposed to peer actors. The finite observation history generally does not determine the full state, making a partially observed multi-agent formulation appropriate.

All emissions for a tick enter one joint transition. Iterating over agents in software must not let a later agent observe an earlier agent’s current-tick action. This one-block feedback contract removes order-dependent lookahead and makes the model-input recording auditable. It also introduces an explicit 80 ms decision boundary in addition to physical propagation and codec delays. This discretization is an engineering choice, not a claim that human conversation or every native decoder operates at 80 ms resolution. Simulator snapshots can expose source attribution for debugging and future centralized critics; the actor observation does not receive clean peer signals or peer transcripts.

3.2 Acoustic dynamics and timing

The engine accepts simultaneous mono float32 emissions of $B = 1,920$ samples at $f_s = 24,000$ Hz, an 80 ms decision interval. Policies consume the previous completed receiver block; current emissions jointly determine the next observations. Initial model input is silence. For source i and receiver j , direct pressure gain and delay are

$$g_{ij} = \frac{1}{\max(1, d_{ij})} \prod_{k \in \mathcal{O}_{ij}} 10^{-L_k/20}, \quad \tau_{ij} = \lceil d_{ij} f_s / 343 \rceil, \quad (2)$$

$$y_j[n] = \sum_{i \neq j} g_{ij} x_i[n - \tau_{ij}] + r_j[n] + \epsilon_j[n]. \quad (3)$$

Here d_{ij} is distance in meters, L_k is an intersected divider’s loss in dB, and ϵ_j is independent Gaussian microphone noise with RMS $10^{\eta/20}$. There is no own-source path. Delay queues cross block boundaries. Geometry is evaluated at emission time, so moving an agent does not rewrite sound already in flight. The optional r_j is one right-wall image-source reflection with gain $0.45\rho / \max(1, d_{ij}^{\text{refl}})$. The reflection control ρ is not a calibrated RT60. Raw superposition remains unclipped; WAV encoding and live inputs clamp to $[-1, 1]$ and expose clipping counters.

Headless CPU steps advance without wall-clock pacing. Live evaluation uses a configurable wall-clock pacing interval and independent asynchronous worker send/receive loops. Bounded queues expose underruns, overflow, dropped samples, and final queued audio. Slow observers lose stale visual updates rather than backpressuring the acoustic loop. Physical propagation, block decision delay, codec buffering, and model inference are distinct latency sources. A fixed acoustic seed does not make asynchronous GPU trajectories bitwise reproducible.

3.3 Three clocks and finite-horizon boundaries

We distinguish sample time, simulation time, and wall time. Sample index n advances at the declared 24 kHz rate within saved audio. Simulation time is $t\Delta$, indexing simultaneous input/output blocks

and acoustic events. If input pacing is $v \in (0, 1]$, the evaluator targets wall deadline

$$D_t = D_0 + t\Delta/v. \quad (4)$$

Thus $v = 0.25$ submits one 80 ms audio block every 320 wall milliseconds. A 60-second WAV then takes approximately 240 seconds to collect, but remains a normal-speed 24 kHz recording. This gives colocated workers more compute time between inputs; it neither slows the waveform nor establishes real-time capacity. Wall-clock scheduler lateness is measured against D_t , with a deadline overrun counted when lateness exceeds Δ/v . Startup/model loading lies outside the recording interval.

The distinction matters for both throughput and behavior. An agent may produce a block after an acoustic event’s nominal time because native buffering and compute introduce additional delay. Lower pacing can change that availability and therefore the closed-loop trajectory; it is not guaranteed to preserve a real-time conversation. We report pacing, wall duration, queue counters, and audio activity together. We do not interpret GPU utilization alone as conversational latency. At a fixed horizon, samples still queued beyond the endpoint are recorded as a boundary condition rather than silently treated as an additional completed turn.

3.4 Inference, trajectories, and visualization

Each participant has a separate endpoint and model state. Two or three workers can converse in one scene; a WAV worker can instead provide controlled stimulation. The current engine has three named slots, not arbitrary agent counts. Independent experiments use separate session state, workers, endpoints, and output directories. A matrix runner coordinates nine room evaluators concurrently, and a matrix dashboard replays their recordings together. This is distinct from a multi-user live experiment scheduler: the dashboard does not share one speech session among rooms or remotely control an allocated GPU job.

The evaluator saves one JSONL snapshot per tick and three PCM16 WAVs per slot: emitted audio at the mixer boundary, the spatially mixed receiver signal, and the previous receiver block supplied to the adapter. Summaries preserve completion/failure, elapsed time, scheduler lateness, and final diagnostics. Partial failures remain identifiable. This same evaluator supplies an acoustic trajectory; there is no separate token-rollout mode.

The parallel Python environment returns each agent’s microphone PCM, own position, and time, plus acoustic rewards and boundary flags. All agents act simultaneously. A fixed horizon is truncation, not natural task termination. Rewards combine source-attributed audibility, an overlap penalty, and emission energy. They are development proxies: useful backchannels may incur a penalty, and audibility is not comprehension. A four-parameter Bernoulli listen/speak gate illustrates REINFORCE [Williams, 1992], logging behavior probabilities and policy versions with parameters frozen during each rollout. It emits synthetic tones and does not update SLM weights.

The 3D viewer shows transmitted and received RMS, activity, propagation labels, and buffer timing. dBFS is not calibrated sound pressure, and activity is a -65 dBFS energy detector. Animated wave rings deliberately slow apparent propagation. Engineering replay plays actual receiver audio; presentation videos mix clean emissions for intelligibility. Their speech bubbles are offline Whisper ASR on emitted audio, revealed at word timestamps, not native text or measured streaming-ASR latency.

4 System architecture and observability

4.1 Model adapters and backpressure

A worker adapter separates native model execution from the acoustic interface. PersonaPlex and Moshi use separate environments despite sharing the Python import name `moshi`; every participant also has its own process and endpoint. Separating environments resolves package collisions, while separating processes resolves native session/state ownership. A shared GPU is therefore a compute placement decision, not a shared conversational memory. The adapter converts native transport audio into mono PCM before the engine accepts an emission.

Independent send and receive loops avoid serializing model generation behind synchronous acoustic stepping. The PCM output pull fills unavailable samples with zeros and counts underrun frames and

samples. An input queue overflow raises an error rather than silently dropping observations. The default bounded input queue contains 32 blocks; output buffering permits 64 blocks before dropping oldest samples and incrementing explicit counters. Text buffering is separately bounded. These policies make failures finite and visible, but do not repair insufficient throughput: zero filling can preserve a deadline while damaging the conversation. Startup underruns are reported separately from sustained underruns, never erased from totals.

Transport readiness alone is insufficient for synchronized experiments. The matrix runner first launches eighteen workers in bounded startup batches, connects their adapters, and prepares WAV writers. Ready evaluators then cross an in-process start barrier. Pre-barrier and post-barrier counter snapshots expose any audio or queue activity that occurred during initialization. Failure in one room is recorded without silently converting it to an echo model or successful room; other rooms can finish, while the overall experiment remains unsuccessful if required rooms fail. The job terminates its worker processes after the finite experiment.

4.2 Artifact contract and independent checks

Three audio streams serve different diagnostic purposes. The emitted WAV captures the model's decoded output at the mixer boundary, so it can expose synthesis defects before spatial rendering. The received WAV captures the local acoustic mixture and its noise. The input WAV captures the clipped previous mixture supplied to the adapter. Comparing input with the previous receiver block tests feedback alignment; comparing emitted and received content diagnoses spatial degradation. Equality after PCM16 encoding verifies the recorded interface boundary, not the exact post-Opus waveform inside a native model.

A JSONL trace contains one snapshot per simulation tick, with agent activity, generated text when exposed, acoustic links, heuristic rewards, worker counters, and wall timing. Summaries preserve completion status, the error if any, recording duration, lateness statistics, and final queues. Matrix manifests add worker placement, independent endpoints, configuration, readiness, source revision, and recording intervals. Device telemetry is summarized over the common recording interval rather than over idle loading time. Raw manifests can contain private paths and prompts and are not automatically publication artifacts.

Verification has three layers. Structural checks test expected room membership, contiguous ticks, WAV format and length, peer paths, silent unused slots, and input/receiver equality. Runtime-health checks examine startup and later underruns, queue loss, clipping, and deadline overruns. Content review considers emitted speech, receiver intelligibility, task accuracy, and interaction quality. The present implementation automates the first two; offline ASR only supplies fallible evidence for the third. This separation prevents a successful process exit or a lively animation from becoming an unsupported conversation-quality score.

4.3 Observer, ASR, and recorded multi-room playback

The live observer renders a single engine's geometry and counters without setting its clock. Separate independent runs can be observed through separate server processes and ports. The matrix replay view shows nine saved rooms on a shared simulation clock, maintaining each room's independent scene and text history. A selected audio focus prevents the soundtrack from combining unrelated rooms. Changing focus changes which existing room is heard at the current timestamp; it does not splice separately timed episodes into a fictitious simultaneous conversation.

Native generated text, emitted-audio ASR, and receiver-audio ASR answer different questions and remain separately labeled. Optional receiver recognition operates on buffered windows, not instantaneous partial hypotheses. Presentation captions use offline word timestamps and reveal recognized words during playback; their apparent streaming appearance is not a latency measurement. Source identity comes from the simulator's routing metadata. It must not be interpreted as diarization inferred from a mixed microphone signal.

Table 1: Measured paired conversations. Overlap counts the two models, excluding the stimulus. Handler timing is not end-to-end response latency.

Measurement	Near / quiet	Distant / cocktail
Slurm job	189794	189798
Audio / wall duration (s)	60 / 60.011	60 / 60.007
Scheduler lateness, p95 (ms)	1.45	2.04
Tick overruns / input overflow / dropped samples	0 / 0 / 0	0 / 0 / 0
PersonaPlex / Moshi startup underrun frames	1 / 2	1 / 2
Moshi handler mean / p95 (ms)	24.59 / 26.15	24.19 / 26.20
Both models active (s)	3.12	6.56
Direct model-to-model pressure gain (dB)	0	−29.12

5 Experiments and results

5.1 Setup and measurements

Experiments ran on NVIDIA B200 GPUs (183,359 MiB each), driver 580.95.05, using isolated Slurm allocations. Paired runs requested two GPUs, 16 CPUs, and 96 GiB RAM; each model occupied one GPU. PersonaPlex and Moshi used separate Python environments because both install a package named `moshi`. The exercised model paths used Torch 2.8.0/CUDA 12.8 and an NVIDIA PyTorch 25.04 worker image for runtime compilation. PersonaPlex’s reused environment had dependency conflicts outside the tested inference path; successful runs do not certify that entire environment.

Two 60-second (750-frame) PersonaPlex–Moshi experiments used the same spoken rice-cooking WAV stimulus in the third slot. The near condition used 1 m model separation, no divider, and −72 dBFS microphone noise. The distant condition used 5.70 m separation, a 14 dB divider, and −35 dBFS noise. These jointly changed factors are not an acoustic ablation. We measured pacing, buffer health, activity overlap, and source/receiver/input consistency. Timing starts after worker readiness, excluding startup and scheduling.

5.2 Routing and runtime evidence

For every receiver in both runs, recorded input at tick t exactly matched PCM16-encoded received audio at $t - 1$, with initial silence. Re-rendering saved emissions through the mixer reproduced received WAVs within maximum normalized amplitude errors of 9.2×10^{-5} and 4.5×10^{-5} , respectively; the small discrepancies reflect source/WAV quantization. This verifies delivery to the adapter, with native transport encoding remaining a separate codec stage.

Both experiments completed at wall-clock rate (Table 1), with no counted ADC or decoder clipping. Underruns occurred before initial generated samples arrived; buffers were not prefilled. One 80 ms block remained queued at the fixed endpoint. Near-run monitoring reported approximately 17,839 MiB for Moshi and 19,625 MiB for PersonaPlex, 36.4 GiB host working memory, and no OOM or CPU throttling. These are sampled readings, not allocator peaks or training-memory estimates. Paced completion does not measure maximum throughput.

Receiver ASR recognized cooking-answer content in the peer’s microphone recording. The second stimulus began while Moshi spoke, with overlap and later stop/resume visible in the traces. This establishes an interruption stimulus and activity, not causal intentional yielding. Noisy receiver ASR also hallucinated repetitions; neither recognition nor native text is ground-truth comprehension.

5.3 Model-only conversations and failure cases

Additional runs used no prerecorded speaker (Table 2). The cafe workers received clerk/customer roles, voice prompts, and menu context, not scripted dialogue. The selected passage contains an iced oat-latte and croissant order and confirmation. The full run includes initial role confusion and an incorrect price. Longer friends and three-agent conversations contain monologues and repetitive backchannels; selecting favorable clips introduces selection bias.

Table 2: Additional independent conversations; all had zero input overflow, output drops, and clipping. Three participants comprise two model families.

	Friends	Clerk / customer	Three agents
PersonaPlex / Moshi instances	1 / 1	2 / 0	2 / 1
Slurm job	189846	189891	189845
Full recording (s)	90	60	90
Scheduler lateness, p95 (ms)	1.57	1.69	1.26
Startup underrun frames	1 + 2	1 + 1	1 + 2 + 1
Selected interval (s)	0–26.92	9.50–30.25	0–14.92

All connected model inputs matched their previous receiver blocks. Full-run overlap measured 10.96 s for friends and 48.08 s for three agents; the selected intervals contain 3.76 s and 6.96 s. These are activity statistics, not dialogue-quality scores.

The direct Raon PyTorch bridge generated audio/text but averaged 207 ms per 80 ms frame and filled its 32-frame input queue after 4.24 simulated seconds (job 189796). An optimized SGLang path, after CUDA runtime/library-path fixes, completed 60 s in 60.012 s with 1.68 ms p95 lateness and zero overflow/drops (job 189820). Raon had six startup underrun frames and 480 ms queued at the endpoint. Monitoring reported about 90,085 MiB for Raon, including preallocated cache. A warmup mismatch and missing intended sliding-window behavior in an attention fallback remained. Later generated audio diverged substantially from native text and contained garbled phrases. These are unresolved quality failures; the warnings are candidate causes, not established causal explanations.

5.4 Nine independent rooms on four GPUs

The larger experiment instantiates the Cartesian product of three families, Moshi, PersonaPlex, and Raon, as nine ordered two-agent rooms. A family appearing in six positions uses six independent worker processes; eighteen workers participate in total. Left/right order affects role and initiation conditioning, so reverse pairings are retained. The third engine slot is muted in every room. No prerecorded stimulus supplies a hidden third speaker. The experiment evaluates concurrent model instances, not eighteen agents sharing one acoustic room.

Job 190030 used four B200 GPUs, 48 CPUs, and 512 GiB host RAM. GPUs hosted 5, 5, 4, and 4 model processes. Every room collected 750 frames at $v = 0.25$, corresponding to 60 audio seconds and 240.013–240.198 wall seconds. Start timestamps spanned 9.636 ms; all nine recording intervals overlapped for 240.004 seconds. This shared interval and the distinct process/endpoint records establish concurrency more directly than a tiled video alone.

All eighteen workers emitted nonzero audio, and every input recording matched its previous receiver block with initial silence. Peer acoustic paths were present on every tick and unused sources remained silent. Table 3 reports the startup and sustained output availability. Every room had zero input overflow, dropped output samples, dropped text messages, decoder clipping, ADC clipping, and deadline overruns. Each participant had one or two initial underrun frames; none occurred after the first simulation second, which is four wall seconds at this pacing. Final queues retained at most two 80 ms blocks beyond the recording boundary. Both startup-counter snapshots were empty of queued input and received samples, so no hidden pre-recording loss was removed from the measurement.

Table 4 reports the placement’s sampled memory and utilization. Raon’s explicit token-cache cap was 8,192, with at most four running requests and four allocated request-pool indices. Its configured static memory fraction of 0.85 is an availability/budget setting; with an explicit cache cap it does not mean each process allocated 85% of device memory. These measurements include weights, runtime workspaces, and all colocated processes. They do not estimate optimizer memory or demonstrate sufficient headroom for training.

5.5 What the larger run did not solve

Earlier real-time baseline. Job 189994 ran the same nine ordered family pairs on six B200s at $v = 1$, placing one worker of each family on each GPU. All eighteen agents emitted audio and structural routing passed. However, each Raon worker had 222–263 underrun frames out of

Table 3: Four-GPU matrix, job 190030. M: Moshi, P: PersonaPlex, R: Raon. L/R entries follow room order. “Later” means after the first simulation second. All deadline overruns are zero relative to a 320 ms wall tick.

Room L/R	Startup underruns L/R	Later underruns L/R	Lateness p95 (ms)
M / M	2 / 2	0 / 0	1.77
M / P	2 / 1	0 / 0	15.62
M / R	2 / 2	0 / 0	15.84
P / M	1 / 2	0 / 0	15.14
P / P	2 / 2	0 / 0	28.91
P / R	1 / 2	0 / 0	25.89
R / M	2 / 2	0 / 0	13.82
R / P	2 / 1	0 / 0	22.90
R / R	2 / 2	0 / 0	16.27

Table 4: Device-wide telemetry over the common recording interval of job 190030: 45 samples per GPU. Maximum sampled usage is not an exact allocator peak.

GPU	Processes	Mean GiB	Maximum GiB	Mean utilization
0	5	98.590	98.669	52.38%
1	5	98.589	98.669	42.56%
2	4	78.252	78.296	33.56%
3	4	78.248	78.296	35.09%

750, including 215–257 after the first second. Moshi’s underruns were confined to startup; three PersonaPlex instances had one later underrun each. Each room also recorded four to six deadline overruns, despite p95 lateness below 19 ms. Input overflow, output/text loss, and clipping remained zero. Thus low percentile lateness and zero drops did not establish adequate audio throughput. Mean GPU utilization during the common interval was 93.33–95.50%, with approximately 98.25–98.47 GiB used per device.

The four-GPU run changed pacing, placement, Raon cache limits, and a spawned-decoder compatibility path together. It is not a causal ablation of any one change. The baseline’s opt-in Mimi attention-window change reached only the parent process; a freshly spawned decoder did not inherit it. In job 190030 all six decoder-child logs confirmed installation of the window-mask bridge. This repairs an identified process-boundary integration defect, but does not prove that this defect caused the baseline’s runtime or speech-quality failures. Both runs retain their own provenance; a later code commit is not substituted for the baseline’s uncommitted working tree.

Compilation warnings. All six Raon workers in the four-GPU run logged an upstream audio-module compile-warmup failure twice, concerning a mismatch between masked positions and audio-embedding element counts. The runtime caught the exception and continued. Successful sample generation therefore does not show that all audio modules compiled successfully. The compile warning and the verified decoder-child attention-window installation are separate observations.

ASR review rather than a quality pass. Dependent job 190032 transcribed 36 WAVs with Whisper-small: both emitted and both received streams from all nine rooms. Reading these outputs supports connected hobby, Dungeons & Dragons, coffee, movie, and weekend-planning exchanges in the Moshi/PersonaPlex combinations. The Raon/PersonaPlex combinations contain recognizable topics but confused details and repeated goodbyes. Raon/Raon shows loose work, family, and ethics discussion with weak continuity. These observations are not human listening judgments, ground-truth transcripts, or calibrated intelligibility measurements.

Both Moshi–Raon orderings contain substantial phonetic-looking or nonsensical ASR text in the Raon stream, in clean emitted audio as well as the peer’s received audio. Neither is accepted as a speech-quality success despite zero sustained underruns. Recognizable text also does not guarantee factual correctness: one coffee exchange describes cold brew as caffeine-free. The nine-room result

is consequently an operational collection success with unresolved content failures, not a universal dialogue-quality result.

Transport inspection found the expected Raon payload: kind byte 0x01, float32 little-endian mono PCM at 24 kHz, consumed in 1,920-sample blocks. It did not identify a format mismatch, low input level, or clipping as the cause. A discriminating next test would retain native text, acoustic codes, and PCM for fixed input, then decode the *same codes* through offline and streaming Mimi paths. This separates acoustic-generation defects from codec/runtime discrepancies. Independent code-temperature/top- k settings are also filtered by an upstream call signature, leaving the first code under a joint sampler; this is an untested potential factor, not a diagnosis. Codec-rate/representation trade-offs merit controlled investigation [Jung et al., 2026b], without importing conclusions from other encoder settings into this failure.

6 Acoustic trajectories and future policy learning

6.1 What can be collected today

The current evaluator produces a physical interaction trajectory

$$\mathcal{T}_{\text{ac}} = \{o_t, x_t, y_t, r_t, \text{geometry}_t, \text{counters}_t, \text{wall time}_t\}_{t=0}^{H-1}. \quad (5)$$

This supports routing verification, receiver-level analysis, visualization, and the design of reward signals. It is not a token-policy rollout. Decoding is many-to-one with respect to possible internal token histories, and re-transcribing PCM cannot recover the sampled actions or their probabilities. Likewise, native model text may disagree with realized speech and cannot replace the acoustic stream in evaluating a listening partner.

The included stochastic gate demonstrates a complete small on-policy loop, with synthetic carriers as emissions, per-agent behavior probabilities, frozen rollout parameters, and a subsequent REINFORCE update. Its actor uses microphone-derived features and time features. This establishes the mechanics of simultaneous environment actions and versioned toy transitions, not a scalable SLM trainer. The acoustic reward is deliberately transparent but insufficient for semantic credit: rewarding audibility and penalizing overlap could discourage appropriate backchannels or favor unhelpful silence.

6.2 Token-level contracts and policy synchronization

For an SLM policy, a future event record must expose the sampled action z_k^i , its valid-action/event mask, policy version v_i , and behavior probability $\log \pi_{v_i}^i(z_k^i | h_k^i)$ under the actual temperature, truncation, and sampling rules. Text and acoustic events need distinct namespaces where their distributions differ. An explicit mapping from event k to emitted PCM interval and receiver interval is required to align delayed rewards. Recurrent reset boundaries and fixed-horizon truncations must be recorded alongside natural task terminals. An actor-critic method also needs aligned value predictions or a well-defined reconstruction procedure.

For example, a PPO-style ratio [Schulman et al., 2017]

$$\rho_k^i(\theta) = \exp[\log \pi_{\theta}^i(z_k^i | h_k^i) - \log \pi_{v_i}^i(z_k^i | h_k^i)] \quad (6)$$

requires the actual behavior action and distribution, not an ASR-derived surrogate. Summing text and audio-code log probabilities without respecting their sampling factorization can produce an invalid ratio. A successful implementation should first recompute frozen-policy probabilities on saved events and test timing/reset consistency before attempting optimization.

Independent workers introduce version synchronization requirements. A room must declare whether it uses a frozen partner, simultaneous learning, or a versioned opponent pool. Publishing new weights requires a barrier that finishes or discards old-policy episodes and explicitly rebuilds state; pausing a WebSocket does not reset a model or certify on-policy data. Privileged geometry and clean-source attribution may support a centralized critic, but must remain absent from actor observations unless intentionally part of the task. Full-weight training further changes memory and compute requirements substantially; the inference placement is not a training allocation estimate. A bounded adapter/LoRA update is a more inspectable first implementation milestone.

6.3 HiPLEX and multi-agent credit assignment

HiPLEX’s hierarchical factorization motivates separating its pad/epad/con control decisions from conditional content generation [Jung et al., 2026c]. This is distinct from MuDuSi’s existing Bernoulli tone gate, which does not implement that factorization. A prospective integration could log each control event separately from content events, preserving the hierarchy’s masks and conditional probabilities. Timing rewards could then be attached to control decisions, while task or content rewards target the appropriate generated events. This is an application design, not an implemented HiPLEX adapter or a reported training result. Causal alignment would require token-event, emitted-sample, and receiver-sample timestamps spanning codec buffering and propagation delay; that token-to-PCM mapping is not currently available.

Even with those interfaces, attribution is nontrivial. Agent i can change when agent j receives useful information, and a reward arriving after an interruption can reflect both speakers, transport delays, and acoustic masking. The known source paths help establish who could be heard, not who caused a semantic success. Candidate evaluations should compare frozen-partner and joint-learning settings, retain event-aligned reward components, and report speaking duration, task outcome, and interruption response jointly. Counterfactual replay with fixed emissions can isolate acoustic effects; once policies react and change emissions, a new closed-loop trial is needed. The 3D replay is useful for auditing such failures, but cannot replace the policy statistics needed to train or evaluate them.

7 Applications and scope of future extensions

Controlled conversational evaluation. A virtual scene makes position, divider loss, and noise repeatable without repeatedly arranging physical microphones and speakers. Initial applications include comparing a model pair across acoustic conditions, eliciting overlapping speech with a controlled source, and retaining all microphone views for review. Full-Duplex-Bench-style event labels would add distinctions among backchannels, interruptions, pauses, and sustained collisions [Lin et al., 2025a,b]. Those labels and semantic task scorers are not presently implemented; current activity overlap is only an energy statistic.

Mixed-speech processing and enhancement. Source-isolated emissions together with receiver mixtures could support evaluation of Sommelier-like preprocessing pipelines [Jung et al., 2026a]. Diarization requires predicted timestamped identities, a reference policy for overlapping speakers, and identity-aware metrics. Known simulator source IDs are not inferred speaker labels. A receiver-side enhancement adapter, potentially informed by SLICE [Moon et al., 2026], would need explicit sample-rate, latency, gain, and state-reset contracts. Enhancement could improve recognition while delaying a response or erasing backchannels; those trade-offs should be evaluated in closed loop. No such pipeline comparison is claimed here.

Acoustic style and tool-using agents. Voice and role conditioning enable tests of whether an agent preserves a requested speaking style as its microphone conditions change. OmniACBench motivates separating acoustic realization from a correct textual answer [Kim et al., 2026]; MuDuSi currently supplies audio and geometry, not a validated image-conditioned model-observation stream. Tool-using cascaded agents could also be added with timestamped partial-ASR, tool-request, cancellation, and result events. Speculative execution [Jung et al., 2026d] makes these boundaries especially important: apparent promptness may hide wrong or cancelled work. The current native audio/text adapter does not log such tool events.

Richer acoustics and scale. Pyroomacoustics and SoundSpaces 2.0 offer relevant approaches to richer acoustic rendering [Scheibler et al., 2017, Chen et al., 2022]. Replacing the mixer would require preserving causal streaming state, calibrated delay conventions, and the source/receiver/input artifact contract. The current per-room engine still has three slots and quadratic source–receiver paths. Nine concurrent rooms and recorded multi-scene visualization demonstrate independent-session scaling, not arbitrary participants per room or vectorized differentiable acoustics.

8 Reproducibility, limitations, and release

The measured four-GPU configuration is `configs/matrix.four-gpu.json`; the matrix runner, verifier, and replay workflow are documented in `docs/MATRIX.md`. Job 190030 records clean revision 29bfd751d253, while job 189994 records 2022c03a0ffc with an uncommitted working tree. Later matrix commits must not be described as the exact clean baseline source. Earlier paired/showcase experiments have their own historical provenance in the B200 verification and showcase documents. The separate six-GPU $0.5\times$ configuration was not measured by either completed matrix run; a pending submission cancelled before execution is not an inference failure.

A reproducible collection must retain the model/checkpoint and runtime revisions, native conditioning, scene configuration, seed, placement, input pacing, initialization boundary, recording intervals, and counter snapshots. Distinct native environments are required for the two `moshi` packages. Blackwell-compatible Torch/CUDA wheels and a compilation-capable worker image address exercised runtime requirements; a working inference path does not certify all packages in a reused environment. Replaying saved PCM can reproduce the recorded acoustic exchange without loading weights, while regenerating a closed-loop episode can differ because of sampling and asynchronous scheduling.

The evidence is bounded: a small number of stochastic episodes, one principal matrix collection at reduced pace, and no repeated-seed confidence intervals, human intelligibility labels, or controlled counterfactual interruption trials. The acoustic model omits directional devices, calibrated SPL, diffuse reverberation, HRTFs, and own-loudspeaker feedback. Physical hardware validation remains necessary for deployment. Acoustic noise is implemented, but noise-conditioned task success and scene understanding are not validated benchmarks. Favorable selected clips must remain distinguishable from full trajectories and failed cases.

The project packages generic configurations, tests, report sources, and curated media. The report and selected demonstration media are publicly accessible; access to the source repository is currently restricted. Weights, credentials, private settings, and large raw recordings are excluded; generated-audio provenance and caption sources should remain labeled. Reproduction depends on access to the underlying models under their own terms. A self-published technical report and a citation file do not imply peer review or conference acceptance. This report uses the NeurIPS 2026 preprint format; the MuDuSi Project is the report publisher, and the authors' affiliation is KAIST AI.

9 Conclusion

MuDuSi supplies a shared virtual acoustic loop for independent full-duplex speech sessions, with explicit timing and aligned recordings. Real-time paired experiments verify emission-to-peer routing; a four-GPU, eighteen-worker collection extends this to nine concurrently recorded rooms at quarter-speed input pacing. The larger experiment has healthy sustained buffers but retains substantial speech-quality failures, demonstrating why operational and semantic evidence must be assessed separately. Its acoustic trajectories and replay tools support controlled model evaluation today. Hierarchical policy experiments, semantic credit assignment, and full SLM reinforcement learning require additional event, probability, and synchronization interfaces before they can be claimed as implemented capabilities.

A Recorded demonstration views

These captures show actual recorded experiments, not synthetic dialogue mockups. Speech bubbles replay offline ASR of emitted audio; the displayed clock is simulation time. The images illustrate observability rather than establish conversation quality. Full videos are available on the project page.

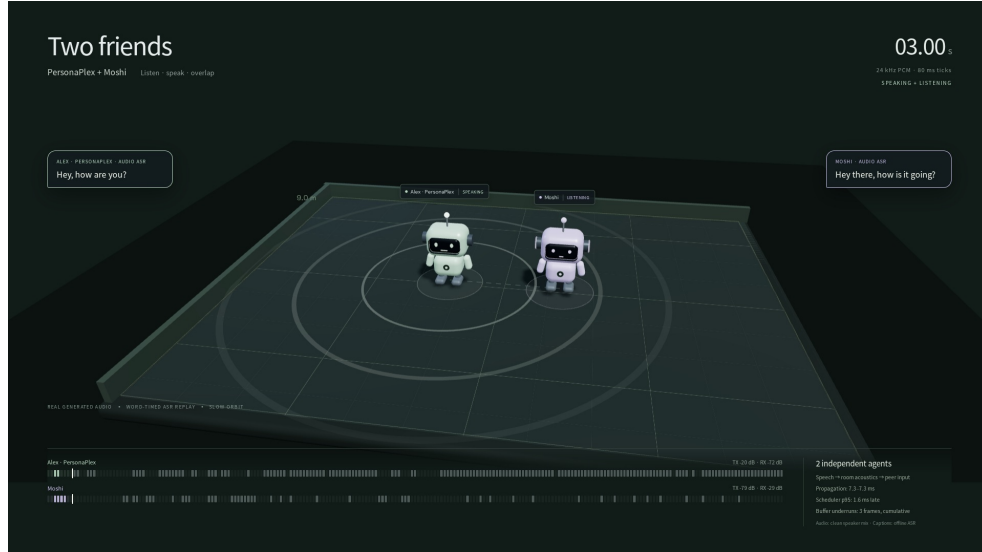


Figure 2: **Two friends.** PersonaPlex and Moshi in the recorded two-agent scene at 3.00 s. The view combines spatial positions, recognized speech, per-agent activity, and receiver levels.



Figure 3: **Clerk and customer.** Two independently stateful PersonaPlex instances with role and voice conditioning, at 16.42 s. The voices and utterances are model-generated; role prompts do not prescribe the transcript.



Figure 4: **Three-way conversation.** Two PersonaPlex instances and one Moshi instance share a single acoustic scene, at 4.96 s. Each hears its own mixture of the other sources. Activity indicators are not inferred diarization labels.

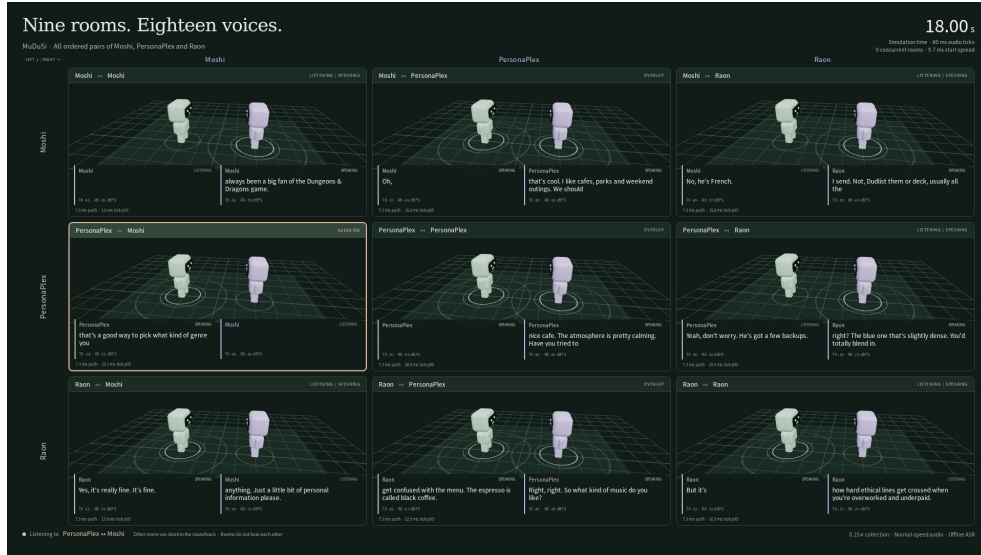


Figure 5: **Nine independent rooms.** All ordered pairs of Moshi, PersonaPlex, and Raon, at 18.00 s. Eighteen workers share four GPUs while each pair remains acoustically isolated. Highlighting selects playback audio only. The 0.25× collection label distinguishes paced collection from normal-speed replay; unresolved Raon speech-quality issues remain visible in ASR.

References

- Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. SoundSpaces: Audio-visual navigation in 3D environments. In *European Conference on Computer Vision*, 2020. URL <https://arxiv.org/abs/1912.11474>.
- Changan Chen, Carl Schissler, Sanchit Garg, Philip Kobernik, Alexander Clegg, Paul Calamia, Dhruv Batra, Philip W. Robinson, and Kristen Grauman. SoundSpaces 2.0: A simulation platform for visual-acoustic learning. *arXiv preprint arXiv:2206.08312*, 2022. URL <https://arxiv.org/abs/2206.08312>.
- Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: A speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024. URL <https://arxiv.org/abs/2410.00037>.
- Kyudan Jung, Jihwan Kim, Soyeon Kim, Jeonghoon Kim, Jaegul Choo, and Cheonbok Park. Sommelier: Scalable open multi-turn audio pre-processing for full-duplex speech language models. *arXiv preprint arXiv:2603.25750*, 2026a. URL <https://arxiv.org/abs/2603.25750>.
- Kyudan Jung, Sehyun Lee, Song-ha Jo, Jaegul Choo, and Sanghyuk Choi. Joint analysis of latent dimensionality and frame rate in continuous audio encoders. *arXiv preprint arXiv:2609.29780*, 2026b. URL <https://arxiv.org/abs/2609.29780>.
- Kyudan Jung, Hyunsin Park, Yoonhyung Lee, Jinhwan Park, Jinhyeok Yang, KiHyun Nam, Jaegul Choo, and Jinkyu Lee. HiPLEX: Hierarchical policy factorization for full duplex speech language models. *arXiv preprint arXiv:2610.07727*, 2026c. URL <https://arxiv.org/abs/2610.07727>.
- Kyudan Jung, Hyunsin Park, Yoonhyung Lee, Jinhwan Park, Jinhyeok Yang, KiHyun Nam, Jaegul Choo, and Jinkyu Lee. Hiding tool latency in on-device cascaded voice agent through speculative execution. *arXiv preprint arXiv:2610.07641*, 2026d. URL <https://arxiv.org/abs/2610.07641>.
- Seunghee Kim, Bumkyu Park, Kyudan Jung, Joosung Lee, Soyeon Kim, Jeonghoon Kim, Taeuk Kim, and Hwiyeol Jo. OmniACBench: A benchmark for evaluating context-grounded acoustic control in omni-modal models. *arXiv preprint arXiv:2603.23938*, 2026. URL <https://arxiv.org/abs/2603.23938>.
- Guan-Ting Lin, Jiachen Lian, Tingle Li, Qirui Wang, Gopala Anumanchipalli, Alexander H. Liu, and Hung-yi Lee. Full-Duplex-Bench: A benchmark to evaluate full-duplex spoken dialogue models on turn-taking capabilities. *arXiv preprint arXiv:2503.04721*, 2025a. URL <https://arxiv.org/abs/2503.04721>.
- Guan-Ting Lin, Shih-Yun Shan Kuan, Qirui Wang, Jiachen Lian, Tingle Li, Shinji Watanabe, and Hung-yi Lee. Full-Duplex-Bench v1.5: Evaluating overlap handling for full-duplex speech models. *arXiv preprint arXiv:2507.23159*, 2025b. URL <https://arxiv.org/abs/2507.23159>.
- Seokhoon Moon, Kyudan Jung, and Jaegul Choo. SLICE: Speech enhancement via layer-wise injection of conditioning embeddings. *arXiv preprint arXiv:2603.05302*, 2026. URL <https://arxiv.org/abs/2603.05302>.
- Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518, 2023. URL <https://proceedings.mlr.press/v202/radford23a.html>.
- Rajarshi Roy, Jonathan Raiman, Sang-gil Lee, Teodor-Dumitru Ene, Robert Kirby, Sungwon Kim, Jaehyeon Kim, and Bryan Catanzaro. PersonaPlex: Voice and role control for full duplex conversational speech models. *Technical report, NVIDIA*, 2026. URL https://research.nvidia.com/labs/adlr/files/personaplex/personaplex_preprint.pdf.

- Robin Scheibler, Eric Bezzam, and Ivan Dokmanić. Pyroomacoustics: A python package for audio room simulations and array processing algorithms. *arXiv preprint arXiv:1710.04196*, 2017. URL <https://arxiv.org/abs/1710.04196>.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL <https://arxiv.org/abs/1707.06347>.
- J. K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis Santos, Rodrigo Perez, Caroline Horsch, Clemens Dieffendahl, Niall L. Williams, Yashas Lokesh, and Praveen Ravi. PettingZoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020. URL <https://arxiv.org/abs/2009.14471>.
- Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. URL <https://roboti.us/lab/papers/TodorovIROS12.pdf>.
- Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL <https://doi.org/10.1007/BF00992696>.