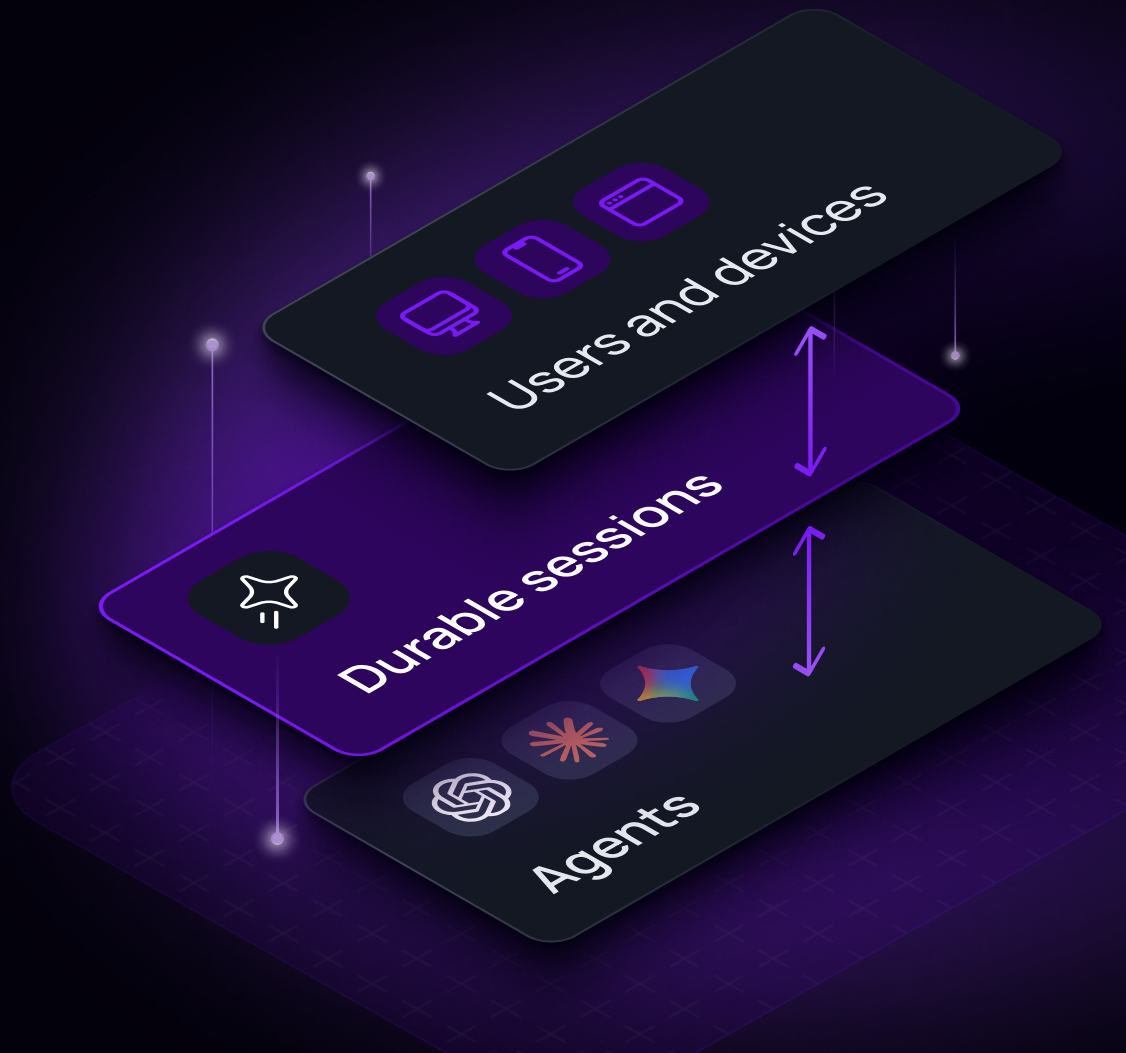
ably

The missing layer in the agentic AI stack:

Why AI applications need durable sessions



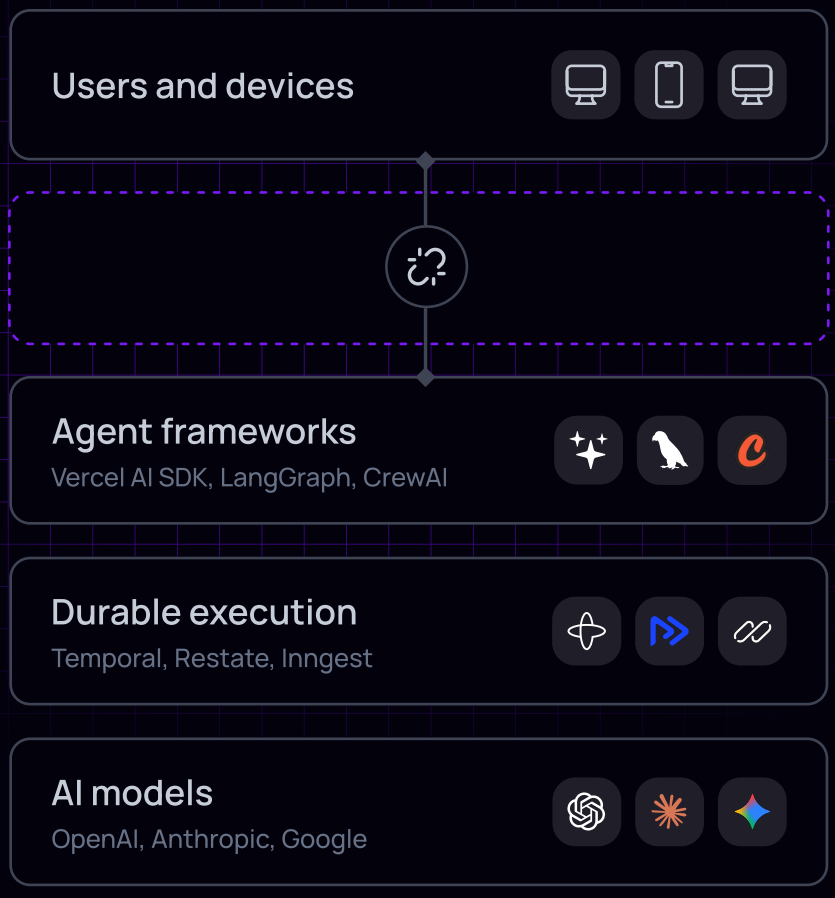
The stack has a hole in it.

Model providers handle inference. Agent frameworks handle orchestration. Durable execution handles backend reliability. But the session between the agent and the user - the live connection through which work actually happens - has no dedicated infrastructure.

Teams building production agentic AI applications are engineering their own session layer. The same infrastructure problem, rebuilt from scratch, by every team.

This is the gap.

FIG 01



The ecosystem has already decided.

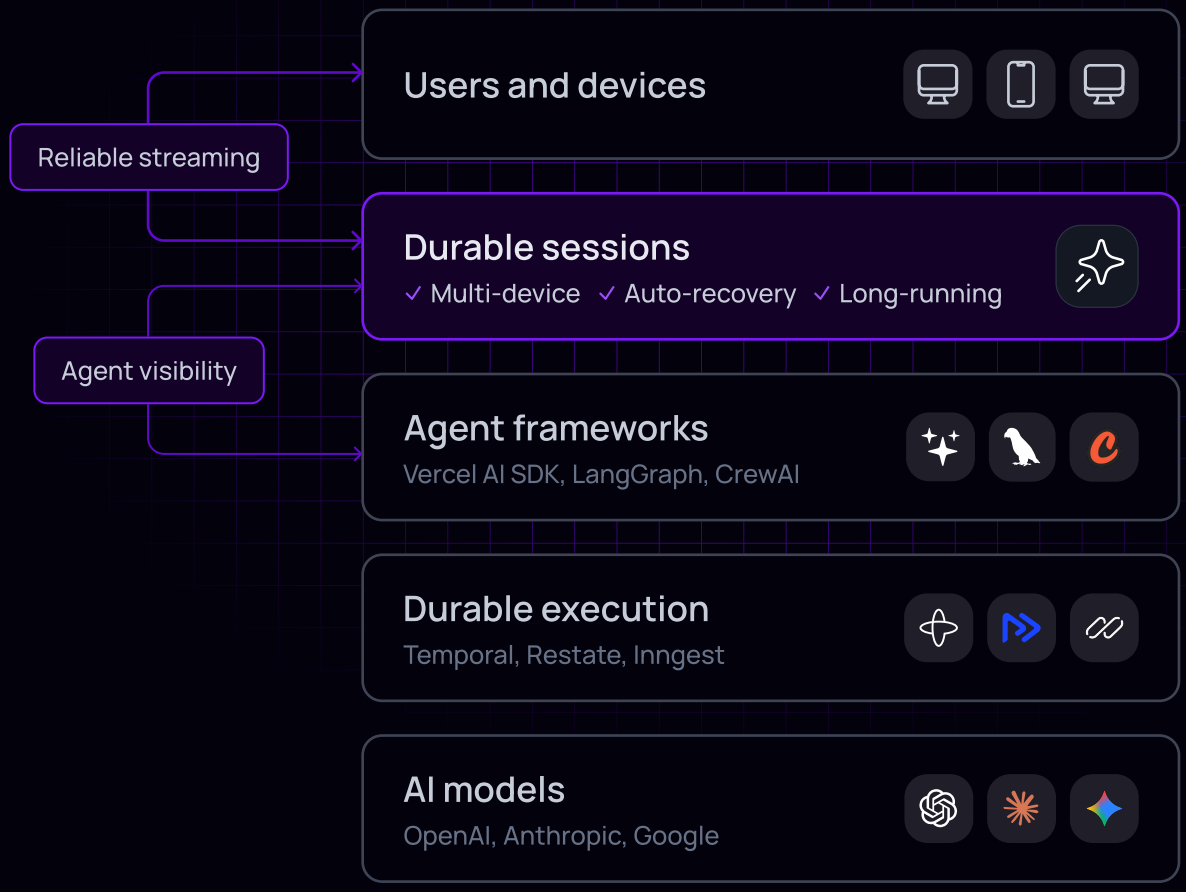
None of the major vendors have claimed this layer as their problem to solve - and they've said so explicitly.

Vercel told developers to bring their own transport. CrewAI closed a WebSocket feature request as "not planned." Temporal has no built-in way to notify frontends of state changes.

These aren't gaps waiting to be filled. They're deliberate boundaries.

What sits between agents and users needs its own layer. It has a name now: **durable sessions.**

FIG 02



The agentic session problem.

Request-response AI apps had simple transport needs. Agentic AI apps have complex durable session problems.

- Prompt-response AI: user sends prompt → model returns response in 2–3 seconds. HTTP streaming handled this adequately.
- Agentic AI: sessions lasting minutes or hours; agents making sequential tool calls; multiple agents coordinating; humans stepping into live sessions. **The connection model fundamentally changed**.

	PROMPT-RESPONSE	AGENTIC AI
Interaction model	Single prompt → answer	Continuous conversation with streaming & steerability
Continuity	Tab-scoped session	Resumable across devices/ sessions
Process visibility	Limited / none	Live tokens, steps, ETA, thinking communicated
Control	Restart	Barge-in, redirect, pause/resume
Background work	Not supported	Runs after you leave (decoupled from app/browser session), notifies on completion
Collaboration	Limited advisor	Agent assistant and multi-user
Notifications	Inline with request	Push updates (in-app, mobile, live activity panel)

Five failure modes

Reconnect wipes the session

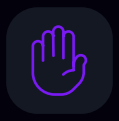
User refreshes or switches networks; streaming breaks. User assumes the agent session is dead.

Device switch = start over

Laptop to phone means re-prompting from scratch.

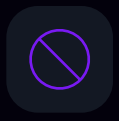
No signal the agent is alive

A 45-second task looks identical to a crashed process.



Handoff to human breaks context

The session resets; user repeats themselves. Trust evaporates.



Agent ignores the user

No steering, no control; user is trapped watching the wrong answer unfold.

These failures compound in production.

In a demo, a dropped session is a retry. In production, it's a lost customer, an unattributable failure, an audit request you can't answer, and a cost you didn't see coming.

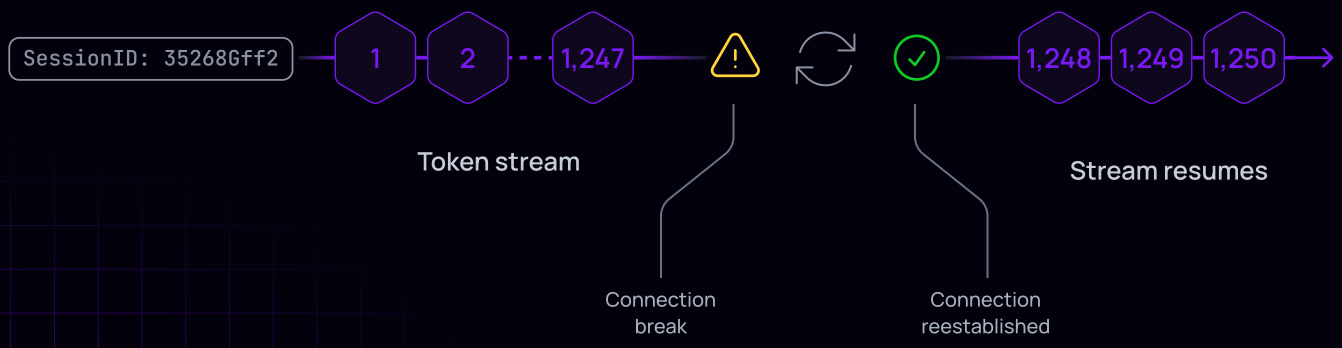
What a durable session actually is.

A durable session is a persistent, addressable session between agents and users that outlives any single connection, device, or participant.

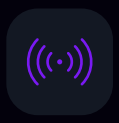
Three properties make a session "durable":

- **Named:** has a persistent ID. Any client - on any device - can rejoin it.
- **Persistent:** state survives disconnection. The session doesn't reset when a WebSocket closes.
- **Resumable:** a reconnecting client receives current state, not a blank slate or a full replay from the beginning.

FIG 03



What durable sessions fix.



Reliable streaming

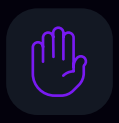
Fault-tolerant WebSocket delivery with exactly-once semantics — no duplicate tokens on reconnect, no silent failures behind corporate proxies.

Session continuity

One session ID, any device. Streaming when connected, catch-up on reconnect, push notification when offline — from the same channel.

Agent visibility

Agent presence as a first-class primitive. Crash detection is a presence event — not an inference from a dead stream.



Human-AI coordination

Bidirectional channels make cancel, redirect, and steer explicit signals. Human agents join the session, get full history, take over — user sees no break.

Integration & ops

Session diagnostics via channel history. Presence-aware cost controls. SOC 2 and HIPAA — shipped, not roadmapped.

The cost of building it yourself.

The most common response when teams hit these problems is to build. Decouple generation from delivery, handle reconnection logic, persist session state, surface agent status. It's all solvable, but it costs months of senior engineering time, and once shipped, it becomes permanent maintenance overhead.

The deeper issue: this is undifferentiated infrastructure. Every team that reaches this point builds roughly the same thing. And every team that builds it eventually hits the same ceiling - usually in production, usually at the worst time.

You no longer have to build this.

When these session infrastructure patterns first emerged, building was the only option. That's changed. Purpose-built infrastructure now exists for exactly this layer, without the months of engineering cost or the ongoing maintenance burden.

ABLY
AI Transport

The session layer, ready to drop in.

Ably AI Transport is the durable session layer for teams building agentic AI products at scale. Reliable streaming, device continuity, and seamless human-AI handover - out of the box, LLM and framework-agnostic.

Explore now:
ably.com/ai-transport

6.5ms message delivery latency	30B+ connections opened monthly
7+ years of 100% uptime	SOCII / HIPAA compliant