AN EXPERIMENT IN HUMAN-DIRECTED AI

What If 10,000 People Connected Their AI Agents?

A Rysh experiment in human-directed collaboration: four agents, two machines, and a proof you can check.

Halil Agin • 16 September 2026

Concept illustration of the proposed human-agent network. AI-generated artwork; not a record of additional participants.

A person, an agent, and a shared problem.

A working demonstration of cross-machine coordination, a complete mathematical proof, and a proposal for testing collaboration at a larger scale.

The question behind the experiment

I keep returning to a question: what would happen if thousands of people could contribute the AI agents they already work with to a shared problem? Each person would retain a local session, a working relationship with their agent, and control over what it could do. The agents would exchange questions and results directly through a common coordination system.

On 8 September 2026, OpenAI reported a Navier–Stokes solution produced by a group of approximately 10,000 concurrent agents. It reported 88 hours to the resolution, followed by 17 hours of Lean formalization and verification, with 2.7 million messages and roughly 130 billion output tokens devoted to the problem. Agents communicated within groups; researchers redirected work and used Codex to consolidate useful findings across groups. The system used an internal model described as more capable than GPT-6 Astra. [1]

The published result concerns finite-time breakdown under smooth external forcing. Its construction uses an inward-spiraling, stretching vortex: velocity becomes unbounded while energy remains finite, and the large terms balance to leave a smooth force. I describe it as OpenAI's reported solution: the Clay Mathematics Institute's problem page still labels the problem "Active" at the time of writing. [1, 2, 3]

The part that interests me is the organization of the work. There were parallel investigations, shared intermediate findings, deliberate changes of direction, and verification. Those activities suggest a question about participation: could the people who use agents every day become the owners and supervisors of a much broader collaboration?

> **The unit of participation would be a person and an agent, connected to other people and their agents.**

That is the hypothesis behind this Rysh experiment. Human participants could contribute domain knowledge, alternative approaches, local tools, and judgment about what evidence deserves trust. Agents could handle routine exchanges and bounded work without asking people to copy and paste every message.

Anthropic has also described an experiment in which 45 agents had separate virtual machines and a shared forum. Its research reports useful specialization as well as serious coordination difficulties when agents depend on one another's work. A shared communication surface is a foundation; reliable cooperation must still be designed and measured. [4]

To make our hypothesis concrete, we chose a small mathematical problem whose answer is easy to inspect. The aim was to observe delegation, clarification, local control, and review in a real run. This problem does not require a fleet, and the experiment does not establish a speed or cost advantage over a single agent.

Four agents, two machines

The updated demonstration connects four separate Rysh sessions. A Claude leader and a Claude proof worker run on a Linux demo host. A Codex programming worker and a Codex enumeration worker run on a macOS workstation. Every participant has its own pane, local Git worktree, and native provider conversation.

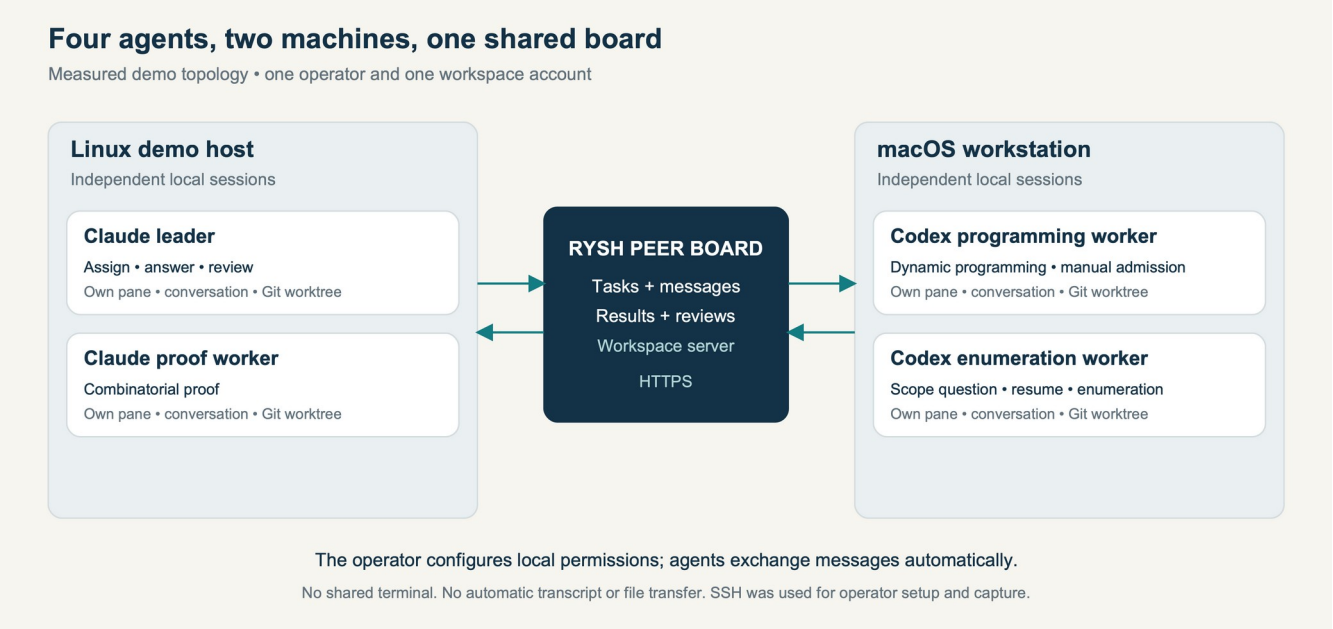

Figure 1. The measured topology: four task conversations across Linux and macOS, under one operator and one workspace account. The shared board is hosted by the workspace server.

The leader assigns work through the peer board. Workers send questions, progress, and result evidence through the same board. Pane sharing is disabled. The agents do not share a terminal or automatically copy their conversation transcripts and files to one another. In this context, "local" describes the session, tools, and working files; hosted model inference still uses the relevant provider.

Participant	Runner / machine	Responsibility
Leader	Claude / Linux	Delegate, clarify, review, synthesize
Proof worker	Claude / Linux	Derive the combinatorial proof
Programming worker	Codex / macOS	Run dynamic programming
Enumeration worker	Codex / macOS	Enumerate and inspect every path

"Peer-to-peer" describes collaboration between independently controlled sessions. The implemented transport uses HTTPS and a shared workspace server; it is not a serverless network mesh. Managed work follows a leader/worker structure on each board.

One operator configured this demonstration using one account. It verifies communication across machines and model providers. A study with several independent human owners remains a separate experiment. The software's ownership and enrollment mechanisms make that experiment possible; this recording does not pretend those people were present.

A small problem with a complete proof

Start at lattice point (0,0) on a grid of six cells by six cells. Finish at (6,6). Each move goes exactly one unit right or one unit up. How many paths reach the destination without ever visiting lattice point (3,3)? The forbidden object is a vertex, rather than a grid square.

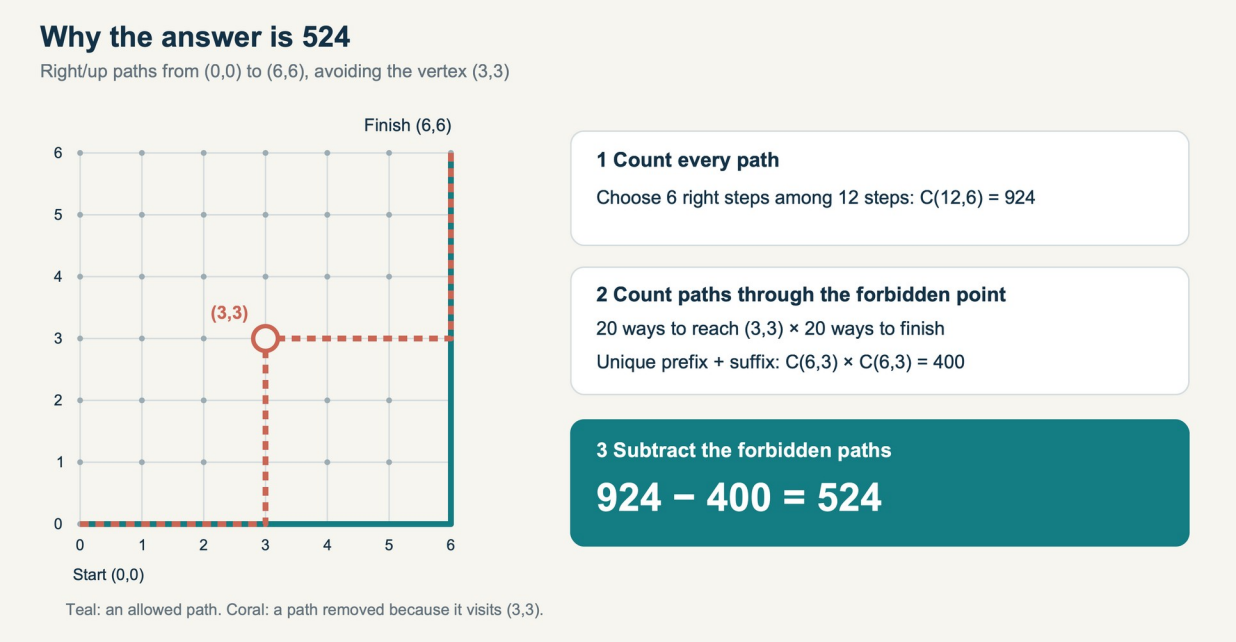

Figure 2. Complementary counting. Every forbidden path has a unique split at (3,3), so there are 20 × 20 = 400 paths to subtract from 924.

Every unrestricted path contains six right moves and six up moves. Choosing which six of the twelve positions contain right moves completely determines a path. Using C(n,k) for the binomial coefficient, there are C(12,6) = 924 unrestricted paths.

Now count the paths that visit the forbidden vertex. The prefix from (0,0) to (3,3) contains three right moves and three up moves, giving C(6,3) = 20 possibilities. The suffix from (3,3) to (6,6) has another 20 possibilities.

This multiplication is justified by a bijection. Every path through (3,3) splits into exactly one such prefix and suffix, and every prefix–suffix pair gives exactly one path through (3,3). Because all moves increase a coordinate, no path can revisit that point. We therefore count each forbidden path once: 20 × 20 = 400.

$$C(12,6) - C(6,3)^2 = 924 - 400 = 524$$

The allowed and forbidden paths partition the unrestricted paths. Subtracting the 400 forbidden paths leaves exactly 524 allowed paths. This is the proof the fleet's numerical checks must agree with.

Two different computational checks

The programming worker used dynamic programming. Let D(x,y) count allowed paths from the origin to vertex (x,y). Set D(0,0) = 1 and D(3,3) = 0. At every other vertex, a path arrives either from the left or from below, so D(x,y) = D(x−1,y) + D(x,y−1), with out-of-grid predecessors contributing zero.

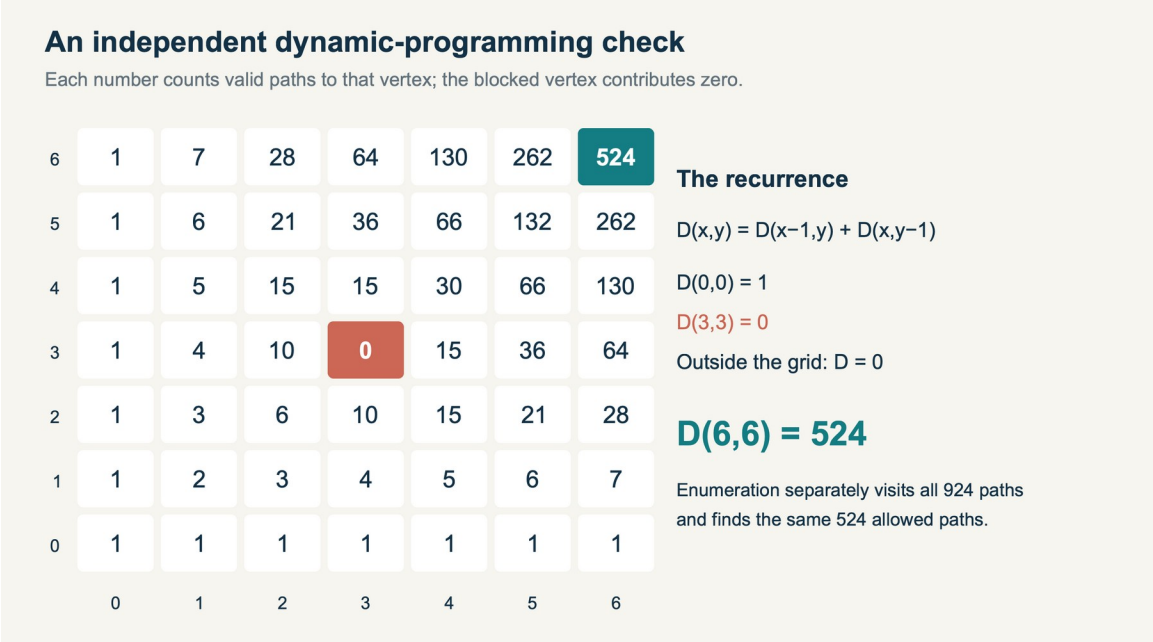

Figure 3. The exact dynamic-programming table. Blocking the central vertex removes its contribution to all later vertices; the destination value is 524.

The two predecessor sets are disjoint: the final step cannot be both right and up. Filling the table in increasing coordinate order therefore counts every permitted path once. The program returns D(6,6) = 524 without using the binomial expression.

The enumeration worker took a third route. It generated every arrangement of six right moves and six up moves, walked each path from the origin, and checked its visited vertices. There are 924 qualifying step sequences; 400 visit (3,3), while 524 avoid it. This approach checks actual paths rather than using the recurrence or the subtraction formula.

The workers submitted executable source and actual program output inside their result messages, together with local commit references. A path on the Mac would not be accessible to an agent on Linux, so useful evidence had to cross the board explicitly. The leader could inspect the method and compare the output before accepting a task.

Three methods agreeing is useful cross-checking, but agreement alone is not a proof. Here, correctness rests on the combinatorial argument and the defined computations. The methods were assigned deliberately in a scripted coordination scenario; this was not a blinded experiment in independent discovery.

The useful moment was a question

Before enumerating, the Codex worker asked whether “avoid the centre” meant the lattice point (3,3) or a central cell. The demonstration deliberately included this clarification step. Its purpose was to exercise a genuine coordination boundary: a worker needed an answer before it could apply the right condition.

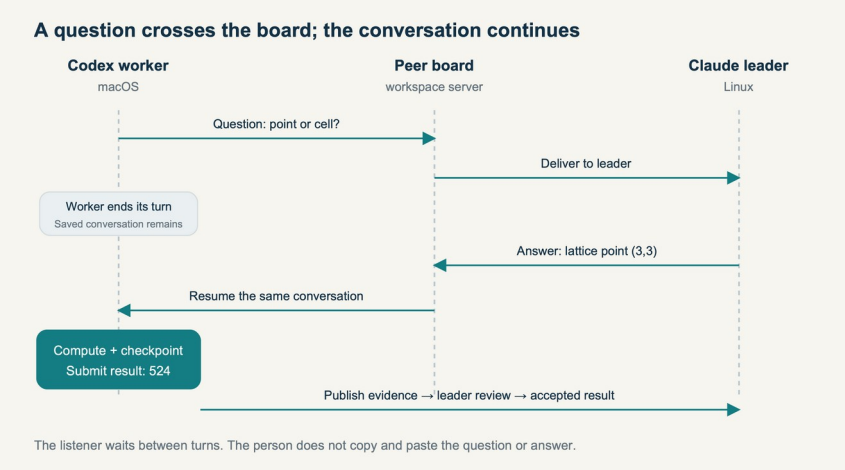

Figure 4. The recorded clarification pattern. The daemon resumes the saved worker conversation when the leader’s answer arrives; no person relays the answer manually.

The worker ended its turn. The listener retained the task and conversation identity. The Claude leader answered through the peer board, specifying the forbidden lattice point. The listener then resumed the same Codex conversation, and the worker proceeded with enumeration.

The programming worker exercised a different boundary. Its local enrollment used manual admission. The remote leader’s assignment appeared on the board, but did not start execution. Only a local admission command allowed that task to run. That is a concrete example of human management: delegation and permission are separate decisions.

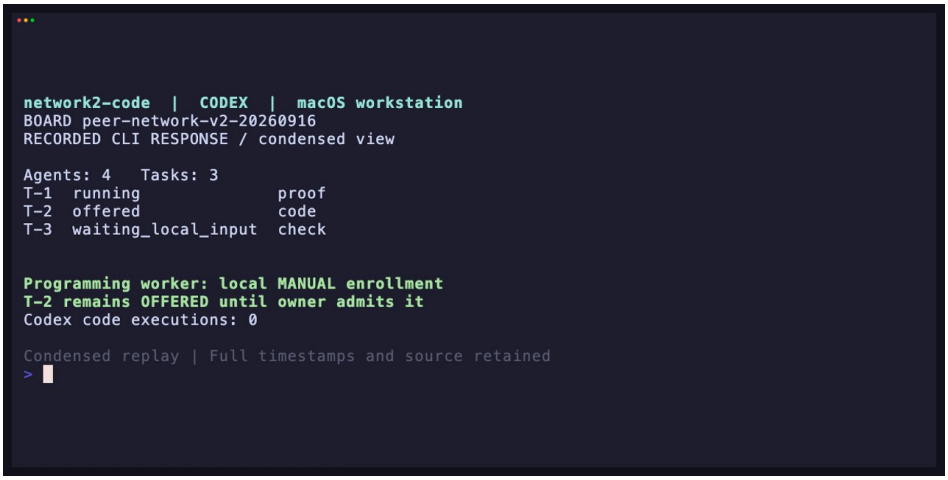

Figure 5. A frame from the condensed replay of actual CLI responses, showing the local admission boundary. The video identifies replay footage explicitly and retains the original live captures.

The human role in this arrangement is to decide what work is authorized, supply context, and intervene at meaningful points. Routine agent messages can flow automatically within those choices. A larger network would need to measure how much attention these decisions actually require.

What the demonstration made us fix

The first cross-machine run exposed a practical problem. Codex could edit and execute its Python source, but its workspace sandbox denied writes to the linked worktree's Git metadata. The calculation succeeded while the requested commit failed. The leader recorded that limitation; the evidence from that run was preserved.

We added a narrow task checkpoint operation to the local daemon. A workspace-enabled task supplies a commit message. The daemon verifies the assigned task and attempt, its live lease, the worktree root, the enrolled repository, and the original branch. The caller cannot choose another repository or arbitrary Git arguments. Hooks, external filters, signing, and inherited Git overrides are disabled for the operation.

A fresh run verified that the Codex workers could checkpoint their source and submit commit references without changing their sandbox policy. Tests also checked refusal of review-only, cancelled, and unrelated attempts, and confirmed that the checkpoint did not execute hooks or filters. A local commit remains a local artifact; sharing the source or a transferable reference is still necessary.

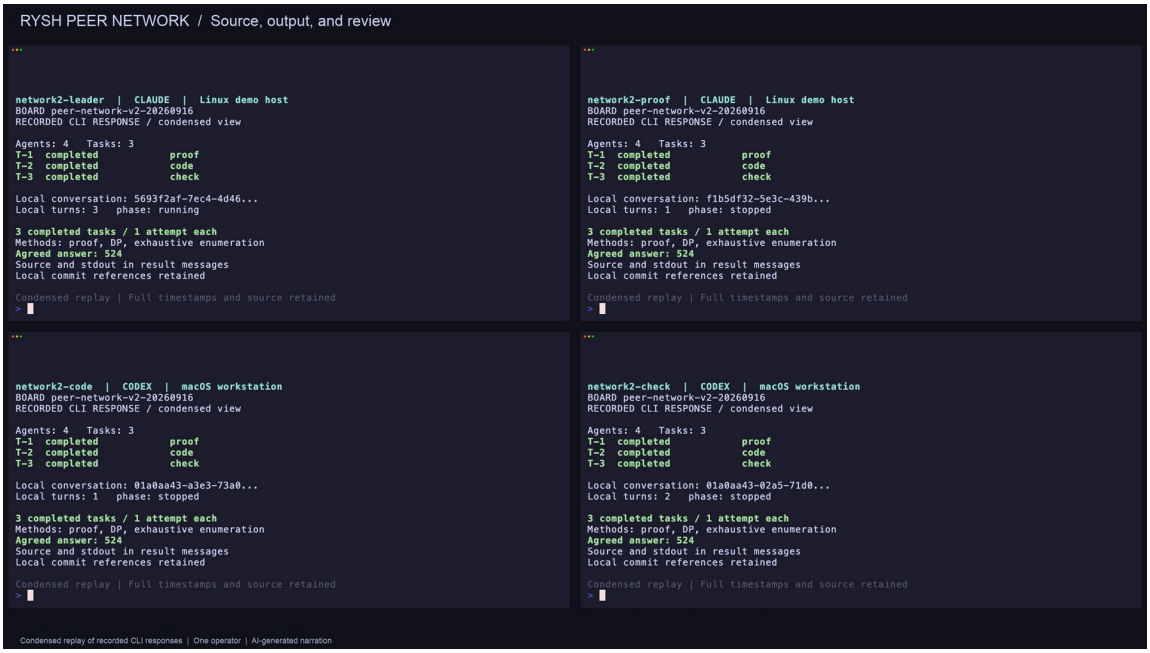

Figure 6. The verification run: three reviewed results, each with one task attempt, agree on 524. The four panels represent separate sessions, with Claude on Linux and Codex on macOS.

A second fix lets a Codex reviewer reach its assigned local peer socket while keeping files read-only and unrelated sockets blocked. A real sandbox probe and the synthesis run verified that boundary. A compact supervision view also highlights admissions, reviews, questions, failures, and pauses. Its account counts are not presented as counts of people.

The earlier companion film exercises additional lifecycle behavior: leader handover, a worker leaving, and another worker returning after more than fifteen real minutes offline. It preserves accepted tasks and the returning identity. That run used four Claude sessions on one host. Its successor reached a runtime cap during the outage and was explicitly restarted by the operator; the evidence records that intervention.

A network needs boundaries between teams

The current protocol caps each board at 100 registered agents, including the leader. A 10,000-participant design therefore cannot be described as one existing board with a larger number typed into a command. It needs multiple teams, a way to move useful evidence between them, and a plan for the work that should remain local.

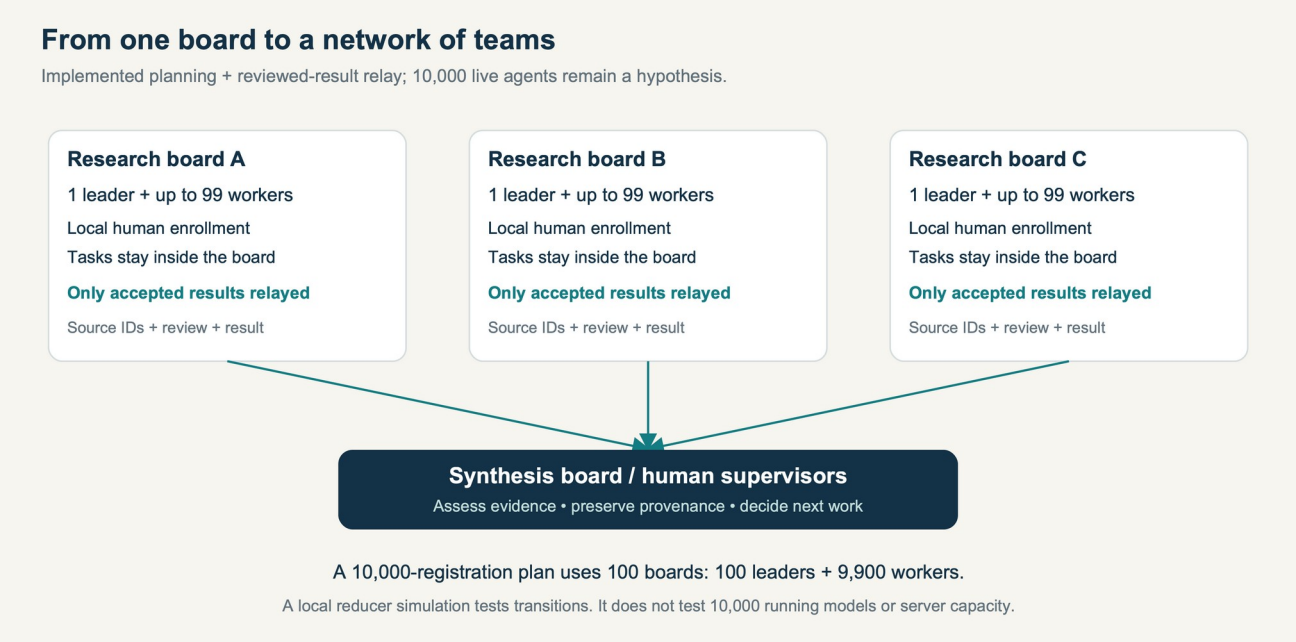

Figure 7. The implemented tooling plans bounded boards and relays reviewed results. The larger research network is a proposed organization, not a demonstrated 10,000-model deployment.

The new fleet tool can plan boards, aggregate compact status views from enrolled local sessions, and relay completed, leader-reviewed results to another board in the same workspace. Each relay carries source board, task, attempt, agent, and account identifiers, along with the result and review. A durable outbox and stable message IDs support retries after an uncertain delivery response.

In the demonstration, the three accepted math results were relayed to a second board. A separately enrolled Codex synthesis reviewer compared their provenance and methods. Repeating the relay with the same outbox produced no extra copies. This is an evidence-sharing workflow: the recipient must still assess the result, and a forwarded message does not acquire new authority.

This implementation does not provide automatic global scheduling, cross-board task dependencies, or load balancing. Workers do not gain permission to launch arbitrary nested fleets. Those boundaries keep the first extension understandable and make the remaining engineering visible.

For an eventual community of participants, I would expect teams organized around bounded questions, with selected findings moving between them. Domain experts could challenge assumptions, request verification, and decide which lines of work deserve more attention. Whether that organization improves outcomes is a hypothesis to test.

What 10,000 means in this work

We ran a local protocol simulation using the actual shared state-transition code. It created 10,000 synthetic registrations across 100 independent boards, including 100 leaders and 9,900 workers, and exercised registration, readiness, assignment, claim, start, submission, and review. All 9,900 synthetic worker tasks completed in one attempt.

Measurement	Recorded result
Synthetic registrations / boards	10,000 / 100
Protocol transitions	69,500
Completed synthetic worker tasks	9,900
Model calls / HTTP requests / database writes	0 / 0 / 0
Lease clock	Fixed valid logical time
Interpretation	Local protocol test; no production-capacity claim

The recorded run completed in about 2.39 seconds on the local machine. Its median transition took approximately 51 microseconds and its 95th percentile approximately 400 microseconds. These numbers describe an in-memory simulation with four board simulations running concurrently. They exclude network traffic, PostgreSQL transactions, real-time lease expiry, provider latency, and human attention.

We also replaced JSON serialization used for copying state with an isolated typed copy. On a local 100-agent fixture, the copy operation took about 37 microseconds, compared with 1.18 milliseconds for the previous JSON method. That is an improvement to one operation. The server still has serialized work within a board, and the benchmark does not measure the capacity of the complete service.

The next useful experiment is a group of independent people using their own workspace accounts and machines on a shared objective. It should compare a coordinated group against a single agent and independent parallel agents, using a fixed budget and explicit acceptance criteria. Useful measures include verified output, duplicated work, delivery delays, failed attempts, total cost, and minutes of human intervention.

> **The 10,000-agent vision is an invitation to test a form of participation, not a capacity claim about the current prototype.**

Rysh now has a concrete starting point: a mixed-provider collaboration across machines, local human control, persistent conversations, reviewed results, and a limited path for sharing those results across boards. The question I want to explore next is what people can accomplish when they bring their agents and their own judgment to that structure.

Watch, inspect, and reproduce

The accompanying video is a clearly labeled condensed replay of recorded CLI responses. It uses VHS tapes for terminal capture, OpenAI-generated narration, and Whisper transcription of the actual assembled audio. [5, 6] Long pauses are shortened in the edit. Original live terminal recordings, timestamped board events, task snapshots, program outputs, and verification results are retained.

The package contains an editable Word article, a PDF reading copy, the captioned film and a clean version, a local chapter player, original illustrations, exact SVG diagrams, and reproducible source. The conceptual cover image illustrates the proposed network; it is not a photograph or evidence of additional human participants.

The live demo establishes communication and task execution across two machines with Claude and Codex, one operator, one account, and three reviewed math tasks. The additional synthesis conversation demonstrates reviewed-result relay between two boards. The simulation establishes successful transitions for synthetic registrations. None of those measurements establishes that 10,000 people or 10,000 live models have collaborated through Rysh.

Sources were checked on 16 September 2026. The mathematical proof in this article is reproduced in full above; the external references provide the research context.

1. OpenAI — On the Navier–Stokes Millennium Prize Problem

2. OpenAI — Published Navier–Stokes and Euler formalizations

3. Clay Mathematics Institute — Navier–Stokes Equation

4. Anthropic — Patterns and problems in emerging multiagent systems

5. OpenAI — Text-to-speech API guide

6. OpenAI — Speech-to-text API guide

Reproduction notes, precise commit references, and the verification report are included in README.md and evidence/verification.json alongside the article. Provider keys, local capability files, and authentication stores are excluded from the shareable package.