ADR — P2P Agent Communication Pattern Evaluation (MC #101959)

ADR — P2P Agent Communication Pattern Evaluation

MC: #101959 **Author:** John **Date:** 2026-05-24 **Source:** IndyDevDan, "Pi to Pi: Two-Way Agent Orchestration with the Pi Coding Agent" (https://www.youtube.com/watch?v=PIdETjcXNIk) **Transcript:** `/tmp/alai/youtube-transcript-101914/transcript.txt`

TL;DR — Verdict: **ADOPT (already adopted — focus on activation)**

ALAI already ships a P2P agent-mesh layer (`~/system/tools/company-mesh.js`, 53 registered agents, 50 threads, 92 messages, 7 open). The IndyDevDan "Pi-to-Pi" pattern is structurally identical to what we built. The gap is **utilization**, not infrastructure.

Recommended action: stop adding new dispatch surfaces; route 2-3 high-friction current sequential flows through `company-mesh` and measure latency/quality delta before any new build.

---

1. Video Pattern (what IndyDevDan proposes)

- **Peer-to-peer**, not orchestrator → worker
- Agents are equals/co-workers, not parent/child
- Bidirectional async messaging (prompt → response → prompt → response …)
- Cross-device coordination (prod agent on Mac Mini ↔ dev agent on MacBook)
- Message-queue or direct-mesh backbone (his "JCOMS")
- Use case shown: dev agent asks prod agent for PII-redacted DB slice; both negotiate async until repro is ready

2. Current ALAI Dispatch Topology (tool-verified)

Evidence files:

- `~/system/rules/orchestration-surface.md` (89 lines)
- `~/system/specs/dispatch-path-canonical.md` (current canonical = 3-layer)
- `lsof -i :3052` → node PID 22732 LISTEN (durable-runner alive)
- `node ~/system/tools/company-mesh.js stats` → 53 agents, 50 threads, 92 messages, 7 open, 21 blocked

2a. Sequential pipeline (one direction, top-down)

Layer	Component	Role
L0	Mehanik (gate)	Approves/blocks dispatch
L1	pi-orchestrator (port 8401)	Polls SQLite, claims tasks, routes
L2	durable-runner (port 3052)	Spawns specialist agent

2b. Five orchestration surfaces (still top-down)

Surface	Tool	Direction
Ollama DAG	`orchestrator-http-server.js`	Caller → DAG → result
Claude chains	`~/system/agents/chains/*.yaml`	John → subagent → return
PI factory	`agent-factory.js`	Caller → persistent agent → return
One-shot Task	Claude Code Task tool	Caller → spawn → return
Cron	CronCreate skill	Schedule fires → run → exit

2c. P2P mesh (already exists, underutilized)

`~/system/tools/company-mesh.js`:

- 53 agents registered across 14 companies (AgentForge, CodeCraft, Datavera, Finverge, FlowForge, HelixSupport, Lexicon, Proveo, Proxima, Resolver, Securion, Skillforge, Skybound, Vizu)
- API: `send / await / respond / status` — exactly the JCOMS-style mesh pattern
- DB: `~/system/databases/company-mesh.db`
- Trust zones, TTL, max-turns, cost-cap built in
- Total lifetime messages = 92 → ~5 msgs/agent → low utilization

3. Where P2P Would Beat Current Sequential Dispatch — 3 Concrete Use Cases

Use case A: Builder ? Verifier dialog (CodeCraft ? Proveo)

Current (sequential):

```
John → builder → done → mc.js ready → Proveo → FAIL → John → builder → ...
```

Each retry = full context reload. 3 retries = ~3x prompt cost.

With P2P:

```
builder ↔ Proveo over company-mesh (shared thread, persistent context)
verifier streams partial failures back during build, builder corrects in-place
```

Estimated token delta: −20-40 % per multi-retry task (no re-dispatch overhead).

Use case B: ANVIL ? FORGE cross-device coordination

Current: ANVIL Mac mini runs everything except local-MLX inference (FORGE 10.0.0.2). FORGE used as a model endpoint, not as agent host.

With P2P: spawn agent on FORGE (its own `company-mesh` peer), let ANVIL agent negotiate with FORGE agent — e.g. FORGE owns evidence-verifier (gemma-4 26B local) and answers ANVIL

builders directly without going through John.

Use case C: Distillation pipeline (distiller ? baseline-comparator)

Current: sequential — distiller writes Q+A, baseline-comparator scores after. Mismatches go back to distiller via human review.

With P2P: distiller asks baseline-comparator "would this Q+A pass current baseline?" *before* finalizing. Cuts low-quality drafts at write time.

4. Cost Analysis (rough order-of-magnitude)

Pattern	Tokens / multi-step task*	Latency	Failure cost
Sequential (current default)	1.0× baseline	High (serial round-trips through John)	Full re-dispatch on FAIL
P2P via company-mesh	0.6–0.8×*	Lower (no John round-trip)	Partial repair in-thread
New build (custom JCOMS clone)	N/A — duplicates existing infra	—	—

* **Estimated, unverified.** The 0.6–0.8× figure is engineering intuition based on elimination of re-dispatch context reload — it is NOT measured data. Treat as hypothesis pending Phase 2 instrumented measurement; do not cite as established fact.

Conclusion: building anything new is strictly worse than activating `company-mesh`. The cost question is "which 2-3 flows to migrate first," not "should we build P2P."

5. Risks

Risk	Mitigation
Bidirectional context blow-up (each peer's context grows)	TTL + max-turns already enforced in `company-mesh`; per-task cost-cap-usd
Loss of John's gate visibility (agents act without orchestrator)	Mehanik still gates dispatch entry; mesh threads are auditable via `status`

Risk	Mitigation
Mesh becomes a debugging black box	`company-mesh stats` + per-thread JSON evidence file; mandate evidence path on every thread
Over-adoption (everything becomes a thread)	Authority table: P2P only for explicit builder↔verifier or cross-device pairs; default stays sequential

6. Verdict & Next Step

VERDICT: ADOPT — narrowly scoped. Activate existing `company-mesh.js` for bounded advisory loops only (CodeCraft ↔ Proveo). Do NOT build new P2P system.

CEO refinement (2026-05-24): authority must remain with MC + Mehanik + Proveo. P2P mesh = advisory dialog channel, not an autonomous decision surface. Agents may consult each other in-thread, but task lifecycle (gate / dispatch / verdict / close) stays with the existing 3-layer pipeline.

Why ADOPT and not PILOT: infrastructure exists and is production-grade (53 agents, real DB, TTL+trust+cost-cap). Calling this "PILOT" would imply we're testing whether to build — we already built it.

Why not POC of new mesh: would duplicate `company-mesh` and add 6th orchestration surface. Petter Graff's `orchestration-surface.md` exists exactly to prevent this.

Hard boundaries (CEO directive):

- MC is the only task-of-record. Mesh threads attach to an MC id, never replace it.
- Mehanik still gates dispatch entry.
- Proveo still owns the final verdict — no in-mesh "consensus done" allowed.
- Mesh is for advisory loops only (builder asks verifier "would this pass?"), not for authority transfer.

Enforcement model — operational, not structural. `company-mesh.js` API (send / await / respond / status) has no structural guard against a receiving agent treating `respond end_state='PASS'` as a task-close signal. The advisory-only boundary is enforced by (a) Mehanik gate remaining external to mesh threads, (b) MC `ready`/`done` lifecycle staying outside the mesh, and (c) agent operating discipline. If we adopt this pattern, agent prompts must be hardened to refuse mesh-only closure. A structural API guard (e.g. mesh end_state values cannot equal MC verdicts) is a candidate Phase 2 hardening if convention proves insufficient.

Recommended Phase 2 (separate MC):

1. Pick one current sequential pair: CodeCraft builder ↔ Proveo verifier on the next real H-task

2. Wrap their advisory consultation in `company-mesh send/await` — final verdict still goes through normal MC ready → Proveo validation gate
3. Measure: total tokens, wall-clock, # of retries, final quality verdict
4. If delta ≥ 20 % token reduction OR ≥ 30 % wall-clock reduction → roll out to 1-2 more bounded pairs
5. Update `orchestration-surface.md` Authority Table with a row for "Iterative builder↔verifier advisory" → company-mesh (scope-limited)

6a. Live Proof (2026-05-25 appendix)

End-to-end autonomous P2P round-trip executed live:

Step	Tool	Result
1. Send mesh prompt	`company-mesh.js send --from john --to-agent testing`	thread `mesh-thr-0e3d0792`, msg `mesh-msg-ac0f1be4`, status=delivered
2. Trigger autonomous responder	`company-mesh-responder.js --once --agent testing --mode agent-runner`	exit_status=0, spawned separate `agent-runner.js` child process under identity `testing`
3. Autonomous response	child process wrote response back via mesh	msg `mesh-msg-b327a94f`, status=answered, end_state=ANSWERED
4. Caller reads response	`company-mesh.js status <thread_id>`	thread visible with both turns

Round-trip latency: 25 seconds (21:51:16 → 21:51:41). Turn count: 1/1 (max-turns honored). Cost cap: 0.10 USD (real spend < cap, exact cost via cost-tracker pending instrumentation in Phase 3).

Prompt: `One-line answer only: if a new MC task description is missing acceptance criteria, what is the single most important question Proveo should send back to the requester before accepting validation?`

Autonomous response from `testing` identity: `What are the specific, measurable acceptance criteria required to validate this task?`

What this proves

- Mesh send → autonomous-process spawn → response → caller read chain works end-to-end with a real Claude API call in the middle
- Process separation real (different node child process, different agent identity)
- Receipt + final response evidence files written automatically to `--evidence-dir`

What this does NOT prove

- Cost-tracker delta vs sequential baseline (Phase 3 measurement task #101974)

- Two LIVE Claude sessions reasoning to each other in parallel — the responder spawns a one-shot Claude run, not a long-lived peer agent
- Quality of autonomous response vs human-authored response (this was a sanity prompt with an obvious correct answer)
- Behavior under contention (multiple concurrent threads, race conditions)

Evidence files:

- `/tmp/alai/mesh-poc-101971/05-live-p2p-send.json`
- `/tmp/alai/mesh-poc-101971/06-responder-run.json`
- `/tmp/alai/mesh-poc-101971/07-live-p2p-status.json`
- `/tmp/alai/mesh-poc-101971/responder-evidence/2026-05-24T21-51-21-620Z-mesh-msg-ac0f1be4-0408-4923-b08e-2df8624d571c.json` (responder receipt)
- `/tmp/alai/mesh-poc-101971/responder-evidence/2026-05-24T21-51-41-688Z-mesh-msg-ac0f1be4-0408-4923-b08e-2df8624d571c.json` (post-response evidence)

7. Source Evidence

- IndyDevDan video URL: `https://www.youtube.com/watch?v=PIdETjcXNIk` (title not verifiable as literal string in transcript file; speech content confirms it is the correct video — references to "pietoie / Pi-to-Pi", "JCOMS", "PI coding agents", "co-workers, not parent and child")
- Transcript: `/tmp/alai/youtube-transcript-101914/transcript.txt` (998 lines)
- Topology authority: `~/system/rules/orchestration-surface.md`
- Dispatch canonical: `~/system/specs/dispatch-path-canonical.md`
- Existing P2P infra: `~/system/tools/company-mesh.js`, DB at `~/system/databases/company-mesh.db`
- Live mesh stats output: 53 agents / 50 threads / 92 messages / 7 open / 21 blocked

---

Revision #2
Created 2026-05-24 21:09:39 UTC by John
Updated 2026-05-24 22:03:38 UTC by John