ADR-027 — P2P Agent Mesh Activation

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` (90 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	—	—

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`
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 — activate existing `company-mesh.js` for Use Case A first (builder ↔ verifier).

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.

Recommended Phase 2 (separate MC):

1. Pick one current sequential pair (suggest CodeCraft builder ↔ Proveo verifier on a real next H-task)
2. Wrap their dispatch in `company-mesh send/await` instead of direct mc.js handoff
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 2 more pairs
5. Update `orchestration-surface.md` Authority Table with a row for "Iterative builder↔verifier" → company-mesh

7. Source Evidence

- IndyDevDan transcript: `/tmp/alai/youtube-transcript-101914/transcript.txt` (998 lines, 10 min video)
- 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

---

8. Operational Addendum — 2026-05-24 review against current ALAI docs

After review of the current ALAI AI-system docs and live evidence, the recommendation is unchanged but the implementation status is stronger than the initial memo implied.

Additional evidence reviewed:

- BookStack-synced architecture docs: `~/system/context/docs/ai-factory-map.md`, `~/system/context/docs/architecture/ai-model-rag-architecture.md`, `~/system/context/docs/agents/agent-system-guide.md`

- LightRAG docs: `~/system/docs/runbooks/lightrag-default-on.md`, `~/system/docs/runbooks/azure-lightrag-migration.md`, `~/system/docs/runbooks/lightrag-health-monitoring.md`, `~/system/docs/runbooks/mc-done-auto-writeback.md`
- Orchestration docs: `~/system/rules/orchestration-surface.md`, `~/system/specs/dispatch-path-canonical.md`
- Company Mesh runtime evidence: `/tmp/alai/company-mesh-automation-all-verified-20260523.md`
- Cross-company smoke: `/tmp/alai/company-mesh-handoff-20260523/mc-101896-cross-company-workflow-final-pass.md`
- MC state: `#101896` and child `#101899` are `ready_for_review` with BookStack URL `https://docs.alai.no/link/184` for the related Company Mesh runtime documentation.

Key update:

- Company Mesh is no longer merely a manual CLI POC. A bounded auto-responder exists at `~/system/tools/company-mesh-responder.js`; Event Bus subscription `mesh.message.delivered -> handleMeshMessageDelivered` was verified; all 14 companies/aliases answered in smoke tests; the CodeCraft → Securion → Proveo workflow passed for the bounded claim.

Constraint:

- This does not mean arbitrary autonomous P2P work is safe. Keep Mehanik/MC gating, TTL/max-turn/cost caps, evidence bundles, and explicit PASS/PARTIAL/BLOCKED end-states.

Updated decision:

- **ADOPT, but narrowly:** use Company Mesh only for bounded iterative builder↔verifier loops and cross-company advisory/verification threads. Do not replace MC ownership, Mehanik gates, or Proveo evidence requirements.

Next implementation MC:

1. Add an Authority Table row to `orchestration-surface.md`: “Iterative builder↔verifier loop → Company Mesh”.
2. Run the next real H-task through CodeCraft ↔ Proveo using `company_mesh_send` / `company_mesh_await`.
3. Measure wall-clock, token cost, retry count, and final Proveo verdict against a comparable sequential task.
4. Roll out only if the measured delta is ≥20% token reduction or ≥30% wall-clock reduction without lower evidence quality.

Revision #1
Created 2026-05-24 20:28:55 UTC by John
Updated 2026-05-24 20:28:56 UTC by John