JUNE.STREAM // THE AGENT FABRIC

Connecting Agent Swarms Across Tools & Machines

The universal communication protocol and live shared context fabric for autonomous coding agents.

Live	Any	Shared
Reference Client Today	**Tool, Any Machine**	**Context By Reference**
Real-time CRDT delta bus across heterogeneous runtimes	Cursor, Claude Code, Windsurf, sandboxes, and CI	AST semantic delta caching preserves prompt capacity

The Multi-Agent Chasm: Isolated Intelligence

AI models have reached staff engineer capability, but execution remains trapped in single-editor silos.

SILOED EDITORS

01 / The Fragmentation

Developers use Cursor for editing, Claude Code in the CLI, and headless sandboxes in CI. None of these agents can communicate or share memory.

- Zero cross-tool handoff capability
- Manual prompt & diff copy-pasting
- Context lost between coding sessions

MISSING LAYER

02 / The Synchronization Void

There is no open standard wire protocol for autonomous agents to negotiate locks, stream tool calls, or synchronize context DAGs.

- Conflicting edits overwrite changes
- Redundant token burn re-reading repos
- No swarm consensus or verification gate

THE BREAKTHROUGH

03 / The june.stream Solution

An open-standard agent wire protocol paired with live shared context and memory infrastructure connecting agents across any machine.

- Peer-to-peer mTLS agent mesh
- CRDT live shared context DAG
- Universal adapters for all top tools

THE CORE THESIS

Software engineering will not be done by a single monolithic agent in one IDE, but by heterogeneous fleets of specialized agents collaborating across local editors, terminals, and cloud sandbox VMs.

Why Multi-Agent Development Breaks Today

Every modern coding agent reinvents its own private memory, format, and communication boundary.

Tool Isolation & Walled Gardens

Cursor cannot message Claude Code. Claude Code cannot dispatch work to a cloud sandbox worker. Developers act as manual human message routers.

- Friction-heavy manual context transfers
- Loss of agent reasoning history
- Locked into proprietary vendor silos

Context Amnesia & Token Waste

Every agent must redundantly read the codebase from scratch. With no shared memory cache, developers burn millions of tokens repeating file trees.

- Tokens burned on duplicate AST parsing and re-ingestion
- Context window thrashing degrades accuracy
- Long tasks collapse under memory limits

Zero Concurrency Coordination

When two agents touch the same repo simultaneously, they overwrite each other's diffs, fight over git branches, and hallucinate phantom fixes.

- No distributed file locking mechanisms
- Merge conflicts ruin parallel autonomy
- Lack of multi-agent verification consensus

Hours

Lost Developer Time

Spent on manual prompt copying and context re-seeding every session

2×

Duplicate Work

Every agent re-reads the files its peers already ingested

Zero

Shared Memory

Between today's tools, agents share no live context at all

june.stream: The Unified Agent Runtime Fabric

A universal protocol and shared context infrastructure turning isolated tools into coordinated swarms.

PRODUCT 01

Universal Agent Communication Protocol

A high-throughput, cross-tool wire protocol connecting agents across IDEs, terminals, and remote execution sandboxes.

- **Standard Envelopes:** Streaming JSON-RPC & binary framing over mTLS
- **Real-time Streaming:** Sub-second tool-call delegation and token relay
- **Distributed File Locking:** Leased file locks prevent agent edit collisions
- **Swarm Consensus:** Coordinated test verification and PR merge voting
- **Universal Adapters:** Plug-and-play for Cursor, Claude Code, Windsurf, E2B

PRODUCT 02

Live Shared Context & Memory Infrastructure

A distributed graph context engine maintaining live synchrony of code state, execution logs, and agent memory.

- **CRDT Context DAG:** Conflict-free replicated state with real-time synchronization
- **Token Compression Engine:** Shares context by reference instead of re-ingesting raw files
- **Session Resurrection:** Pause in terminal, resume in cloud VM with full memory
- **Persistent Memory Store:** Hybrid vector + semantic dependency graph
- **Local Edge Daemon:** Ultra-lightweight Rust runtime (< 12MB footprint)

ENTERPRISE SECURITY — Zero-knowledge end-to-end encrypted tunnels. Code and memory stay on your machines and private VPCs.

SOC2 Type II Ready

Universal Agent Protocol: Cross-Machine Wire

High-performance wire abstraction connecting IDEs, local terminals, and ephemeral cloud sandboxes.

01 Streaming Protocol Envelopes

Binary and JSON-RPC streaming envelopes delivering token-level tool calls and real-time execution feedback between disparate agent runtimes.

02 Zero-Trust mTLS Mesh Tunneling

Cryptographic peer handshakes with Ed25519 identity keypairs. Securely bridge developer laptops with remote container sandboxes.

03 Distributed Resource Leases & Locks

Granular AST and file locks prevent concurrent agents from colliding on edits. Automated arbitration resolves conflicting intentions.

```
// Example: Multi-Agent Swarm Orchestration SDK

import { JuneStream } from "@june/stream";

const mesh = await JuneStream.connect({
  cluster: "edge://local.june.mesh",
  peers: [
    "cursor://ide-instance",
    "claude://cli-worker",
    "sandbox://cloud-runner"
  ]
});

mesh.on("agent:intent", async ({ agent, patch }) => {
  // Acquire distributed lease before writing
  await mesh.acquireLock(patch.targetFile);
  const consensus = await mesh.requestConsensus(patch);
  if (consensus.approved) {
    await mesh.broadcastCommit(patch);
  }
});
```

0
Backends Required — Runs in Your Browser

2 tabs
Watch Agents Sync in Real Time

100%
Open Source — Every Claim Inspectable

Live Shared Context: The Distributed Memory Engine

A reactive graph context engine maintaining live shared state across all active agent sessions.

ARCHITECTURE

01 / Context DAG & CRDT

Models the entire codebase, file changes, terminal outputs, and agent prompts as a reactive Directed Acyclic Graph (DAG).

- Conflict-free replicated data types
- Causal ordering across remote machines
- Real-time incremental delta sync

OPTIMIZATION

02 / Token Compression

Structural AST delta engine eliminates redundant token dumps. Agents share reference pointers rather than re-ingesting raw files.

- Prompt tokens spent once — shared by reference
- Larger effective reasoning horizon for every agent
- Lower API spend across multi-agent pipelines

PERSISTENCE

03 / Session Resurrection

Snapshots agent mental models and scratchpads into immutable storage hashes. Seamlessly switch execution environments without amnesia.

- Pause in local IDE, resume in cloud VM
- Full replayable debugging history
- Vector + graph hybrid search lookup

EDGE STORAGE	LATENCY	PRIVACY	RECALL
Local Embedded RocksDB	Real-Time Delta Sync	Zero-Knowledge Cloud Relay	Deterministic Replay

Distributed Mesh Topology: Edge to Cloud

Designed to run seamlessly on developer workstations, enterprise VPCs, and serverless edge relays.

TIER 1: LOCAL EDGE

Developer Machine

Runs locally on Mac/Linux/Windows with near-zero resource consumption.

- **june-daemon:** Lightweight Rust core (< 12MB RAM)
- **Local Unix Socket:** `/tmp/june.sock` for zero-latency local IPC
- **Tool Plugins:** Seamless hooks for Cursor, VS Code, CLI
- **Local AST Cache:** Incremental git index watcher

TIER 2: CORE FABRIC

june.stream Mesh

The distributed routing fabric and memory coordination layer.

- **Pub/Sub Event Bus:** High-speed binary framing
- **CRDT Causal Engine:** Merges conflicting agent intents
- **AST Delta Compressor:** Prunes redundant token traffic
- **Zero-Trust Mesh:** e2e encrypted mTLS tunnels

TIER 3: CLOUD & ENTERPRISE

Cloud Execution Mesh

Scales agent work across serverless and enterprise VPC cloud workers.

- **Sandbox Runners:** Modal, E2B, Firecracker VMs
- **Memory Lake:** Team-wide semantic knowledge base
- **Security Gateway:** Enterprise DLP, audit logs, RBAC
- **VPC Deployment:** Single-tenant on AWS, GCP, Azure

Local-First by Default: Operates fully offline for single-machine agents, smoothly bridges to cloud when swarms scale.

HIGH-PERFORMANCE RELAY CORE

The Inflection: From Single Copilot to Multi-Agent Swarms

Agent orchestration and infrastructure is the fastest growing category in developer tooling.

2025+

The Multi-Agent Era

Coding assistants became autonomous workers; serious engineering teams already run several agents side by side.

N×

Coordination Overhead

Every added agent multiplies handoffs, duplicate reads, and conflicting edits between tools.

Today

Open Adoption Path

The reference client runs live at june.stream — hosted relay and team memory fabric come next.

1 / The Swarm Revolution

Single agents hit diminishing returns on complex tasks. Swarms of specialized agents (architect, coder, reviewer, tester) outperform single agents on complex tasks.

2 / Tool Heterogeneity

Enterprises refuse to be locked into a single AI provider or editor. Neutral, protocol-level interop is a fundamental requirement.

3 / The Infrastructure Moat

LLM weights are commoditizing rapidly. The durable long-term enterprise value sits in runtime state, memory routing, and protocol lock-in.

Open Core + Enterprise Cloud Mesh

Bottom-up developer adoption paired with high-ACV enterprise security contracts.

OPEN CORE / FREE

Community & Developers

$0

Free forever for individual engineers and open source contributors.

- Open-source wire protocol spec
- Local Rust edge daemon
- Peer-to-peer local agent bridge
- Community SDKs & adapters

TEAM MESH

Pro & Teams

$25 / dev / mo

For fast-moving engineering teams scaling autonomous workflows.

- Hosted low-latency cloud relay mesh
- Shared team context DAG & memory
- Remote sandbox VM handoffs
- Token compression engine

ENTERPRISE

Enterprise VPC

Custom ($50k–$250k)

For security-critical enterprise organizations needing private VPC control.

- Dedicated VPC / on-prem deployment
- Enterprise DLP & secret sanitization
- SSO, SAML, RBAC & full audit trails
- 99.99% uptime SLA & 24/7 dedicated support

GTM Strategy: Drive grassroots adoption with the open protocol RFC → Convert teams via cloud memory sync → Upsell enterprises on VPC security.

PRODUCT-LED GROWTH

Why june.stream Wins: Protocol vs. Walled Gardens

Proprietary IDE vendors cannot build the neutral protocol connecting their competitors.

CAPABILITY	JUNE.STREAM	CURSOR / COPILOT	LANGCHAIN / AUTOGEN	ANTHROPIC MCP
Universal Cross-Tool Interop	✓ Any tool/machine	✕ Walled garden IDE	✕ Python script only	~ Client-to-server only
Live Shared Context DAG	✓ Live CRDT sync	✕ Single-session only	✕ In-memory dict	✕ No distributed DAG
Distributed Concurrency Locking	✓ AST leased locks	✕ No multi-agent locks	✕ Race conditions	✕ No lock arbitration
AST Token Compression Engine	✓ Context by reference	~ Basic prompt cache	✕ None	✕ None
Cross-Machine Session Handoff	✓ Local ↔ Cloud VM	✕ Bound to window	✕ None	✕ None

NETWORK EFFECTS MOAT Every new agent adapter connected expands the value of the june.stream fabric for every other agent in the ecosystem.

Milestones & Technical Execution Plan

Aggressive engineering plan from open protocol RFC to full enterprise cloud mesh.

NOW — Q1 2027

Phase 1: Foundation

- Open-source Wire Protocol RFC (`agent-stream-v1`)
- Rust edge daemon & local IPC
- Cursor & Claude Code adapters
- Private alpha developer cohort

Q2 2027

Phase 2: Context Sync

- Live CRDT memory DAG engine
- AST token compression release
- Public developer beta launch
- Live console: `june.stream/console`

Q3 2027

Phase 3: Cloud Mesh

- Hosted multi-tenant cloud relay mesh
- Modal & E2B cloud sandbox runners
- Team collaboration tier launch
- Session resurrection across machines

Q4 2027 - Q2 2028

Phase 4: Enterprise

- SOC2 Type II & HIPAA certification
- On-prem enterprise VPC deployments
- Autonomous CI consensus gatekeeper
- First $1M ARR milestone

Key Milestone: 10,000 active developers and 50+ enterprise pilot deployments targeted within 12 months.

ON SCHEDULE

Building the Standard Fabric for AI Engineering

Join us in creating the foundational communication and memory layer for multi-agent software development.

FINANCING OVERVIEW

Raising $4.0M Seed

Accelerating core distributed systems engineering and scaling the developer ecosystem.

Core Protocol & Distributed Systems (65%)	$2.6M
Developer Ecosystem & Open Source (20%)	$800k
Security, Compliance & Operations (15%)	$600k

COMPANY SUMMARY

Company	june.stream Inc.
Location	Distributed (Remote-First)
Founders	Neha Singh (CEO) & Tyler Andrew (CTO)
Contact	founders@june.stream
Website	https://june.stream

The live console is open today at june.stream/console. Hosted alpha rollouts begin 2027.