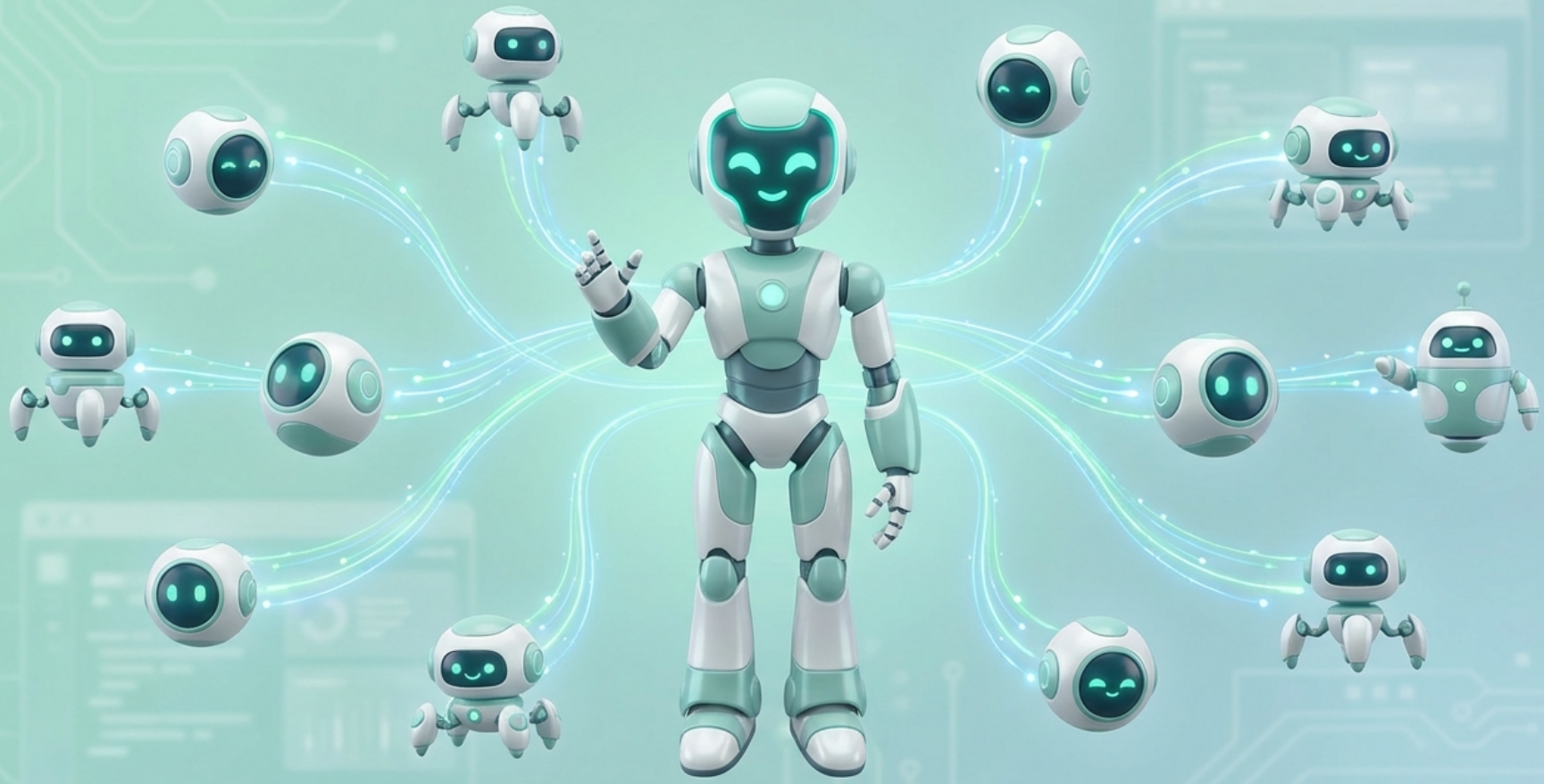
CC: Swarm Mode

How 10 AI Agents Built a macOS App in 13 Minutes

Workshop 5 Supplementary Material



What We'll Cover

- How claudesp orchestrates 10+ parallel agents
- The Teammate() tool and inbox-based communication
- Dependency management with blockedBy
- Parallel execution patterns
- Conflict detection and resolution
- Graceful shutdown protocol
- Real case study: AgentControlTower macOS app

Swarm Architecture

Team Lead + 10 specialized worker agents

What 10 Agents Built in 13 Minutes

22 Swift Files	**6** Build Scripts	**12** Tasks	**10** Agents
13 Minutes	**1** Build ✓	**0** Errors	**∞** Potential

Agent Specialization

Each agent has a focused responsibility

`team-lead`	Orchestration & coordination
`models`	AgentType, LauncherState
`hotkey`	GlobalHotkeyManager
`ui-settings`	Settings, Onboarding views
`updater`	Sparkle UpdateController
`foundation`	Package.swift, project setup
`spawner`	ProcessSpawner, DirectoryDetector
`ui-panel`	QuickCapturePanel UI
`intents`	App Intents for Siri/Spotlight
`scripts`	Build & distribution scripts

The Teammate() Tool

Core coordination primitive

Teammate() Operations

`write`

Send message to agent inbox

```
Teammate(
  target_agent_id: "ui-panel",
  operation: "write",
  value: "Task done!"
)
```

`requestShutdown`

Request graceful termination

```
Teammate(
  target_agent_id: "scripts",
  operation: "requestShutdown",
  reason: "All done"
)
```

`approveShutdown`

Confirm and exit

```
Teammate(
  request_id: "shutdown-123",
  operation: "approveShutdown"
)
```

Message Types in the Swarm

Agent → Agent

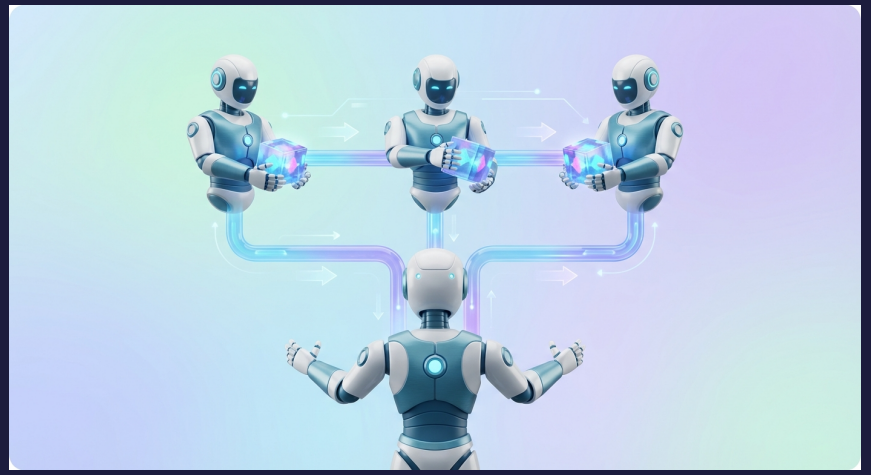
- Task completion notifications
- Status check requests
- Work coordination messages
- Dependency unblock alerts
- Help requests

System Messages

- team_permission_update
- idle_notification
- shutdown_request
- shutdown_approved
- teammate_terminated

Dependency Management

blockedBy enforces execution order

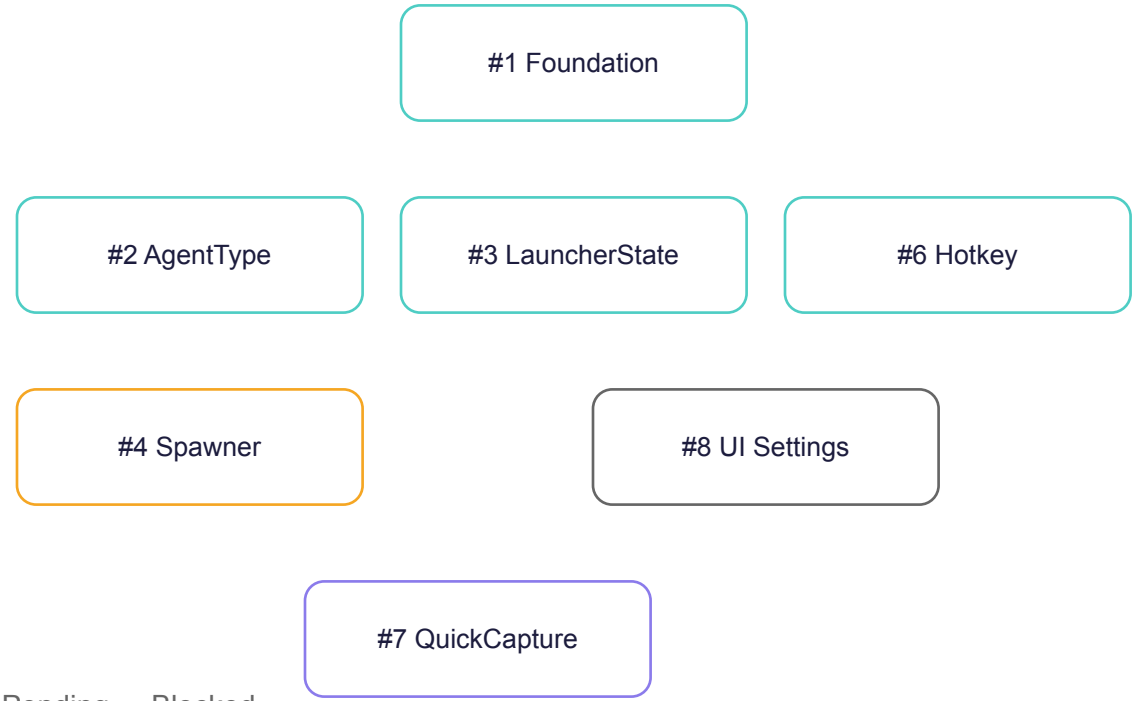


TaskList Output from Swarm

```
#1  [completed]   Create Package.swift and project foundation
#2  [completed]   Implement AgentType model
#3  [completed]   Implement LauncherState model
#4  [in_progress] Implement ProcessSpawner          [blocked by #2]
#5  [pending]     Implement DirectoryDetector       [blocked by #1]
#6  [completed]   Implement GlobalHotkeyManager     [blocked by #1]
#7  [pending]     Implement QuickCapturePanel       [blocked by #2, #3, #4, #5]
#8  [pending]     Implement Settings/Onboarding     [blocked by #2, #3, #6]
#9  [pending]     Main App entry point              [blocked by #7, #8, #11]
#10 [pending]     Implement App Intents             [blocked by #2, #4]
#11 [completed]   Implement UpdateController        [blocked by #1]
#12 [completed]   Create build scripts
```

Tasks form a DAG - blocked tasks literally cannot start until dependencies complete

Dependency Chain Visualization



● Done ● In Progress ● Pending ● Blocked

Parallel Execution

Multiple agents working simultaneously

Pattern: Proactive Parallel Work

> *"I'll proceed with creating my files. The dependencies should match the reference implementation patterns, so I can write my code to use those expected interfaces."*

Agents don't just wait for dependencies - they code against expected interfaces, trusting teammates will follow established patterns. This reduces total execution time significantly.

Conflict Resolution

When two agents create the same file

Conflict Detection & Resolution

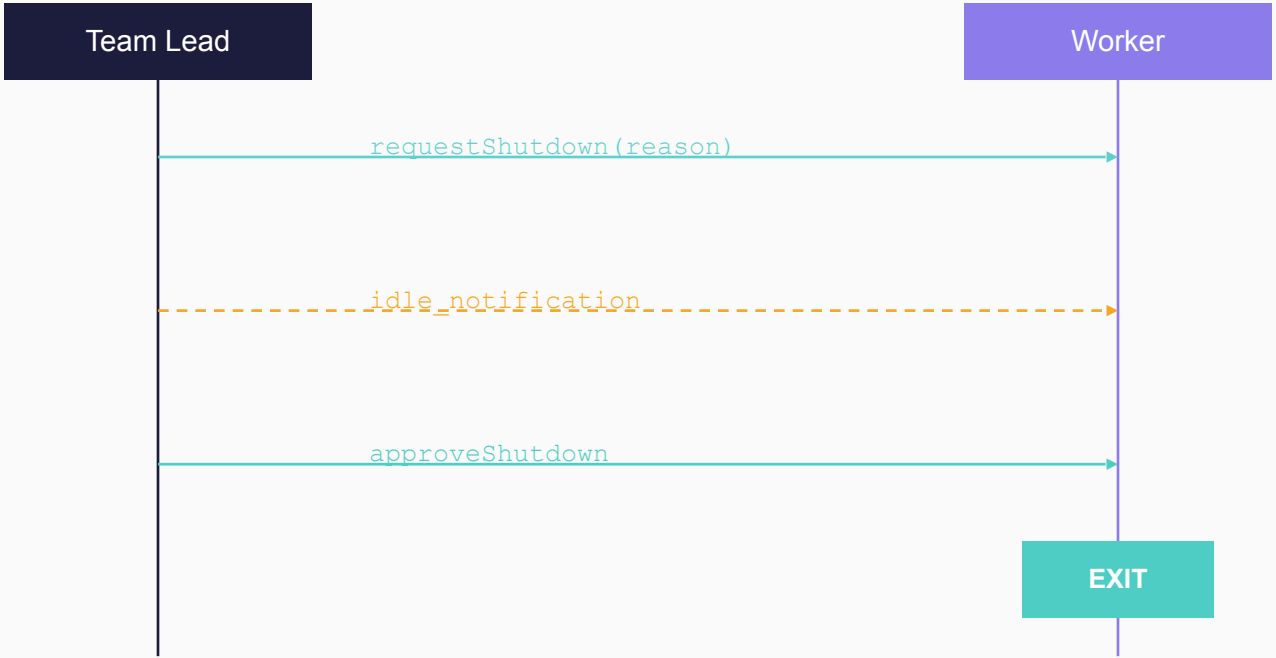
```
[Agent: app-main]
This is a placeholder file that conflicts with my
AgentControlTowerApp.swift which uses the @main attribute.
I need to remove this placeholder since I've already
created the proper entry point.

[Tool]: Bash • command: rm /Users/.../App/main.swift

[Tool]: Teammate • target_agent_id: team-lead
        value: "Resolved conflict: Removed placeholder main.swift
               that was conflicting with @main attribute"

[Tool]: Tool • result: { "success": true, "message": "Message sent" }
```

Agents detect and resolve conflicts autonomously

Graceful Shutdown

Two-phase termination protocol

Shutdown Protocol



Dynamic Permissions

Just-in-time access grants

Permission Grant Message

```
{
  "type": "team_permission_update",
  "permissionUpdate": {
    "type": "addRules",
    "rules": [{
      "toolName": "Edit",
      "ruleContent": "//Users/.../AgentControlTower/Models/**"
    }],
    "behavior": "allow",
    "destination": "session"
  },
  "directoryPath": "/Users/.../Models",
  "toolName": "Edit"
}
```

Permissions granted just-in-time as agents need them - principle of least privilege

Idle Detection & Reallocation

- Agents broadcast idle status when tasks complete
- {"type":"idle_notification", "from":"scripts", "timestamp":"..."}
- Team lead can reallocate idle agents to remaining work
- Prevents wasted compute resources
- Enables dynamic load balancing across the swarm

Reference Implementation

Port from existing codebase

Pattern: Port from Reference

- Agents constantly read from AgentControlTowerLite
- Serves as the source of truth for patterns
- Ensures consistency across all generated files
- Reduces decision-making overhead
- "I'll follow the reference implementation patterns"
- Agents understand how code should look by example

Continuous Build Verification

- Agents run swift build after significant changes
- Catches compilation errors early
- Ensures components integrate correctly
- Build errors guide next steps
- Final result: "Build complete! (0.15s)"
- All 22 files compile together successfully

The Result

AgentControlTower macOS App

Generated Project Structure

```
Sources/AgentControlTower/
├── App/
│   ├── AgentControlTowerApp.swift    # @main, MenuBarExtra
│   └── AppDelegate.swift             # Lifecycle, onboarding
├── Models/
│   ├── AgentType.swift               # Claude, Codex, Gemini
│   └── LauncherState.swift           # @Observable state
├── Services/
│   ├── DirectoryDetector.swift       # Smart directory detection
│   ├── GlobalHotkeyManager.swift     # ^⇧A hotkey
│   ├── ProcessSpawner.swift          # Terminal AppleScript
│   └── UpdateController.swift        # Sparkle auto-updates
├── UI/
│   ├── QuickCapturePanel.swift       # Main prompt UI
│   ├── SettingsView.swift            # Settings tabs
│   └── OnboardingView.swift          # First-launch flow
└── Intents/
    ├── LaunchClaudeIntent.swift      # Spotlight/Siri
    └── AgentShortcuts.swift          # AppShortcutsProvider
```

Key Takeaways

- Inbox-based communication enables loose coupling
- blockedBy enforces correct execution order
- Agents work proactively, not just reactively
- Conflicts are detected and resolved autonomously
- Graceful shutdown ensures clean termination
- Continuous verification catches issues early
- Reference implementations guide consistent patterns

Try It Yourself

```
# Install claude-sneakpeek
npx @realmikekelly/claude-sneakpeek quick --name claudesp

# Start a session
claudesp

# Propose a project
> propose a plan to implement a habit tracker CLI

# Trigger swarm mode
> use swarm mode to implement this

# Watch the magic happen...
# 10+ agents spawn and work in parallel
# Files appear in real-time
# Build verification runs automatically
```

github.com/mikekelly/claude-sneakpeek

Questions?