Agent Lifecycle & Dormancy Management

1. Abstract:

A method for managing the lifecycle of autonomous agents in a decentralized memory-driven system, where agents automatically instantiate, enter dormancy, and reactivate in response to emergent memory-flow and pressure signals, without reliance on external schedulers.

2. Technical Field:

This invention relates to autonomous agent orchestration in distributed ledger environments, and more particularly to theory-level methods for agent instantiation, dormancy management, and reactivation driven by on-chain memory and pressure dynamics.

3. Background:

Traditional decentralized systems rely on external scheduling services or manual triggers to manage agent lifecycles, leading to inefficiencies and centralization risks. There is a need for a self-regulating framework where agents respond solely to the system's intrinsic memory-flow and pressure signals.

4. Summary:

Detecting emergent memory-flow or pressure signals to automatically instantiate agents.

Transitioning agents into a dormant state after predefined inactivity intervals or pressure drop-offs.

Reactivating dormant agents when memory-flow thresholds or new pressure events occur.

Recording lifecycle transitions as one-action-one-mint events for auditability.

Coordinating multiple agents via shared memory-stream signals to optimize resource usage.

5. Detailed Description:

At the conceptual level, each autonomous agent subscribes to one or more memory-stream or pressure-stream feeds. When the aggregated flow signal exceeds an activation threshold, the system automatically instantiates the agent in a VM or protocol module. Following a period of quiescence—measured by sustained flow below a dormancy threshold—the agent records a dormancy event and ceases active processing. Upon detection of new pressure spikes, dormant agents reactivate, ensuring that agent resources align dynamically with the system's informational demands.

6. Method Flow:

Step 1: Activation Monitoring – Continuously monitor memory-flow and pressure metrics across on-chain events.

Step 2: Agent Instantiation – When flow > activation threshold, spin up the agent and mint an activation event.

Step 3: Dormancy Trigger – If flow remains below a dormancy threshold for a grace period, the agent mints a dormancy event and halts.

Step 4: Reactivation Trigger – Upon new flow or pressure spike above threshold, re-instantiate the agent and mint a reactivation event.

Step 5: Lifecycle Auditing – Maintain an immutable log of activation, dormancy, and reactivation events for each agent.

7. Narrative Worked Example:

An analytics agent watches transaction tension streams. When tension rises above threshold at t=100s, the agent activates and logs its activation. After processing until t=200s with tension below dormancy level, the agent logs dormancy and stops. At t=250s, a sudden tension spike reactivates the agent, which logs reactivation and resumes processing.

8. Algorithmic Worked Example:

Pseudocode:

```
flow_signal = subscribeToFlow(stream_id)
for event in flow_signal:
  if not agent.active and event.flow > ACT_THRESH:
    agent = instantiateAgent()
    mint_event('AgentActivated', agent.id, event.flow)
  elif agent.active and event.flow < DORM_THRESH for DORM_PERIOD:
    mint_event('AgentDormant', agent.id, event.flow)
    agent.halt()
  elif not agent.active and event.flow > REACT_THRESH:
    agent = reactivateAgent(agent.id)
    mint_event('AgentReactivated', agent.id, event.flow)
```

9. Potential Embodiments:

Hierarchical agent lifecycles where supervisors spawn sub-agents based on multi-tier pressure signals.

Privacy-preserving activation via ZKPs to attest agent eligibility without revealing sensitive flow data.

Cross-chain agent networks where pressure on one mesh node triggers agents on another.

Adaptive thresholds learned via reinforcement feedback from NGAC reflections.

10. Implementation Notes:

Lifecycle transitions use one-action-one-mint transactions for on-chain traceability. Agent code resides in protocol modules or as VM functions; no external scheduler is required.

11. Claims:

1. A method for autonomous agent lifecycle management in a decentralized memory-driven system, comprising:

a. continuously monitoring, by a processor, memory-flow and pressure signals from on-chain events;

b. instantiating, by the processor, an agent and minting an activation event when signals exceed an activation threshold;

c. detecting, by the processor, that signals remain below a dormancy threshold for a predefined grace period and minting a dormancy event;

d. reactivating, by the processor, the agent and minting a reactivation event when signals again exceed a reactivation threshold;

e. maintaining, by the processor, an immutable log of agent lifecycle events.

2. The method of claim 1, wherein activation and dormancy thresholds are dynamically tuned based on historical flow patterns.

3. The method of claim 1, wherein lifecycle events are recorded as one-action-one-mint transactions in a dedicated module.

4. The method of claim 1, wherein agents are instantiated within a VM context embedded in the core application module.

5. The method of claim 1, further comprising hierarchical spawning of sub-agents based on multi-tier pressure signals.