Title: The MTOR Event Bus: A Unified WebSocket-Driven Intent Framework Using Standardized JSON, N-gram Modeling, and Elastic AI Compute Fabric

Author: Jim Ames, N2NHU Labs

Abstract: This paper introduces the MTOR Event Bus, a stateless, unified AI operating protocol built atop WebSockets and standardized JSON. It provides real-time, decentralized communication for AI modules, driven by N-gram-style intent encapsulation. This infrastructure allows for elastic scale-out via dynamic worker allocation using services like NGROK, and it includes health monitoring, worker blacklisting, and readmittance protocols. The system is a foundational component of the RENT A HAL architecture, enabling modular, intent-driven computation in distributed AI ecosystems.

1. Introduction: The progression toward general-purpose AI requires new communication fabrics — ones that support real-time, intent-oriented messaging, scale elastically, and behave like organic systems. MTOR (Multi-Tronic Operating Realm) solves this with a decentralized event bus that mimics a neural network in behavior but uses modern web-native standards: WebSockets and JSON.

2. Unified Event Bus: The MTOR Event Bus is a WebSocket-based message-passing framework. Each participating AI module (speech, vision, GUI, memory, intent-core) connects to the bus and exchanges JSON N-grams. These packets are atomic thought units — expressing intent, weight, timestamp, and context.

Example MTOR Packet:

```
{
  "event": "vision.update",
  "intent": "continueToFunction",
  "weight": 0.91,
  "origin": "vision",
  "target": "intent-core",
  "data": {
    "detected_object": "fire",
    "confidence": 0.98
  },
  "timestamp": 1691421393
}
```

3. The MTOR N-Gram: Each packet is a self-contained state and action proposition. It mirrors biological signaling: no shared memory, no dependencies, just spikes of meaning. This ensures:

- Stateless modularity
- Real-time processing
- Natural suppression/emphasis based on `weight`

4. Elastic Compute Fabric via NGROK: MTOR enables elastic scaling with NGROK (or similar tunneling/ overlay tools):

- Workers auto-connect to the MTOR bus via secure tunnels
- Bus broadcasts their presence and capabilities
- Tasks are assigned based on declared intent-responsiveness
- Disconnected workers auto-fail and are blacklisted

5. Worker Health Monitoring and Blacklisting:

- Each worker reports heartbeat packets at set intervals
- If health signal is lost beyond a threshold, the worker is blacklisted
- If health signal resumes and passes integrity test (latency, memory use, CPU spike checks), the worker is re-allowed onto the fabric
- This mechanism mimics immune system logic: detect, quarantine, retest, reintegrate

6. System Diagram (conceptual):

- [Intent-Core] <--WS--> [Speech Worker]
- [Intent-Core] <--WS--> [Vision Worker]
- [MTOR Bus] <--WS--> [Elastic Pool: ngrok-tunneled workers]
- [Monitor Daemon] <--WS--> [Health log, Blacklist DB]

7. Use Case: RENT A HAL All HAL modules (text, voice, GUI, IMAGINE, memory) connect to the MTOR bus. HAL maintains local autonomy while transparently scaling out compute-heavy tasks (e.g., vision synthesis or NLU parsing) to elastic workers.

8. Conclusion: The MTOR event bus is a foundation for autonomous AI ecosystems. Its N-gram packet model, WebSocket delivery, and elastic self-healing fabric solve key issues in scaling and autonomy. Paired with the HAL architecture, it provides a path toward robust, ethical, scalable machine intelligence.

Appendix:

- JSON Packet Schema
- Health Check Parameters
- Sample Elastic Worker Lifecycle
- NGROK Integration Instructions

References:

- Ames, J. (2025). MTOR Intent Dynamics
- Ames, J. (2025). The Mathematics of Malignant Intent
- Rent-A-HAL source documentation
- N2NHU Labs internal logs

(End of document)