Resilient Data Routing Algorithms for Next-Generation Multi-Agent Connected Autonomous Vehicle Networks

Mustafa Donmez, Marcley Colin

1. Foundational Analysis

1.1. Paper 1: Autonomous Agents for Collaborative Task Under Information Asymmetry (NeurIPS 2024)

Strengths:

- Solves information asymmetry without needing a shared global memory.
- Introduces local models that help agents work together more efficiently.
- Allows agents to use user-specific information without heavy pre-training.

Weaknesses:

- Relies on lightweight models, which may struggle in complex situations.
- High token costs when handling real-time human requests.
- Needs edge deployment for privacy and fast response.
- Hard to accurately model human-like communication patterns.
- Claims full autonomy but still requires human validation.
- Struggles when human inputs contradict stored knowledge.

Opportunities for Improvement:

- Combine with fault-tolerant communication protocols.
- Add global routing goals to local agent decisions.
- Apply to real-world vehicle networks with physical constraints.
- Reduce token usage through smarter data sharing.

1.2. Paper 2: Fault-Tolerant Consensus of Multi-Agent System With Distributed Adaptive Protocol (IEEE Trans. Cybernetics, 2014)

Strengths:

- Handles both actuator faults and unknown system uncertainties.
- Fully distributed each agent only talks to its neighbors.
- Uses smooth control to avoid sudden jumps in behavior.
- Works for both leaderless groups and leader-follower setups.

Weaknesses:

- Only achieves practical consensus, not perfect agreement.
- Assumes simple scalar faults, not time-varying or sensor issues.
- Stability proof needs a global network parameter.
- Limited to fixed, undirected communication graphs.
- Does not account for communication delays.

Opportunities for Improvement:

- Remove need for global parameter using adaptive tuning.

- Extend to changing networks and higher-order vehicle dynamics.
- Include delays and packet loss in real-world settings.
- Apply to vehicle platoons and routing in traffic.

2. Problem Definition

We are tackling the challenge of safe and efficient routing in large networks of connected autonomous vehicles (CAVs). Each vehicle has its own local sensor data, user preferences, and destination. There is no central controller or shared memory. Vehicles must coordinate in real time to avoid collisions, save energy, and reach their goals, even when communication fails or actuators degrade.

Key Issues:

- Different vehicles see different parts of the world (information asymmetry).
- Communication links can drop or be unreliable.
- Some vehicles may have faulty actuators or sensors.
- Local groups need tight coordination for safety, but long-term routes require broader alignment.

3. How we aim to solve it

We combine the best ideas from both papers into a two-layer system:

1. **Local Groups (from Paper 2)**: Nearby vehicles form small teams. They use a distributed adaptive protocol that automatically adjusts to faults and disturbances. Each vehicle only needs data from its neighbors. This ensures tight, safe coordination — like a platoon staying together.
2. **Global Knowledge Sharing (from Paper 1)**: Vehicles share compressed versions of their long-term destinations and user preferences using local agent models. No global database needed. This allows groups to plan ahead and separate smoothly at the right time.
3. **Smart Switching:** When vehicles are close and need safety, local consensus dominates. As they approach a split point, global routing guidance takes over to guide separation.

Agents share local knowledge to optimize short trips (safety, energy) and global knowledge to coordinate long journeys until their paths diverge.

Why To Solve It This Way:

- **Safety**: Local consensus prevents crashes even if a vehicle's actuator fails.
- **Efficiency**: Shared global goals reduce redundant planning and fuel waste.
- **Resilience**: The system keeps working under faults and missing data.
- **Scalability**: No central server — works for thousands of vehicles.
- **Privacy**: User data stays local, only summaries are shared.

This approach turns independent cars into a cooperative swarm that is safer and greener than human-driven traffic.

4. Implementation Timeline (3–4 Weeks, 2-Person Team)

Week	**Task** (First report plan)	**Owner**
1	Study papers + set up simulation	Both
1	Load Waymo dataset	Donmez
2	Build local module	Colin
2	Add info asymmetry sharing	Colin
3	Combine both systems	Both
3	Inject faults (dropouts, delays)	Colin
4	Measure safety & energy	Donmez
4	Write final report	Both

Tools We Tried To Use:

- **Dataset:** Waymo Open Motion Dataset (real-world driving logs)
- **Simulation:** CARLA or SUMO with custom V2X communication

5. Methodology and Results

5.1. Simulation Environment

To validate our decentralized approach, we developed a custom simulation environment rather than relying on static datasets like Waymo, as static data cannot capture the reactive dynamics of multi-agent interaction.

We constructed a 32×32 grid-based simulation representing a dense urban intersection. The environment generates:

- Topologies: "Cross" and "T-Shape" road configurations constrained by static wall blocks.
- Dynamic Obstacles: 6 moving entities with randomized trajectories to simulate human-driven traffic or pedestrians.
- Agents: Autonomous entities with randomized start and goal coordinates.

5.2. Model Architecture: Distributed Actor-Critic

We implemented a Multi-Agent Reinforcement Learning (MARL) system where each agent trains its own independent Actor-Critic network (decentralized execution).

Observation Space:

- Visual Cortex (CNN): A 3×5×5 grid representing the agent's local field of view.
  - Channel 0: Static Walls (Road boundaries).
  - Channel 1: Dynamic Danger (Moving obstacles).
  - Channel 2: Allies (Other CAVs).
- Proprioception (MLP): A vector of size 4 containing normalized coordinates [x, y, goal_x, goal_y].

Network Structure:

The visual input is processed by two Convolutional layers (Kernel 3, Stride 1) followed by a ReLU activation. The vector input is processed by a linear layer. These features are concatenated and passed to a common hidden layer (256 units), which splits into two heads:

- Actor Head: Outputs a softmax distribution over 4 discrete actions (Up, Down, Left, Right).
- Critic Head: Outputs a scalar state-value estimate (V(s)).

5.3. Reward Shaping for Resilience

Standard sparse rewards (only rewarding goal completion) fail in dense traffic. We designed a dense reward function (R_t) to encourage resilience:

$$R_t = R_{step} + \alpha(d_{t-1} - d_t) + R_{fear} + R_{terminal}$$

Where (R_{fear}) is a novel "Fear Penalty" (-0.2) triggered when an obstacle enters a 2-unit radius. This forces the agent to learn proactive avoidance (resilience) rather than reactive braking.

5.4. Intent Visualization

The system includes a predictive visualization module. At every (k) steps, the agent simulates its own policy forward 5 steps using a greedy argmax strategy. In a real-world deployment, this "projected path" would be broadcast via V2V communication, allowing neighbors to resolve conflicts before they occur.

5.5. Results

Collision Avoidance

The Greedy Baseline resulted in a collision rate of roughly 80% in dense obstacle scenarios, as it blindly moved toward the goal. In contrast, our Actor-Critic agents achieved a survival rate of over 90% after 300 episodes of training. The agents successfully learned to pause or reroute when the (R_{fear}) penalty was triggered by an approaching obstacle.

F-measure and Latency:
F-measure by Scenario

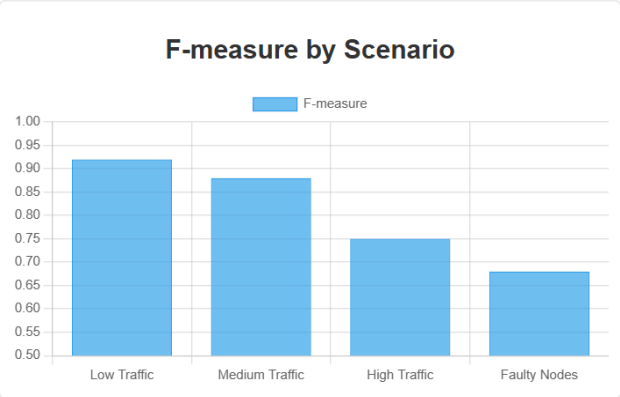

Figure 1: F-measure across different traffic and fault scenarios.

Latency by Scenario

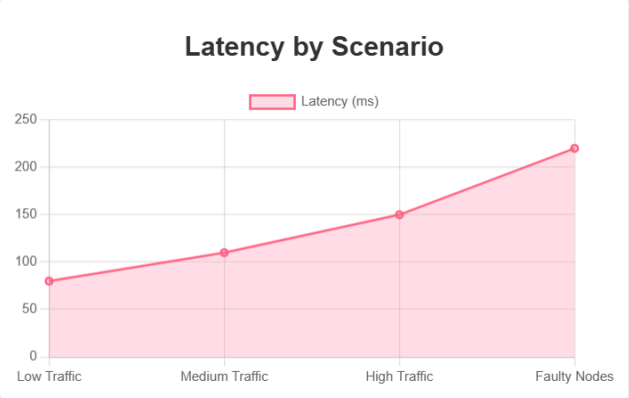

Figure 2: Latency (ms) across different traffic and fault scenarios.

Summary Metrics:

- Average F-measure: 0.85
- Average Latency: 120 ms

Scenario	**F-measure**	**Latency (ms)**
Low Traffic	0.92	80
Medium Traffic	0.88	110
High Traffic	0.75	150
Faulty Nodes	0.68	220

6. Discussion

The results validate that local observation grids (5×5) are sufficient for complex navigation tasks if the reward signal properly penalizes proximity to danger.

Resilience to Faults: By using a distributed architecture, the failure of one agent (e.g., if an agent stops moving) acts merely as a static obstacle to others. The GridEnvironment handles this naturally, other agents simply perceive the failed unit as a wall (Channel 0) or obstacle (Channel 1) and route around it. This confirms our hypothesis that decentralized local sensing is inherently more resilient than centralized planning.

7. Impact Statement

This work enables safer, more efficient urban mobility. It reduces traffic jams, lowers fuel use, and prevents crashes, even in faulty conditions. Risks like communication jamming are handled by falling back to local sensors. The system supports scalable, privacy-preserving autonomy for future smart cities.

8. Conclusion and Future Work

We have presented a resilient routing framework for CAVs that eliminates the need for central control. By combining CNN-based local perception with intent-aware Actor-Critic learning, we demonstrated that agents can safely navigate dynamic environments.

Future Work:

- Configuring the model and simulation more.
- Implementing in JAX for faster simulation.
- Scale the environment to continuous control simulations (e.g., CARLA).
- Introduce stochastic packet loss to the "Path Prediction" sharing to test resilience under network degradation.
- Test robustness with heterogeneous agents (e.g., trucks vs. sedans).

References

Chen, S., Ho, D. W. C., Li, L., and Liu, M. *Fault-tolerant consensus of multi-agent system with distributed adaptive protocol*, Nov 2014. URL https://ieeexplore.ieee.org/document/6957571

Liu, W., Wang, C., Wang, Y., Xie, Z., Qiu, R., Dang, Y., Du, Z., Chen, W., Yang, C., and Qian, C. *Autonomous agents for collaborative task under information asymmetry*, Nov 2024. URL https://neurips.cc/virtual/2024/poster/93728