A Systems Level Measurement Study of Consensus Overhead for Time Sensitive Vehicle Collaboration

By
Hui Yu
29th January, 2026

Department of Computer Science and Engineering
University at Buffalo
State University of New York

Project Supervisor: Dr. Haonan Lu

Abstract

Collaboration among autonomous vehicles and edge devices (V2X/V2V) is becoming increasingly common as vehicles gain stronger onboard sensing and compute. Recent work has explored computer vision and machine learning pipelines for detecting objects and sharing perceptual information across agents. However, many existing approaches assume reliable timestamps and tightly synchronized clocks, which can be difficult to maintain in dynamic, mobile networks.

This study evaluates whether a consensus protocol can provide a practical synchronization backbone for collaborative perception under two common mobility patterns: (1) a minority of vehicles gradually drifting away from the main group, and (2) a majority of vehicles gradually drifting away from the main group. We deploy an EPaxos-based system on CloudLab using 3 independent nodes representing vehicles and inject artificial network delays to emulate increasing separation. We measure latency across three layers: L1 ping latency, L2 baseline RPC latency, and L3 EPaxos commit latency under injected delays. By comparing L3 against L2 under both scenarios, we characterize the overhead introduced by consensus as network conditions degrade.

Overall, our focus is on measuring how consensus latency evolves as network delay increases in two common driving scenarios. These observations can inform future work on synchronization and coordination mechanisms for collaborative perception, particularly those aiming to reduce coordination overhead at the system level.

1 Introduction

1.1 Motivation and Goal

Recent studies increasingly promote collaboration between autonomous vehicles and roadside infrastructure, where multiple agents share perceptual information to improve safety and efficiency. Instead of relying on a single vehicle's limited view, collaborative perception aims to fuse observations from different viewpoints into a more complete understanding of the local scene. However, this collaboration is fundamentally constrained by timing and ordering: messages arrive late, clocks drift, and agents may describe different moments in time. This work focuses on measuring how widely used distributed consensus protocols such as Multi-Paxos[1] and EPaxos[2] behave under mobility-inspired delay patterns, as a first step toward understanding whether systems-level coordination can practically support time-sensitive collaborative perception.

1.2 What is Collaboration

Collaboration refers to multiple agents working together toward a shared goal. The intuition is simple: there is strength in numbers. However, as seen in parallel and multi-threaded systems, collaboration is not free, it introduces coordination and communication costs. If agents cannot communicate efficiently and reliably, collaborating can underperform compared to operating independently.

In the context of collaborative perception for autonomous vehicles, collaboration means sharing sensing information such as camera images or features, and LiDAR point clouds or derived detections. By combining information across agents, the system can construct a more complete view of the local scene, often represented as a Bird's Eye View (BEV) of the vicinity. This shared view helps overcome the limitations of any single vehicle's perspective, especially under occlusion, obstruction, or sensor interference.

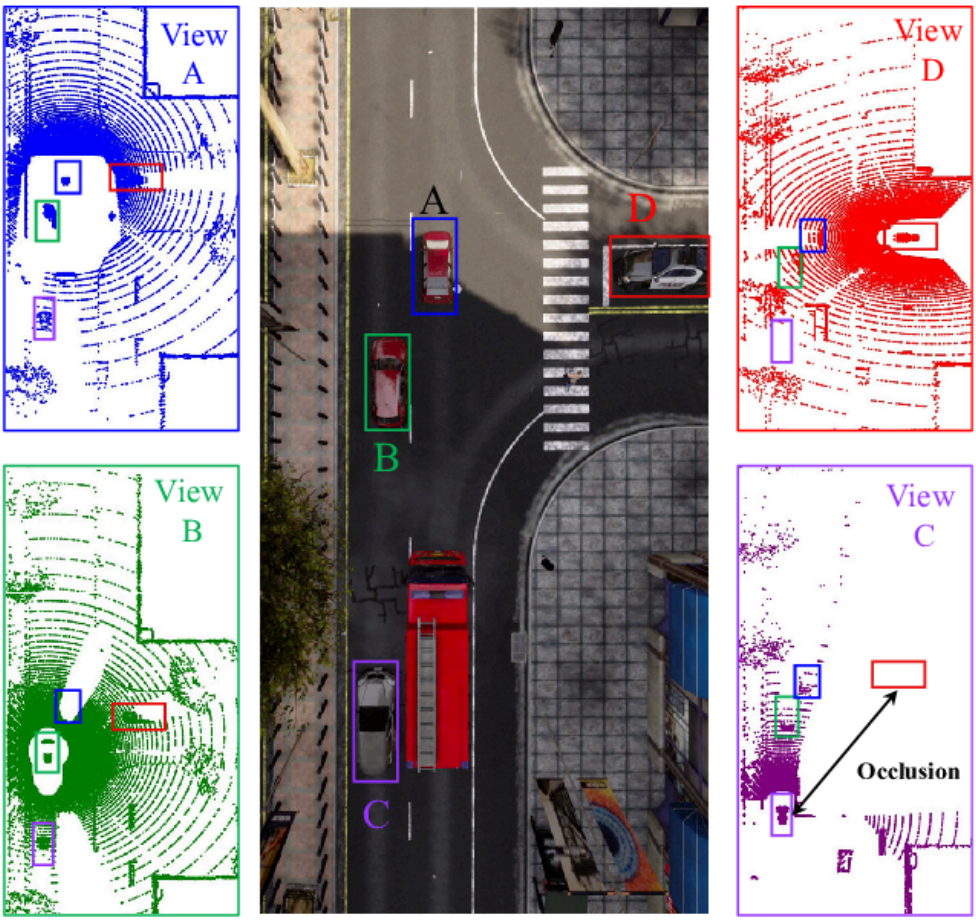

Figure 1: Collaborative perception for autonomous driving at an intersection[3]

Figure 1 illustrates why this matters. The figure shows a BEV style view of an intersection along with what each vehicle can perceive individually. Vehicles A and B have a clear line of sight to the scene and can observe vehicles C and D. In contrast, vehicles C and D each perceive only partial information because a large obstacle (the fire truck) blocks the direct line of sight between them. In this situation, a collision becomes plausible if vehicle D merges into the lane of vehicle C while vehicle C continues along its current trajectory without awareness of D. In a collaborative setting, vehicles A and B can share their observations with both C and D, allowing the two occluded vehicles to become aware of each other and enabling safer decisions. Beyond collision avoidance, collaborative perception can also improve driving quality in common maneuvers such as merging, lane changes, and coordinated passage through intersections.

1.3 Why Timing and Ordering Matter

Achieving the desired collaborative outcome depends not only on what agents share, but also on when information is produced and in what order it is consumed. Returning to the intersection example, vehicle C may receive perceptual updates from both vehicles A and B. Even if those updates are individually correct, they can appear conflicting at C if they correspond to different

moments in time. In real deployments, perfect synchronization across agents is difficult, clocks can drift, wireless links introduce variable transmission latency, packets may be delayed or retransmitted, and interference can cause bursty delivery. As a result, two messages that are generated close together may arrive far apart, and messages generated at different times may arrive nearly simultaneously.

This can lead to incorrect fusion. For instance, a message from vehicle A might arrive at C with stale information, placing vehicle D farther from the intersection and implying low risk. At nearly the same time, vehicle B might deliver a more recent update indicating that vehicle D is already entering the intersection on a collision course. If C fuses these updates without accurately accounting for timing, the resulting shared view can be inconsistent and may misrepresent the current state of the world. In addition to raw timing mismatch, ordering matters: the system must decide which updates should be treated as the latest, and how to integrate observations that arrive out of order.


Figure 2: Image stitching

To illustrate the impact of timing mismatch on perception outputs, we conducted a simple toy example. We captured two images from different time frames in which the toy cars had moved, and then attempted to fuse the images using standard computer vision alignment and blending. Because the two inputs represented different states of the scene, the fused output produced "ghosting" artifacts (two apparent instances of the same car), making the result unreliable for determining the true object location. This example highlights a broader point that collaborative perception is highly sensitive to temporal alignment, and small timing errors can propagate into large fusion errors.

1.4 Related Work

To address imperfect synchronization between agents, recent studies have proposed a range of techniques within the vision and learning pipeline to reduce the impact of timing mismatch[4]. One representative direction is to explicitly realign information across time before fusion. For example, CoBEVFlow[5] models scene motion in BEV space by estimating a BEV flow field and then uses this motion estimate to shift asynchronous features, so that messages from different agents correspond more closely to a shared reference time prior to fusion.

A second direction is to change the communication representation so that fusion becomes less sensitive to misalignment and bandwidth constraints. CPPC[6] proposes communicating sparse, object-centric point clusters rather than dense feature maps, and then aggregating information by matching and merging these clusters. This design reduces the amount of transmitted data and

incorporates mechanisms intended to improve robustness under practical issues such as latency and pose errors.

Together, these works demonstrate that timing mismatch is a well-recognized challenge in collaborative perception. They primarily mitigate the problem through compensation or robustness strategies inside the perception pipeline. At the same time, these approaches generally still rely on timestamps or relative time offsets being sufficiently meaningful to support alignment decisions, such as estimating how far information should be shifted and determining which messages can reasonably be fused.

2 Methodology

2.1 Systems Support

Collaborative perception frameworks often treat timestamps as the glue that make multi-agent fusion possible. In practice, delay, packet drops, and clock drift make that alignment unreliable, which turns fusion into a coordination problem (what to fuse and in what order). This is where distributed systems mechanisms can help, they provide principled ways to reason about asynchrony and ordering across agents, motivating our evaluation of consensus protocols as a potential "support layer" and measuring the cost of doing so.

2.2 Consensus

Consensus can be viewed as a coordination tool that helps a group of distributed agents maintain a consistent view of shared decision despite asynchrony. At a high level, a consensus protocol ensures that participating nodes agree on a single outcome, and the order of the outcome, even when messages are delayed, reordered, or some nodes temporarily fail. This is useful when agents must make compatible decisions based on partial and time sensitive information.

In the context of collaborative perception, consensus does not improve perception accuracy directly, but it can act as a support mechanism for coordination. For example, agreeing on which updates are considered "committed," enforcing a consistent ordering of shared information, stabilizing a shared state that downstream fusion can depend on. The key tradeoff is cost, stronger agreement typically requires additional communication rounds and coordination, which can introduce latency. Because collaborative driving is time sensitive, the central question is whether the coordination benefits of consensus can be achieved at an acceptable latency overhead under mobility inspired network delays.

2.3 Multi-Paxos vs EPaxos (Leader vs Leaderless)

In the context of collaborative perception, we chose Multi-Paxos[5] and EPaxos[6] as the main consensus algorithm for testing as they are widely adopted consensus protocols that present two

contrasting coordination styles, which makes them useful baselines for studying coordination cost in time sensitive collaboration.

Muti-Paxos is a widely adopted leader based protocol and serves as a natural reference point for understanding the cost of enforcing a single agreed order. Its centralized leader structure provides a clear, simple coordination path, but also creates a potential bottleneck and single point of failure under network variability.

EPaxos provides a complementary leaderless design. Instead of routing all coordination through one leader, any agent can accept requests, which better matches collaborative perception where agents exchange updates in a peer-to-peer style. In vehicle collaboration, maintaining a single permanent coordinator is often unrealistic because connectivity and relative positions change over time, and no single agent is consistently best placed to coordinate. Including EPaxos alongside Multi-Paxos therefore gives a clean comparison between leader based and peer style coordination.

3 Experimental Evaluation

3.1 Experiment Setup

To simulate real world scenarios of vehicle collaboration, we used CloudLab[7] from University at Utah as our testbed. By reserving physical machines to conduct our testing, we were able to create a controlled environment that closely resembles real world conditions. Specifically, we reserved three independent nodes, and treated each node as a standalone “vehicle” participating in collaborative perception. All three nodes were placed on the same experimental network so they could communicate directly, allowing us to study coordination behavior under identical hardware conditions while controlling only the network delay between nodes.

To achieve this, we inject artificial network delays into selected communication links between nodes. The intuition is that as vehicles separate physically, wireless links typically experience higher latency and increased variability. For example, vehicles on the road rarely maintain the same speed. A slower vehicle may fall farther and farther behind the group, and over time its communication with the group becomes increasingly delayed before it effectively leaves the collaboration range. By injecting delays, it provides a controlled way to reproduce that effect while keeping the rest of the system unchanged.

We evaluate two primary scenarios:

1. In Minority drifts away, only one of the three nodes becomes increasingly delayed relative to the other two, representing a single vehicle slowly separating from the main group.

2. In Majority drifts away, two nodes become increasingly delayed relative to the remaining node, representing most of the group moving away from a minority. To Model progressive separation, delays are injected over 10 intervals with added 300 ms per interval starting at 0 ms.

Multi-Paxos introduces additional cases because it relies on a leader. Beyond which node(s) drift, performance may depend on whether the drifting node(s) include/exclude the current leader. We therefore evaluate Muti-Paxos under both cases while EPaxos remains defined solely by which node(s) drift because it is leaderless.

3.2 Measurements (L1/L2/L3, median vs tail)
To isolate network effect from coordination overhead, we report latency at three layers and summarize each layer using both typical (median) and worst-case behavior (tail).

L1 Network latency (Ping):
L1 measures the raw network latency between nodes, independent of any application logic. We use ICMP ping to estimate round trip time (RTT) between node pairs. Conceptually, L1 answers the question of "How long does it take for a packet to travel between two agents in the current network condition?" As injected delay increases, L1 reflects the baseline connectivity cost that higher level protocol must pay. Reporting L1 is important because it provides the ground truth network signal that later layers (RPC and consensus) build on.

L2 Baseline messaging latency (RPC):
L2 measures the latency of the normal communication path used by the system without running consensus. In our setup, L2 is measured by sending a request via an RPC call and timing the end-to-end response. Conceptually, L2 answers: "If agents just exchange messages directly, what is the latency cost under the current network delay?" This layer includes serialization, RPC framework overhead, and one request/response round trip, but it excludes any multi-replica coordination. L2 serves as the baseline for "best effort" communication, and it is the reference point we compare against when interpreting consensus overhead.

L3 Coordination latency (Consensus commit):
L3 measures the end-to-end latency to commit an operation using a consensus protocol. We time for when a client submits a request to when the protocol reports the operation as committed according to its correctness conditions. L3 answers: "How long does it take to reach agreement and make an update durable/visible under the current network delay?" L3 includes protocol overhead: additional message rounds, quorum communication, leader involvement and potential leader sensitivity, and any retransmissions or timeouts.

For EPaxos, our evaluation is limited to the fast path only, meaning we measure the commit latency when commands proceed without triggering the slow path. We did not include the slow path because generating reliable, repeatable command conflicts that consistently trigger EPaxos' slow path behavior proved difficult in our setup. As a result, the EPaxos L3 results do not include dependency resolution or other slow path coordination costs and should be interpreted as fast path latency under low conflict conditions.

3.4 Results

L1 (Avg): 219.58µs
L2 (Avg): 385.67µs

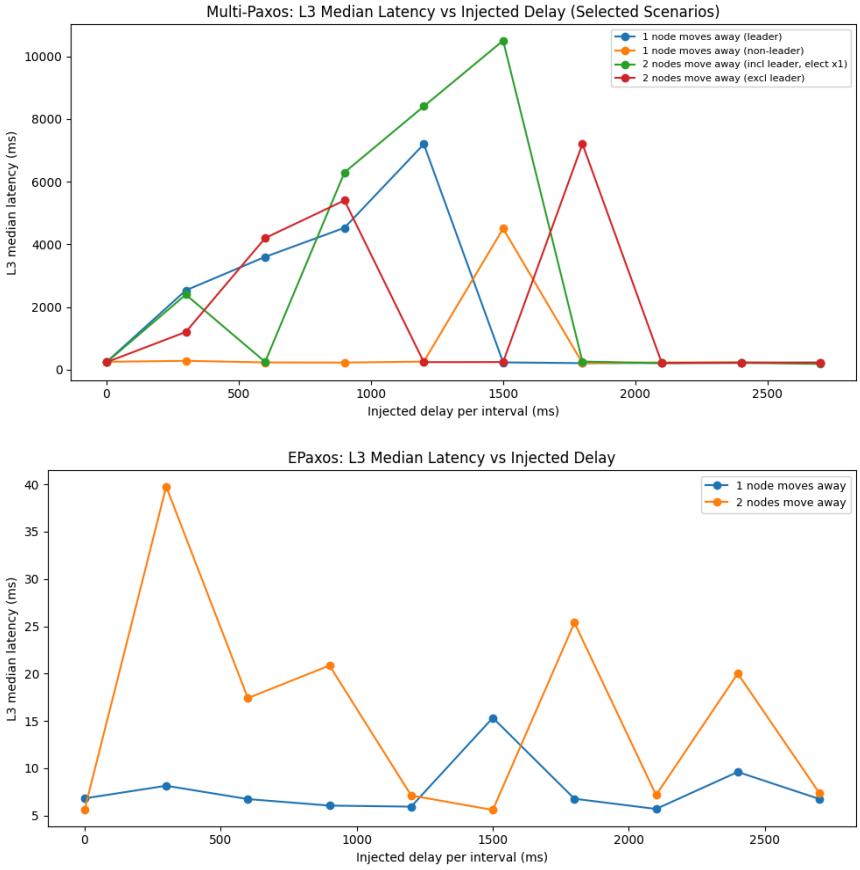

Figure 3: L3 (Median)

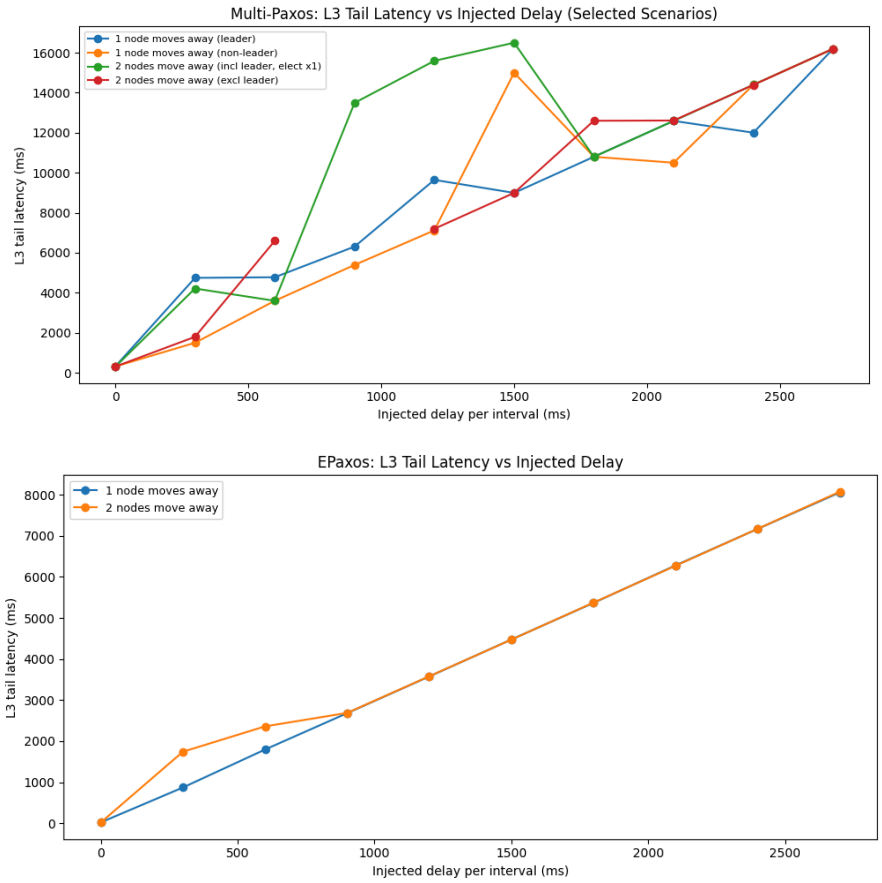

Figure 4: L3 (Tail)

The L1 (ping) and L2 (baseline RPC) measurements behave as expected in CloudLab, since the testbed runs on reserved physical machines in a controlled environment with minimal external interference. As injected delay increases, L1 and L2 primarily reflect the delay we intentionally introduced.

For Multi-Paxos, the L3 median curve shows gradual increases followed by sharp drops. This pattern is consistent with Multi-Paxos' leader based structure: when the current leader becomes slow or effectively unreachable, a leader re-election can occur. Once a new leader is chosen with better connectivity, commit latency improves noticeably, which explains the sudden drops. The L3 tail latency shows a similar rise-and-drop behavior, but with larger spikes because timeouts and re-election events disproportionately affect worst-case behavior. One notable case is when two nodes drift away while the leader stays, where the system may struggle to complete re-election reliably. As delays grow, election attempts can repeatedly time out and fail to reach quorum, leading to persistently high tail latency.

For EPaxos, the L3 median latency is substantially lower than Multi-Paxos across the tested delays. This is expected because EPaxos is leaderless and avoids the fixed route through a leader

cost present in leader based protocols. In our evaluation, EPaxos is measured on the fast path only, which further reduces coordination overhead compared to scenarios that would trigger slow-path behavior. The L3 tail latency for EPaxos increases with injected delay and, beyond roughly ~1000 ms, becomes dominated by the network delay itself, indicating that under large separation the protocol's worst-case behavior is constrained primarily by communication latency rather than compute.

4 Conclusion and Future Work

Even under the most favorable EPaxos conditions tested, the consensus commit latency (L3) remains meaningfully higher than the baseline RPC latency (L2). This reflects a fundamental coordination cost: L2 represents best-effort messaging, while L3 includes additional protocol communication needed to reach agreement. The persistent L3-L2 gap suggests that, although leaderless fast-path EPaxos reduces overhead compared to Multi-Paxos, further progress is needed to bring coordination latency closer to baseline communication. Future work can explore more lightweight coordination mechanisms or application aware designs that exploit the structure of collaborative perception workloads to reduce consensus overhead while maintaining acceptable correctness guarantees.

As a next step, we plan to bring this evaluation beyond CloudLab by replicating the same experiments on a physical, scaled-down autonomous vehicle platform using RoboRacer[8] toy cars. Unlike a controlled testbed, real deployments introduce additional variability from wireless interference, changing link quality, sensor and compute delays, and environmental dynamics. By running the same minority/majority drift scenarios on RoboRacer vehicles and comparing the resulting L1/L2/L3 latency behavior to our CloudLab measurements, we can assess how well the observed trends generalize under more realistic conditions and identify new bottlenecks that do not appear in a controlled environment.

Bibliography

1] L. Lamport, “The Part-Time Parliament,” ACM Transactions on Computer Systems, vol. 16, no. 2, pp. 133–169, May 1998.

[2] I. Moraru, D. G. Andersen, and M. Kaminsky, “There is More Consensus in Egalitarian Parliaments,” in Proc. 24th ACM Symposium on Operating Systems Principles (SOSP), Nov. 2013. doi: 10.1145/2517349.2517350.

[3] G. Luo et al., “EdgeCooper: Network-Aware Cooperative LiDAR Perception for Enhanced Vehicular Awareness,” IEEE Journal on Selected Areas in Communications, vol. 42, no. 1, pp. 207–222, Jan. 2024. (Figure 1).

[4] Little-Podi, “Collaborative_Perception,” GitHub repository. https://github.com/Little-Podi/Collaborative_Perception.

[5] S. Wei, Y. Wei, Y. Hu, Y. Lu, Y. Zhong, S. Chen, and Y. Zhang, “Asynchrony-Robust Collaborative Perception via Bird’s Eye View Flow,” in Advances in Neural Information Processing Systems (NeurIPS), 2023.

[6] Z. Ding, J. Fu, S. Liu, H. Li, S. Chen, H. Li, S. Zhang, and X. Zhou, “Point Cluster: A Compact Message Unit for Communication-Efficient Collaborative Perception,” in International Conference on Learning Representations (ICLR), 2025.

[7] D. Duplyakin et al., “The Design and Operation of CloudLab,” in Proc. USENIX Annual Technical Conference (USENIX ATC), 2019.

[8] RoboRacer Foundation, “RoboRacer,” https://roboracer.ai/.

[9] L. Lamport, “Paxos Made Simple,” ACM SIGACT News, vol. 32, no. 4, pp. 51–58, Dec. 2001.