

Queue-Sync: A Queue-Aware Delay-Compensated Time Synchronization Framework for High-Dynamic V2X Networks

Zonghao Feng^{1,2}, Shuohan Liu^{1,2*}, Huiling Shi^{1,2*}, Xu Zhang³, Hongji Fang⁴, Yongke Xi⁴

¹ Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, China

² Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Jinan, China

³ School of Computing Sciences, University of East Anglia (UEA), Norwich

⁴ Shandong High-Speed Information Group Co., Ltd. Jinan, China

Email: 10431240243@stu.qlu.edu.cn, shuohan.liu@qlu.edu.cn, shihl@sdas.org, x.zhang27@uea.ac.uk, fanghongji@sdhsg.com, xiyk_living@sdhsg.com

Abstract—Vehicle-to-Everything (V2X) systems that support cooperative perception and control require precise network-wide time synchronization. In dense traffic, however, existing schemes often decouple synchronization from communication delay. As a result, rapidly varying uplink queueing delays remain uncompensated and introduce persistent asymmetric bias that degrades synchronization accuracy. To address this issue, we have proposed Queue-Sync, a delay-compensated time synchronization framework for a Cloud-RSU-EV (CRE) architecture. Queue-Sync consists of a roadside Dynamic Delay Prediction Module (DPM) and an on-vehicle Delay-Compensated Synchronization Module (DCM). The DPM combines an M/M/1 queueing model with an adaptive extended Kalman filter (AEKF) to estimate the uplink arrival rate and queueing delay online. The DCM performs asymmetric timestamp-based synchronization. It injects the predicted delay into the offset calculation for feedforward compensation, and feeds back the measured round-trip time (RTT) to calibrate the DPM. Based on an extended fine-grained clock model, simulation results show that Queue-Sync consistently yields a lower average synchronization error compared to baseline schemes. This improvement holds under different synchronization intervals and traffic densities. It achieves a smoother convergence and more stable clock synchronization performance at the cost of a modest increase in synchronization message overhead.

Keywords—Vehicle-to-Everything (V2X), time synchronization, queueing delay, delay compensation, extended Kalman filter

I. INTRODUCTION

Vehicle-to-Everything (V2X) systems are evolving toward cooperative perception and computation [1]. They are becoming a key part of future 5G/6G intelligent transportation systems [2]. Applications such as cooperative perception, cooperative control, platooning, and vehicular task offloading require tight temporal and network-wide consistency. Therefore, vehicles and roadside units (RSUs) must share a common time reference for timestamping, sensor fusion, and decision making.

However, V2X wireless links are strongly affected by traffic load and scheduling in practice [3]. Contention, retransmissions, and priority access on the uplink create deep queues. As a result, uplink queueing delay can dominate the round-trip time (RTT) under heavy load and induce asymmetric, time-varying one-way delays [4]. These delay fluctuations invalidate the symmetric-path assumption underlying conventional four-timestamp synchronization. Consequently, uplink queueing bias cannot be canceled and synchronization error grows quickly in dense traffic. Simply shortening the synchronization period or increasing the message rate may temporarily reduce error. However, it further increases uplink load and queueing.

Existing studies have identified this contradiction from both control and networking viewpoints. Under high density or blockage, vehicular links exhibit large delay randomness and reliability loss [3]. In addition, multihop offloading and forwarding in Internet-of-Vehicles (IoV) scenarios further reduce connection time and increase end-to-end delay jitter [5]. In cooperative control, communication delay, packet loss, and topology changes can break platoon stability and spatio-temporal coherence [6]. Moreover, cooperative adaptive cruise control (CACC) cannot guarantee string stability when the delay reaches 0.1–0.3 s [7]. In cooperative merging, uncertain vehicle-to-infrastructure (V2I) delay may destabilize trajectories and increase collision risk [8].

From the networking perspective, queueing delay is the most uncertain component of end-to-end latency, as it is driven by arrival rate, service rate, scheduling, and traffic bursts [9]. Dynamic offloading of multi-access edge computing (MEC), digital twin, and IoV tasks amplifies load fluctuations [10], [11]. In addition, V2X cooperative perception and autonomous driving services further introduce bursty, time-varying traffic load [12].

Time synchronization is highly vulnerable to these link dynamics. Surveys report that synchronization errors in vehic-

ular ad hoc networks stem from propagation delay, uplink-downlink asymmetry, queueing residence time, and observation gaps due to mobility [13]. Degraded synchronization precision harms cooperative perception and control and weakens security mechanisms that depend on information freshness [14].

Existing vehicular time-synchronization schemes are mainly protocol-based or filter-based. Protocol-based methods exchange a few timestamps. They assume negligible queueing or symmetric one-way delays. When the uplink to the RSU is congested, the offset estimate becomes biased. Filter-based methods use Kalman filtering. They include the extended Kalman filter and the median Kalman filter. These filters smooth noisy RTT observations. They mitigate random jitter and mobility effects [15]–[17]. However, both categories treat RTT as a black-box. They ignore queue-aware delay modeling. Thus, uplink-queueing-induced bias persists under heavy load. This motivates our queue-aware, delay-compensated framework.

In summary, existing queue-unaware time-synchronization research followed several lines. The work in [13] surveyed protocol-level designs for vehicular networks. It showed that most schemes still relied on symmetric or slowly varying delay assumptions when deriving offset estimators.

Building on such protocol models, the work in [15] introduced a Kalman-filter-based scheme that explicitly tracked clock offset from noisy bidirectional delay measurements. This scheme thereby reduced the impact of random propagation jitter. The work in [16] further extended this filtering idea by jointly estimating clock offset and skew with an EKF. This extension ensured that long-term drift could also be compensated. To enhance robustness under highly dynamic and abnormal links, the work in [17] replaced the standard Kalman filter with a median-KF structure. This structure suppressed outliers observed on mobile ad hoc links.

However, despite these advances, all these works shared a common limitation: they treated the RTT as a black-box noisy measurement. They did not build queue-aware delay models that exposed arrival rate, service rate, or scheduling discipline for explicit compensation. As a result, the systematic bias introduced by uplink queueing persisted under high traffic load. This situation left a gap for queue-aware, delay-compensated time-synchronization frameworks.

To address this gap, We innovatively construct a queue-aware closed-loop synchronization Queue-Sync framework, within which two key innovations are introduced:

- A queueing-delay-driven feedforward compensation mechanism that predicts uplink queueing delay and injects it into timestamp-based offset estimation to eliminate persistent asymmetric bias.
- A closed-loop calibration mechanism where RTT feedback continuously refines the queueing-delay predictor, enabling stable synchronization under highly dynamic V2X traffic.

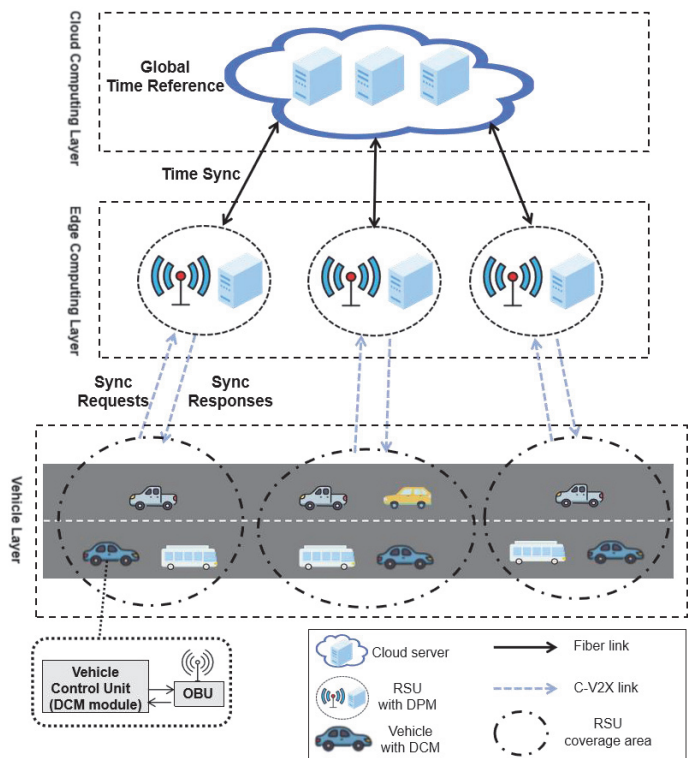


Fig. 1. System architecture of Queue-Sync showing cloud synchronization, RSU-EV links, and delay-affected timestamp exchange.

II. PRELIMINARY

This section models and describes the system upon which Queue-Sync relies. It includes the three-layer CRE architecture, the node clock model, and the dynamic communication delay model dominated by uplink queueing. It also covers the interaction workflow among entities and the formal problem formulation.

A. System Architecture

The system architecture is shown in Fig. 1. It consists of a cloud server, multiple RSUs, and a large number of EVs. The cloud server connects to all RSUs via wired backhaul links and provides a global time reference. Each RSU communicates with EVs through cellular vehicle-to-everything (C-V2X) links and acts as the local master clock for the EVs within its coverage area. It maintains per-link synchronization state and collects RTT measurements. Each EV is equipped with an on-board unit (OBU) that can exchange timestamped messages with the serving RSU. It can adjust its local hardware clock based on received synchronization information. The specific prediction and compensation mechanisms built on top of this architecture are described in Section III.

B. Node Clock Model

We formalize the clock asynchrony among nodes. In a distributed V2X network, the local hardware clock of any node

i (EV) is not ideal. Its local time $C_i(t)$ is modeled by a linear affine relation with the true physical time t :

$$C_i(t) = \alpha_i t + \beta_i, \quad (1)$$

where α_i is the clock skew and β_i is the clock offset. Time synchronization aims to estimate and compensate these parameters so that the corrected local clock approaches the reference clock. In this work, vehicle and RSU oscillators are assumed relatively stable over the considered synchronization intervals. Therefore, we set $\alpha_i = 1$ and focus on estimating and correcting β_i .

C. Communication Delay Model

In the V2X scenario, the RTT of synchronization messages between an EV and an RSU can be decomposed into two one-way components. The uplink delay is denoted by D_{up} (from the EV to the RSU), and the downlink delay is denoted by D_{down} (from the RSU to the EV). Each one-way delay consists of propagation, transmission, processing, and queueing components. For modeling convenience, the relatively stable parts are grouped as $D_{fixed} = D^{prop} + D^{trans} + D^{proc}$, where D^{prop} , D^{trans} , and D^{proc} denote the propagation, transmission, and processing delays, respectively. The traffic-dependent queueing waiting time is denoted by D_{queue} . Under this decomposition, the one-way delays can be expressed as

$$D_{up} = D_{fixed} + D_{queue}, \quad (2)$$

$$D_{down} \approx D_{fixed}, \quad (3)$$

where D_{queue} denotes the queueing delay in the RSU uplink buffer. As the traffic load increases, D_{queue} grows and induces uplink-downlink asymmetry, violating the symmetric-delay assumption of PTP/NTP and biasing four-timestamp offset estimation.

D. Interaction Workflow

Under the CRE architecture in Fig. 1, Queue-Sync operates in periodic synchronization rounds. The cloud first synchronizes all RSUs so that each RSU serves as a local time reference for its covered EVs. During each round, the serving RSU and an EV perform a brief two-way timestamp exchange over the C-V2X link. The RSU embeds its current reference time in the downlink message. The EV responds with a timestamped message, updates its local clock, and reports the measured RTT. The RSU then uses these RTT reports to update its internal view of the uplink delay state. This operation forms a lightweight closed-loop interaction on top of the underlying V2X protocols. The prediction and compensation logic of this closed loop is detailed in Section III.

E. Problem Formulation

In high-dynamic V2X networks, random and asymmetric D_{queue} breaks the symmetric-delay assumption, so four-timestamp offset estimation suffers persistent bias. Load variations further make high-precision synchronization hard to

sustain. Under the CRE architecture with a fixed synchronization interval, we aim to design a closed-loop, queue-aware, and delay-compensated mechanism. This mechanism integrates RTT with queueing modeling by actively tracking the uplink D_{queue} . It then injects this prediction into the offset estimation to mitigate queueing-induced asymmetry.

III. QUEUE-SYNC FRAMEWORK

This section presents the Queue-Sync framework. It includes its closed-loop structure and the functions and information exchange between the roadside DPM and the vehicle-side DCM.

A. Framework Overview

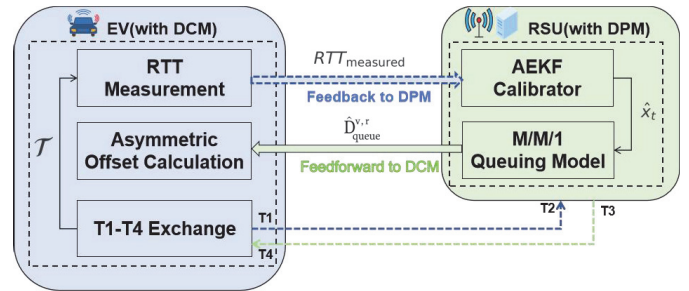


Fig. 2. Closed-loop operation of the Queue-Sync framework.

The closed-loop operation of Queue-Sync is illustrated in Fig. 2. Its main functional components are as follows.

On the EV side, the OBU integrates the DCM. The DCM periodically exchanges timestamps T_1 – T_4 with the serving RSU. From each exchange, the DCM computes the measured RTT, denoted by $RTT_{measured}$. The RSU then returns the predicted uplink queueing delay, denoted by $\hat{D}_{queue}^{v,r}$. The DCM embeds $\hat{D}_{queue}^{v,r}$ into the asymmetric delay-compensation formula. It obtains a corrected clock-offset estimate. It then updates the local clock.

On the RSU side, the DPM models and predicts the queueing behavior online. It adopts an M/M/1 queue. It uses the vehicle-reported $RTT_{measured}$ as observations. The DPM applies an adaptive extended Kalman filter (AEKF). The AEKF calibrates the model parameters. The DPM then outputs the predicted uplink queueing delay $\hat{D}_{queue}^{v,r}$.

In Queue-Sync, the EV-to-RSU direction carries $RTT_{measured}$ as feedback. This feedback helps the DPM track the uplink queueing state. The RSU-to-EV direction delivers $\hat{D}_{queue}^{v,r}$ as a feedforward term to the DCM. This design enables delay-compensated offset estimation. It mitigates uplink queueing fluctuations under a fixed synchronization period.

B. DCM

The DCM is deployed in the OBU. It serves as the execution and sensing unit of Queue-Sync on the EV side. In each synchronization session, the EV and the RSU perform one two-way timestamp exchange. The EV sends a request at T_1 .

The RSU receives it at T_2 and sends a response at T_3 . The EV receives the response at T_4 . The timestamp set is denoted by $\mathcal{T} = \{T_1, T_2, T_3, T_4\}$. Recalling the delay decomposition, the uplink delay from vehicle v to RSU r is written as $D_{\text{up}} = D_{\text{fixed}} + D_{\text{queue}}^{v,r}$. The clock offset estimate after queueing compensation is

$$\beta_v^{\text{asym}} = \frac{(T_2 - T_1) - (T_4 - T_3) - \hat{D}_{\text{queue}}^{v,r}}{2}, \quad (4)$$

where $\hat{D}_{\text{queue}}^{v,r}$ is provided by the RSU-side DPM. It serves as a feedforward compensation term. Based on this estimate, the DCM performs local time correction according to (1).

The DCM also measures the link state. The measured RTT for one session is

$$RTT_{\text{measured}} = (T_4 - T_1) - (T_3 - T_2), \quad (5)$$

where RTT_{measured} contains the current uplink queueing component. The DCM sends RTT_{measured} back to the RSU. It serves as the observation input for the AEKF in the DPM. It is used to update the queueing model parameters online. The DCM thus corrects the local clock and reports RTT measurements.

During the cold-start phase, the DCM can initiate a few synchronization rounds without compensation. It obtains initial RTT_{measured} values. These values help the DPM complete parameter initialization. When mobility causes synchronization timeouts or RSU handovers, a valid RTT_{measured} may be missing. In that case, the DPM performs only the prediction step. The closed-loop process resumes once a new synchronization session is established.

C. DPM with AEKF

The DPM is deployed on the RSU side and acts as the predictor in the Queue-Sync framework. It estimates the uplink queueing delay online based on queueing modeling and an AEKF. It outputs the feedforward compensation amount to the vehicle side.

To balance modeling capability and real-time performance, the DPM adopts an M/M/1 queue to characterize the RSU uplink buffer. Let the arrival rate at time t be $\lambda_{\text{arr},t}$ and the service rate be μ . When the traffic intensity $\rho_t = \lambda_{\text{arr},t}/\mu < 1$, the average queueing delay is

$$\mathbb{E}[D_{\text{queue}}] = \frac{\lambda_{\text{arr},t}}{\mu(\mu - \lambda_{\text{arr},t})} = \frac{\rho_t}{\mu(1 - \rho_t)}, \quad (6)$$

where $\lambda_{\text{arr},t}$ and ρ_t denote the instantaneous arrival rate and traffic intensity, respectively.

Although the M/M/1 assumption may not fully hold under bursty or self-similar V2X arrivals, the resulting model mismatch is partially mitigated by the closed-loop design. Specifically, the AEKF continuously re-calibrates the delay estimate using the measured $RTT_{\text{measured},t}$. This feedback helps mitigate long-term error accumulation.

Let x_t denote $\lambda_{\text{arr},t}$. Let $\hat{x}_t$ be the state estimate produced by the AEKF at time t . The predicted uplink queueing delay for vehicle v and RSU r , denoted by $\hat{D}_{\text{queue}}^{v,r}$, is obtained by

substituting $\hat{x}_t$ into (6). It is then sent via a control channel to the EV-side DCM. There, it is used for the asymmetric delay compensation in (4).

The DPM employs an AEKF to calibrate the queueing model online. It uses the measured $RTT_{\text{measured},t}$ returned by the inner-loop DCM as the observation z_t . The arrival rate is treated as a slowly varying random process and modeled as a random walk:

$$x_t = x_{t-1} + w_{t-1}, \quad (7)$$

where w_{t-1} is the process noise with zero mean and covariance Q_{t-1} .

The measurement equation exploits the fact that the RTT contains both fixed delay and queueing delay and is written as

$$z_t = RTT_{\text{measured},t} = h(x_t) + v_t, \quad (8)$$

with

$$h(x_t) = D_{\text{fixed}} + \frac{x_t}{\mu(\mu - x_t)}, \quad (9)$$

where v_t is the measurement noise with zero mean and covariance R .

In each synchronization interval, the AEKF first propagates the *a priori* state according to (7). It then uses the innovation between the actual observation z_t and the model prediction $h(\hat{x}_t^-)$ to obtain the *a posteriori* estimate $\hat{x}_t$. Here, $\hat{x}_t^-$ denotes the *a priori* estimate. The covariance matrices Q_{t-1} and R can be adaptively adjusted according to the innovation statistics to enhance robustness. The updated $\hat{x}_t$ is then substituted into (6) to compute $\hat{D}_{\text{queue}}^{v,r}$. Together with the compensation and measurement performed by the inner-loop DCM, this forms a closed loop of prediction, calibration, compensation, and feedback.

IV. PERFORMANCE EVALUATION

A. Simulation Configuration

We use the Opportunistic Network Environment (ONE) [18] to emulate a three-layer CRE architecture. The cloud connects to all RSUs via wired links and provides a global time reference. RSUs communicate with passing EVs over wireless links. EVs initiate time-synchronization sessions when entering RSU coverage.

The road topology uses the Helsinki city map of about $4.5 \text{ km} \times 3.4 \text{ km}$. Vehicles move along shortest paths at 30–50 km/h. They repeatedly undergo an attach-synchronize-depart process.

To model clock drift and link asymmetry, each RSU and EV maintains an independent logical clock and generates timestamps locally. One-way delay includes a fixed component and a queueing component. On the uplink, synchronization messages share the RSU access queue with data traffic and experience load-dependent queueing. On the downlink, only a small random jitter is added, yielding pronounced uplink queueing and stable downlink delay. All experiments keep this setting and vary only one factor at a time.

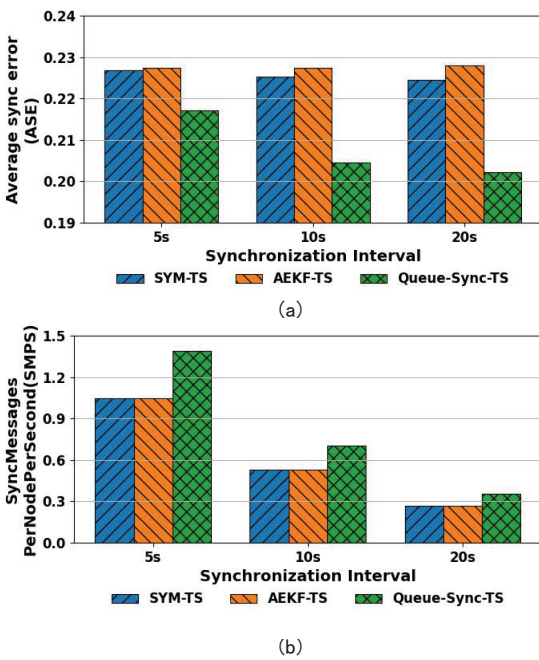


Fig. 3. Influence of synchronization interval. (a) ASE. (b) SMPS.

B. Comparison Configuration

To assess Queue-Sync, we compare three synchronization schemes under the same simulation environment:

1) **SYM-TS**: A standard symmetric-delay synchronization scheme. It follows the standard four-timestamp exchange used in NTP/PTP-style synchronization [13]. It assumes equal uplink and downlink delays, ignores RSU queueing, and does not filter the RTT, serving as a baseline.

2) **AEKF-TS**: A queue-aware but filtering-only scheme. The RSU models the uplink as an M/M/1 queue [9] and applies an AEKF to smooth the measured RTT, following EKF-based clock-parameter tracking ideas in wireless sensor networks [16]. The EV still runs symmetric-delay synchronization with a fixed interval and no feedforward compensation.

3) **Queue-Sync-TS**: A queue-aware and delay-compensated synchronization scheme. The RSU predicts the uplink queueing delay using the M/M/1 model and AEKF and sends this prediction to the EV. The EV adopts an asymmetric delay model and injects the predicted delay into the offset estimation.

Performance is evaluated using three metrics:

1) **Average synchronization error (ASE)**: The mean clock offset between each EV and the reference clock over the whole simulation, reflecting synchronization accuracy.

2) **Synchronization messages per node per second (SMPS)**: The average number of synchronization packets per node, indicating communication overhead.

3) **Temporal evolution of ASE**: The time series of ASE, which reveals both convergence speed and steady-state error.

C. Influence of Synchronization Interval

In this set of experiments, the number of EVs, the traffic configuration, and the scenario are fixed. Only the synchro-

nization interval T_{sync} is varied. As shown in Fig. 3(a), when T_{sync} increases from 5 s to 20 s, the ASE of all three schemes grows significantly. This behavior indicates that overly infrequent synchronization aggravates the accumulated clock drift and thus degrades accuracy. For all intervals, the ASE of Queue-Sync-TS is consistently lower than that of SYM-TS and AEKF-TS. This implies that, under the same synchronization frequency, queueing modeling and feedforward compensation are beneficial. They help mitigate the impact of uplink queueing delay on synchronization accuracy.

Fig. 3(b) shows that SMPS is approximately inversely proportional to T_{sync} . Shortening the interval markedly increases the synchronization message overhead. Because Queue-Sync-TS needs to deliver predicted delay values, its SMPS is slightly higher than that of the two baselines at the same T_{sync} . Considering both synchronization accuracy and messaging overhead, $T_{\text{sync}} = 10$ s provides a reasonable compromise in this scenario. This setting is adopted in the remaining experiments.

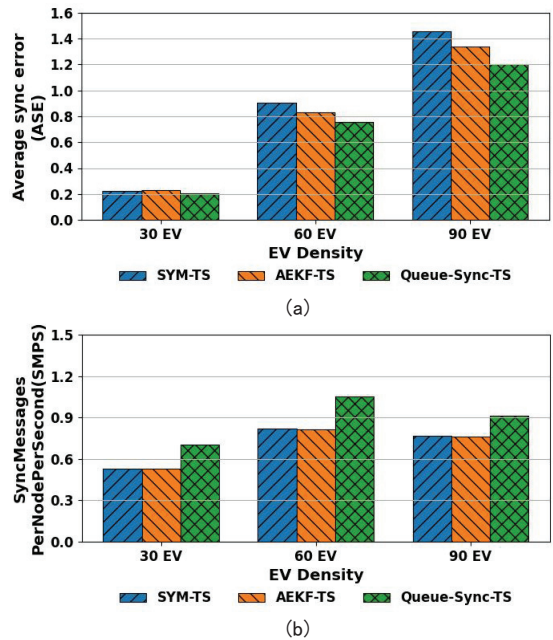


Fig. 4. Influence of EV density. (a) ASE. (b) SMPS.

D. Influence of EV Density

This set of experiments uses the default configuration and varies only the number of EVs on the road segment to control the traffic density. As shown in Fig. 4(a), the ASE of all three schemes increases with EV density, indicating that heavy traffic intensifies queueing-delay fluctuations and link asymmetry, thereby degrading synchronization accuracy. SYM-TS degrades fastest, AEKF-TS benefits from RTT filtering and achieves slightly smaller steady-state error, while Queue-Sync-TS consistently attains the lowest ASE and the flattest growth curve. These results show that, under the same synchronization frequency, queueing modeling and delay compensation more

effectively suppress synchronization errors in dense-traffic scenarios.

Fig. 4(b) shows that the normalized SMPS changes little with EV density, but Queue-Sync-TS is consistently slightly higher than the other two schemes because it transmits compensation information. Overall, as EV density increases, Queue-Sync-TS trades a moderate increase in message count for significantly more stable synchronization performance.

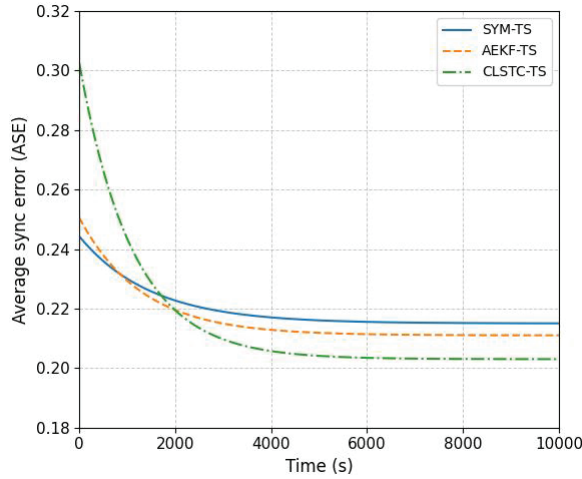


Fig. 5. ASE evolution over time for different schemes.

E. Temporal Evolution of ASE

In the final set of experiments, the temporal evolution of ASE is observed over a longer simulation horizon. As depicted in Fig. 5, the average synchronization error of all three schemes quickly decreases at the beginning and then enters a steady region. However, their convergence processes and steady-state levels differ markedly. SYM-TS converges rapidly but stabilizes at a relatively high error level. AEKF-TS leverages RTT filtering to slightly reduce the steady-state error. Yet its long-term improvement is limited.

Queue-Sync-TS, in contrast, exhibits a somewhat larger error at cold start because the queueing model is not yet fully calibrated. As the DPM and AEKF learn online, the predicted delay becomes more accurate. The error curve continues to decrease, and Queue-Sync-TS eventually reaches the lowest steady-state level among all schemes. Overall, Queue-Sync-TS outperforms the baselines in both convergence speed and steady-state accuracy. This makes it more suitable for long-term operation in high-dynamic V2X environments.

V. CONCLUSION

This paper has proposed Queue-Sync, a queue-aware closed-loop delay-compensated time-synchronization framework for V2X networks. Queue-Sync jointly models uplink queueing delay and performs asymmetric compensation based on delay prediction. Simulation results show that Queue-Sync-TS reduces ASE and improves stability compared with SYM-TS and AEKF-TS.

Future work will extend Queue-Sync to more general queueing models and hardware testbeds. It will also investigate adaptive synchronization intervals under mixed V2X traffic.

ACKNOWLEDGMENT

This research was partially funded by the National Natural Science Foundation of China (62501339), the Natural Science Foundation of Shandong Province under Grant (ZR2025QC713, ZR2023LZH011), the Taishan Scholar Program of Shandong Province, China (TSQN202306258, TSN202312230).

REFERENCES

- [1] M. Noor-A-Rahim, Z. Liu, H. Lee, M. O. Khyam, J. He, D. Pesch, K. Moessner, W. Saad, and H. V. Poor, "6g for vehicle-to-everything (v2x) communications: Enabling technologies, challenges, and opportunities," *Proceedings of the IEEE*, vol. 110, no. 6, pp. 712–734, 2022.
- [2] Q. Cao, J. Li, and P. Trucco, "Intelligent transportation aiding urban sustainability: An empirical analysis with reinforcement and deep learning," in *2025 28th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2025, pp. 7–12.
- [3] L. Bai, Z. Huang, Y. Ge, R. Yu, L. Wang, and X. Cheng, "Cellular Vehicle-to-Everything (C-V2X) Testing: From Theory to Practice."
- [4] C. Chen, J. Luo, D. Shen, Z. Xu, and R. Xiong, "Achieving low queuing latency in time-slotted lora networks," in *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2024, pp. 3237–3242.
- [5] C. Chen, Y. Zeng, H. Li, Y. Liu, and S. Wan, "A Multihop Task Offloading Decision Model in MEC-Enabled Internet of Vehicles," vol. 10, pp. 3215–3230, 2023.
- [6] P. Wang, H. Deng, J. Zhang, L. Wang, M. Zhang, and Y. Li, "Model predictive control for connected vehicle platoon under switching communication topology," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7817–7830, 2021.
- [7] B. Tian, G. Wang, Z. Xu, Y. Zhang, and X. Zhao, "Communication delay compensation for string stability of CACC system using LSTM prediction," vol. 29, p. 100333, 2021.
- [8] Y. Fang, H. Min, X. Wu, W. Wang, X. Zhao, and G. Mao, "On-Ramp Merging Strategies of Connected and Automated Vehicles Considering Communication Delay," vol. 23, pp. 15 298–15 312, 2022.
- [9] A. Roy, J. L. Pachua, and A. K. Saha, "An overview of queuing delay and various delay based algorithms in networks," vol. 103, pp. 2361–2399, 2021.
- [10] D. Ren, X. Gui, and K. Zhang, "Adaptive Request Scheduling and Service Caching for MEC-Assisted IoT Networks: An Online Learning Approach," vol. 9, pp. 17 372–17 386, 2022.
- [11] J. Yang, F. Lin, C. Chakraborty, K. Yu, Z. Guo, A.-T. Nguyen, and J. J. P. C. Rodrigues, "A Parallel Intelligence-Driven Resource Scheduling Scheme for Digital Twins-Based Intelligent Vehicular Systems," vol. 8, pp. 2770–2785, 2023.
- [12] T. Huang, J. Liu, X. Zhou, D. C. Nguyen, M. R. Azghadi, Y. Xia, Q.-L. Han, and S. Sun, "Vehicle-to-everything cooperative perception for autonomous driving," *Proceedings of the IEEE*, 2025.
- [13] K. F. Hasan, C. Wang, Y. Feng, and Y.-C. Tian, "Time synchronization in vehicular ad-hoc networks: A survey on theory and practice," vol. 14, pp. 39–51, 2018.
- [14] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing Vehicle-to-Everything (V2X) Communication Platforms," vol. 5, pp. 693–713, 2020.
- [15] X. Ni, T. Lu, S. Ye, Y. Zheng, P. Chen, and L. Chen, "Pair Nodes Clock Synchronization Algorithm Based on Kalman Filter for Underwater Wireless Sensor Networks," vol. 21, p. 4426, 2021.
- [16] H. Wang, R. Lu, Z. Peng, and M. Li, "Timestamp-Free Clock Parameters Tracking Using Extended Kalman Filtering in Wireless Sensor Networks," vol. 69, pp. 6926–6938, 2021.
- [17] Y. Jeon, T. Kim, and T. Kim, "Fast and Robust Time Synchronization with Median Kalman Filtering for Mobile Ad-Hoc Networks," vol. 21, p. 590, 2021.
- [18] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *Proceedings of the Second International ICST Conference on Simulation Tools and Techniques*. ICST, 2009.