Channel-Aware QUIC Control for Enhanced CAM Communications in C-V2X Deployments Over Aerial Base Stations

George Amponis, *Graduate Student Member, IEEE*, Thomas Lagkas, *Senior Member, IEEE*, Vasileios Argyriou, *Member, IEEE*, Panagiotis Radoglou-Grammatikis, *Member, IEEE*, Konstantinos Kyranou, Ioannis Makris, and Panagiotis Sarigiannidis, *Member, IEEE*

***Abstract*—The proliferation of vehicular ad hoc networks necessitates efficient data transfer protocols, particularly in the context of Cellular Vehicle-to-Everything (C-V2X) communications. This paper focuses on enhancing the performance of the Quick UDP Internet Connections (QUIC) protocol, focusing on cooperative vehicular networks supported by aerial drone relays. While QUIC outperforms traditional protocols, its default congestion and flow control mechanisms do not adequately address the unique challenges posed by volatile networks spanning the terrestrial and aerial domains, as they are characterized by frequent topology changes, and high propagation delay volatility. We analyze QUIC's congestion and flow control and propose enhancements to optimize its performance in such networks, specifically designed for C-V2X communications in Open Radio Access Networks (O-RAN). Our proposal adjusts connections' congestion window size and individual streams' flow control windows in a channel-aware manner. Simulation experiments assess the performance of our proposal, comparing it with QUIC's default mechanisms. Our proposal can be seamlessly integrated into existing implementations, making it a viable approach for improving performance and addressing the challenges specific to vehicle-to-drone communications. By addressing QUIC's limitations and optimizing its performance for C-V2X applications in O-RAN, our enhancement offers a valuable contribution towards enabling low-latency, and resource-aware vehicular communications for the realization of autonomous driving and advanced vehicular services.**

***Index Terms*—Connected vehicles-to-everything, QUIC, aerial base stations, channel-aware flow & congestion control.**

Manuscript received 30 June 2023; revised 6 March 2024 and 19 April 2024; accepted 21 April 2024. Date of publication 25 April 2024; date of current version 16 July 2024. This work was supported by the European Union's Horizon Europe research and innovation programme under Grant 101096456. The review of this article was coordinated by the TVT Administrator. *(Corresponding author: Thomas Lagkas.)*

George Amponis is with the Department of R&D, K3Y Ltd., 1000 Sofia, Bulgaria, and also with the Department of Computer Science, Democritus University of Thrace, 65404 Kavala, Greece (e-mail: gamponis@k3y.bg, geaboni@cs.duth.gr).

Thomas Lagkas is with the Department of Computer Science, Democritus University of Thrace, 65404 Kavala, Greece (e-mail: tlagkas@cs.duth.gr).

Vasileios Argyriou is with the Department of Networks and Digital Media, Kingston University, KT2 7LB London, U.K. (e-mail: vasileios.argyriou@kingston.ac.uk).

Panagiotis Radoglou-Grammatikis is with the Department of R&D, K3Y Ltd., 1000 Sofia, Bulgaria, and also with the Department of Electrical and Computer Engineering, University of Western Macedonia, 50150 Kozani, Greece (e-mail: pradoglou@k3y.bg).

Konstantinos Kyranou is with the SIDROCO Holdings Ltd., 1082 Nicosia, Cyprus (e-mail: kkyranou@sidroco.com).

Ioannis Makris is with the MetaMind Innovations, 50100 Kozani, Greece (e-mail: makris@metamind.gr).

Panagiotis Sarigiannidis is with the Department of Electrical and Computer Engineering, University of Western Macedonia, 50150 Kozani, Greece (e-mail: psarigiannidis@uowm.gr).

Digital Object Identifier 10.1109/TVT.2024.3393614

I. Introduction

As DEVELOPMENTS in the domain of transport-layer protocols have pivoted towards ensuring end-to-end delivery of application-sensitive data through an increasingly dynamic and transient network layer, new requirements and transmission provisioning necessities have emerged. In the current state of the art, we can categorize relevant protocols in two main groups, namely the reliable protocols, which offer trustworthiness at the expense of responsiveness e.g., TCP and SCTP, and the unreliable datagram protocols capable of offering great responsiveness at the cost of packet reception dependability, such as UDP, DCCP, and RTP. As a means of keeping the best of both worlds, the QUIC protocol was finalized by J. Iyengar and M. Thomson [1] in May 2021 and aims to improve upon traditional transport layer protocols. While TCP is a reliable protocol that ensures data is delivered without errors, it suffers from high latency due to its three-way handshake and slow start algorithm. UDP, on the other hand, is a much faster protocol that does not guarantee reliable delivery but is often used for real-time applications such as video streaming and gaming. QUIC was developed with several pivotal features, such as multiplexing, encryption, and importantly, a flow control mechanism.

The reasoning behind resorting to QUIC (which is a connection-oriented protocol) instead of adopting a connectionless approach) or adopting TCP directly can be further elaborated. For cooperative awareness messages (CAMs), QUIC, is able to overcome the persistent constraints of TCP's congestion and flow control mechanisms (e.g., slow start and head-of-line blocking) through stream multiplexing and per-stream flow control which results in the preservation of high throughput and low latency in environments that are characterized by diverse topologies and dynamic network characteristics. Contrary to TCP, QUIC manages congestion windows and flow control parameters for each data stream independently, therefore making the data transmission more agile and flexible in comparison

to TCP. Additionally, QUIC integrates TLS 1.3 directly into the transport layer, dramatically improving the security and leading to reduction of connection and transport latency-the most exceptional advantage in C-V2X communication where state messages are immediately exchanged within CAMs. The motivation to utilize QUIC for CAMs lies not only in its performance superiority against TCP, but also in comparison to connectionless protocols (with UDP being its transport-layer foundation). Choosing QUIC, in this case, can be seen as optimal for CAM delivery under the condition of complex and dynamic environments of vehicular communication. This connection-oriented strategy is the key point of it that offers more efficient method of communication management in high density networks where vehicles experience frequent link loss and links are either closed or reconnected. As a result, the network performance gets significantly degraded. QUIC's internal architecture is innately prepared to troubleshoot issues such as packet loss, changing bandwidth and spikes in latency, swiftly, thereby ensuring the timely and reliable delivery of the security-related data. What is critical is the presence of these features in order to be appropriate for services that depend on the ultra-reliable low-latency communications (URLLC), which is why QUIC is a very important element in vehicular networks where the traffic is always continuous, coherent, and synchronized to ensure real time.

In order to support the wider adoption of QUIC for mobile ad hoc environments, we aim to revisit the flow and congestion control mechanisms. Currently, QUIC's flow and congestion control mechanisms are sub-optimal for transient networks, which are characterized by frequent topology changes, unpredictable bandwidth, and high packet loss rates. We aim to propose an enhanced Flow Control function to implement stream multiplexing in a more efficient manner, considering the global connection optimum. Lastly, to improve congestion control, we propose using a modified version of the Swarm-HTCP (S-HTCP) additive increase-multiplicative decrease (AIMD) scaling algorithm for increased performance in high-mobility ad hoc network [2]. Our approach creates a solid abstraction layer between the volatile network medium and the QUIC overlay, effectively highlighting the protocol's deterministic behavior. By isolating the protocol from the volatile network medium, we are able to ensure that the protocol's behavior is consistent and deterministic, which is essential for reliable and efficient data transfer. Overall, our approach provides a robust and reliable solution for enhancing the congestion and flow control functions in QUIC for volatile, vehicular networks.

We emphasize the significance of CAMs in the context of O-RAN and its potential impact on the simulation and modeling of Open Cloud and C-V2X networks. Our research aims to enhance the existing frameworks by integrating improvements that address key aspects such as the design of test-bed architectures and the simulation of Open Cloud and C-V2X scenarios. In our work, we utilize amongst others, the O-RAN E2 NS-3 module [3] facilitating support for running multiple terminations of an O-RAN-compliant E2 interface within a simulation. This integration allows us to explore the dynamics of O-RAN networks more comprehensively and assess the performance of CAM message exchanges in a realistic environment. By leveraging this capability, we can evaluate the effectiveness of O-RAN deployments, identify potential bottlenecks, and optimize the network configurations. Furthermore, our research incorporates the C-V2X mode 4 ns-3 module [4] specifically tailored for C-V2X Mode 4 communications. This model builds upon the ns-3 (Device-to-Device) D2D model from NIST, providing a reliable foundation for simulating and evaluating C-V2X scenarios within the O-RAN context. By employing this model, we can assess the efficiency of CAM message delivery, analyze the impact of network dynamics on communication reliability, and explore potential optimizations to enhance the overall performance of C-V2X networks.

Our proposed improvements enable us to push the boundaries of Open Cloud and C-V2X simulation and modeling frameworks. Accurate representations of O-RAN architectures (including support for O-RAN-compliant E2 interfaces and C-V2X Mode 4 communication) allow us to simulate realistic network conditions, evaluate performance metrics, and gain insights into the behavior and interactions of various network components. We utilize said modelling capacity to propose a C-V2X application-specific variant of the QUIC protocol, which is in turn evaluated using the same emulation framework. Ultimately, our research aims not only to propose a new variant of QUIC, but also to contribute to the advancement of O-RAN technologies, the development of efficient testbed architectures, and the simulation and modeling of Open Cloud and C-V2X networks. By bridging the gap between novel application-specific protocols and vehicular communications modelling solutions, we can facilitate the optimization of network designs and pave the way for the realization of reliable and high-performing vehicular communication systems.

The remainder of this work is structured as follows. Section II provides an outline regarding the reasoning and motivation for conducting this research while it also clarifies the contributions of our work to the underlying modules it utilizes. Section III provides a comprehensive description of QUIC's features and key mechanisms placing particular emphasis on the protocol's congestion and flow control mechanisms. Section IV describes the followed methodology for implementing our proposed enhancement, providing mathematical modelling for all involved components and elaborating on our proposed bi-fold improvement. Section V provides detailed information regarding the evaluation of the proposed scheme, including obtained results and a comparative analysis. Lastly, Section VI concludes our work after a comprehensive discussion and summary of the improvements achieved.

II. Rationale and Motivation

The rapid evolution of vehicular networks and their increasing role in enabling intelligent transportation systems necessitates the development of communication protocols that can adapt to the dynamic and often unpredictable nature of these environments. The QUIC protocol, originally designed to optimize web traffic efficiency through reduced connection and transport latency, presents a compelling foundation for vehicular

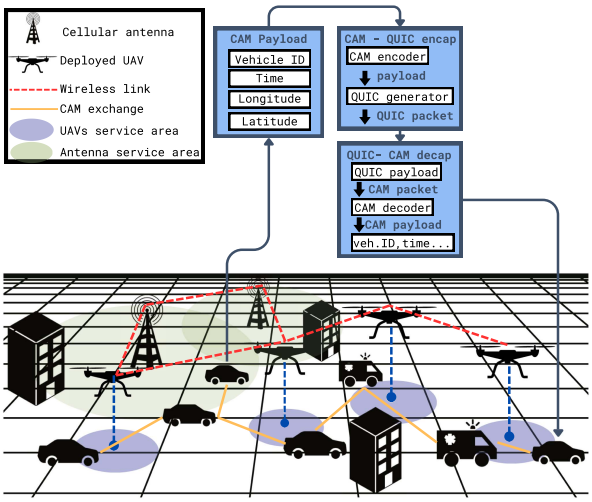

Fig. 1. Envisioned system model.

communication due to its inherent advantages over traditional protocols like TCP. However, the direct application of QUIC in vehicular networks without considering the unique challenges posed by these environments could lead to reduced performance. Research in optimization methods specifically targeting the QUIC protocol's mechanisms in order to fine-tune them for usage in vehicular and/or beyond-5G scenarios have already showcased promising results [5], [6]; our work draws motivation from this research and takes it one step further through the introduction of channel awareness and aerial link-enabled relaying. We additionally consider the outputs of authors in [7] who also considered potential enhancements in QUIC's flow control mechanisms to achieve greater responsiveness and lower latency in next-generation cellular communications.

The inclusion of Unmanned Aerial Vehicles (UAVs) in vehicular communication scenarios is highlighted by recent research, reflecting their growing importance in the wireless network landscape. UAVs serve as aerial platforms, enhancing network resilience, extending communication ranges, and facilitating rapid network deployment [8], particularly in scenarios where ground infrastructure is lacking or compromised. Their utility is especially pronounced in emergency situations where traditional communication networks may be impaired by natural disasters or catastrophic events, allowing for swift establishment of ad-hoc networks to ensure uninterrupted communication for rescue and coordination efforts [9]. Further advancing the integration of UAVs, recent studies have demonstrated their role in C-V2X communication between Connected and Autonomous Vehicles (CAVs) and UAVs [10]; utilizing a combination of communication technologies including Dedicated Short Range Communication (DSRC), User Datagram Protocol (UDP), internet-based WebSocket, and Transmission Control Protocol (TCP), aerial links can support various use cases, including accident location information sharing and real-time photo transmission from UAV-mounted cameras to traffic management systems. Fig. 1 showcases the envisioned system model, showing the involved entities and an example of the data flow process for exchanging a CAM over ad hoc aerial links.

Such methods are expected to significantly enhance network density, support ultra-reliable low-latency communications, and enable advanced services, such as smart city applications, autonomous driving, and remote sensing [11]. Our research aligns with these forward-looking perspectives, employing UAVs not just as theoretical elements but as practical enablers of robust, adaptable, and far-reaching vehicular communication systems. The primary motivation behind this study stems from the observation that the default congestion and flow control mechanisms of QUIC are not tailored for the high mobility and variable network conditions characteristic of vehicular networks, specifically considering aerial node-enabled vehicular communication relaying, which is both a next-generation use case involving 5G networks where drones will play a pivotal role in ensuring constant overage of connected (vehicular) nodes [12], [13], [14], and a disaster-relief use case where already aerial nodes have seen relaying scenarios. These environments are marked by rapid changes in network topology, variable signal quality, and fluctuating network densities, all of which can severely impact the reliability and efficiency of communication protocols. We aim to enhance the QUIC protocol by integrating adaptive mechanisms that leverage real-time network conditions to dynamically adjust its operational parameters. This approach seeks to maintain optimal flow control and congestion management, thereby ensuring robust and efficient communication even in the face of volatility.

Our QUIC enhancements significantly advance vehicular communications, catering to the stringent demands of 5G networks and URLLC requirements [15]. Tailored for vehicular network dynamics, these modifications ensure robust, low-latency communications essential for autonomous driving and intelligent traffic systems. By integrating advanced flow and congestion control mechanisms, our protocol optimizes data transmission in highly mobile environments, crucial for effective emergency response and vehicular IoT applications. This not only demonstrates the protocol's real-world applicability but also its pivotal role in evolving vehicular networks towards enhanced efficiency and reliability. *a) Our Contribution:* Considering the motivation described above, we introduce significant advancements to the congestion and flow control mechanisms within the QUIC protocol, tailored to the unique and dynamic challenges of vehicular networks. Our contributions extend into the development and implementation of a novel congestion control algorithm, based on a modified version of our previous work (S-HTCP [2], and a pioneering flow control method, "Global-optimum Aware Dynamic Flow Control", which is a unique addition proposed by our present research.

1) A novel Congestion Control Algorithm: The S-HTCP algorithm which we have further enhanced, represents a paradigm shift in the approach to congestion control within QUIC. This algorithm employs an AIMD strategy which considers the time elapsed after the last observed congestion event and the minimum Round-Trip Time (RTT) of a given flow. It has now been further developed to adjust its behavior based on a nuanced analysis of RSSI volatility. This allows for a more granular and adaptive response to network conditions. Our approach significantly

TABLE I
CONTRIBUTIONS OF OUR WORK TO EXISTING RESEARCH

Contribution	Relevant Works	Current work
Congestion Control	S-HTCP (AIMD-based) [2]: 1. Time elapsed since last congestion event 2. Minimum RTT	Enhanced S-HTCP (AIMD-based): 1. Time elapsed since last congestion event 2. Minimum RTT 3. RSSI volatility (adjusting for shadowing-induced spikes/drops)
Flow Control Mechanism	Native QUIC Flow Control Implementation [16]: 1. Stream level data transmission mechanism 2. Stream-level flow control mechanism	Enhanced QUIC Flow Control Implementation: 1. Stream level data transmission mechanism 2. Stream-level flow control mechanism 3. Connection-level global optimum-aware window optimization mechanism 4. Consideration of ACK reception rate for stream window self-modulation
Simulation Medium	NS3 C-V2X Simulator Framework [4]: 1. Mode-4 Communications 2. Native vehicular node mobility models	Utilized NS3 C-V2X Simulator Framework: 1. Mode 4 Communications 2. Native vehicular node mobility models 3. Realistic propagation and shadowing models 4. Custom aerial relay mobility model [17] 5. Vehicular CAMs using actual node parameters 6. Integration with real-world street map data

improves upon traditional methods, providing a robust solution that addresses the rapid changes inherent in vehicular networks. This new congestion control mechanism is meant to replace the default algorithms supported by QUIC.

2) Global-optimum Aware Dynamic Flow Control: Beyond congestion control, we also introduce a novel flow control mechanism designed to maximize the efficiency and fairness of resource allocation among multiple streams within a QUIC connection. This method addresses the common issue of individual streams under-utilizing their allocated flow control and congestion windows due to diminished Acknowledgement (ACK) rates, which can lead to bottlenecks. By considering the global optimum of at a connection level, our approach mitigates these bottlenecks and ensures a more efficient and equitable distribution of resources. This innovation enhances the shared congestion window's utilization.

These contributions collectively represent a comprehensive effort to address the challenges of vehicular network communications through the QUIC protocol, and can be visualized in Table I. By replacing the QUIC congestion control algorithm with our (now further enhanced) S-HTCP, and by improving the flow control mechanism of QUIC, our work lays the groundwork for more reliable, efficient, and fair communication in highly dynamic network environments.

III. BACKGROUND

The QUIC protocol offers several features that enhance its functionality and make it suitable for various communication scenarios. One key feature is its ability to multiplex different streams over a single UDP connection, allowing for concurrent and independent data transmission. QUIC also significantly reduces connection establishment latency, with a best-case scenario of one RTT and the potential for zero-RTT connection establishment (when the client has interacted with the server before), which significantly outperforms traditional TCP-based connections. QUIC ensures the security of data delivery through authenticated and encrypted header and payload. QUIC It offers diverse flow control mechanisms at both the connection and stream levels, allowing the sender to adjust the amount of data transmitted based on the receiver's advertised capacity [7]. This efficient flow control prevents overwhelming the receiver and ensures smooth data transmission. QUIC provides flexible congestion control mechanisms, including the default CUBIC algorithm as well as that of Bottleneck Bandwidth and RTT (BBR) [6]. It also employs a packet pacing mechanism to effectively manage data bursts and handle network traffic. Additionally, QUIC supports connection migration, which is particularly useful in scenarios involving IP address changes. By using a 64-bit connection ID and maintaining the same session key, QUIC ensures seamless connection maintenance and allows migrating clients to maintain authentication and cryptographic verification. These features collectively contribute to the efficiency, security, and adaptability of the QUIC protocol, making it a promising choice for various communication scenarios, including vehicular communications [18]. In this article, we will delve into the fundamental mechanisms and operational principles of the QUIC protocol, with a specific focus on the congestion and flow control control mechanisms. The ultimate goal of this work is to propose a set of channel-aware mechanisms for QUIC, in order to facilitate the secure, timely and reliable exchange of CAMs.

A. QUIC: Congestion Control

Congestion control is a critical component of any transport protocol, including QUIC. It aims to ensure that the amount of data sent by a sender does not exceed the capacity of the network and avoids congestion collapse. Most QUIC implementations employ a variant of the CUBIC congestion control algorithm to regulate the sending rate and adjust the congestion window size. CUBIC is a popular congestion control algorithm that aims to provide a scalable, stable, and fair mechanism for managing congestion in the network. It utilizes a cubic function to estimate the available network capacity and adaptively adjust the sending rate accordingly. However, it's important to note that CUBIC is primarily used for connection-level congestion control in QUIC, rather than stream-level congestion control. There also exist implementations leveraging the BBR algorithm, which probes the network to accurately estimate the available bandwidth and

delay, and then adjust the sending rate accordingly. BBR's main advantage is its capability to fully use bandwidth, despite high packet losses [19].

B. QUIC: Flow Control

In QUIC, each stream within a connection has its own flow control mechanisms. Flow control ensures that a receiver does not get overwhelmed by data from a sender, while congestion control regulates the sending rate to avoid network congestion. While QUIC provides stream-level flow control, which dynamically adjusts the receive window for each stream based on available buffer space, it currently lacks a mechanism for re-allocating the congestion window in case of delayed ACKs at the stream level. In the absence of a re-allocation mechanism, if a stream does not receive timely ACKs for the data it has sent, the neighbouring streams and the overlaying connection may not be able to fully utilize their allocated congestion window. This can lead to under-utilization of network-wide resources and sub-optimal performance, especially for streams experiencing delays in receiving ACKs. This will be discussed in greater detail in Section IV-B.

C. Cooperative Awareness Messages (CAMs)

In our work, we propose the utilization of QUIC as a transport-layer protocol, the payload of which are CAMs. Those messages are essential for facilitating effective communication among vehicles and infrastructure in intelligent transportation systems - they provide crucial information about a vehicle's state, enabling cooperative functionalities (e.g., collision avoidance and traffic management). CAMs are broadcasted periodically, at a given frequency defined by application requirements and network limitations [20]. Currently, CAMs are traditionally transmitted over UDP due to its real-time capabilities and overall suitability for real-time applications. However, UDP lacks any form of reliability mechanisms, necessitating additional error detection and recovery at the application layer; using our modified version of QUIC as a transport-layer protocol enables built-in reliability, congestion control, and security, improving the efficiency and integrity of CAM transmission. This development is in aligned with the goals of O-RAN, promoting seamless and secure communication between vehicles and infrastructure.

IV. Methodology - Enhancing the QUIC Protocol

Our proposed enhancement of the QUIC protocol is twofold. Firstly, we propose the utilization of the S-HTCP congestion control algorithm which we proposed in [2], which has been further enhanced to consider the received singular strength indicator (RSSI) metric. Secondly, we propose a new methodology to allocate the available congestion window of a given QUIC connection's individual streams.

A. QUIC - Congestion Control Enhancement

By default, QUIC uses either CUBIC or BBR as the default congestion control algorithm. CUBIC, which is a heuristic algorithm driven by sender-side events, utilizes real-time scaling metrics instead of RTT-based metrics. BBR considers channel characteristics and parameters to adjust pacing rate and congestion window gain, starting from a given value. Equations (1) [21] concerns the Pacing Rate parameter (rate at which packets are sent), while (2) [21] concerns model the way in which BBR implements channel awareness in practice, considering that at a given time t in a considered period T, we can define BtlBw as seen in (2).

$$\text{Pacing} = \min(\text{Gain} \cdot \text{BtlBw}, \max(\text{Pacing})) \tag{1}$$

$$\text{BtlBw} = \max(\text{deliveryRate}_t), \quad \forall t \in [T - W_b, T]. \tag{2}$$

where:
$Gain$ = Pacing scaling factor, determines pacing rate.
$BtlBw$ = Estimated bottleneck bandwidth.
W_b = Bandwidth filter window.
T = Time period considered by BRR.

Similarly, the congestion window calculation process in CUBIC can be seen in (3) [7], and is a direct function of the K time offset parameter. Said offset parameter represents the time when the cubic curve started. It is calculated as the third root of the product of the maximum window size (W_max) and the complement of the beta parameter ($1 - beta$), divided by the scaling constant (C). The value of K essentially sets a baseline for the congestion window's growth and reduction. It influences how quickly the congestion window increases during the additive increase phase and how aggressively it decreases during the multiplicative decrease phase in response to congestion signals. Equation 4 correspondingly describes this metric.

$$\text{CWND} = C \cdot (T - K)^3 + w_{\max} \tag{3}$$

$$K = \sqrt[3]{\frac{w_{\max} \cdot (1 - \beta)}{C}} \tag{4}$$

where:
C = Scaling constant determining aggressiveness.
T = Time elapsed since the last cwnd reduction.
K = Offset parameter depicting the cubic curve start.
$w_{\max}$ = Window size before the last reduction.
β = Decrease factor of the AIMD algorithm.

1) S-HTCP: Our Congestion Control Algorithm: In our previous work in [2] we proposed a new set metrics for the AIMD algorithm of the H-TCP protocol and managed to achieve an improvement in total throughput and end-to-end delay. D. J. Leith et al in [22] define the α and β factors which we used to formulate our own algorithm variant. Equations (5) and (6) show the baseline of our two previously proposed AIMD metrics. Note that we are assuming that TCP has crossed the threshold for switching from standard TCP operation to the new increase function. Normally, the second component of the product shown in (5) would be expressed as $1 + 10(\Delta - \Delta_L) + (\Delta - \Delta_L/2)^2$, where Δ_L is the aforementioned threshold after which the protocol switches to the operation of interest. In our case, we disregard the previous operational phase and thus replace Δ_L with a null

value. Simplifying the expression yields $\Delta^2/4 + 10\Delta + 1$.

$$\alpha_{S-HTCP} = e^{\Delta/\lambda_1 - RTT_{\min}/\lambda_2} \cdot \left(\frac{\Delta^2}{4} + 10\Delta + 1\right) \quad (5)$$

$$\beta_{S-HTCP} = e^{-\Delta/\lambda_1} \frac{RTT_{\min}}{RTT_{max}} \quad (6)$$

where:
$\Delta =$ Time since the last congestion event.
$RTT_{min} =$ Minimum round trip time.
$RTT_{max} =$ Maximum round trip time.
$\lambda 1 =$ Exponential decay constant 1
$\lambda 2 =$ Exponential decay constant 2

In this case, assuming a successful acknowledgement, the congestion window is defined as $cwnd \leftarrow cwnd + \alpha/cwnd$, while in the case of congestion event $cwnd \leftarrow \beta \cdot cwnd$.

2) Enhancing S-HTCP Using RSSI Volatility Parameters: In the context of this work we have additionally enhanced the S-HTCP AIMD algorithm to consider the volatility of the RSSI value, as well as the direction thereof. In order to capture the behaviour of the links in the same medium through the RSSI, we follow a coherent methodology described below. The methodology has been implemented in NS-3 and can be adopted by public implementations of the QUIC protocol with relative ease and minimal overhead. In addition to considering the time elapsed since the last congestion event and the min/max RTT, we do the following: First we calculate the RSSI difference (ΔRSSI) between consecutive measurements: $\Delta\text{RSSI} = RSSI[n]$ - $RSSI[n-1]$, where $RSSI[n]$ represents the RSSI value at time step n and $RSSI[n-1]$ represents the RSSI value at the previous time step. Secondly, we calculate the time difference (Δt) between consecutive measurements: $\Delta\text{t} = t[n] - t[n-1]$, where $t[n]$ represents the time at time step n and $t[n-1]$ represents the time at the previous time step. Thirdly, we calculate the rate of change of RSSI (RSSI slope) using the derivative formula: $\text{RSSI}_{\text{slope}} = \Delta\text{RSSI}/\Delta\text{t}$, outputting the rate at which the RSSI is changing per unit time. Fourthly, we apply smoothing to the $\text{RSSI}_{\text{slope}}$ to reduce noise or fluctuations. Our approach is to calculate the cumulative average (CA): Smoothed RSSI slope $= CA(RSSI) = \frac{1}{k}\sum_{i=n-k}^{n} \text{RSSI\_slope}[i]$, where k is the number of previous RSSI slope values to be considered in the moving average calculation. The eventually computed metric is utilized by means of sigmoid scaling for the α AIMD parameter, considering inverse scaling for the β expression.

What we have thus achieved is the fact that the protocol can now re-adjust the additive increase rate to be greater for connections with a larger congestion window. By scaling the α parameter accordingly, this approach offers resilience against abrupt RTT changes, similarly with the observed convergence time - which in turn further reduces RTT unfairness between competing flows.

Using this information, we can re-write the expressions in (5) and (6) as follows in (7) and (8), representing the enhanced α and β parameters previously analyzed.

$$\alpha_{enh} = \frac{e^{\Delta/\lambda_1 - RTT_{\min}/\lambda_2} \cdot \left(\frac{\Delta^2}{4} + 10\Delta + 1\right)}{e^{-\lambda_3 \cdot (\frac{1}{k}\sum_{i=n-k}^{n} \text{RSSI\_slope}[i])}} \quad (7)$$

$$\beta_{enh} = \frac{e^{-\Delta/\lambda_1} \frac{RTT_{\min}}{RTT_{\max}}}{e^{\lambda_4 \cdot (\frac{1}{k}\sum_{i=n-k}^{n} \text{RSSI\_slope}[i])}} \quad (8)$$

Experimentally, we found that the relationship betweenλ_1, λ_2, λ_3 and λ_4 yielding the best results in terms of congestion window maximization (considering the elapsed time since the last congestion event) were:$\lambda_1 \approx \lambda_2/3$, $\lambda_3 \approx \lambda_2/6$, and $\lambda_4 \approx \lambda_2/18$. By applying these definitions and setting $\lambda_1 = \lambda$, and $CA(RSSI) = (\frac{1}{k}\sum_{i=n-k}^{n} \text{RSSI\_slope}[i])$ the expressions can be simplified as follows in (9) and (10).

$$\alpha_{enh} = e^{\frac{6\Delta - 2RTT_{\min} + 3\lambda^2 CA(RSSI)}{6\lambda}} \left(\frac{\Delta^2}{4} + 10\Delta + 1\right) \quad (9)$$

$$\beta_{enh} = e^{\frac{-6\Delta - \lambda^2 CA(RSSI)}{6\lambda}} \frac{RTT_{\min}}{RTT_{\max}} \quad (10)$$

where:
$\Delta =$ Time since the last congestion event.
$RTT_{min} =$ Minimum round trip time.
$RTT_{max} =$ Maximum round trip time.
$\lambda =$ Simplified exponential decay constant.
$CA(RSSI) =$ Cumulative average of a connection's RSSI.

The inclusion of RSSI volatility as a parameter in our extended AIMD algorithm provides valuable insights into the stability and quality of wireless connections. By monitoring RSSI fluctuations, our approach can dynamically adjust the congestion control strategy based on the reliability of the wireless link. This adaptability allows a connection's congestion window to respond more effectively to potential network congestion or interference, thereby ensuring uninterrupted data transmission and mitigating performance degradation. Additionally, the previously proposed consideration of the time elapsed since the last congestion event, offers a dynamic perspective on network stability. By incorporating this parameter, the algorithm intelligently adapts the above-described AIMD parameters based on the historical behavior of the network. This enables QUIC's congestion control mechanism to avoid unnecessary CWND reductions during transient congestion periods, resulting in improved network efficiency and faster recovery after congestion events.

In the context of improving the reliability and lowering the latency of autonomous driving and V2X communications, our proposed approach of considering RSSI volatility instead of Signal-to-Noise Ratio (SNR) for congestion window optimization in O-RAN and its integration with C-V2X technology offers significant advantages: reliability and low latency are crucial for C-V2X systems, where real-time and reliable communication between vehicles, networks, and infrastructure is essential. By incorporating RSSI volatility, our approach captures the dynamic changes in the RSSI, providing a comprehensive understanding of the wireless link's stability. Unlike SNR, which

offers a static measure of signal quality using Rx power estimations [23], RSSI volatility reacts swiftly to sudden variations in the link, enabling proactive congestion control measures. By accurately monitoring and considering RSSI volatility, our algorithm facilitates precise congestion window adjustments that reflect real-time network conditions. This enables quicker response times and reduces CAM delays. The ability to make informed congestion control decisions based on nuanced RSSI variations enhances the reliability and responsiveness of autonomous driving systems. To further elaborate on the reasoning behind choosing RSSI over SNR volatility as a metric, focusing on RSSI aligns our work with prevalent practices in real-world cellular and vehicular communication systems, where this metric is commonly used for various operational decisions e.g., adaptive rate control or (cellular) handover mechanisms. This choice is poised to enhance the practical applicability of our findings to (current and future) C-V2X technologies. Additionally, given that RSSI is noise-model agnostic, it provides a valuable measure of link quality volatility without the need for detailed noise level estimation. This approach simplifies the simulation setup while still allowing for accurate modeling of signal propagation and reception under varying conditions. This ensures the broad applicability of our research findings across diverse vehicular and wireless scenarios, not limited by specific characteristics of the noise environment or detailed signal quality assessments.

It is important to note that we differentiate between the effects of fast fading and large-scale fluctuations on RSSI measurements. Fast fading, characterized by rapid, short-term changes in signal strength due to multipath scattering, presents a challenge to maintaining stable communication channels. To mitigate its impact and prevent the algorithm from reacting to these transient variations, we employ sophisticated smoothing techniques on RSSI values as discussed previously which "cancel" out the effects of fast fading. This approach ensures that our congestion control adjustments are based on more stable trends in signal strength, primarily influenced by factors such as shadowing and the mobility of nodes instead of. This distinction enables it to make informed decisions about congestion window adjustments, thereby enhancing overall reliability. Summarizing, the deliberate application of smoothing functions to RSSI measurements allows us to extract meaningful trends from the link quality volatility whilst minimizing the effects of sudden RSSI drops.

B. QUIC - Dynamic Flow Control and Resource Allocation Optimization

QUIC inherently creates several streams per connection. At the connection level, QUIC uses CUBIC or BBR to avoid congestion. At the stream level, QUIC uses a limit-based flow control mechanism. A receiver advertises the limit of total bytes it is prepared to receive on a given stream or for the entire connection. Stream flow control attempts to prevent a single stream from monopolizing the entire receive buffer. It also prevents senders from exceeding the buffer capacity of a receiver - this is done by limiting the total bytes of stream data sent in `STREAM` frames. For each stream, QUIC maintains a sending rate to ensure fair sharing of available bandwidth among streams within a connection. It regulates the sending rate based on feedback received from the receiver, such as ACKs and information about the available buffer space at the receiver. This approach allows QUIC to adapt to varying network conditions, handle congestion effectively, and deliver improved performance for real-time applications, which is of particular importance in low-latency C-V2X environments.

The interconnection between streams' congestion control and flow control mechanisms in QUIC plays a crucial role in ensuring efficient and reliable data transmission. The congestion control mechanism regulates the rate at which data is sent over the connection, while the flow control mechanism manages the amount of data that can be sent on individual streams. These two mechanisms work together to prevent congestion and ensure fair resource allocation within the QUIC protocol. At the stream level, flow control prevents a single stream from monopolizing the receive buffer by limiting the amount of data that can be sent on each stream. The receiver advertises the initial flow control limits for all streams during the handshake, and subsequently sends `MAX_STREAM_DATA` frames to increase the limits. This mechanism allows the receiver to control the rate at which data is received on each stream, ensuring that a stream does not consume more resources than allocated.

At the connection level QUIC maintains a congestion window, which represents the allowed number of packets in flight at any given time, which is shared among all streams. This ensures that the overall transmission rate does not overwhelm the network and prevents congestion. The interaction between congestion control and flow control is evident in the way ACKs affect the allocated proportion of the connection CWND for each stream. When a stream receives acknowledgments, the flow control window is updated to increase the allowed transmission rate for that stream. This allows well-performing streams to utilize more of the available bandwidth. However, if a stream does not receive acknowledgments, its allocated proportion of the connection CWND (supplied to the stream in the form of the flow control window) remains unchanged, ensuring fairness among streams and preventing a poorly performing stream from consuming a larger share of the available resources.

The existing state of flow control in QUIC presents a profound challenge wherein an individual stream, due to a diminished acknowledgment rate, may result in sub-utilization of its allocated flow control and congestion window. This, in turn, creates a bottleneck situation as the remaining streams within the same congestion group currently lack the capability to dynamically update their congestion windows and adjust sending rates in response to the reduced ACK rate of the affected stream. To overcome this limitation, we propose a refined approach to the stream-level flow control resource management system within QUIC.

1) Consideration of Connection-Level Global Optimum: Our proposition involves considering the global optimum of the connection, instead of treating each stream in isolation. By empowering the other streams to adapt their windows and sending rates based on feedback received from the affected stream, we can effectively mitigate the bottleneck caused by

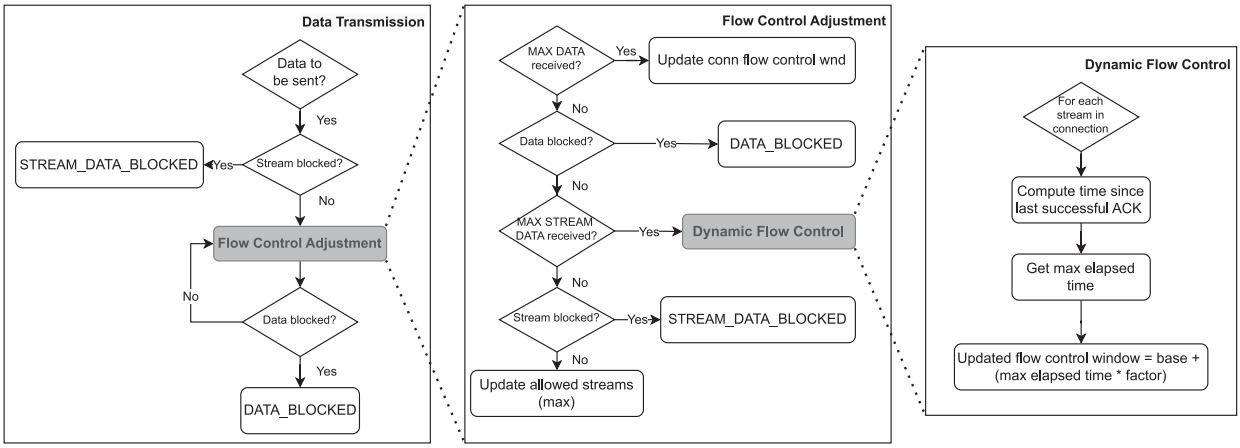

Fig. 2. Overview of the proposed QUIC flow control mechanism enhancement.

Algorithm 1: Data Transmission Function.

Require: Stream data block status
Ensure: Singular stream data transmission
while data to be sent **do**
  if $StreamBlocked(stream)$ **then**
    $STREAM\_DATA\_BLOCKED(frame)$
    Break
  FLOWCONTROLADJUSTMENT()
  if $DataBlocked()$ **then**
    $DATA\_BLOCKED(frame)$
    Break
  FLOWCONTROLADJUSTMENT()
  Send data packets

Algorithm 2: Flow Control Adjustment Function.

Require: Time elapsed since acknowledgement
Ensure: Adjusted flow control window
Upon receiving $MAX\_DATA(frame)$:
$FC\_WND_{conn} \leftarrow frame.maxData$
if $DataBlocked()$ **then** :
  $DATA\_BLOCKED(frame)$
Upon receiving $MAX\_STREAM\_DATA(frame)$:
$FC\_WND[frame.streamID] \leftarrow$ DYNAMICFLOWCONTROL$(frame.streamID)$
if $StreamBlocked(frame.streamID)$ **then** :
  $STREAM\_DATA\_BLOCKED(frame)$
Upon receiving $MAX\_STREAMS(frame)$:
$AllowedStreams_{\max}[frame.streamType] \leftarrow frame.maxStreams$
if StreamBlocked(stream) **then** :
  $STREAMS\_BLOCKED(frame)$

Algorithm 3: Dynamic Flow Control Function.

Require: Flow control window size.
Ensure: Data transmission, flow control management
function DynamicFlowControlstreamID
  for each $streamID$ **in** $Connection$ **do**
    $lastAckTime \leftarrow LastAckTime(streamID)$
    $currentTime \leftarrow CurrentTime()$
    $elapsedTime \leftarrow currentTime - lastAckTime$
  $adjustedWindow \leftarrow MAX\_STREAM\_DATA\_BASE + (elapsedTime_{\max} * STREAM\_DATA\_FACTOR)$
  return $adjustedWindow$

the under-utilization of the allocated window. Implementing this modification necessitates enhancements to the existing flow control mechanisms of QUIC. It entails establishing robust communication and coordination mechanisms among streams to facilitate the exchange of ACK-related information and enable collective decision-making regarding sending window adjustments. By incorporating insights from all streams within the connection, we can achieve a harmonized allocation of network resources and optimize the overall sending rate of the connection. This approach not only mitigates the bottleneck effect induced by a single stream but also maximizes the utilization of the shared congestion window, thereby elevating performance and fairness across all streams within the connection. Currently, a QUIC sender is set to ignore any `MAX_STREAM_DATA` or `MAX_DATA` frames that do not increase flow control limits. [1]. Effectively, a stream has consumed its allocated flow control limit (which is a function of the connection's shared resources, the number and type of concurrent streams, as well as the prioritization thereof), it will be blocked from increasing its sending rate. If a stream (usually of a higher priority) fails to timely receive ACKs, its allocated window will still occupy the shared resources and will reduce connection-wide throughput. Varying ACK rates in QUIC streams will generally result in this connection-wide resource under-utilization.

Considering the above, we are proposing the addition of the `DynamicFlowControl` function in the QUIC flow control mechanism. Algorithms 1, 2 and 3 offer a detailed view of how the proposed enhancement is implemented in practice, while Fig. 2 gives a high-level overview of our

proposed function. The proposed function is to be called within the `FlowControlAdjustment` routine within the same mechanism. The algorithm consists of three main functions: `DataTransmission`, `FlowControlAdjustment`, and `DynamicFlowControl`. These functions work together to enable data transmission and dynamically adjust flow control parameters. The `DataTransmission` function handles the process of sending data packets over the network. It operates in a loop, continuously checking if there is data to be sent. If the flow control window for a specific stream (`StreamBlocked(stream)`) is full, indicating that the receiver cannot accept more data, the function notifies the receiver by sending a `STREAM_DATA_BLOCKED` frame. This action halts the data transmission temporarily, allowing for flow control adjustment. `FlowControlAdjustment` is then called to adjust the flow control parameters based on received frames. Upon receiving a `MAX_DATA` frame, which indicates the maximum flow control window size at the connection level, the algorithm updates the flow control window size (`FC_WND`) accordingly. If the flow control window is still blocked (`DataBlocked()`), meaning that the receiver cannot accommodate more data, a `DATA_BLOCKED` frame is sent to notify the receiver about the limitation. When a `MAX_STREAM_DATA` frame is received, representing the maximum flow control window size for a specific stream, the algorithm updates the flow control window size (`FC_WND`) for that stream using the `DynamicFlowControl` function which calculates the adjusted window size based on the elapsed time since the last acknowledgment received for the streams of the connection at hand. If the flow control window is blocked (`StreamBlocked(frame.streamID)`), indicating that the receiver cannot receive more data for that stream, a `STREAM_DATA_BLOCKED` frame is sent to inform the receiver. Similarly, upon receiving a `MAX_STREAMS` frame, which specifies the maximum number of streams allowed for a given stream type, the algorithm updates the maximum allowed streams ($\texttt{AllowedStreams}_{\max}$) accordingly. If the stream limit is reached (`StreamBlocked(stream)`), indicating that no more streams can be created, a `STREAMS_BLOCKED` frame is sent to notify the receiver. The `DynamicFlowControl` function is responsible for calculating the adjusted window size for stream-level flow control. It iterates over each stream in the connection and determines the elapsed time since the last acknowledgment (`LastAckTime(streamID)`) for each stream. The elapsed time is then used to calculate the adjusted window size by multiplying it with a predefined `STREAM_DATA_FACTOR` and adding it to the base window size (`MAX_STREAM_DATA_BASE`). This dynamic adjustment takes into account the varying network conditions and the receiver's ability to handle data.

V. Evaluation

This section is dedicated to the analysis of the evaluation of our proposed enhancements to the discussed QUIC mechanisms. It entails an analysis of our simulation environment, considered mobility models, the structured of the exchanged benchmark messages, as well as the actual results. A total of two emulation scenarios have been designed, each based on a combination of different mobility models for aerial and terrestrial nodes. Our simulation environment is built around several NS-3 modules as well as additional software such as Simulation of Urban Mobility (SUMO) [24], specialized in modelling vehicular mobility in emulated real-world environments.

TABLE II
Description of Simulation Parameters

Attribute	Value
Mobility model: Vehicular Node	Constant Acceleration, SUMO waypoints
Mobility model: Aerial Node	Anchored self similar Gauss-Markov
Terrestrial simulation grid	2.5 X 2.5 km
Number of aerial nodes	1 - 4
Number of vehicular nodes	2 - 24
Offered load	0.5 - 2.5 Mbps
Relative velocity: Vehicular - aerial nodes	0 - 30 m/s
QUIC congestion control algorithm	CUBIC, BBR, Modified S-HTCP
QUIC flow control mechanism	Native, Dynamic flow control
STREAM_DATA_FACTOR	4 (moderately increased aggressiveness)
Path loss Models	Two Ray Ground Propagation Loss Model
Shadowing Model	Lognormal Shadowing Model, $\sigma = 8$ dB

TABLE III
Exchanged CAM Attributes

CAM Attribute	Description	Variable Type
Vehicle ID	ID of the vehicle derived from the node ID.	string
Simulation time	Time of the simulation in milliseconds.	int
Position (x-coordinate)	X-coordinate of the vehicle's position.	int
Position (y-coordinate)	Y-coordinate of the vehicle's position.	int

A. Simulation Environment

Regarding our utilized NS3 modules, firstly, F. Eckermann et al in [4] introduce an NS-3 based C-V2X simulator which constituted the foundation of our emulation framework.

Secondly, we used the QUIC implementation in NS-3 offered by A. De Basio et al in [16]. The implementation (available in GitHub [25]) is aligned with version 13 of the IETF QUIC drafts and is based on the NS-3 TCP implementation and includes improved acknowledgement mechanism, multiplexing of different streams in a single connection, 0-RTT handshake, possibility for custom stream schedulers, as well as BBR which is of utmost importance for our application.

Thirdly, our simulation environment consist of the O-RAN E2 interface NS-3 module [3]. If our case, it is used to facilitate the exchange of CAMs between the RAN infrastructure and the connected vehicle systems, through aerial relays. Table II offers an overview of the most important simulation parameters.

Fourthly, as already mentioned, for one of the considered scenarios, we have utilized SUMO to model an accurate and realistic vehicular traffic scenario, based on the TAPASCologne [26] scenario.

B. Vehicular CAMs

Each exchanged CAM message is constructed as a string and converted to a packet for transmission. The message format includes the fields described in III. Received CAM messages are logged, along with other relevant information to separate CSV files. The logged information includes the transmitted and

Algorithm 4: GenerateCAMs.

Require: $vehiclesList$
Ensure: $camMessagesList$
procedure GenerateCAMs$vehiclesList$
  $camMessagesList \leftarrow$ empty list
  for each $vehicle$ **in** $vehiclesList$ **do**
    Create a new CAM message object
    $CAM_{vehicleID} \leftarrow id_{vehicle}$
    $CAM_{simulationTime} \leftarrow simTime$
    $CAM_{xPosition} \leftarrow posTx.x$
    $CAM_{yPosition} \leftarrow posTx.y$
    $camMessagesList$.add(CAM)
  return $camMessagesList$

received CAM messages themselves, simulation time and statistics (simulation time, total received rackets and total transmitted packets.

Algorithm 4 provides a high-level overview of how CAMs are generated. Specifically, the algorithm takes a list of vehicles as input and generates a list of CAM messages as output. It processes each vehicle in the input list individually, following a series of steps. Firstly, it creates a new CAM message object. Then, it assigns the Vehicle ID attribute to the ID of the current vehicle. Subsequently, the algorithm sets the Simulation Time attribute to the specified simulation time. Next, it sets the Position (x-coordinate) attribute to the x-coordinate of the current vehicle's position, and the Position (y-coordinate) attribute to the y-coordinate of the current vehicle's position. After completing these attribute assignments, the algorithm adds the generated CAM message to the list of CAM messages. Finally, the algorithm returns the complete list of generated CAM messages.

C. Mobility Models

a) Scenario 1: Linear Vehicular Mobility: For this scenario, we consider that terrestrial vehicles' mobility vectors are lineal and can be described by the constant acceleration mobility model.

Aerial ad hoc nodes are tasked with implementing cellular communication relaying, leveraging the anchored self-similar 3D Gauss-Markov mobility model which we proposed in [17]. This mobility model is geared towards modelling communication-relaying scenarios leveraging aerial nodes and updates the process of calculating the new velocity of nodes by introducing the relative velocity between two directly associated nodes as a weighted and exponentially decaying positional index. The observed outcome is a more fluid and stabilised relative acceleration, which leads to a more position-oriented deployment of the swarm, a key enabler in communications relaying. Equation (11) [17] describes the process of setting the new speed for a node using a randomness index which has been enhanced to accurately model communications-relaying applications. Similarly, (12) [17] models the process of setting the new direction, while (13) [17] mathematically models the process of assigning a new pitch value for a node. The foundation of all three expressions is the Gauss-Markov mobility model which has been modified as we document in detail [17].

$$s_n = ae^{-\frac{1}{j}\sum_{n=1}^{n}|s_{n-1}-s_n|/\lambda_1}s_{(n-1)} + (1-ae^{-\frac{1}{j}\sum_{n=1}^{n}|s_{n-1}-s_n|/\lambda_1})\bar{s} + \sqrt{(1-a^2e^{-\frac{2}{j}\sum_{n=1}^{n}|s_{n-1}-s_n|/\lambda_1})}s_{x_{n-1}} \quad (11)$$

$$d_n = ae^{-\frac{1}{j}\sum_{n=1}^{n}|d_{n-1}-d_n|/\lambda_2(\partial s/\partial t)}s_{(n-1)} + (1-ae^{-\frac{1}{j}\sum_{n=1}^{n}|d_{n-1}-d_n|/\lambda_2(\partial s/\partial t)})\bar{d} + \sqrt{(1-a^2e^{-\frac{2}{j}\sum_{n=1}^{n}|d_{n-1}-d_n|/\lambda_2(\partial s/\partial t)})}d_{x_{n-1}} \quad (12)$$

$$p_n = ae^{-\frac{1}{j}\sum_{n=1}^{n}|p_{n-1}-p_n|/\lambda_3(\partial d/\partial t)}s_{(n-1)} + (1-ae^{-\frac{1}{j}\sum_{n=1}^{n}|p_{n-1}-p_n|/\lambda_3(\partial d/\partial t)})\bar{p} + \sqrt{(1-a^2e^{-\frac{2}{j}\sum_{n=1}^{n}|p_{n-1}-p_n|/\lambda_3(\partial d/\partial t)})}p_{x_{n-1}} \quad (13)$$

where:

s_n	= Node speed for a given iteration.
d_n	= Node direction for a given iteration.
p_n	= Node pitch for a given iteration.
α	= Gaussian Randomness Index.
$\partial s/\partial t$	= Rate of change of node speed with respect to time, assuming constant direction and pitch.
$\partial d/\partial t$	= Rate of change of node direction with respect to time, assuming constant speed and pitch.
$\lambda_{1,2,3}$	= Exponential decay constants.

Equation (14) describes $\vec{V}_{xyz}$, which is the mobility vector in 3D space of a given networked entity, where for the mobility vector component projected in the x axis it is true that $\vec{V}_x = (s_n \cos(d_n)\cos(p_n))\hat{i}$, while regarding the component projected in the y axis we have $\vec{V}_y = (s_n \sin(d_n)\cos(p_n))\hat{j}$, and for the component projected in the z axis we have $\vec{V}_z = (s_n \sin(p_n))\hat{k}$. The applied technique results in spatial "anchoring" in regards to nodes' previous speed and direction respectively. Effectively, this smooths the rate at which direction changes whilst also maintaining the Gaussian randomisation attribute of the overarching positional mechanism.

$$\vec{V}_{xyz} = \sum(\vec{V}_x, \vec{V}_y, \vec{V}_z) \quad (14)$$

b) Scenario 2: TAPASCologne-based Mobility: This scenario constitutes an additional, highly realistic hybrid NS3-SUMO-based scenario, considering actual vehicular mobility as derived from TAPASCologne. It provides a comprehensive simulation environment designed to model a realistic vehicular traffic within Cologne, Germany, across a day. The original data, initially aligned with a proprietary road network, has been meticulously mapped to a network derived from OpenStreetMap. The scenario package includes road networks from OSM, Points of Interest (POIs), polygons, and mapped trips covering early morning to late evening hours. In our application, we focused on a subset of the modelled vehicles, in an area of south-western Cologne, as shown in the SUMO environment in Fig. 3.

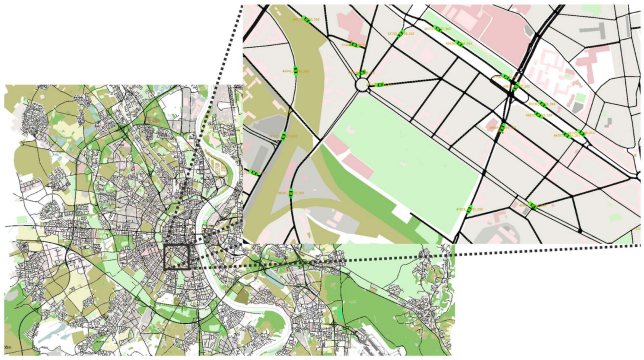

Fig. 3. TAPASCologne SUMO Environment.

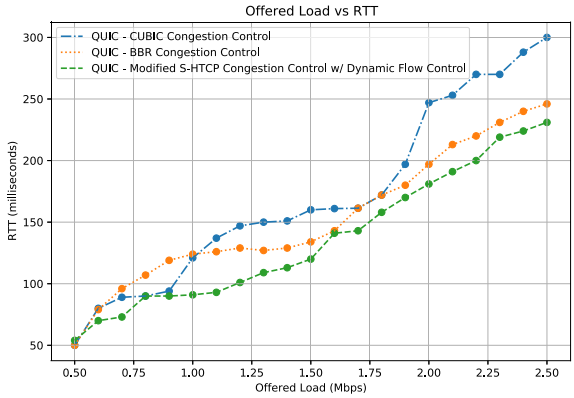

Fig. 4. Achieved RTT as a function of offered load.

This scenario can be considered hybrid in terms of mobility, as it utilizes two sources for the definition of node mobility vectors. Specifically, all aerial nodes are assigned the same anchored self-similar Gauss-Markov-based mobility model, as described in the previous scenario. However the initial locations of the aerial nodes are defined as a function of the outline of the locations given by the vehicular nodes. Regarding the vehicular nodes, their mobility parameters are not given by a mobility model as before, but are rather extracted from TAPASCologne.

To facilitate the integration of SUMO-derived mobility models into NS-3, we employed the NS3 `traceExporter` tool to export SUMO mobility traces into NS-2 trace format, suitable for NS-3 consumption. This conversion is pivotal for harnessing realistic vehicular mobility patterns within network simulations. After running the simulation with the desired parameters, we generate the NS-2 trace file (`simulation.ns2.tr`) encapsulating vehicle movements. Subsequently, within NS-3, the `Ns2MobilityHelper` class reads and applies these mobility patterns to network nodes directly from the `.tr` file. This approach replaces mobility models for vehicular nodes.

D. Results

The conducted experiments considered a use case where mobile vehicular nodes unicasted streams of CAM data to a networked entity, over aerial nodes functioning as mobile relays. The results constitute averaged values of numerous experiments and clearly demonstrate that the proposed enhancements to the QUIC protocol can yield tangible performance increases in a C-V2X scenario. Our end-goal is to practically show that;

1) The proposed enhancement of the S-HTCP congestion control algorithm, considering RSSI volatility on top of the ratio between minimum and maximum RTT and time elapsed since the last congestion event, manages to increase total throughput whilst maintaining substantially lower latency when coupled with native QUIC implementations. This statement should be valid while the relative velocity (as measured between mobile networked entities and their corresponding relays) is increased.
2) The proposed enhancement of the native QUIC flow control mechanism, considering an active connection's streams' time elapsed since the last received ACK will yield increased total throughput per connection as the offered load increases. Again, as the relative velocity (as measured between mobile networked entities and their corresponding relays) is increased, total throughput should be less affected.

In terms of achieved communication quality, we consider the relationship between offered load and RTT, as well as that between offered load and connection throughput.

Fig. 4 visualizes the performance of the benchmarked QUIC variants with the three examined congestion control algorithms (CUBIC, BBR, and Modified S-HTCP, the last of which was also enhanced with the proposed dynamic flow control mechanism) in terms of RTT as the offered load is increased. It becomes evident that the variant of Modified S-HTCP Congestion Control w/ Dynamic Flow Control measurably outperforms the other two QUIC implementations based on CUBIC and BBR respectively. Our proposed mechanisms enable QUIC to achieve lower round trip times as the offered load increases, and does so in a less volatile and more deterministic manner. Regarding the CUBIC-based QUIC variant, it can be observed that despite initially having better performance compared to BBR-based QUIC, its behaviour is more erratic. Regarding the BBR-based QUIC implementation, we observe a behaviour similar to that of our proposed implementation, though with a decreased performance and reduced linearity. This performance increase can mainly be attributed to the proposed S-HTCP-based congestion control algorithm.

The measured RTT values are averages achieved after multiple simulations per scenario, per protocol/algorithm combination. As to the differences in trends observed between the contender protocols, those can be mainly attributed to the impact of congestion control algorithms (varying responses to network condition changes), network dynamics and the impact of relative velocity. Considering the rate at which RTT increases as we adjust the offered load, we can deduct that our proposed protocol and algorithm combination achieves a statistically significant lower end-to-end delay. In practice, RTT is measured by utilizing simple acknowledgements and measuring the total time elapsed for such an event. Lastly, we need to remark that our implementation of the congestion control algorithm utilizes by itself the measured end-to-end delay (both the minimum and maximum values, as well as the ratio thereof) as a metric to

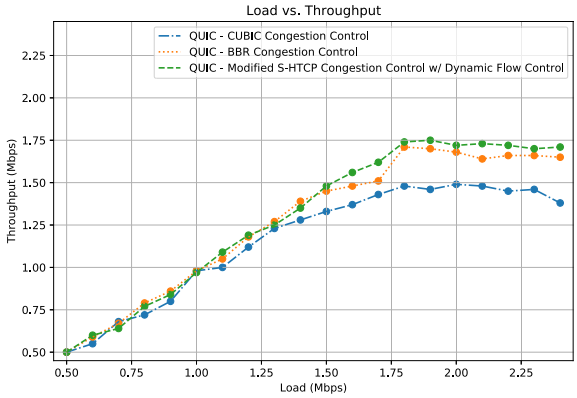

Fig. 5. Achieved throughput as a function of offered load.

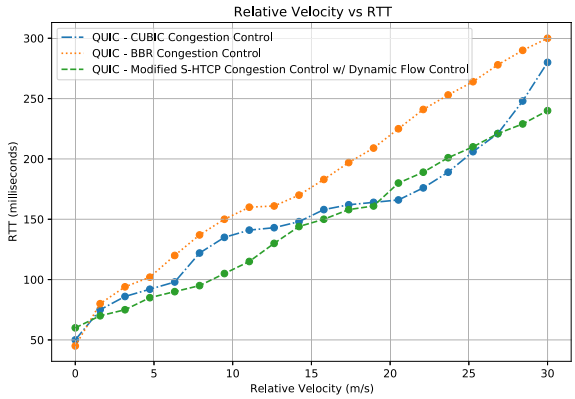

Fig. 6. Achieved RTT as a function of relative node velocity.

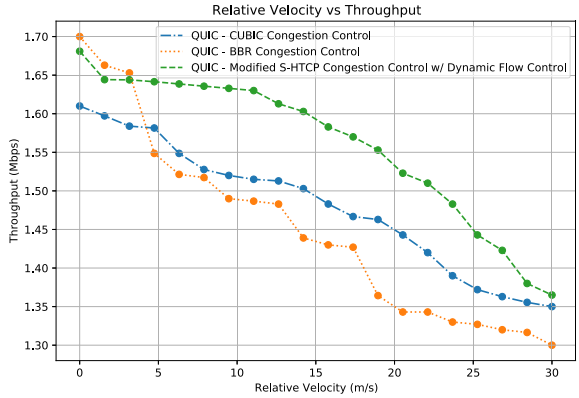

Fig. 7. Achieved throughput as a function of relative node velocity.

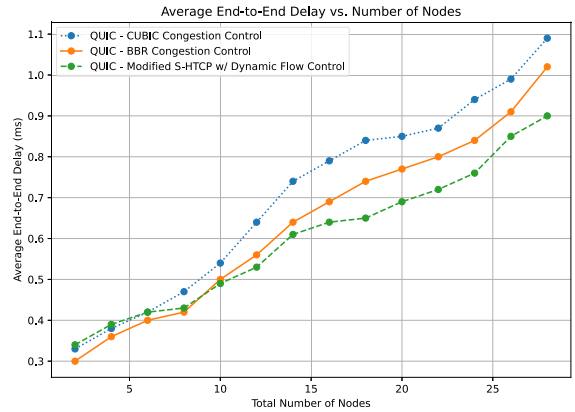

Fig. 8. End-to-end delay as a function of network size.

dynamically adjust sending rates. Therefore, it is expected that when offered greater loads, the variant of the protocol capable of converging faster to the optimal operating point will yield superior results in terms of RTT.

Continuing, Fig. 5 visualizes the performance of the same QUIC variants in terms of throughput as the offered load is increased. Again, our proposed enhancements to the QUIC congestion and flow control mechanisms outperform the native CUBIC- and BBR-based QUIC benchmarks. Specifically, our proposed mechanism enhancements enable QUIC to achieve higher throughput as the offered load increases, which can be attributed to the way in which data streams now implement flow control and calculate their corresponding windows. While minimal, the consistent increase in throughput of our proposed mechanisms indicate that C-V2X applications would greatly benefit from similar dynamic flow-control implementations.

As highlighted in [27], dynamic propagation delay introduced by higher degrees of mobility increases the possibility of packet loss. By default, transport-layer protocols consider all packet loss events as indications that the network is experiencing congestion. Thus, the sending rate accordingly is appropriately reduced. In our evaluation, we thus considered RTT and throughput performance as a function of propagation delay volatility, expressed through relative node velocity. Fig. 6 visualizes the performance of the benchmarked QUIC implementations in terms of RTT as the relative node velocity is increased. Our proposed enhancements demonstrate a comparatively better performance, being surpassed by BBR-based QUIC only for a temporary phase, after which the proposed mechanisms significantly outperform all other implementations. It is worth mentioning that the S-HTCP based QUIC incorporating the proposed Dynamic Flow Control mechanism follows a clearly linear behaviour with a high degree of determinism while the other variants exhibit greater variance coupled with under-performance.

Fig. 7 illustrates the performance of the three QUIC variants in terms of throughput as the relative node velocity is increased. Assuming a completely static network, the BBR-based variant demonstrates increased performance. However, as relative node velocity is increased BBR's performance is shown to greatly deteriorate, and is surpassed by both the CUBIC-based implementation and our own proposed mechanisms. More specifically, the S-HTCP-based variant coupled with Dynamic Flow Control appears to be comparatively less affected by changes in relative node velocity. This also translates to increased resilience of the network against volatile propagation delay which is by definition a common occurrence in vehicular networks in C-V2X environments.

Fig. 8 illustrates the achieved end-to-end delay as a function of the total network size. The second mobility scenario of those described in Section V-C was used to generate these results, involving up to a total of 28 nodes (including the deployed aerial relays). In order to conduct this experiment, each set of (a

maximum of) 6 vehicles was connected to a network of UAVs, and different point-to-point paths were established to benchmark the network. Considering the nature of the deployment (half duplex CAM unicasting) and the negligible propagation delay, the end-to-end delay is generally in acceptable levels (less than 1 ms in almost all cases) even considering repeated relaying and the channel propagation and shadowing models. Our proposed variant seems to outperform the native QUIC implementations in terms of scaling better as the network becomes more congested.

VI. Conclusion

We have considered highly advanced C-V2X scenarios incorporating aerial relays, engaging in realistic mobility using both custom relevant models, as well as actual street map data derived from SUMO, as described in Section V-C. In our scenario, vehicular nodes exchange CAMs which are implemented as custom frames in NS-3, over established QUIC connections. We propose a set of enhancements for the implementation of QUIC, utilizing a novel congestion control mechanism and a global-optimum aware dynamic flow control algorithm. Our evaluation proves that enabling channel-awareness in two core mechanisms of the QUIC protocol yields increased efficiency in C-V2X deployments over aerial base stations.

To further validate our findings, we considered an anchored self-similar Gauss-Markov-based mobility model for aerial nodes, capable of accurately modelling the behaviour of drones engaging in communication relaying. Coupled with dual mobility scenarios for vehicular nodes, our work presents a strong case in terms of realism, both from a physical and a network perspective. Similarly, we consider the impact of relative speed on throughput and RTT by introducing constant acceleration to the corresponding vehicular entities' models, as mobility is a critical consideration for mobile ad hoc networks [27].

Our implementation demonstrates a high degree of resilience against speed volatility, hinting that such improvements can bring about the realization of truly advanced autonomous vehicular services networked. The proposed enhancements are relatively easy to implement in NS-3 and are compatible with existing QUIC implementations, making our proposed work a viable solution for improving the performance of QUIC in transient networks.

Overall, our research paves the way for the next generation of vehicular communication technologies bringing about reliable yet UDP-like fast communications for the control and user planes alike, taking into account future developments in the domain of ad hoc networking and vehicular mobility. By meticulously addressing the intricacies of V2X networks, including the integration of aerial relays and the adoption of sophisticated mobility models, we underscore the potential for QUIC protocol adaptations to significantly elevate network performance. The introduction of our novel congestion control and dynamic flow control algorithms not only showcases a marked improvement in network efficiency but also emphasizes the protocol's adaptability to the fluctuating dynamics and size of vehicular networks and their channel parameters. This adaptability is crucial for supporting the diverse and evolving needs of modern vehicular applications, from autonomous driving to real-time traffic management systems and crisis mitigation endeavours. Furthermore, our approach demonstrates a harmonious balance between theoretical innovation and practical applicability, offering a scalable and forward-thinking solution for future vehicular communications and heralding a new era of mobility and connectivity.

References

[1] J. Iyengar and M. Thomson, "QUIC: A. UDP-Based multiplexed and secure transport," RFC 9000, Internet Engineering Task Force (IETF), RFC Editor, May 2021. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9000.txt

[2] G. Amponis, T. Lagkas, K. Tsiknas, P. Radoglou-Grammatikis, and P. Sarigiannidis, "Introducing a new TCP variant for UAV networks following comparative simulations," *Simul. Modelling Pract. Theory*, vol. 123, 2023, Art. no. 102708, doi: 10.1016/j.simpat.2022.102708.

[3] A. Lacava et al., "Programmable and customized intelligence for traffic steering in 5G networks using open RAN architectures," 2022, *arXiv:2209.14171*.

[4] F. Eckermann, M. Kahlert, and C. Wietfeld, "Performance analysis of C-V2X mode 4 communication introducing an open-source C-V2X simulator," in *Proc. IEEE 90th Veh. Technol. Conf.*, 2019, pp 1–5.

[5] Y. Chen, H. Shi, Q. Weng, and Z. Shi, "Congestion control design of multicast QUIC based on reinforcement learning," in *Proc. IEEE Int. Conf. Ubiquitous Commun.*, 2023, pp. 232–236, doi: 10.1109/Ucom59132.2023.10257593.

[6] H. Kawasaki, K. Ibuka, H. Murakami, and T. Matsumura, "Performance evaluation of congestion control for QUIC in asymmetrical latency environment," in *Proc. IEEE 26th Int. Symp. Wireless Pers. Multimedia Commun.*, 2023, pp. 1–6, doi: 10.1109/WPMC59531.2023.10338958.

[7] S. Lee and D. An, "Enhanced flow control for low latency in QUIC," *Energies*, vol. 15, no. 12, 2022, Art. no.4241, doi: 10.3390/en15124241.

[8] G. Chopra, S. Rani, W. Viriyasitavat, G. Dhiman, A. Kaur, and S. Vimal, "UAV-Assisted partial co-operative NOMA based resource allocation in C2VX and TinyML based use case scenario," *IEEE Internet Things J.*, early access, Jan., 09, 2024, doi: 10.1109/JIOT.2024.3351733.

[9] V. Jain and K. Liu, "Hybrid IoT system for emergency responders," in *Proc. IEEE 11th Int. Conf. Mobile Cloud Comput., Serv., Eng.*, 2023, pp. 59–66, doi: 10.1109/MobileCloud58788.2023.00015.

[10] O. Kavas-Torris, S. Y. Gelbal, M. Ridvan Cantas, B. A. Guvenc, and L. Guvenc, "V2X communication between connected and automated vehicles (CAVs) and unmanned aerial vehicles (UAVs)," *Sensors*, vol. 22, no. 22, 2022, Art. no. 8941, doi: 10.3390/s22228941.

[11] L. Miao, J. J. Virtusio, and K.-L. Hua, "PC5-Based Cellular-V2X evolution and deployment," *Sensors*, vol. 21, no. 3, 2021, Art. no. 843, doi: 10.3390/s21030843.

[12] M. Arif and W. Kim, "Analysis of U-V2X Communications with Non-Clustered and Clustered Jamming in the Presence of Fluctuating UAV Beam Width," *Mathematics*, vol. 11, no. 15, 2023, Art. no. 3434, doi: 10.3390/math11153434.

[13] X. Ma, L. Wang, W. Han, X. Wang, and T. Shang, "A UAV-assisted V2X network architecture with separated data transmission and network control," *China Commun.*, vol. 20, no. 6, pp. 260–276, Jun. 2023, doi: 10.23919/JCC.2023.00.030.

[14] O. Chughtai, N. Nawaz Qadri, Z. Kaleem, and C. Yuen, "Drone-assisted cooperative routing scheme for seamless connectivity in V2X communication," *IEEE Access*, vol. 12, pp. 17369–17381, 2024, doi: 10.1109/ACCESS.2024.3359273.

[15] Annu and P. Rajalakshmi, "Towards 6G V2X sidelink: Survey of resource allocation - mathematical formulations, challenges, and proposed solutions," *IEEE Open J. Veh. Technol.*, vol. 5, pp. 344–383, 2024, doi: 10.1109/OJVT.2024.3368240.

[16] A. De Biasio, F. Chiariotti, M. Polese, A. Zanella, and M. Zorzi, "A QUIC implementation for NS-3," in *Proc. Workshop NS3-3*, 2019, pp. 1–8, doi: 10.1145/3321349.3321351.

[17] G. Amponis et al., "Anchored self-similar 3D gauss-markov mobility model for ad hoc routing scenarios," *IET Netw.*, vol. 12, no. 5, pp. 250–259, 2023, doi: 10.1049/ntw2.12089.

[18] P. Kharat and M. Kulkarni, "Modified QUIC protocol with congestion control for improved network performance," *IET Commun.*, vol. 15, no. 9, pp. 1210–1222, 2021, doi: 10.1049/cmu2.12154.

[19] I. Mahmud, T. Lubna, Y.-J. Song, and Y.-Z. Cho, "Coupled multipath BBR (C-MPBBR): A efficient congestion control algorithm for multipath TCP," *IEEE Access*, vol. 8, pp. 165497–165511, 2020, doi: 10.1109/ACCESS.2020.3022720.

[20] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, ETSI EN 302 637-2 V1.4.1, European Telecommunications Standards Institute, 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.04.01_30/en_30263702v010401v.pdf

[21] H. Zhang, H. Zhu, Y. Xia, L. Zhang, Y. Zhang, and Y. Deng, "Performance analysis of BBR congestion control protocol based on NS3," in *Proc. IEEE 7th Int. Conf. Adv. Cloud Big Data*, 2019, pp. 363–368, doi: 10.1109/CBD.2019.00071.

[22] D. J. Leith, R. N. Shorten, and Y. L. Hamilton, "H-TCP: A framework for congestion control in high-speed and long-distance networks," 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:6989052

[23] T. Jiawei, C. J. Pawase, and K. Chang, "Adaptive sidelink open loop power control optimization strategies for vehicle-to- vehicle communications in 5G-NR-V2X," *IEEE Access*, vol. 12, pp. 25079–25089, 2024, doi: 10.1109/ACCESS.2024.3365133.

[24] DLR, "Simulation of urban mobility (SUMO)" GitHub, 2019. Accessed: May 2, 2024. [Online]. Available: https://github.com/eclipse-sumo/sumo

[25] Signetlabdei, "QUIC implementation for ns-3 - GitHub repository," GitHub, 2019, *GitHub repository*. [Online]. Available: https://github.com/signetlabdei/quic-ns-3

[26] DLR TAPASCologne GitHub, 2019. Accessed: May 2, 2024. [Online]. Available: https://github.com/eclipse-sumo/sumo/blob/main/docs/web/docs/Data/Scenarios/TAPASCologne.md

[27] M. Allman and J. Griner, "TCP behavior in networks with dynamic propagation delay," in *Proc. IEEE Globecom Glob. Telecommun. Conf. Conf. Rec.*, 2000, vol. 2, pp. 1103–1108, doi: 10.1109/GLOCOM.2000.891308.