Title

Adaptive C-V2X Sidelink Communications for Vehicular Applications Beyond Safety Messages

Authors

Ku, Yu-Jen

Flowers, Bryse

Thornton, Samuel

et al.

Publication Date

2022-06-22

DOI

10.1109/vtc2022-spring54318.2022.9860580

Peer reviewed

Adaptive C-V2X Sidelink Communications for Vehicular Applications Beyond Safety Messages

Yu-Jen Ku*, Bryse Flowers*, Samuel Thornton*, Sabur Baidya†, and Sujit Dey*
*Department of Electrical and Computer Engineering, University of California, San Diego
†Department of Computer Science and Engineering, University of Louisville
e-mail: {yuku, bflowers, sjthornt}@ucsd.edu, sabur.baidya@louisville.edu, sdey@ucsd.edu

2022 IEEE 95th Vehicular Technology Conference (VTC2022-Spring) | 978-1-6654-8243-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/VTC2022-Spring54318.2022.9860580

***Abstract*—The current Cellular Vehicle-to-Everything (C-V2X) Sidelink communication protocol provides a low latency interface for sharing short safety messages among Road-Side Units and vehicles. However, while its packets are broadcasted in the channel, the throughput is vulnerable to channel conditions and cannot meet the needs of the emerging connected and autonomous vehicles applications (e.g. vehicular fusion tasks), which require multi-modal and multi-source sensor data sharing. In this work, we establish a C-V2X testbed on the campus of the University of California, San Diego, to study the feasibility of using C-V2X Sidelink communications for transmitting sensor data in real-time. We implement an end-to-end RGB sensor data (i.e. camera image frames) transmission mechanism on the C-V2X Sidelink testbed and explore the corresponding Quality of Service (QoS) characteristics under two configurable link-level parameters, the Modulation and Coding Scheme (MCS) and packet size. We then propose a cross-layer predictive and adaptive framework which adjusts, in real-time, the MCS and packet size settings based on side-channel information to optimize the QoS for image frame transmission. The real-world trace-driven emulation shows that the proposed policy improves the average frame *goodput* performance by 28% compared to fixed configuration policies that are used in current Sidelink communications.**

***Keywords*—** C-V2X, Link Adaptation, Sidelink, Testbed, Vehicular Data Fusion

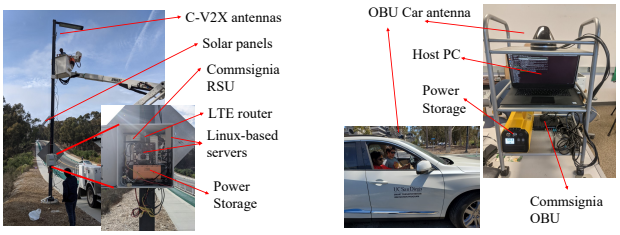
I. Introduction

The emergence of connected and autonomous vehicles (CAVs) include capabilities of multi-sensor data acquisition, processing, and connectivity to support advanced vehicular applications. As a result, vehicles are now smarter and can coordinate with other vehicles, bicyclists, pedestrians, road side sensors, and infrastructures over vehicle-to-everything (V2X) [1] communications. Traditional V2X communications for real-time vehicular applications involve sharing telematics data and passing the Basic Safety Messages (BSM) [2] with other vehicles, or the Road-Side Unit (RSU), over short-range wireless communications. However, in the realm of CAVs, many vehicular fusion applications require sending raw or preprocessed sensor data to the vehicular edge computing (VEC) nodes at one-hop wireless distance for computing different degrees of data fusion. Therefore, these V2X communications can also carry complex multi-modal and multi-source sensor data for vehicular fusion tasks. Some of these fusion tasks demand real-time performance constraints for data-transmission and computing involving complex data-driven algorithms. The V2X communications has evolved from Wi-Fi based vehicular ad hoc network (VANET) to dedicated short range communications (DSRC) [3] in recent past to satisfy real-time requirements over short rage wireless communications. In recent times, Cellular Vehicle-to-Everything (C-V2X) [4] has been developed as a promising next generation technology that provides a better link budget than DSRC by incorporating V2X over cellular communication technology. For real-time applications, it uses PC5 Sidelink [5] interface, thus also obtaining low latency. Like its predecessor V2X technologies, C-V2X Sidelink transmits short messages over broadcast to communicate with other vehicles, RSUs, or infrastructure. However, as C-V2X Sidelink is designed for broadcasting the BSMs or small telematics data, *we need efficient and adaptive protocols that can support the Quality of Service (QoS) of the sensor data transmission for these complex tasks*.

The motivation of sharing and exchanging vehicular sensor information and fusion in real-time arises from the limitations of current Advanced Driver Assistance System (ADAS). While ADAS provides useful benefits to the driver, they are still far from perfect due to utilizing *only* the vehicles' on-board sensors. Although modern vehicles come equipped with advanced sensors (e.g., RGB cameras, radar and lidars), each of these sensors has inherent range and FoV limitations and can also be affected by environmental conditions such as weather and occlusion. However, this problem can be circumvented by sharing the sensor data from multiple vehicles over V2X communications - filling in gaps in each individual vehicle's sensor coverage. This would need C-V2X's continued support of low-latency but also much higher throughput data transmission compared to transmitting BSM which are of low data size ($< 1\,KB$). Vehicular sensor data are of much larger sizes, e.g., RGB images ($1 - 100\,KB$), lidar point-cloud images ($10 - 1000\,MB$). However, the current C-V2X Sidelink protocol has low throughput and is unreliable for uploading fusion data. To explore and mitigate the issue, and enable fusion data transmission over C-V2X Sidelink, in this work, we established a real-world C-V2X testbed. We chose to begin this study with the use of C-V2X for mid-size data load (i.e. RGB images) and implemented a RGB frame transmission mechanism using WAVE Short Message Protocol (WSMP) packets over the C-V2X Sidelink interface. We explored its frame level QoS characteristics under different configurable parameters, i.e. the Modulation and Coding Scheme (MCS) and WSMP packet size. A cross-layer predictive and adaptive framework is then proposed for enhancing the frame level QoS (e.g. frame transmission *goodput*, which is an interplay of frame transmission delay or frame loss rate) of C-V2X Sidelink. Real-world trace-driven emulation result shows that our proposed link adaption policy achieves more than 28% improvement of average frame transmission *goodput* compared to fixed configuration strategies, showcasing its advantages in supporting data-rich real-time vehicular applications.

The research contributions of our paper are the following:

- We study the feasibility of using C-V2X Sidelink for fusion data transmission by establishing a C-V2X testbed and implementing an image frame transmission mechanism over the broadcast channel of the C-V2X Sidelinks.
- We have developed a Deep Learning (DL) based frame-level QoS estimation model, which estimates the current frame transmission delay and reliability based on the vehicle's location, speed, and the MCS and packet size settings of the C-V2X Sidelink.
- We have proposed a link adaption policy from side-channel information, which adjusts in real-time the MCS and packet size



(a) RSU (b) OBU

Fig. 1: C-V2X testbed setup at the UCSD campus.

settings to improve frame transmission *goodput* performance over the C-V2X Sidelink channel without explicit coordination between RSU and vehicles.

In the following of this paper, we will present our C-V2X testbed setup and initial QoS characterization results under different configurable parameters in Section II. In Section III, a mechanism of using the WSMP packets over the C-V2X Sidelink channel to transmit fusion data is introduced. A cross-layer predictive and adaptive framework is then proposed in Section IV and evaluated using real-world trace-driven emulation result in Section V.

II. System Setup

In this work, we aim at real-world demonstration of the feasibility of using C-V2X Sidelink beyond transmitting safety messages. Therefore, we build a C-V2X Sidelink testbed at the University of California, San Diego (UCSD) campus. The external appearance of this testbed is shown in Fig. 1.

A. C-V2X Testbed

This testbed consists of two major parts, the RSU and On-Board Unit (OBU) modules. The RSU module is shown in Fig. 1a, which consists of Commsignia RSU kit [6] and two 8m height C-V2X Urban Antennas. The OBU is shown in Fig. 1b, which consists of Commsignia OBU kit [7] and a V2X OBU car antenna. Both Commsignia kits are powered by Qualcomm C-V2X 9150 radio [8] running 3GPP Release 14 C-V2X standard. In addition, we have also installed a LTE router and Linux-based edge computing servers which collocate with the RSU module to facilitate our future studies on the edge-based data fusion applications. The entire RSU module, including the edge computing servers, are powered solely by solar panels and a portable battery for the purpose of establishing a green communication and computing environment.

B. Parameter Configurations

In this work, we focus on studying the characteristics of C-V2X channels under different configurations of the following two configurable parameters: MCS and packet size.

1) Modulation and Coding Scheme: MCS determines the number of bits that can be transmitted within one resource block. The MCS configuration and corresponding Transport Block Size (TBS) of 3GPP release 14 C-V2X standard is determined based on Table 7.1.7.2.1-1 in 3GPP TS 36.213 document [9]. Higher MCS index allows higher value of TBS, reducing packet transmission time. However, higher MCS index is more vulnerable to channel noise and interference. In this work, we use MCS index up to 17, which is the maximum MCS index available for Commsignia RSU and OBU.

2) Packet Size: Packet size is another critical configurable parameter which determines the performance of C-V2X channels. In this study, we transmit packets following the WSMP, which is a common protocol used in Wireless Access for Vehicular Environment (WAVE) [10]. Based on our observation during exploring the C-V2X testbed, we found that higher packet size is more vulnerable to packet loss when the C-V2X channel condition is degraded. However, lower packet size induces more transmission overhead due to its fixed header size as it requires sending more packets for a given size of fusion data. We will discuss in detail in Section IV.

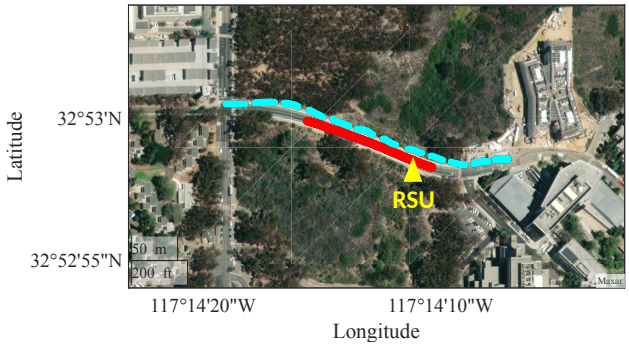


Fig. 2: GPS record of the trace used in emulation

C. Data Collection

The results presented in the following sections in this work are based on real world data that we recorded using our C-V2X testbed on the UCSD campus. We focus on characterizing the following QoS metrics: the packet transmission delay and packet loss rate (as packets are broadcasted over Sidelink with no link-level recovery mechanism). The packet transmission delay is measured by sending WSMP packets between OBU and RSU multiple times and averaging the observed delay. To collect the packet loss rate data, we placed the OBU inside of our research vehicle with the corresponding antenna placed on the top of the vehicle as shown in Fig. 1b. After we begin transmitting WSMP packets from the OBU, the vehicle was then driven along a road segment besides the RSU from one end of the RSU's range to the other. The road segment is shown by the cyan dash line in Fig. 2. The packet loss rate is measured by recording how many packets are successfully received at the RSU. During the data recording, the vehicle is driven at as close to a constant speed ranging from 10 to 40 miles per hour (mph) as possible, while still adhering to the traffic laws and traffics along the route. This drive was repeated two times in each direction on the road for each of the different combinations of MCS and packet size configurations.

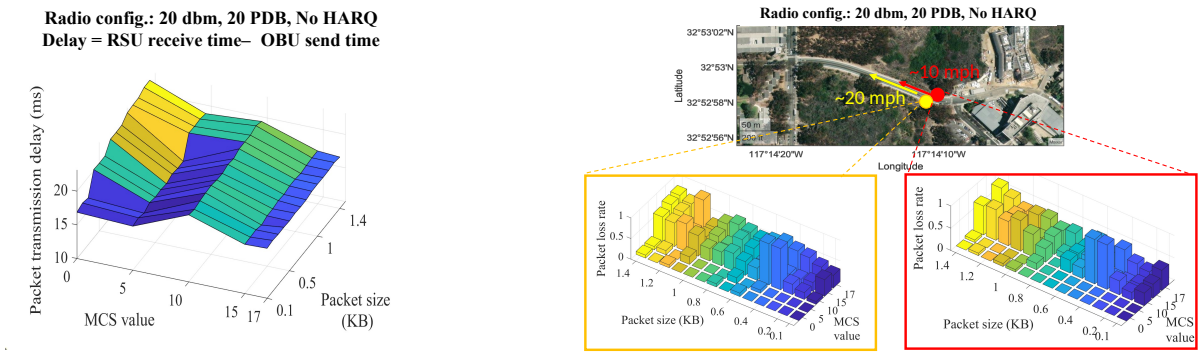
D. C-V2X Characterization: Packet Level QoS

Fig. 3 shows the QoS characteristic of C-V2X with respect to MCS index ranges from 0 to 17, and packet size setting ranges from 0.1 to 1.4 KB. The QoS characteristics studied in this work are packet transmission delay and packet loss rate. Because packets are transmitted by the broadcast mechanism in C-V2X Sidelink, packet loss rate is a critical factor to effective link rate besides packet transmission delay.

Fig. 3a shows transmission delay performance, where higher MCS value leads to lower delay. The observed effect is strongest at large packet size regions. On the other hand, for lower MCS value, delay increases with the packet size. But similarly it is not obvious for high MCS values. On the other hand, Fig. 3b shows the packet loss rate characteristic, where we can see that higher MCS and higher packet size lead to high packet loss rate. But it is worth noting that packet loss rate varies significantly at different locations and speeds. For example, we measure the packet loss rate characteristic at the same location with different speeds, with 10 mph shown by the red dot and 20 mph shown by the yellow dot in Fig. 3b. Although these two dots are very close to each other, the resulting packet loss rate varies with different speeds.

III. Realizing RGB Sensor Sharing Over C-V2X

Although current C-V2X Sidelink protocol is designed for BSMs, the characteristics shown in Fig. 3 present an opportunity for sharing



(a) Packet transmission delay

(b) Packet loss rate

Fig. 3: C-V2X characterization of packet level QoS

data of vehicular fusion applications if we can keep the delay and packet loss rate performance as low and stable as possible. In this work, we use uploading data collected by the RGB sensors, namely the on-board cameras, as an example to demonstrate the feasiblity of using C-V2X channel for fusion data sharing. We plan to send the camera captured image using the WSMP packets. However, the image size is usually larger than the maximum WSMP packet size allowed by the commercial Commsignia's RSU and OBU devices. Therefore, we have implemented a WSMP packet-based end-to-end image transmitting application on our testbed, as shown in Fig. 4.

We first read the binary file of an image as a character array and chunk the array into equal length segments. Each segment is put into the payload of a WSMP packet before being sent out by OBU. The total number of segments and the index of the current segment is saved in the header of each WSMP packet. After the RSU receives the packets, it combines the payloads according to the segment number and indices indicated in the header and saves the resulting array as an image. Therefore, the transmission of an image frame includes the transmission of multiple WSMP packets, with the required number of packets depending on the WSMP packet and image size. In addition, the transmission delay for sending a frame will include the delay for sending multiple WSMP packets for this frame. Although this end-to-end application provides a direct way to send image data using the existing WSMP packets without changing the current C-V2X data transmission mechanism, it is sensitive to packet loss if there is no retransmission control for the lost packets; even losing a single packet from an image frame results in significant corruption to the image.

Therefore, when the camera fusion data is sent over the C-V2X channel, the frame level QoS need to be carefully considered and optimized, which includes frame transmission delay and frame level loss rate. In the following section, we will demonstrate a link adaption policy to maximize a utility metric (e.g. frame *goodput*) by changing the MCS and WSMP packet size configurations of the OBU based on the current estimation of frame transmission delay and loss rate.

IV. Link Adaptation from Side-Channel Info

The primary design constraint for C-V2X is the *broadcast* transmission of basic safety messages. As a result, there is essentially no ability to adapt transmission parameters to current link quality and instead these parameters are often updated in relation to vehicle speed, to combat doppler effects, or geographic location, to comply with local regulations. More traditional communications (i.e. WiFi or cellular) typically provide feedback on channel state that allows the sender to adapt their MCS to greatly increase the spectral efficiency, and therefore link rate, that can be achieved. The question we seek to answer in this section is *how can we add this link adaptation capability to C-V2X without the need to change the underlying protocol to provide feedback?*

There is a growing trend of utilizing the so called *side-channel* or *out-of-band* information for optimizing communications; these are catch-all terms to describe information that is obtained from outside the communications link. The systems employing radios are often sensor-rich and more holistic integration offers optimizations that aren't possible to do on the radio alone. In the current work, we utilize vehicle location and speed - two variables that are highly predictive of channel quality while being easily obtainable on a vehicle. While there can be some generalizable relationships between these parameters and channel quality (e.g. longer distance links have lower Signal-to-Noise Ratio (SNR) due to free space path loss), many important smaller scale effects are site specific (e.g. a static Line of Sight (LoS) blockage from foliage). Thus, we chose to pursue a DL based approach where the methodology can still generalize to any site by simply re-training on a new dataset collected at that location. This methodology is quite simple, and robust, to implement and test, as we show in Section V, however, it is unable to adapt to more dynamic challenges such as network congestion or LoS blockage - we leave these challenges to future work.

While many wireless communications techniques are often mostly agnostic of the traffic they carry, the current work explores a tighter integration between the application, transmitting image frames from vehicle sensors, and the communications. This tight integration allows for an additional degree of freedom in link optimization - *how should we fragment image frames into underlying packets for transmission?* As shown in Fig. 3, smaller packet sizes result in higher reliability, but can potentially induce higher frame transmission delay (as there are more packets to transmit for a given frame and more overhead from headers at various levels); thus there is a clear trade space between lowering frame transmission delay while ensuring the frame is actually able to arrive reliably.

This section first provides the details of how the current work utilizes side-channel information for predicting the packet loss that will be exhibited for specific MCS and packet size actions and then provides our rudimentary capability of estimating frame delay from the actions alone. The section then concludes with a description of how this predictive capability can be included into a policy that performs link adaptation to satisfy arbitrary QoS constraints (e.g. frame reliability, frame latency, etc.) while maximizing an utility metric (e.g. frame *goodput*).

A. Estimating Packet Loss Rate

The goal of this section is to predict the packet loss rate, r, that would be observed in the current state, the location coordinates denoted as l_{lat} and l_{lng} and the velocity vector denoted as v, if the radio is configured to take a specific action, where the MCS is denoted as m and packet size is denoted as p. Put more formally, the current work models the following equation

$$\hat{r} = f(\underbrace{l_{\text{lat}}, l_{\text{lng}}, v}_{\text{“State”}}, \underbrace{m, p}_{\text{“Action”}}; \theta) \tag{1}$$

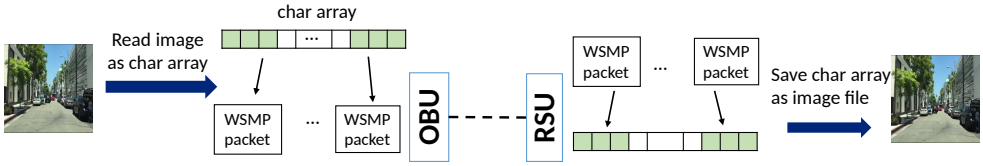


Fig. 4: Implementation of camera data transmission through C-V2X interface using WSMP protocols

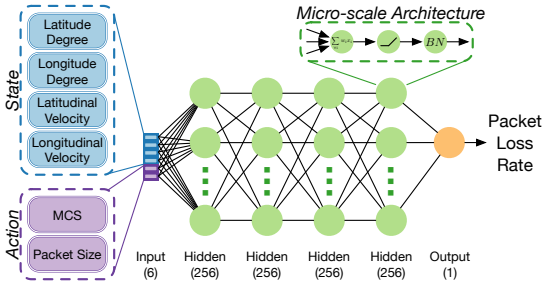


Fig. 5: Deep neural network architecture used for predicting packet loss rate.

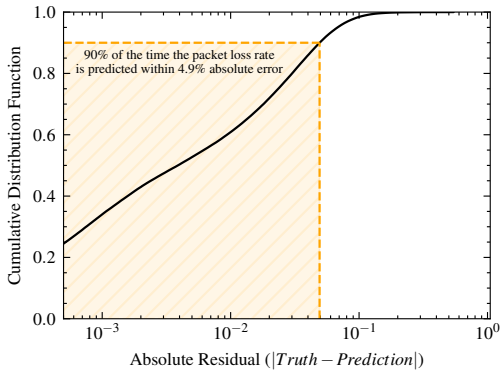


Fig. 6: Performance of the packet loss rate prediction model on the test set.

using a deep neural network where the parameters, θ, are learned through stochastic gradient descent to model the roadway scene of interest. A prediction of the packet loss rate, can be implicitly transformed into a prediction for the frame loss rate through the following expression:

$$\hat{r}_{\text{frame}} = 1 - (1 - \hat{r})^n \tag{2}$$

where n denotes the number of packets needed to transmit the entire frame after fragmentation. Note that (2) implies, as was discussed in Section III, that successfully receiving a frame requires that *all* of the underlying fragments were successfully received to avoid corrupted images.

The specific network architecture used is outlined in Fig. 5. It is a simple feed-forward neural network consisting of five layers with 256 neurons each[1], except for the output layer which contains a single neuron as only one variable is being predicted. All layers, except for the output layer, are followed by the ReLU non-linearity and batch normalization. Each of the inputs are also normalized to zero-mean, unit-variance, as is standard practice. Mean squared error is used as the loss function and Adam as the optimizer. The model was trained for 100 epochs, with a batch size of 512, and a learning rate of $1e{-}3$. The randomly sampled training-test sets were split 80-20, consisting of ≈ 2.6M and ≈ 660k examples respectively.

The model is able to achieve nearly perfect results on the collected dataset, as shown in Fig. 6, and indicated by its achievement of an R^2 value of 0.99 on the test set. Putting this into more intuitively interpretable terms, the model has less than 4.9% absolute error on 90% of the test set.

B. Estimating Frame Delay

End-to-end packet delay in wireless communications is affected by a slew of different variables, which are typically simplified to the following: i) transmission delay, ii) propagation delay, iii) processing delay, and iv) queuing delay. Transmission delay, or the time needed to put the packet onto the link, is modeled entirely (under the assumption of constant bandwidth) by the MCS and packet size used - the *action* variables already utilized in Fig. 5. Propagation delay, or the time needed for the signal to travel from sender to receiver, would be modeled (in LoS conditions) entirely by the distance between the two radios - this can be easily extracted from the *state* variables already utilized in Fig. 5. However, C-V2X only utilizes very short range links making this propagation delay exceptionally small compared with total observed latency; thus, the current work chooses to omit modeling it which greatly simplifies the problem of predicting packet delay. While the current work cannot bisect the final two, processing and queuing delay, from one another it is nearly certain that the queuing delay is the dominant source of delay in C-V2X.

Queuing delay is state dependent; recent packets queued within the radio along with current network congestion can each impact it. To handle the former, the current work chose to instead model the *frame level* delay[2], in which a single image frame is fragmented into multiple WSMP packets (as described in Section III), and the delay is characterized as the time from when the first fragment is sent to when the last fragment is received. As previously mentioned, the latter factor, network congestion, is important, but not modeled in the current work's experimental setup; furthermore, it is not observable given our current decision making variables which we plan to address in future work.

Given the constraints above, the delay can then be estimated from a set of three parameters: MCS, WSMP packet size, and frame size. As each parameter is *discrete valued*, the current work chose to simply create a table of the mean values observed for each parameter combination rather than unnecessarily complicate the prediction problem with a DL model. Put more formally

$$\hat{d}(m, p, s) = \mathbb{E}[D_{m,p,s}] \tag{3}$$

where $D_{m,p,s}$, is a random variable for the frame level delay following an empirical probability distribution, $A_{m,p,s}$. $A_{m,p,s}$ is defined as the empirical distribution of all observed packet delays, d, in our dataset, given that the specific MCS, m, packet size, p, and frame size, s, were used during data collection. Fig. 7 shows an example of the mean frame delay observed for varying MCS and packet sizes when transmitting an image frame size of 6.5KB. As would be expected, as MCS and packet size increases, the observed frame delay is decreased.

[1]A smaller network with three layers and only 64 neurons per layer was also evaluated. As there was, unsurprisingly, a small performance loss and DL models are becoming increasingly computationally cheap due to hardware accelerators, the current work only discusses the larger (but still minuscule compared to state-of-the-art models in other domains) model for brevity.

[2]It was initially surmised that characterizing individual packet level delay and multiplying by the number of packets needed to be sent for any given image frame size would provide a sufficient estimate (and is more generalizable). However, the current work found that this significantly overestimated the frame level delay, leading to pessimistic predictions.

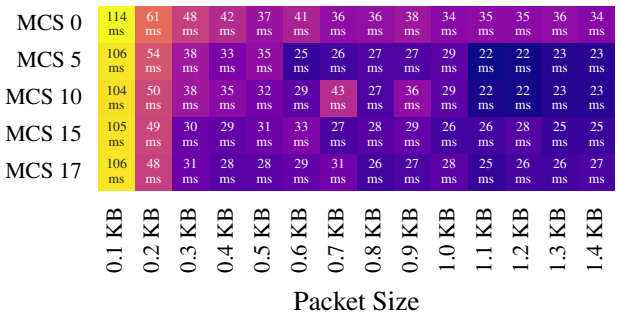


Fig. 7: Example table of the mean frame delay observed for specific MCS values and packet sizes when the image frame size has been fixed to 6.5KB.

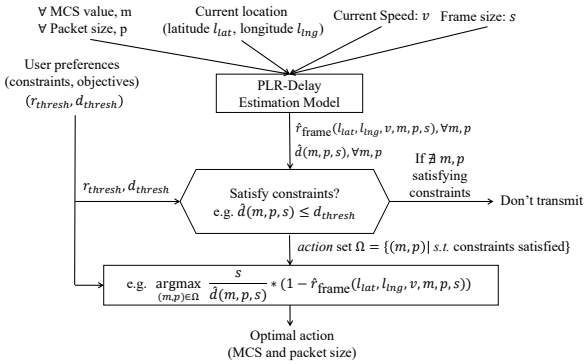


Fig. 8: MCS and packet size adaptation methodology

C. Predictive Link Adaptation for Arbitrary QoS Satisfaction

Given the predicted values of frame delivery rate and frame delay, as were outlined in the prior sections, selecting the optimal MCS and packet size becomes a simple search amongst the action space. The methodology is outlined pictorially in Fig. 8 and textually described in more detail below.

While the methodology presented could be generalized to optimize for other metrics (e.g. minimize frame delay, maximize frame delivery rate, or maximize a weighted utility of both, etc.), the primary criterion used in the current work is the achievable frame rate, or more colloquially, the *goodput*. Additionally, the application, or network operator, is enabled to provide constraints on when the communication should be attempted; a maximum frame delay or minimum frame delivery rate can be specified to the optimization that ensures that any *service outage* does not unnecessarily create congestion in the network (i.e. utilize spectrum resources without satisfactorily completing a link). In short, the link adaptation policy is the solution to the following optimization problem:

$$\begin{aligned} \arg\max_{m,p} \quad & \frac{s}{\hat{d}(m,p,s)}(1 - \hat{r}_{\text{frame}}(l_{\text{lat}}, l_{\text{lng}}, v, m, p, s)) \\ \text{s.t.} \quad & \hat{r}(\cdot) \leq r_{\text{thresh}} \\ & \hat{d}(\cdot) \leq d_{\text{thresh}} \end{aligned} \tag{4}$$

where the estimates for the frame loss rate, $\hat{r}_{\text{frame}}$, and frame delay, $\hat{d}$, are provided by (2) and (3) respectively.

V. Performance Evaluation

To evaluate the proposed link adaptation policy, we have established a real-world trace-driven emulation framework. We then use this framework to compare the *goodput* performance of the proposed link adaptation policy with other static link configuration policies.

A. Real-world Trace-driven Emulation Framework

In the current real-world trace-driven emulation framework, we aim at generating the result frame loss rate, frame transmission delay, and hence the utility metric (e.g. frame *goodput*) for any given instance of vehicle location, speed, and MCS and packet size configurations of the C-V2X link. Note that frame *goodput* is defined as the optimization objective of (4), but with $\hat{r}_{\text{frame}}$ and $\hat{d}$ being substituted by actual observed frame loss rate and transmission delay, respectively, instead of their predicted values.

Without loss of generality, in this evaluation we assume the frame size to be transmitted is 6.5 KB. In one of our previous studies [11], vehicular applications, like object detection, can achieve more than 90% accuracy with this level of RGB image size. To generate frame transmission delay, we transmit a 6.5 KB image from OBU to RSU and record the transmission delay. The experiment is repeated 2000 times for every MCS and packet size *actions* and each frame transmission delay is recorded. During emulation, the ground truth frame transmission delay is generated by picking a delay value uniformly at random among the recorded delays of the experiment which corresponds to the actual MCS and packet size *actions*.

While our packet loss rate estimation model and proposed link adaption policy apply to different vehicle speed conditions, we chose to fix the vehicle speed as close to 10 mph as possible in this emulation framework. For this purpose, we chose the road segment shown by the red line in Fig. 2 as our test drive path as the vehicle speed is more consistent within this segment. The GPS coordinates and speed along this drive path is picked from one of our data collection drives described in Section II, with 10 mph driving speed. The ground truth packet loss rates associated to any given GPS coordinates and speed instance above, for different MCS and packet size *actions*, are derived from the dataset we collected; converting from packet to frame loss rate is done through (2).

Then we use the proposed link adaptation policy to estimate the frame delay and frame loss rate, and pick the MCS and packet size *actions*. The decisions might not necessarily be optimal for each temporal data point due to the estimation deviation. Therefore, we will compare the our policy with the optimal policy which adapts the MCS and packet size *actions* based on the ground truth knowledge. For comparison, we also provide frame *goodput* performance of fixed configuration policies (i.e. the MCS and packet size *actions* will not change for adaptation). These policies are keeping 1) 17 MCS, 1.4 KB packet size, 2) 0 MCS, 1.4KB packet size, and 3) 5 MCS and 0.7 KB packet size.

B. Experiment Result

Fig. 9a shows in temporal domain the result *goodput* of each policies and their corresponding MCS and packet size *actions*, where the purple dash line is the proposed link adaption policy and green line is the optimal result, which is obtained by assuming OBU has the knowledge of actual packet loss rate and transmission delay. Blue, red, and yellow dot lines are for the fixed configuration policies. The x-axis includes the indices of time points of the whole drive on the chosen segment, the interval between each point is 10 ms.

Among the fixed configuration policies, 0 MCS and 1.4KB packet size performs the best and can achieve around 20 fps *goodput* most of the time. However, with link adaption, the proposed policy achieves even higher *goodput* than the fixed configuration policies. As shown in Fig. 9b and 9c, the MCS and packet size *actions* chosen by the proposed policy changes across time. Therefore the corresponding *goodput* is higher than the fixed configuration policies as long as the estimation of frame delay and packet size are precise. Note that the *goodput* of the optimal policy is always higher than the proposed policy because its adaption is based on precise estimation of frame delay and packet loss rate.

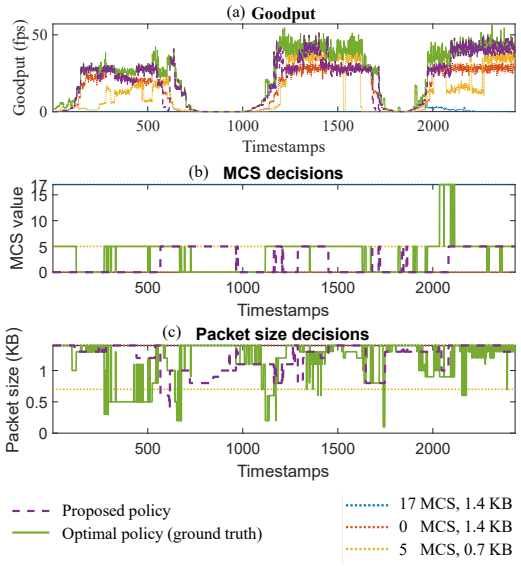


Fig. 9: The temporal emulation result under different link adaption policies of (a) goodput result, (b) MCS actions, and (c) packet size actions

Furthermore, the proposed policy extends the length of period when the link is able to achieve non-zero goodput, which means a few image frames can still be sent out while fixed configurations are not able to achieve the same. For example, among regions between timestamps 600 to 750, 1000 to 1200, and 1850 to 1950, while the *goodput* results of all the fixed policies are effectively zero, the proposed adaptation policy still keeps high *goodput* by adapting to other MCS and packet size *actions*.

Fig. 10 shows the cumulative distribution function (cdf) of *goodput* across all time point for each policy and the corresponding average *goodput*. For example, *the proposed policy outperforms the best fixed configuration policy by 28%* and the cdf also shows that given for any goodput milestone below 30 fps, the proposed policy will always have more time points achieving that milestone than other fixed configuration policies.

Note that there are two dead zones in Fig. 9a, i.e. regions between timestamps 750 to 1000 and 1750 to 1850, which were respectively taken place when the vehicle was at the bottom of the hill and when the vehicle was right underneath the RSU antennas. The link would not support any transmission in these two zones. The performance optimization within these regions is beyond the scope of link adaptation. Further tuning of C-V2X radio interface, or adopting other data transmission policies are required. We will leave this as future work.

VI. Conclusion

In this paper, we have established a C-V2X testbed and demonstrated that C-V2X Sidelink can be used beyond just transmitting BSMs. We have shown an intelligent way to achieve this over broadcast channel without explicit coordination between RSU and OBU. By leveraging the prediction of frame delay and delivery rate, the proposed link adaption methodology does not require feedback information from RSU. Therefore, if the vehicle is capable of measuring its location, speed, and data size, this link adaptation methodology can be implemented on OBU, without any modification to the current protocol, like the fixed configuration policies.

The proposed predictive link adaption strategy dynamically changes MCS and packet size configuration and demonstrates a more than 28% improvement of average *goodput* compared to fixed configuration policies over real-world trace-driven emulations. For future work, we plan to expand current site-specific QoS estimation method to a general

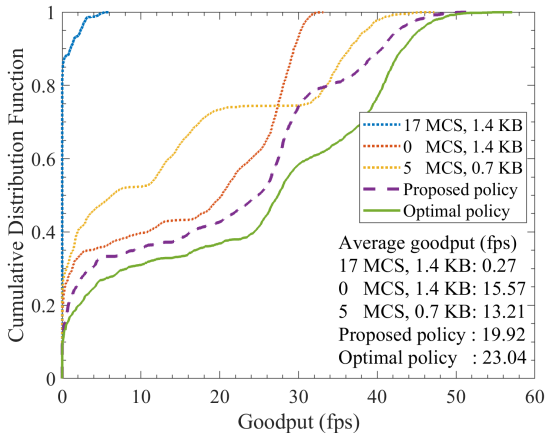


Fig. 10: Cumulative distribution function of *goodput* result using different adaption policies.

model using online and unsupervised learning techniques, so that it can be adapt to more dynamic changes like network congestion and LoS blockage. Additionally, we will study the solutions for the transmission of fusion data to adapt to the dead zones as shown in Fig.9a.

Acknowledgment

This material is partially supported by Qualcomm Inc., the UC San Diego Center for Wireless Communications, and the Smart Transportation Innovation Program (STIP). The authors would also like to thank the staff of Qualcomm Inc. led by Mr. Rashmin Anjaria, Mr. Sean Maschue, Mr, Aasif Dingankar, and Mr. Jim Misener for their help in the technical support of operating the C-V2X devices.

References

[1] E. Uhlemann, "Initial steps toward a cellular vehicle-to-everything standard [connected vehicles]," *IEEE Vehicular Technology Magazine*, vol. 12, no. 1, pp. 14–19, 2017.

[2] M. Kamrani, R. Arvin, and A. J. Khattak, "Extracting useful information from basic safety message data: An empirical study of driving volatility measures and crash frequency at intersections," *Transportation research record*, vol. 2672, no. 38, pp. 290–301, 2018.

[3] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[4] S. Chen, J. Hu, Y. Shi, L. Zhao, and W. Li, "A vision of c-v2x: Technologies, field testing, and challenges with chinese development," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3872–3881, 2020.

[5] R. Molina-Masegosa, J. Gozalvez, and M. Sepulcre, "Configuration of the c-v2x mode 4 sidelink pc5 interface for vehicular communication," in *2018 14th International conference on mobile ad-hoc and sensor networks (MSN)*. IEEE, 2018, pp. 43–48.

[6] Commsignia Inc. , *High performance V2X enabled roadside unit with edge computing*, (Accessed: Feb. 18, 2022). [Online]. Available: https://www.commsignia.com/products/rsu/

[7] Commsignia Inc., *Powerful V2X Onboard Unit*, (Accessed: Feb. 18, 2022). [Online]. Available: https://www.commsignia.com/products/obu/

[8] Qualcomm Inc., *C-V2X 9150*, (Accessed: Feb. 18, 2022). [Online]. Available: https://www.qualcomm.com/products/qualcomm-c-v2x-9150

[9] T. 3rd Generation Partnership Project, "Evolved universal terrestrial radio access (e-utra); physical layer procedures (3gpp ts 36.213 version 14.17.0 release 14)," *3GPP-REF-36213, rel. 14.17.0, 2021. [Online]. Available: http://www.3gpp.org/. Accessed: Feb. 21,*, 2022.

[10] R. A. Uzcategui, A. J. De Sucre, and G. Acosta-Marum, "Wave: A tutorial," *IEEE Communications Magazine*, vol. 47, no. 5, pp. 126–133, 2009.

[11] Y.-J. Ku, S. Baidya, and S. Dey, "Adaptive computation partitioning and offloading in real-time sustainable vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 13 221–13 237, 2021.