A Multi-layered Framework for Informing V2I Deployment Decisions Using Commercial Hardware-in-the-loop Testing of RSUs

Bryson Schiel*, Sam Swindler*, Alek Farmer*, Daniel Sharp*,
Anup Hassan Murali†, Brielle Corry† and Philip Lundrigan*
*Dept. of Electrical and Computer Engineering
Brigham Young University, Provo, UT
Email: {schielb, samswin, afarmer8, dansharp, lundrigan}@byu.edu
†Panasonic - Cirrus, Denver, CO
Email: {anup.murali, brielle.corry}@us.panasonic.com

***Abstract*—Vehicle-to-Everything (V2X) technologies are seeing dramatic growth as smart cities join the Internet of Things (IoT). Overseeing the adoption of this technology, governing bodies such as State Departments of Transportation (DOTs) are starting to contemplate Vehicle-to-Infrastructure (V2I) deployments, and they need to know which V2I devices will perform best in their specific jurisdictions. In this paper, we develop several methods to test commercial, off-the-shelf (COTS) Cellular-V2X (C-V2X) roadside units (RSUs) as a way to inform government deployment. These methods compare performance between different COTS RSUs side-by-side. We present these methods all together as an open source framework which evaluates an RSU multilaterally: at specification compliance, network management, and transmission range layers.**

***Index Terms*—RSU, V2I, Hardware-in-the-loop V2X testing, IoT, Open Source**

I. Introduction

Vehicle-to-Everything (V2X) is a growing technology in the Internet of Things (IoT) with great potential to make cities more efficient and reduce vehicular accidents and associated fatalities [1]. In order to fully take advantage of this technology, enhanced roadside infrastructure needs to be able to interact with smart vehicles for services such as traffic monitoring and guidance. This side of V2X is commonly referred to as Vehicle-to-Infrastructure (V2I). As cars report information to each other and communicate with V2I deployments, the network creates an important ecosystem of safe and autonomous transportation.

V2I deployments are typically run by government organizations, such as a department of transportation (DOT). These organizations will deploy Roadside Equipment (RSE; individual devices are called Roadside Units, or RSUs) next to a road or intersection and connect them to the DOT's larger traffic infrastructure. V2X-enabled vehicles, which are equipped with On-board Units (OBUs), send and receive messages to and from RSUs and other vehicles. This communication between vehicles and infrastructure allows for compelling uses like red light violation warnings, traffic light preemption, and traffic shaping [2].

Unfortunately, V2X technology is still developing, and deployments can require a huge monetary and technology investment. Nevertheless, these infrastructural devices need to be implemented before OBUs could be broadly adopted in vehicles. More than ever, DOTs need to understand V2X and the devices available to them and begin the process of deploying them. To that end, a lot of resources are being put forward to helping DOTs adopt V2X technology and find best uses for it [3].

Ultimately, deployment decisions are an uphill task for these organizations. As DOTs decide between V2I devices, they must be informed about the performance, reliability, and security of the various RSU options available to them. This is complicated as different RSUs may have different ways of being interfaced with and characterized. RSU standardization has helped, but not all devices follow the standards perfectly, and, since this is such a new area of development, many recent features have not yet been standardized or are late to be implemented [4]. While proprietary organizations such as OmniAir [5] can verify specification compliance for various protocols, there is a lack of publicly available tools that measure how compliant devices are with those management standards. Even with specification compliance, it may be unclear how well an RSU may meet the needs of a particular region.

This is where our work innovates. Our goal is to build a methodology that DOTs and other organizations can mimic to run hardware-in-the-loop tests with different COTS RSUs, enabling them to evaluate RSUs for V2I deployment decisions. To this end, we design an automated testing framework for DOTs (or any other stakeholders that are interested in deploying RSUs) to characterize important aspects of RSU performance. Our work takes a holistic approach, focusing on three key requirements of an RSU: specification compliance, network manageability, and transmission range performance (see Figure 1). Rather than relying on theoretical or simulated data, we perform hardware-in-the-loop testing on commercial, off-the-shelf (COTS) RSUs. As our work matures, perhaps it

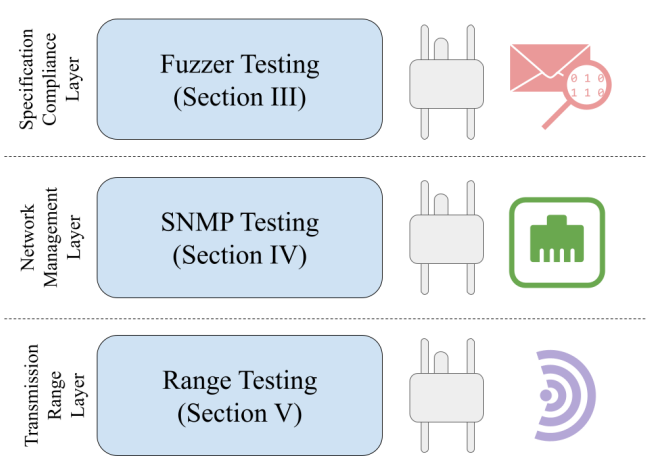

Fig. 1. A visual representation of our framework. We approach the RSU network capabilities from a top-down perspective (relative to the OSI networking model), starting with an application layer attribute in Section III, a network layer attribute in Section IV, and lastly a physical layer attribute in Section V.

can help to inform standards organizations and manufacturers regarding the behavior of future commercial RSU products.

We use our framework to test four commercially available and widely deployed RSUs. Using our over-the-air protocol fuzzer, we identify a message that can be used to perform a denial-of-service attack on V2X devices; we also identify non-standard behavior when an RSU receives an invalid message. Using our networkability testing, we show weaknesses in RSUs that would cause barriers to deploying RSUs at scale. Using our wireless range testing, we characterize RSUs wireless performance through both indoor and outdoor experiments.

The body of this paper explains our framework and how our work may be replicated. In Section III, we test specification compliance by running a library of fuzzed V2X Traveler Information Messages (TIMs) and report on various behaviors showcased by the RSUs. In Section IV, we test network manageability by observing each RSU's response to standard V2X Simple Network Management Protocol (SNMP) queries and commands. Lastly, in Section V, we test transmission range by measuring how well each RSU can wirelessly send and receive messages with different signal strengths. Together, these tests form a single framework that measures an RSU at the specification compliance, network management, and transmission range layers.

Our framework is designed to be expandable so that more tests or RSUs can be integrated into it. We have open sourced our framework as an application for others to use and build on [6]. This is discussed in Section VI with screenshots included.

II. Related Work

As mentioned in the previous section, OmniAir [5] is an organization that certifies compliance of RSU devices. In theory, this provides assurances that the RSU will adhere to certain minimal behavior that is specified by the RSU specification. Information about OmniAir's testing procedure is limited, so it is hard to know what tests they are performing to compare with our framework. While certifying compliance is important, our framework focuses on performance relative to other RSUs. These methods are complementary to each other. Our framework also allows organizations to do their own in-house testing to assure that the RSU performance matches their needs and expectations.

In previous work, Cellular-V2X (C-V2X) device capabilities have been measured using computer simulations [7], [8], and while this provides scalability and performance estimations, simulations do not always reflect realistic environments. In [9], research was performed outside the virtual environment using open-source software-defined radio solutions. However, this is limited by employing non-commercial hardware and is not deployable at scale. Real world testing has been performed on commercial hardware [10], [11], though this testing focused on comparing performance of different V2X high-level protocols and utilized special access to driving courses. While our work also utilizes commercial hardware, it focuses on comparing devices instead of protocols.

Little prior work has been done on evaluating the scalability and management of commercial V2X device deployments. Management of such devices is an important, yet challenging problem. This is made clear by the research done on the management of IoT devices [12]. As more RSUs get deployed, management and maintenance of those devices will become a huge problem. Characterizing the management performance of RSUs will make a significant impact on the maintainability and scalability of V2X deployments.

Using our protocol fuzzer (Section III), we identify a data frame that can cause an OBU to stop functioning until it is rebooted and multiple RSUs to go offline for 30 seconds. While such a packet and fuzzing capability is novel, others have found exploits in V2X communication. For example, denial-of-service (DoS) vulnerabilities in the V2X ecosystem have been previously demonstrated. [13] presents a vulnerability on the cellular protocol used by V2X devices, and [14] simulates an exploitation in message prioritization that causes network delays. Although these and other works raise legitimate concerns, their focus has been on design flaws in the protocol itself or limitations in infrastructure resources. Our work contributes to this discussion by demonstrating vulnerabilities in device-specific software implementations and presenting a framework to discover further vulnerabilities.

III. Specification Compliance: Fuzzer Testing

We test the specification compliance of RSE to the V2X communication protocol, specifically the messages contained in the SAE J2735 V2X Communications Message Set Dictionary [15]. Knowing that an RSU is unable to receive or transmit certain V2X messages would be valuable to a DOT or other prospective buyer, since any limitations will severely affect the available use cases for that device. Additionally, knowing a device's resilience to unusual messages produced

by error-prone or malicious actors is valuable in evaluating the road-worthiness of such a device. Evaluating both the device's compatibility and resilience help stakeholders to make an informed decision.

To do this, we create an over-the-air V2X protocol fuzzer. This fuzzer is first of its kind in exploring RSU's ability to transmit and receive a large variety of V2X messages. We evaluate compatibility by transmitting several types of valid messages with a large variation in possible values stored in each message field. We evaluate resilience by transmitting a selection of invalid or unusual messages, verifying that devices accept valid messages, reject invalid ones, and produce no unusual behaviors. In the following sections we present the methodology used to perform testing and report our results.

A. Methodology

In our work, we take the principles of fuzzer testing and apply them to the V2X interfaces presented by each RSU. Fuzzing is an approach to finding design flaws in computer systems [16] or network protocols [17], [18]. We generate randomized V2X messages, transmit them to our RSUs, and record results by observing the packet forwarding behavior of our RSUs. Initially, we focus on transmitting TIM messages, but this can be expanded in the future. Each RSU has a programmable packet forwarding protocol, where it automatically transmits received V2X packets over an Ethernet connection. These packets are forwarded to a single host machine. Using Wireshark, a packet-level network monitoring program, we identify which RSU is forwarding a packet that it has received. This setup is represented in Figure 2. Devices fail a test if they forward these packets to our machine at a rate below 90% of the original transmission rate, or if they make significant changes to valid message bodies. Additionally, a device fails a test if it forwards packets that are non-compliant with J2735.

In order to perform this kind of testing, we need tools capable of generating messages with specific values, encoding them into hexadecimal, transmitting them from one RSU to another, and comparing them against original messages for discrepancies.

Using the "asn1tools" library [19] and the ASN.1 files distributed by the SAE [20], we create a Python codec that can encode human-readable representations of V2X messages into their corresponding hexadecimal value, and vice versa; this codec is included as a submodule to our application. While we encounter some encoding issues caused by limitations in the underlying "asn1tools" library, these concerns are minor enough that we can proceed. We then expand our codec, making it easy to create large quantities of valid V2X messages. This expansion includes generators that produce messages with randomized content, messages with specifically overwritten values, and invalid messages whose value exceeds bounds defined by J2735.

Finally, we create a tool to automatically transmit messages and record results. For each test, a controlling machine assigns a message to be transmitted at a rate of 10 messages per second by one of our RSUs. The remaining RSUs, having been configured to forward incoming messages back to the controlling machine over UDP, receive the message. Figure 2 depicts the wired and wireless connections used between our devices. The controlling machine makes a 15 second observation of forwarded messages, performing two calculations to determine if a message is being correctly received (and is thus compliant with the protocol). First, the controlling machine calculates a packet reception rate (PRR). If this PRR is below 90%, the message is flagged for further review. Second, the controlling machine compares the packet content of the forwarded messages against the content of the message that should have been transmitted, flagging any major discrepancies between the two. We discover that RSUs can change the contents of transmitted messages when they are received. We take messages that cause problems on the RSUs and transmit them to an OBU to see how the OBU is impacted.

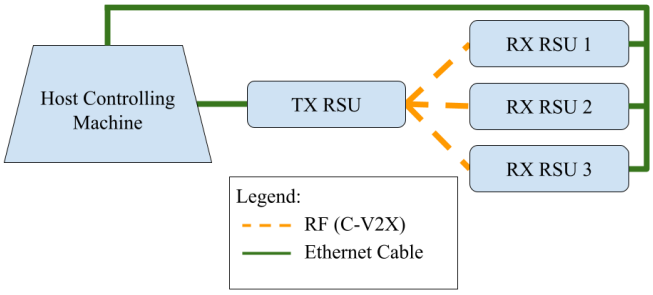

Fig. 2. A representation of the devices and connections in our compliance testing setup.

Using these tools to create messages and transmit them in an automated way, we analyze the results of our experiments and discover that some messages cause malfunctions when received by both RSUs and OBUs.

B. Analysis

Over the course of our testing, we observe two notable behaviors from our commercial device. First, we discover an implementation bug in two RSUs and one OBU that *creates a Denial of Service vulnerability*. Second, we discover that receiving RSUs fail to check message contents for several out-of-bounds field values.

1) Denial of Service Vulnerability: Using our fuzzing framework, we find a packet that disrupts the availability of V2X services on our OBU and two of our RSUs. TIMs are messages commonly used to notify drivers of situations such as adverse road conditions. Contained within every TIM is a description of the physical geographic space that an advisory is applicable to. There are many ways to describe this space, such as using circular or rectangular regions. One of the ways this is done is by describing the region relative to a previously specified traffic lane. This method, called a `ComputedLane`, is what our particular packet contained.

When an OBU receives a TIM, it examines its contents, determining if the vehicle is physically located in the region being described by that TIM. If this is the case, the OBU

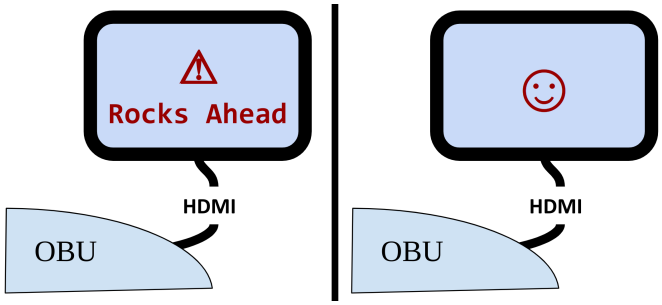

Fig. 3. On the left, under normal conditions, an OBU that receives a TIM from a deployed RSU will display the TIM message via HDMI (presumably to a vehicle's dashboard). On the right, when an adversarial RSU is transmitting a TIM with a *ComputedLane*, the message disappears from the dashboard, potentially depriving the driver of a critical warning.

HDMI will produce some kind of output. Our OBU can be attached to a screen and display applicable TIMs as messages for an end-user to read. In our lab testing we find that when we transmit messages containing a `ComputedLane`, our OBU stops displaying any TIMs it is receiving and also stops transmitting it s standard Basic Safety Messages (BSMs). The OBU recovers only after being power cycled. See Figure 3 for a visual of this phenomenon.

While previous work has demonstrated denial of service attacks within the V2X protocol [13], [14], our message demonstrates a denial-of-service attack found in the implementation of the commercial hardware itself. We achieve this by manipulating the content of a V2X message into a state apparently overlooked by application developers, but still within the bounds established by J2735. These results show that a message can be crafted either by accident or on purpose that disables RSUs for an extended time, and OBUs for an entire power cycle. This brings to light a major safety and security risk while also highlighting the importance of the fuzzing testing framework that we have developed.

We verify our results in an outdoor environment to ensure that nothing indoors is the true cause of this behavior. Using two different RSUs, one RSU transmits two TIMs which cover distinct regions. When our OBU is moved into either of the regions, it displays the corresponding advisory message on an output screen. The other RSU then transmits a message containing a TIM containing a `ComputedLane`. When this happens, our OBU stops displaying the advisory messages, even when moved into either region. Normal operating behavior only resumes after the OBU is powered cycled, if it leaves the space that the `ComputedLane` TIM describes. We also find that the other RSU ceases operation once it receives that TIM, and it only resumes operations after the TIM is no longer being transmitted.

While these findings are probably the result of an easily repairable programming oversight, it demonstrates that our framework is capable of discovering issues that are present only in real-world implementations of the protocol.

2) RSU Transmission Validation: A second specification compliance issue was identified by our fuzzer. The SAE J2735 specification assigns fixed bit lengths to many fields in a V2X message. When a value exceeds this length, our transmitting RSU fails to invalidate the contents of the message, proceeding to transmit them anyway. Receiver RSUs also fail to invalidate the contents of their message, freely forwarding their contents to the controller machine. This behavior of freely transmitting, receiving, and forwarding messages not in compliance with J2735 and calls into question whether RSUs should validate the contents of TIMs as part of their operations. While §2.5.2.9.2 of [21] states that RSUs should be capable of doing on-device processing of BSMs, it makes no requirements regarding the pre-processing of TIMs. If this behavior were to remain, the system would rely on the higher-layer applications that RSUs interact with to properly handle TIMs with invalid bounds. Doing so leaves the possibility for errors in the design of that software to produce potentially hazardous situations.

For example, if a DOT employee were to attempt to transmit a TIM but erroneously encode a bad value, the RSU would not contribute toward catching the error in the message. This could lead to life-saving advisories being disregarded by OBUs because of the bad value. Therefore, we believe this behavior represents an issue that needs to be resolved. Every device in the communication chain should contribute to the integrity of that communication, thus producing a more robust and modular system.

As has been demonstrated, validating compatibility with the complete J2735 message set is an important metric for evaluating COTS RSUs. Any omissions in compatibility would represent a serious reduction in available use cases for devices, so discovering them early could prevent unnecessary costs. These preliminary results demonstrate the usefulness of the fuzzing technique in finding implementation issues and validating device compatibility with the SAE J2735 Message Set. While the focus of our work has been on fuzzing TIM messages, there are many other messages that need to be fuzzed. Our framework provides the flexibility to do just that.

IV. Network Manageability: SNMP Testing

Following our fuzzer analysis, we examine the design of our RSE in terms of network manageability. RSUs are intended to be used in deployments that can be very large. Such deployments are most effective when any individual device requires little manual effort in order to maintain. Minimized management at an individual RSU level makes an entire deployment more scalable and adaptable to system-wide change.

While we do not have access to a large deployment of devices to perform scalability experiments on, each of the devices we have available is a unique model from a different vendor. We can evaluate the effort required to manage a single device and then make conclusions about the effort required to manage hundreds of that same device. In the following subsections, we present the methodology used to assess these devices and analyze the results, rating the networkability of these devices. A DOT could take this and determine the scalability that a network of these devices would afford in a deployment.

Topic	Reasoning	RSU 1	RSU 2	RSU 3	RSU 4
1. Message forwarding	Major functionality of RSUs to forward messages to the traffic management system based on different applications.	✗	✓	✓	∼
2. Read manufacturer codes	A V2X network may be made up of many different types of RSUs.	∼	✓	✗	∼
3. Read serial number	Used to uniquely identify each RSU.	∼	✓	✗	∼
4. Read GPS coordinates	Used in visualizing a V2X network.	∼	✓	✓	∼
5. Read operational status	Used to track RSU health and can alert a DOT when an outage occurs.	∼	✓	✓	∼
6. Read firmware version	Used to ensure RSU firmware is up to date.	✗	✓	✓	∼
7. Read CPU statistics	Used in debugging poor RSU performance and managing RSU health.	∼	✓	✓	∼
8. Read memory statistics	Used in debugging poor RSU performance and managing RSU health.	∼	✓	✓	∼

TABLE I: Evaluation metrics and results from networkability testing. ✓means pass, ∼ means partial pass, and ✗ means failure.

A. Methodology

We evaluate RSUs by examining their networkability. We define networkability to mean that the main functions of the RSU can be managed remotely and automatically. To provide quantitative results of our analysis we use the RSU standard [21] to define key performance functions. We then perform these functions on four production RSUs, rating them on their level of networkability. Table I lists the evaluation metrics we use to evaluate these devices as well as the results of our trials.

To measure networkability, we test each RSU's responsiveness to Simple Network Management Protocol (SNMP) commands. SNMP is an industry standard protocol to monitor and manage networks of all types, verifying device health and functionality. It is also required in RSUs. In our framework, a Python script runs an SNMP command for each evaluation metric and records the message that is returned from the RSU. If an RSU returns an appropriate SNMP response, we say that that it passes that evaluation metric. If it fails to do so, we examine its operation more closely to see if there is a non-SNMP way of performing the same operation. Sometimes there is, and sometimes there is not, and this has a great impact on the networkability of an RSU and the scalability of a deployment of that RSU.

B. Analysis

We run networkability tests on the same 3 RSUs from Sections III and V as well as an additional RSU we had access to at this stage of testing (labeled RSU 4). We indicate a passing result when that device's feature is functioning according to the RSU SNMP standard [21]. A partial result indicates that the functionality is available, but with manufacturer-specific steps (for example, some of an RSU's data might be retrieved through SSH instead). A failing result indicates the device provides the functionality through a proprietary interface, manually through a GUI, or does not provide the functionality at all. Table I presents the results of our findings along with our rankings for the individual RSUs based on our SNMP tests.

Overall, these devices provide a serviceable level of network management. Much of the reduced scores are a result of manufacturer-specific variations. We suggest that these differences would be most impactful in heterogeneous manufacturer deployments. However, as long as a deployment relies on a single device manufacturer, adaptations could be made to create a functioning system.

Our results contribute a snapshot view of the networkability of real V2X infrastructural hardware. Networkability is an often over looked indicator of the readiness of a technology despite its profound ramifications on scalability and management. Having it as a part of our testing framework adds to the overall value of the testing suite and how informed a user of the suite would be when evaluating potential RSE for massive deployments.

V. Transmission Range: Testing Signal Responsiveness

This last section of our framework describes tests that measure PHY layer capabilities of the devices: specifically, their communication range. This section is divided into two sets of testing: subsections V-A and V-B refer to tests that are run in an indoor environment; and subsections V-C and V-D refer to tests that are run in an outdoor environment.

We again use an OBU in these test, in this case as a form of C-V2X message traffic generation. It transmits ten BSMs every second, providing a steady and predictable stream of messages to parse in each trial.

A. Indoor Testing Methodology

In order to create replicable range tests in an indoor environment, we need to be able to manually control the physical link between each of the devices. The indoor experiments are thus performed using RF coaxial cables to connect the RSUs and the OBU, via a mesh attenuator [22], which manages the attenuation on each connection. This allows us to

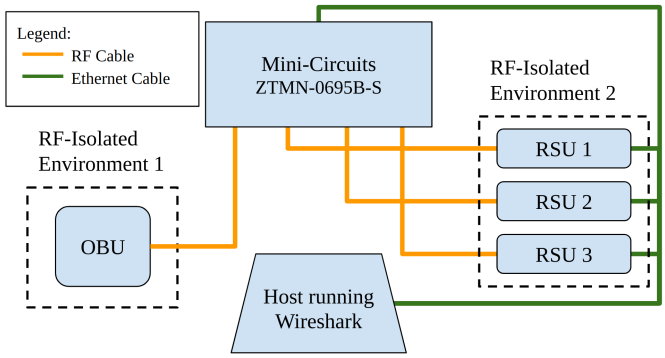

Fig. 4. A representation of the devices and connections in our indoor setup.

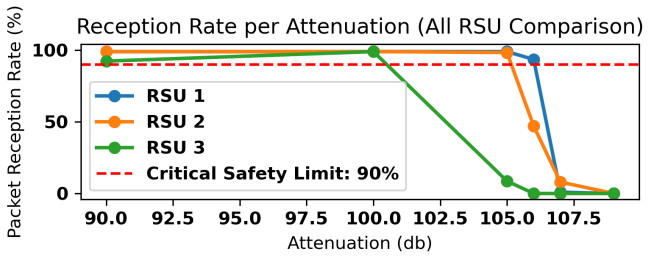

Fig. 5. The results from running all three RSUs against each other over multiple trials. Each dot represents the RSU's performance during a given trial, when a specific attenuation was applied for 15 minutes. Note the "knees" where the reception rates drop below the critical safety limit.

control the signal strength between transmitter and receivers by manually changing the attenuation in each trial. We shield our devices from radio frequency (RF) leakage by placing our transmitter and the receivers in separate RF-isolated cages. This successfully isolates them from each other and forces all communication to be conducted on our manually adjustable cable connection. Figure 4 shows the entire indoor setup.

As the OBU transmits the BSMs, on a low enough attenuation, the RSUs can capture and record them. As in Section III, we record packet reception rate (PRR) by forwarding these messages to an external computer. When this host machine stops receiving forwarded packets, we determine that the RSU has started dropping BSMs from the OBU due to too-high attenuation.

In the experiment, a trial is run for 15 minutes with a constant, identical attenuation value between the OBU and each RSU. During the trial, we count how many packets are forwarded by each RSU during that time and divide it by the number expected to get the PRR. The OBU transmits ten V2X messages every second, so we expect around 9000 received messages from each RSU during the trial. These PRRs are what we compare between RSUs to measure performance. We then modify the attenuation value and repeat for the next trial. This process continues until all RSUs reach a low PRR (usually 0%).

B. Indoor Testing Analysis

After all the trials have been run, we see at what attenuation value each RSU starts to lose reception and eventually lose it entirely. We set a PRR of 90% as the safety minimum for each device and identify the "knee" of the graph as the point where the PRR drops below that safety limit. Figure 5 shows the results from one such experiment. Looking at the knee locations in this trial, RSUs 1 and 2 both outperform RSU 3.

The key to all of these range tests is not the determination of a given RSU's performance under certain specific conditions, but rather an RSU's performance compared to that of a competing RSU when in identical (or at least very similar) conditions. In the results (Figure 5), it is not as important that RSU 1 can achieve a high PRR despite the highly attenuated signal as it is that RSU 1 outperforms RSUs 2 and 3 in identical conditions.

While these results are clearly beneficial, there remains the question as to whether or not indoor tests can accurately reflect RSU performance in an outdoor setting. To answer this question, we take our tests outside.

C. Outdoor Testing Methodology

For the outdoor experiments, we use a road that is typical of where deployments for these devices would take place. We station the RSUs at one end of a road and then drive a vehicle away from them with the OBU transmitting V2X messages ten times a second. This differs from our indoor test in that we cannot accurately measure the signal attenuation over time as the vehicle drives away. We are, however, able to record the vehicle's progress through time and determine at what times it reaches successive intersections, as shown in see Figure 6. This figure shows the map of our route along with a label for each intersection crossed. Using this route allows us to use a changing distance as an intermediary for attenuation of a signal. We use a camera to capture the time stamp of crossing intersections while driving. We use Wireshark to cross reference packet timestamps with how far away the vehicle has gotten and generate an RSU's PRR for each intersection. We can then show how the vehicle's distance (and the associated signal attenuation) affects each RSU's PRR.

D. Outdoor Testing Analysis

As an OBU-mounted vehicle is driven away from the RSUs, we collect the forwarded packets that each RSU is able to receive. Again, definitive signal strength cannot be determined at this time, but the important detail is that each RSU has equal access to the signal provided by the OBU. This comparative performance test thus reflects well the setting for the indoor testing environment, as described in Section V-A.

We perform multiple trials of the test as outlined. At the end of this testing, we can inspect each RSU's comparative performance based on where the OBU is located on the route. Figure 7 shows how well each RSU performs on average based on the the distance it is from the OBU. The results are grouped by the intersection the car was in when the packets were captured, and box-plots show how well each RSU did

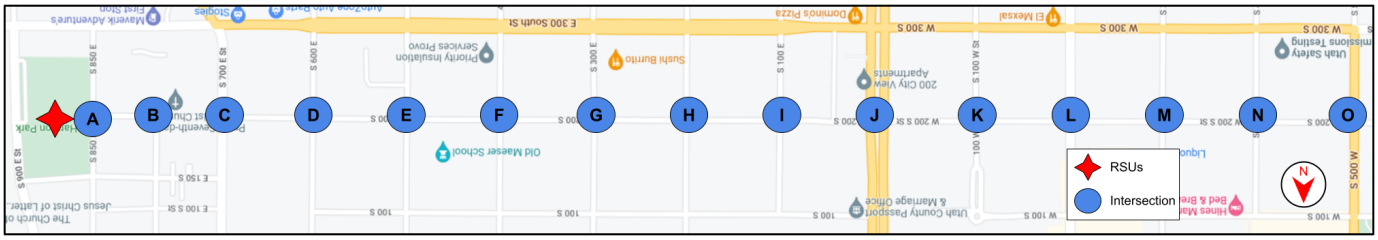

Fig. 6. The route taken by the vehicle while driving away from the RSUs; each intersection is labeled with a letter that is then used to reference distance in Figure 7. The distance between each intersection and the RSUs is also recorded in Figure 7.

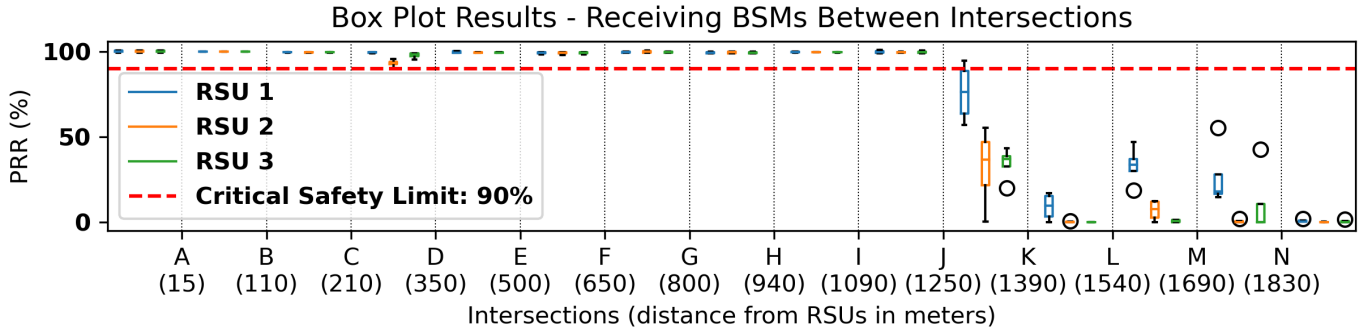

Fig. 7. The box plot results from three trials. The horizontal axis indicates intersections as labeled in Figure 6. Between each intersection, a box plot indicates the performance distribution of each RSU as the OBU moved from one intersection to the next. It can be seen that, as distance increases, the PRR of each RSU eventually falls to 0 as the vehicle gets too far away for a signal.

over the multiple trials. The key takeaway from Figure 7 is the general performance of every RSU compared to each other as the OBU gets further away from where they are stationed. The RSU labels correlate to those of Figure 5, and so we can determine that, as indicated in the indoor testing, RSU 1 leads performance, followed by RSU 2 and then RSU 3.

Subsections V-C and V-D serve to validate the usefulness of indoor testing as recorded in subsections V-A and V-B. The indoor tests, thus being validated, may be considered a cheap and time-efficient method of testing RSU performance while getting useful results. These results serve well to inform a DOT as it strives to make V2I deployment decisions.

VI. The Framework Application

After developing these tests, our final goal is to incorporate them all into a single open-source framework and make it freely available [6]. We created an application that runs on Python and uses Qt GUI development software. A user of this application can have several locally-networked RSUs, input the device network characteristics, and then run the same tests we did automatically.

Each section from the body of this paper has its own tab in the application, and tests can be run after putting the correct RSU credentials (SSH, SNMP, etc.) in these tabs. Figure 8 includes screenshots for these tabs and the information required.

It should be noted that the application, as it stands, requires access to certain tools. For example, our Fuzzer testing requires a user to have previously gained access to the SAE J2735 ASN.1 protocol files [20]; this is proprietary property, and we do not distribute it. For our range testing, we use the Mini-Circuits mesh attenuator [22], and our application operates remotely best with this device. If a different device is desired for this type of testing, the code would need to be modified to interact with that device. While the application has these various requirements, the abstract framework we have attempted to define can be implemented in many different ways, and the application provides a good building block to perform other tests without these specific tools.

No matter what resources an organization like a DOT might have, this framework, and the application that houses it, can very flexibly support many different tests that would be needed to characterize their region. While this application runs the tests we have demonstrated in this paper, it also makes it easy to develop new tests and add them to the framework. Coding in Python and using Qt, two widely popular tools, allows anyone to use this code and add custom functionality, either to our current tests or to any other digital feature they wish to test RSU functionality from the perspective of a network administrator. A stakeholder like a DOT could build and run whatever tests they need on individual RSUs to characterize them, determine the differences, and decide which RSU best matches their needs for a large-scale V2I deployment.

VII. Conclusions and Future Work

Our research initiative is to help DOTs determine the best-performing RSE for their V2I deployment. To accomplish this, we present a testing suite that uses hardware-in-the-loop testing on COTS RSUs that comprehensively and comparatively tests each RSU at the specification compliance, network management, and transmission range layers.

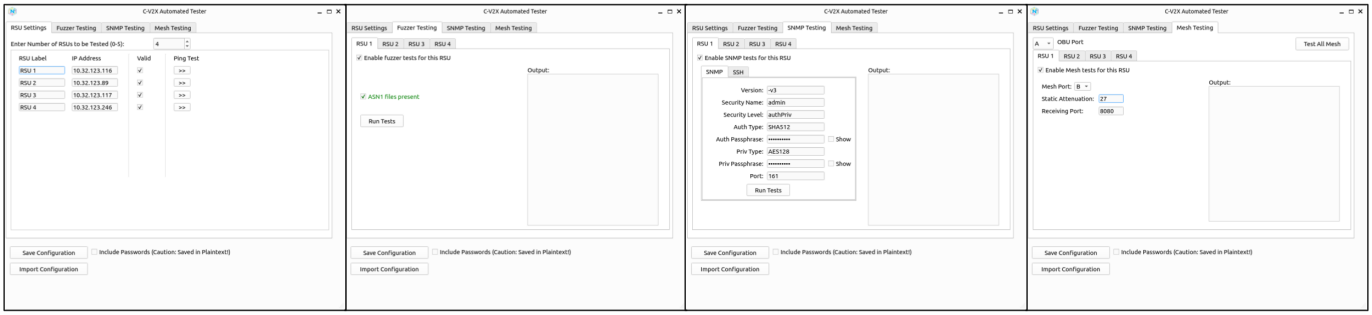

Fig. 8. Screenshots from our open-source application. On the left, our four RSUs are registered with the application. This can be done manually and then saved locally for importing later. Middle left, the system is setup to run the fuzzer tests on the RSUs (they must be in the configuration shown in Figure 2). Middle right, the system is setup to run SNMP tests (or SSH versions if SNMP is disabled on the RSU); these need only be connected to the same network as the device running the application. Far right, the system is setup to run the range tests (they must be in the configuration shown in Figure 4).

We present a GUI application and provide the code as open source. Any organization, like a DOT, could take this application as is or augment it to meet their needs when making V2I deployment decisions. They can test any digital interface an RSU would have that they would need to interact with and characterize.

We are confident in the potential of these methods and the results they have already yielded; in fact, our testing results have already been considered by a state DOT in their own V2I deployment decision making process. While refinements to the methodology will be made over time to address limitations, the methods themselves already show great promise.

This framework is a relatively new method for independent testing of COTS devices as DOTs and other organizations are preparing for large V2I deployments in their jurisdictions. Future work will involve refinement of the testing process, refinement of the GUI application, and improving the modularity of the application. Ultimately, we are confident in the potential this framework has and the ability another user would have to use it for their own V2I deployment decisions.

REFERENCES

[1] "FMVSS No. 150 Vehicle-To-Vehicle Communication Technology For Light Vehicles," Nov. 2016. [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.gov/files/documents/v2v_pria_12-12-16_clean.pdf

[2] H. Noori and M. Valkama, "Impact of vanet-based v2x communication using ieee 802.11p on reducing vehicles traveling time in realistic large scale urban area," in *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, 2013, pp. 654–661.

[3] "Saving lives with connectivity: Accelerating vehicle to everything (v2x) deployment," Jan. 2024. [Online]. Available: https://www.grants.gov/search-results-detail/350731

[4] N. R. U. W. Group, "NTCIP 1218 - National Transportation Communications for ITS Protocol Object Definitions for Roadside Units (RSUs)," Sep. 2020.

[5] OmniAir, "Omniair – industry association promoting interoperability and certification in its, tolling, and connected vehicles." [Online]. Available: https://omniair.org

[6] B. Schiel, "V2X_App," Mar. 2024. [Online]. Available: https://github.com/NET-BYU/V2X_App

[7] F. Eckermann, M. Kahlert, and C. Wietfeld, "Performance analysis of c-v2x mode 4 communication introducing an open-source c-v2x simulator," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5.

[8] A. Nabil, K. Kaur, C. Dietrich, and V. Marojevic, "Performance analysis of sensing-based semi-persistent scheduling in c-v2x networks," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–5.

[9] J. Manco, G. G. Baños, J. Härri, and M. Sepulcre, "Prototyping v2x applications in large-scale scenarios using openairinterface," in *2020 IEEE Vehicular Networking Conference (VNC)*, 2020, pp. 1–4.

[10] A. Rayamajhi, A. Yoseph, A. Balse, Z. Huang, E. M. Leslie, and V. Fessmann, "Preliminary Performance Baseline Testing for Dedicated Short-Range Communication (DSRC) and Cellular Vehicle-to-Everything (C-V2X)," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, Nov. 2020, pp. 1–5, iSSN: 2577-2465.

[11] V. Maglogiannis, D. Naudts, S. Hadiwardoyo, D. van den Akker, J. Marquez-Barja, and I. Moerman, "Experimental v2x evaluation for c-v2x and its-g5 technologies in a real-life highway environment," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1521–1538, 2022.

[12] A. E. Braten, F. A. Kraemer, and D. Palma, "Autonomous IoT Device Management Systems: Structured Review and Generalized Cognitive Model," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4275–4290, Mar. 2021, conference Name: IEEE Internet of Things Journal.

[13] G. Twardokus and H. Rahbari, "Vehicle-to-nothing? securing c-v2x against protocol-aware dos attacks," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, May 2022, pp. 1629–1638.

[14] Y. Li, R. Hou, K.-S. Lui, and H. Li, "An MEC-Based DoS Attack Detection Mechanism for C-V2X Networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*. Abu Dhabi, United Arab Emirates: IEEE, Dec. 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8647323/

[15] V2X Core Technical Committee, "V2X Communications Message Set Dictionary," SAE International, Tech. Rep., 2022. [Online]. Available: https://www.sae.org/content/j2735_202211

[16] B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of unix utilities," *Commun. ACM*, vol. 33, no. 12, p. 32–44, dec 1990. [Online]. Available: https://doi.org/10.1145/96267.96279

[17] V.-T. Pham, M. Böhme, and A. Roychoudhury, "Aflnet: A greybox fuzzer for network protocols," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 460–465.

[18] H. Cao, L. Huang, S. Hu, S. Shi, and Y. Liu, "Owfuzz: Discovering wi-fi flaws in modern devices through over-the-air fuzzing," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 263–273. [Online]. Available: https://doi.org/10.1145/3558482.3590174

[19] E. Moqvist, "asn1tools," https://github.com/eerimoq/asn1tools, 2023.

[20] V2X Core Technical Committee, "V2X Communications Message Set Dictionary™ ASN file," SAE International, Tech. Rep., 2022. [Online]. Available: https://www.sae.org/content/j2735asn_202211

[21] RSU Standardization Working Group, "Cti 4001 v01.01 - roadside unit (rsu) standard amendment," Sep 2022.

[22] Mini-Circuits. 6-Port Mesh Network ZTMN-0695B. [Online]. Available: https://www.minicircuits.com/pdfs/ZTMN-0695B_Datasheet.pdf