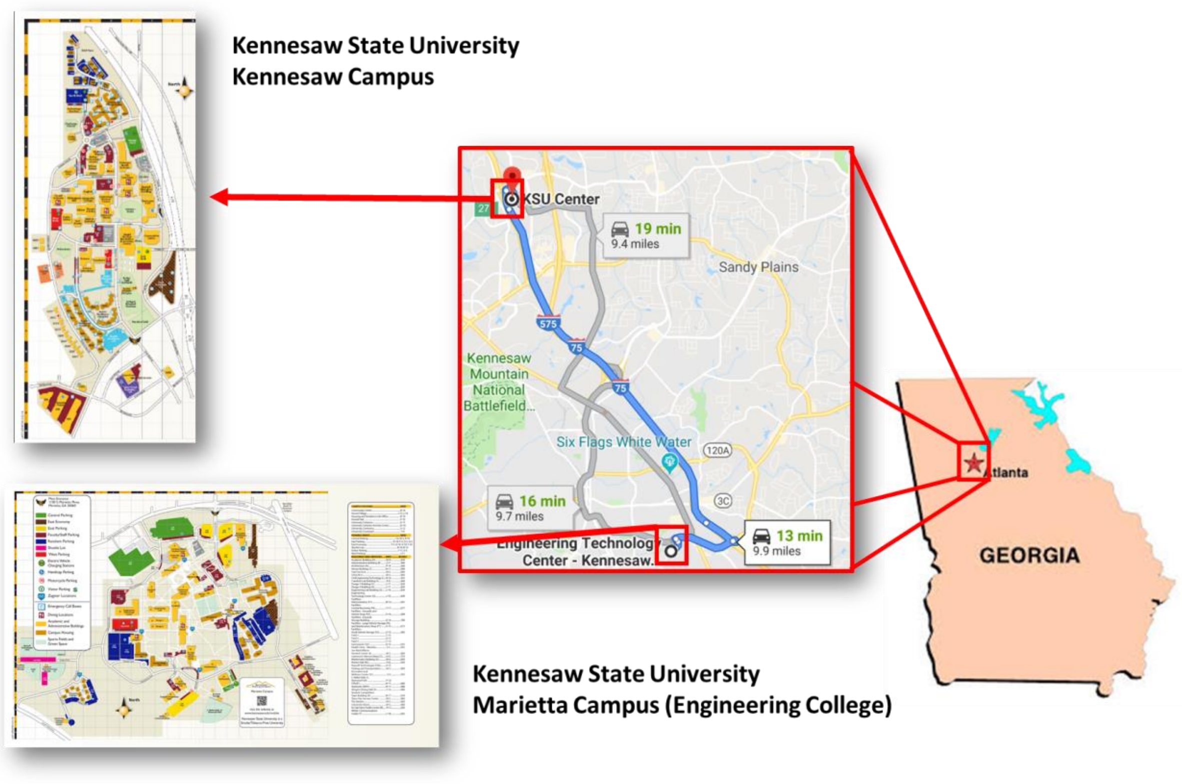
**Proposed Deployment:**

Our deployment is all about testing V2X applications using technologies that are unreachable with the current equipment available. To date, it is near impossible to work with a vendor, understand their API, and write an application using DSRC/C-V2X vendors software stacks. They are essentially locked boxes, and it is stifling scientific discovery. We have bootstrapped and developed our own DSRC radio using open-source hardware, open-source software, and COTS wi-fi cards, we will hereafter refer to our platform as "**research equipment**". We want to study technologies such as blockchain, machine learning/deep learning, ect…applied to the edge. Currently these technologies are implemented in programming languages supported by the web-development community. We would like to conduct research into deploying applications for V2X, that are currently nearly impossible to develop working with a vendor. We only need access to the packets entering the radio, and then build new applications using other programming languages than C/C++.

We seek to deploy Dedicated Short Range Communications (DSRC) radios, Cellular-Vehicle-to-Everything (C-V2X) radios, and research equipment in the 5.9GHz band.

We desire to conduct various studies in partnership with various organizations such as: Departments of Transportations (DOT) in the metro-Atlanta area. Including, but not limited to: Marietta, DOT, Cobb County DOT, Atlanta DOT, Gwinett DOT, Georgia DOT. We also intend to partner with regional partners such as Curiosity Labs at Peachtree Corners, and Alpharetta, GA iATL, and the Atlanta Regional Commission. We are also seeking a proposal with Ford Motors research division to deploy our research equipment and experiment with new V2X applications.

We have research equipment that we would like to first deploy on our campus located in Marietta, GA and our Kennesaw, GA campus, and along Highway I-75 connecting our campuses.


Kennesaw State University
Kennesaw Campus
KSU Center
19 min
9.4 miles
Sandy Plains
575
75
Kennesaw Mountain National Battlefield...
Six Flags White Water
120A
16 min
9.7 miles
3C
13 min
9.9 miles
Engineering Technology Center - Kennesaw...
Atlanta
GEORGIA
Kennesaw State University
Marietta Campus (Engineering College)

**The purpose of our deployment is to:**

Conduct application testing using our research equipment. We have a custom kernel which allows us to use COTS Wi-Fi cards in the 5.9GHz band. We desire to use this COTS equipment as research equipment which can be configured as a: U-NII-4 Unlicensed Wi-Fi equipment and/or DSRC equipment. Our research equipment will allow us to rapidly develop V2X applications quickly for performing academic research.

According to the recent NPRM (Notice of Proposed Rulemaking – ET Docket No. 19-138), we also intend to conduct various tests between configuring our device operating similar to a U-NII-4 unlicensed devices at (5.850-5.895), similar to a DSRC radio at (5.895-5.905 GHz), and potentially similarly as a C-V2X radio at (5.905-5.925 GHz). The DSRC and C-V2X radios will be bought from commercial vendors. We currently have 7 DSRC OBUS, 1 RSU and we are purchasing C-V2X radios this month (Feb 2020). Our research equipment are custom COTS devices which are open-source hardware platforms with COTS Wi-Fi cards. Our open-source software will run containerization software.

**Benefits of Containerization in V2X**

Rapid prototyping using COTS Wi-Fi cards can allow research institutions, or municipalities to roll-out new application prototypes alongside automotive applications. Universities can deploy campus-wide custom spectrum networks using inexpensive hardware already available to the wider developer community. Containerization allows for real world testing of V2X applications to be conducted by researchers while being as open as they choose.

**Future Proof:** New technologies can be prototyped quicker, using containers. The monolithic architecture would be locked into the vendors API and run-time environment. To prototype new technologies such as a new machine learning model or distributed ledger application in the V2X context, the monolithic architecture needs to have an entire code base developed implementing that technology for that devices specific run-time environment. With microservices, widely available open-source libraries can be deployed within a container implementing the new technology. This is what we mean by leveraging the wider development community.

**Easier to test and maintain:** With containers it is easier to add new developers to traditionally large code bases. Developing on a microservice application is much easier than developing on a monolithic application. When hiring new developers to the application team, the developer does not need to configure their own machine to be exactly like the rest of the team. The container provides the runtime environment, the new developer just needs to run the container manager locally on their machine with the correct image (i.e. host operating system). Also, the new developer only needs to understand their microservice to build the application. While there is no set standard for microservice code base size, compared to monolithic code base size, the new developer only has to focus on one scope of the application, and not worry about how their function affects every global variables or data structure.

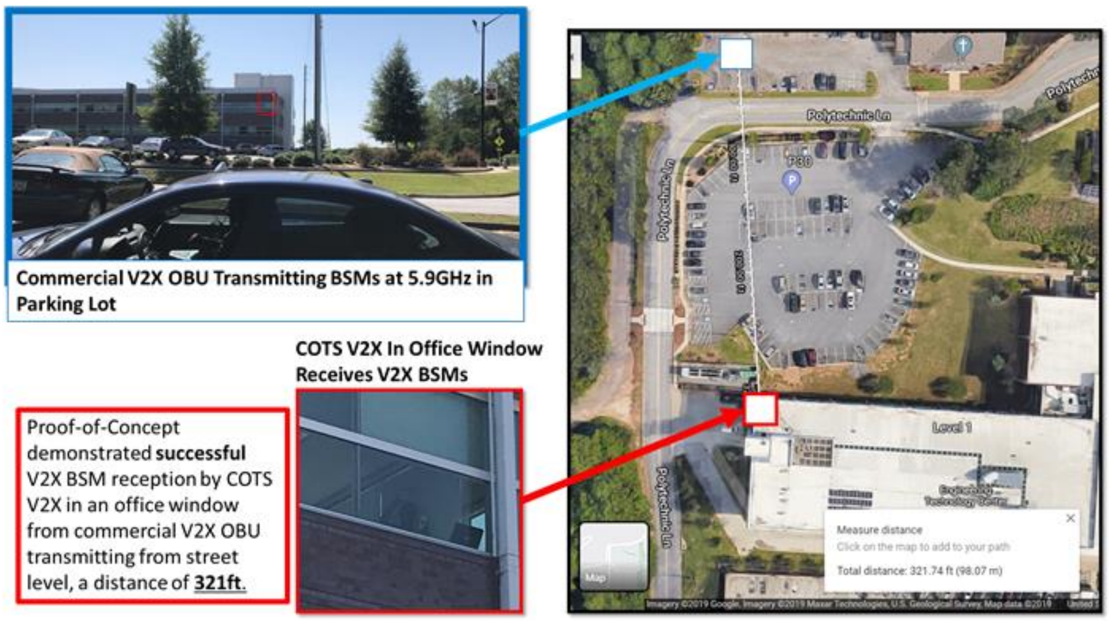
**Easier to scale and support interoperability:** Containerization allows a diversity of developers to participate in developing a V2X application. Whereas a V2X application development team would be limited to C/C++ software engineers in the monolithic paradigm, the same V2X application can be written in Javascript, C#, Python. There are significant implications to the application development strategy with this architecture. Primarily, that application code is easier to standardize across a wider developer community. This is how interoperability can be achieved, through using programming languages that have enabled the world wide web and the Internet of Things. With containerization, a roadmap can be more easily defined through more widely used runtime environments.

**Improved security and trustworthiness:** Containers can be privileged/denied hardware resource capabilities, independent of other containers. What this means is that a dedicated developer team can work on a safety microservice in a trusted environment, and still interact with an infotainment microservice which can be outsourced to a third-party developer. The different teams and their microservices are siloed from each other (from a developers perspective), such that their contributions can be developed and tested in isolation from each other. This provides added security and reliability in the application.

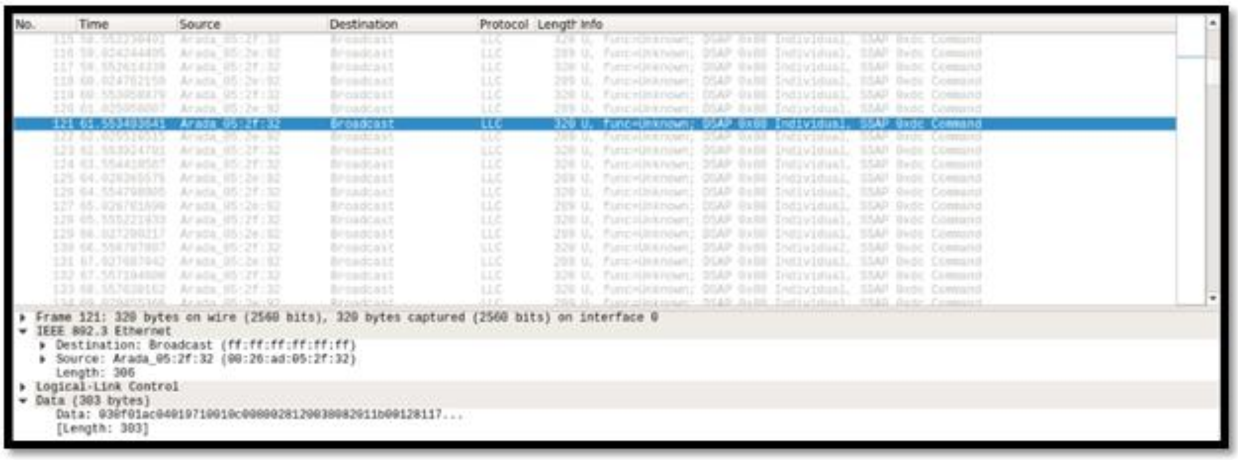
**Granular control of deployment:** Containerization allows for built-in versioning of applications as a whole and/or individual microservices. In practice, a road authority may want to test a new message format for Signal Phase and Timing (SPaT). A container can be delivered to the RSU that runs alongside the incumbent SPaT application. The test application can be improved upon during operation, without destroying the operation of the stable version of the application. If the test application throws an exception, the stable SPaT application will continue to run, while the defunct test application reports back a log before the container manager spins-down the instance, or resets the instance prior to failure for traceability.

**Proof-of-Concept Deployment**

In Marietta, GA, we are currently planning a campus wide deployment of our research equipment. This will be the first truly open V2X deployment in the world, at this scale, and a place for V2X innovation unlike any other. We are currently in the planning phase where we have validated a linux distribution running a custom linux kernel. This kernel enables a laptop to receive DSRC packets. We plan to introduce containerization on vendor procured C-V2X devices as well.

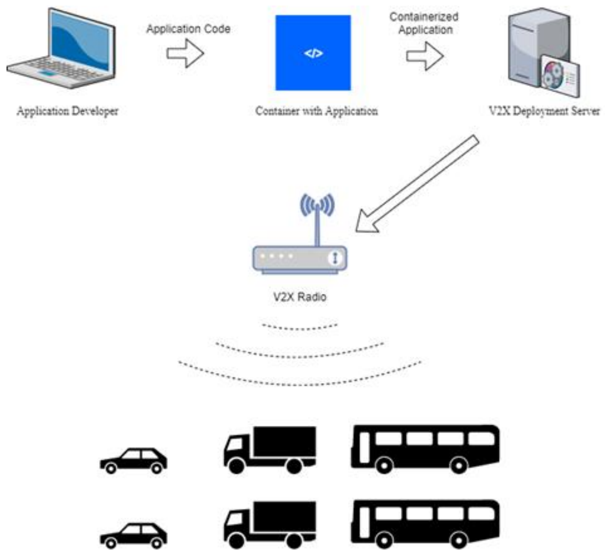

Commercial V2X OBU Transmitting BSMs at 5.9GHz in Parking Lot
COTS V2X In Office Window Receives V2X BSMs
Proof-of-Concept demonstrated successful V2X BSM reception by COTS V2X in an office window from commercial V2X OBU transmitting from street level, a distance of 321ft.
Polytechnic Ln
Level 1
Measure distance
Click on the map to add to your path
Total distance: 321.74 ft (98.07 m)
Map

Initial experiments use **custom linux kernel on laptop** to receive BSMs from test vehicle on street.



```
No.   Time          Source           Destination   Protocol  Length Info
  115 58.552230401  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  116 59.024244405  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  117 59.552614339  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  118 60.024762159  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  119 60.553958479  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  120 61.025958087  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  121 61.553403641  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  122 62.025516515  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  123 62.553924701  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  124 63.554418567  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  125 64.026365579  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  126 64.554798805  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  127 65.026781699  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  128 65.555221933  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  129 66.027290217  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  130 66.556797087  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  131 67.027487942  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  132 67.557104808  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  133 68.557630162  Arada_05:2f:32   Broadcast     LLC       320 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
  134 69.029455166  Arada_05:2e:92   Broadcast     LLC       299 U, func=Unknown; DSAP 0x88 Individual, SSAP 0xdc Command
▸ Frame 121: 320 bytes on wire (2560 bits), 320 bytes captured (2560 bits) on interface 0
▾ IEEE 802.3 Ethernet
  ▸ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Source: Arada_05:2f:32 (00:26:ad:05:2f:32)
    Length: 306
▸ Logical-Link Control
▾ Data (303 bytes)
    Data: 030f01ac04019710010c0000028120038082011b00128117...
    [Length: 303]
```

Validating kernel receiving DSRC BSMs on laptop wi-fi card from inside a building.



Overview of V2X Application Deployment

The research equipment expected to be deployed around campus are focused on the infrastructure side of the network where there is constant connectivity to the containerization server through the campus Wi-Fi. Our devices will be deployed inside buildings at windows facing the streets or outside on a light pole or at an intersection (working with our local DOTs). As V2X equipped vehicles enter the reception range of our equipment, containerized applications running locally on our equipment facilitate packets received through the V2X radio layer. Containers are able to access the packet and run logic over connection streams.

Planned deployment of research equipment.

From a development perspective, programmers can develop applications locally on their own machines regardless of underlying hardware or operating system. This is the key enabling feature that containerization allows. A single developer can write an application in python 2.7 and send that containerized app with all dependencies included to any other developer to contribute to. This saves time due to system configuration and interoperability. Once a container is ready for deployment on our V2X network, all devices receiving the container can immediately begin running the application without a system admin needed to configure the target specifically for the python 2.7 application. The apps are developed and deployed in near-real-time, and their operations are monitored remotely. If an application fails, the device is not locked into a dead app, the container fails, and the device reports back to the developer that the container fails, but other applications running in other containers are not affected.

As an example, a V2X car counting application written in C for traffic management services can be deployed next to an experimental blockchain application written in python for security credential management services. These applications would run isolated from each other, at the edge. But if need be, traffic management operations could allow the blockchain service to communicate with the car counting application. These applications can be written by different vendors operating under different contractual obligations. This is enabled by containerization software running locally on the V2X edge device.

The development of applications using V2X standards will require testing of the applications in the field. In the case of a university research team, campus vehicles and buses may provide a starting point for the researchers to deploy their application and test the communication between the deployment test systems. As the nature of V2X includes more than vehicle-to-vehicle communication, use cases such as vehicle-to-infrastructure, network, or pavement monitoring can be considered. To use a college campus bus as an example target for development, the method for delivering application test builds to the bus will have to be resilient to signal attenuation and radio proximity.

The communication between the bus and the application developer will rely on wireless standards like WiFi or cellular data connections. While the deployment target is on campus at the university, V2X devices operating over standards like 5.8 GHz Unlicensed ISM (industrial, scientific and medical), consumer WiFi bands, or DSRC can deliver the application build. This use case is where the necessity of containerized applications is most obvious, due to the lightweight wrapper of containers and their ability to precisely dictate the runtime environment within the container. To update a virtual machine and integrate the new application build would necessitate a longer connection time between the deployment target on the bus and the deployment radio. Where virtual machine images can be sized in the GBs, containers can be delivered in a few hundred MBs.

We have not implemented a microservice application yet, due to initial baseline configuration and validation first. We are currently exploring new use-cases of fog computing and Cloud-of-Things paradigm. Benchmarks will be obtained and reported in future publications.

**Safety as Priority Number One**

One of the most important topics surrounding V2X infrastructure is the safety of V2X systems and their ability to protect drivers and pedestrians from collisions through vehicular awareness. As V2X systems deploy into production vehicles, the intent of bad actors to harm or destroy using these connected vehicles must be kept at the forefront of all development efforts. The goal of lessening roadway fatalities, ideally to zero deaths, is called "Vision Zero" and is generally used to describe a safety initiative by using V2X communications. In order to achieve Vision Zero goals, application prototyping must prioritize security from the onset of development, not as an afterthought. Containers provide a strong focus on security by isolating applications from the system and other local microservices, unless explicitly told otherwise. By isolating the applications from the lower level systems in the vehicle, a hacker could be kept from getting into the steering or braking systems of the vehicle and taking control. The trust of the public and of regulatory bodies must be considered one of the most valuable assets to a V2X research entity experimenting with containerized application development.

**Conclusion**

Vision Zero is unlikely to be achieved without the adoption of V2X standards by all major automotive manufacturers and legislative bodies. The agreement between these entities will drastically increase the speed at which a set of communication standards is agreed upon. Once a set of standards is in place, development of new functionality and safety features will gain momentum, and the capability of vehicles to use V2X standards will become a selling point to consumers. In order to get there, containerization provides a lightweight and secure framework for developing and testing V2X systems, and is realistic for budget conscious research institutions due to the support of the open source community. In tandem to open source efforts, containerization is becoming increasingly popular in the commercial data center market, guaranteeing maintenance and development for the foreseeable future. V2X can learn from the mistakes of the Information Technology field, and be given a proper roadmap into future proofing edge applications.