University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

Design of a Throughput Test Automation Platform for a Telematic Control Unit

Ford Motor Company
Ottawa, Ontario, Canada

Prepared by
Nicholas William Shields
20626940
nwshield@uwaterloo.ca
2B, Computer Engineering

8 May 2018
confidential-1

350 Lester Street
Waterloo, Ontario, Canada
N2L, 3L1

May 8, 2018

Vincent Gaudet, Chair
Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario
N2L 3G1

Dear Sir,

This report, entitled "Design of a Throughput Test Automation Platform for a Telematic Control Unit", was prepared as my 2B Work Report for the University of Waterloo. This report is in fulfillment of the course WKRPT 201. The purpose of this report is to outline the requirements of a throughput test automation platform and to detail the creation process of this platform that can execute and process the results of a number of given test cases. This is a confidential-1 report as some of the information mentioned is proprietary.

Ford Motor Company is an American automaker based out of Dearborn, Michigan. The company is well known for its various automobiles, which include the Ford F-150, Mustang and Focus to name a few.

I was hired at Ford as an Assistant Software Developer on the Connected Vehicles team by Shahid Chaudry. Shahid is the Product Development Manager of the team, and he was responsible for overseeing the work I was doing as well as the work being conducted by the nine members of the team. The Connected Vehicles Team is responsible for developing and testing solutions that allow Ford to better collect car data by sending this data to the cloud using in-car cellular technology. The team is also responsible for further developing and testing SYNC5, which is the product name for the infotainment system found in the front console of all modern Ford vehicles.

I would like to thank Shahid for his guidance throughout my term at Ford. I would also like to acknowledge one of my coworkers, Florian Khanghi, for his assistance in selecting this topic as well as serving as a great mentor to me throughout the work term. I hereby confirm that I have received no further help other than what is mentioned above in writing this report. I also confirm this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

Nicholas William Shields
ID 20626940

Contributions

During my co-op work placement, I was employed at Ford as an Assistant Software Developer. The Connected Vehicles Team, for which I was member of, was a relatively small team, consisting of six core team members, as well as three contractors. The Connected Vehicles Team belonged to a larger team called the Ford Fully Networked Vehicle team. This larger team consisted of over 400 developers and engineers all working on projects related to making Ford vehicles and all of their components cloud connected. Throughout the work placement, I worked on one main project which was related to testing and reporting the network throughput on the Telematic Control Unit that Ford will use inside their future vehicles.

The team's main goal was to ensure that the performance of all the components used in the Fully Networked Vehicle always met a certain threshold. This was done through the continuous development of software tools that interact directly with the various hardware components found inside modern Ford vehicles. To further explain this, it should be noted that there are three critical components that our team was responsible for ensuring that performance standards were met. These components are:

1. The Telematic Control Unit which is responsible for connecting the car to the internet;
2. The Electronic Control Unit which is responsible for collecting data from the car sensors; and
3. The SYNC Infotainment Center which is the touch screen display found in the front of the car.

The team's secondary goals aimed at helping further develop the above-mentioned components through assisting the teams that were responsible for developing each component. In helping further develop these components, the team puts careful consideration into researching ways to maximize the performance of each component. The team is also responsible for reporting software bugs to respective teams and reporting performance updates to higher executives at Ford.

One of my roles at Ford was to submit weekly bug fixes and improvements for an automation platform built using Python that tested the time it took for Ford's Telematic Control Unit to establish a connection with a cellular network. The bug fixes that I normally conducted were usually fairly small and easy, without requiring much thought or analysis. Hence, no further discussion about this task will be made in this report.

The biggest part of my job was designing and creating a software package that automated throughput testing on Ford's Telematic Control Unit. This required an extensive amount of planning as there was a lot of requirements that needed to be met for this software package. For the first two months of my work term, I spent hours building prototypes that met each individual requirement. The last two months is when these prototypes that I created were combined into one larger software automation platform. Extensive thought and consideration was necessary in deciding how these prototypes would be combined.

At Ford, I was the only person working on the project that I was assigned to. Similarly, I was also the only person working on this report. This serves as one of the core elements where my job relates heavily to this report. In both this report and in my job, I have been forced to think on my own and come up with and test my own solutions. Being the only person working on a project at Ford also forced me to continuously record information pertaining to what I was doing so that it was easier to explain to coworkers when required.

One of the very important elements of this report is that it is the first time where I've been able to fully document a project that I was assigned during a work term. In previous work placements, I would develop good code with thorough inline documentation that explained how the code worked and what purpose it served, but it always fell short of explaining the code decisions that were made. This report has allowed me to document, detail and reflect on my thought process when making design decisions. Furthermore, this report has helped me to improve my overall technical writing ability and it has helped allow me to explain a topic that is very technical in nature to a non-technical audience through the use of background information. Conducting this report has also helped me better understand the importance of the project to Ford as it forced me to put more consideration and thought towards why my team at Ford thought this project was necessary and it has given me a better understanding about Ford's vision towards building a fully networked vehicle.

In the broader scheme of things, the throughput test automation platform that is detailed in this report will continue to serve as a crucial tool to the Connected Vehicles Team at Ford. Even though I am no longer a member of the team, the platform that I developed will continue to help provide continuous throughput data to the team, allowing them to catch potential problems far sooner than they previously were able to. Though the platform is certainly far from perfect, it still serves as the fundamental basis towards fully eliminating the need to ever run a manual throughput test. The platform, as it exists now, will continue to run a long list of throughput tests each day after new firmware becomes available for the Telematic Control Unit.

Summary

The main purpose of the report is to document the analysis and design of a Throughput Test Automation Platform for Ford's Telematic Control Unit. This involves the analysis of a hardware setup suitable to accomplish throughput testing as well as the selection and design of a software platform that can automatically run throughput tests on the device and report the results to a common location that can be reviewed by an engineer at a later time.

The scope of this project is to eliminate the need for manual throughput testing to ensure that the Connected Vehicles Team can use their time as effectively as possible. This platform being developed is intended to be used by internal Ford Employees only, specifically by employees that are current or future members of the Connected Vehicles Team.

The major points covered in this report are as follows. Section 1.0 introduces the report and covers the background information necessary for a reader to better understand the content in this report. Section 2.0 details the Ford Fully Networked Vehicle and reasoning behind the need for an automation platform. Section 3.0 introduces the requirements of the throughput test automation platform and it details the test bench being used. Section 4.0 details the development of the automation platform and further explains some of the important decisions made throughout the course of the creation process that helped meet the requirements of the project. Lastly, section 5.0 details the thought process that went behind solving some of the critical problems that occurred throughout the course of the project development. Section 5.0 is also followed by a conclusion and recommendation section which recaps everything discussed in the report.

The major conclusions in this report are that the chosen design and implementation for the throughput test automation platform meets the standards with the required functionality. The hardware setup chosen is suitable and the software can reliably run throughput tests and return the results to a common web location for an engineer or developer to review at his or her convenience. Though difficult problems did arise throughout the development process, these difficult problems have been solved using proper engineering analysis and have been documented extensively in the hope that they can be easily solved if they ever occur again.

The major recommendations in this report are that the throughput test automation platform should be deployed and used immediately. Doing so will free up time for team engineers and developers to focus their time and effort on other projects. In saying this, there are additional software elements that should be added to the platform that have yet to be added. By adding these additional elements, the overall reliability of the platform can be improved, and it can also be tweaked to run more complex tests that are outside of the scope of this report.

Table of Contents

List of Figures

List of Tables

1 Introduction

This report aims to provide a non-technical audience with a complete breakdown of how a throughput test automation platform was designed for a Telematic Control Unit. The idea of throughput testing has been mentioned several times in the preceding front matter of this report. Unfortunately, for a non-technical audience, the idea of throughput may be unfamiliar. Furthermore, there are several subtopics discussed that are related to throughput testing that will also appear complex in nature. Careful consideration has been taken in providing a non-technical audience with the required knowledge to understand each element of this report. This section aims at providing a non-technical audience with all the information required to formulate a proper understanding of the content that follows in the later sections of this report.

1.1 Throughput Testing

Throughput is defined as the amount of material or items passing through a system or process. However, in the context of this report, network throughput is actually the term of interest. Network throughput is a measure of the rate of successful messages delivered through a communication channel. The unit of measurement used for throughput tests in the case of this report is Megabits per second (Mbit/s). For the remainder of this report, it should be made aware to any reader that when the term throughput is mentioned, it should be inferred that network throughput is what is actually meant. The client-server model can be used to help better understand the idea of throughput and simple network communications.

1.1.1 Client-Server Model

To further grasp the concept of throughput, a step back needs to be taken to explain the idea of the server-client model. Figure 1 shows what a simple client-server model looks like.

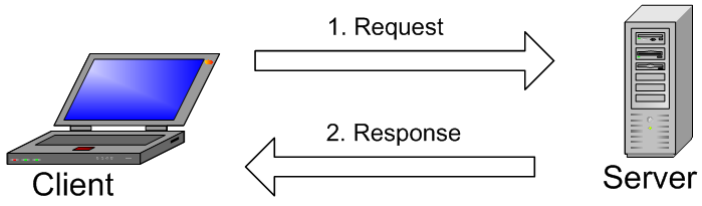

Figure 1. A simple client-server model [1].

In the client-server model, the end user is the client. When the end user wants to access content from the server, they can send a network request to the server. The server will respond with the information that was requested by the client. A less technical analogy to help familiarize a non-

technical reader with this concept is as follows. Consider the use of a mobile device to access any social media platform. In this case, the example of Facebook will be used. When the mobile device wants to display the news feed from Facebook, it first needs to request it from the Facebook server through the use of the internet. When the Facebook server receives this request, it sends a response to the mobile device, which contains the news feed that the mobile device initially requested. Clearly in this example, the mobile phone acts as the client and the Facebook server acts as the server. With that in mind, the rate at which data is transferred between the client and the server and vice-versa is the throughput.

1.2 Throughput Test Types

Provided that an understanding of what throughput is has been made, the concept can be broken down into three different test types, which are:

- Uplink Throughput Tests: Throughput of the data being sent to the server from the client.
- Downlink Throughput Tests: Throughput of the data being sent to the client from the server.
- Bidirectional Throughput Test: An uplink and downlink test are executed simultaneously.

Each one of these test types is important in assessing the throughput performance of any device. Emphasis is placed on how well the bidirectional test performs because most devices send and receive data at the same time, so the bidirectional throughput test is a good indicator as to how well a device will perform from a network throughput standpoint.

1.3 Throughput Test Protocols

Beyond the three different types of throughput tests that exist, there is also the protocol being used that needs to be taken into account. The actual specifics of each protocol remain outside the scope of this report, but it's important to take note that each of the three tests mentioned in section 1.2 can be executed as a TCP (Transmission Control Protocol) throughput test or as a UDP (User Datagram Protocol) throughput test. Given this information, it can be concluded that a total of six different throughput tests types now exist:

- TCP Uplink Throughput Test
- TCP Downlink Throughput Test
- TCP Bidirectional Throughput Test
- UDP Uplink Throughput Test
- UDP Downlink Throughput Test

- UDP Bidirectional Throughput Test

All six of these tests need to be considered, but as previously noted, the bidirectional tests are the best indicator has to how well the device under test (DUT) can perform under stress, so these are the tests that are most important.

2 Ford FNV and the Throughput Testing Problem

The preceding section already hints at the idea that the coming sections of this report will oversee the design and analysis of a throughput test automation platform. However, the motive behind producing such a platform should be made clear beforehand. This section aims to provide an understanding of the Ford Fully Networked Vehicle eco-system, as well as the motive behind the need for this automation platform.

2.1 Ford Fully Networked Vehicle

Ford FNV (Fully Networked Vehicle) is an initiative that is current being handled by the Ford FNV team, located in Ottawa, Ontario, Canada. The team is currently working on solutions that bring better connectivity into modern Ford vehicles. Some of the core developments within this team include:

- Continued development of the touch-screen infotainment system found in the front console of all modern Ford Vehicles. The name of this infotainment system is called SYNC5, which is not an abbreviation. The current initiatives include prepackaging internet services such Spotify into the system, as well as innovating ways to make the device more futureproof and useful to the consumer. Previous iterations of SYNC have received various criticisms from being slow and too small for comfortable use.
- Development of a new ECU (Electronics Control Unit) which will replace the existing ECU that is currently being used by Ford. The ECU is a critical component to all vehicles as it serves as a central hub for information sharing. The ECU currently being used by Ford uses the CAN (Controller Area Network) bus to connect to all the sensors throughout the vehicle. CAN is widely accepted as the standard communication bus system used in all vehicles due to its multiplexing ability, which allows multiple analog signals to be sent using one wire. Some examples of these sensors that the ECU connects to include oil temperature sensors and fluid level sensors. The main purpose of the ECU is to collect data from the sensors around the car. The ECU is then supposed to notify the operator of any problems if these sensors report unideal measurements. The secondary purpose of the ECU is to also serve as an input

for the operator to control various car functions, including turn signals, vehicle headlights, and windshield wiper controls to name a few. The newest ECU design currently being worked on by the FNV team aims at moving away from the CAN bus and instead replacing it with ethernet. The reasoning behind this switch is that switching to ethernet will work better in accordance to connecting the car to the internet. It will also give Ford the availability to send software updates to the vehicles components, which is something that has never been made possible before. Since CAN is a lightweight protocol [3], it becomes difficult to send large amounts of data. This is where replacing all the connections with ethernet cabling is beneficial as it will allow Ford to send large updates with ease. With ethernet, even updating the vehicle's sensors is now possible.

- Development of a new TCU (Telematics Control Unit) to allow new Ford vehicles to come with cellular connectivity built in. The current TCU that Ford currently deploys in today's vehicles only contains a GPS (Global Positioning Unit) to track the vehicle in the case of it getting stolen. New efforts have been made to include cellular connectivity into the TCU, allowing it to serve not only as a GPS, but now as a cellular modem. The TCU is the most critical part in fulfilling the vision of the FNV team as it provides the internet connectivity to the vehicle, which allows the car to effectively communicate with Ford's internal databases.

As far as this report is concerned, the only area of interest is TCU. To further illustrate the Fully Networked Vehicle, Figure 2 provides a high-level block diagram of the FNV ecosystem.

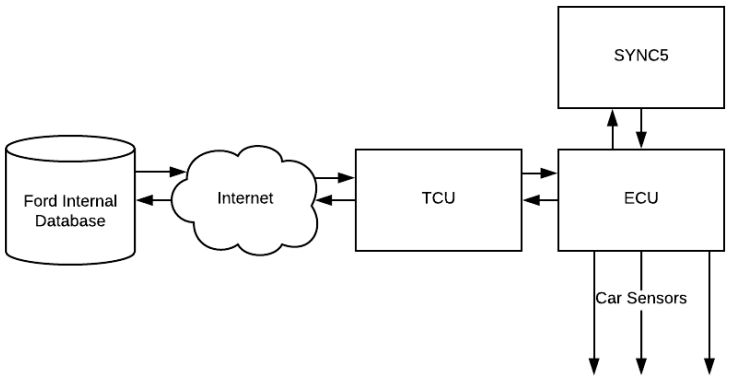

Figure 2. Block Diagram of the Ford FNV Ecosystem.

Evidently, the ECU acts as the central hub and it communicates with the car's sensors and the SYNC5 infotainment system. It's also responsible for passing data to and from the TCU, which is then sent and received from a Ford Internal Database via the internet.

The motive behind a Fully Networked Vehicle is that it serves as major benefit for Ford. Having a fully networked vehicle allows Ford to collect continuous metrics and data related to the vehicle, which includes all of the vehicle's sensor data, as well as information pertaining to the operators driving habits. This is useful because it gives Ford the opportunity to collect data about their vehicles much faster than before. This data can allow Ford to assess the quality of their vehicles. It can help pinpoint trends between car owners, and it can also help Ford identify parts in the vehicle that need improvement. Furthermore, with the FNV, Ford can quietly send software updates to components in the car such as the TCU, ECU and SYNC5. Through the use of the ethernet instead of the CAN bus, car sensors can also theoretically receive updates because ethernet can handle much larger amounts of data.

The Fully Networked Vehicle also serves as a huge benefit to the consumer of the vehicle. Having a vehicle that is connected to the internet can serve as a mobile internet hotspot for passengers in the vehicle, allowing them to surf the web while commuting. The other major benefit is that Ford will keep the vehicles software up to date without the operator being aware, allowing for a better overall experience. Ford also has the ability to send personal push notifications to operator by displaying them on the SYNC5 LCD display. Some useful things that Ford could inform the operator include vehicle recall information and vehicle service requests.

Although the entire Ford FNV ecosystem has been introduced, focus will be shifted to the TCU, as serves as the fundamental component in regard to this report.

2.2 Development of the TCU

With a functional understanding of the Ford FNV ecosystem, more emphasis will now be placed on the TCU as it serves as the fundamental topic of this report. The TCU currently being developed by Ford is a custom Qualcomm Snapdragon chip that features V2C communication, which allows for seamless communication between the TCU and nearby cellular communication towers. The actual specifications of this chip is outside of the scope of this report, so no further discussion will follow. What is important to be aware of is that the TCU has a firmware on it that is actively being developed by Ford, with new firmware build releases occurring daily. These firmware releases can affect the overall cellular transmission performance of the TCU, especially if bugs exist in the code. This happens because the actual computing resources that exist on the

TCU are minimal, so it is important that the code is optimal or else there could be a decrease in the transmission performance of the TCU. Poor transmission performance translates directly to a loss of throughput, which is what the Connected Vehicles Team is responsible for reporting.

3 Design Requirements and Constraints

Provided that a fair understanding of the concepts discussed in both Section 1.0 and Section 2.0 is made, this section details the lab station and hardware that are used for the throughput automation platform. Project requirements will also be detailed as provided by the Connected Vehicles Team.

3.1 Lab Station Setup

A simple diagram outlining the hardware being used in this project is portrayed in Figure 3.

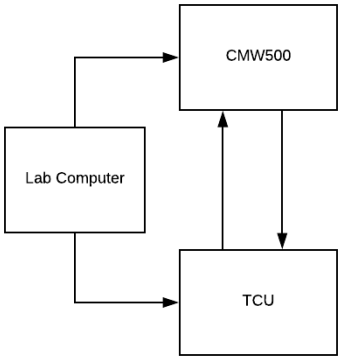

Figure 3. A block diagram of the Lab Station used for throughput testing.

Evidently, the lab station contains only three components. To fully understand the setup as depicted in the diagram, an extensive effort has been made to explain each component.

3.1.1 CMW500

The Rohde and Shwarz CMW500 is a wideband radio communication tester that supports various cellular technologies including LTE (Long Term Evolution) and WCDMA (Wideband Code Division Multiple Access). It's most important feature is that it can generate a cellular signal that the TCU can establish a connection to. It plays the role of a cellular communication tower. There are two reasons the CMW is used instead of simply connecting the TCU to a communication tower for testing. The first reason is that Ford doesn't actually have access to a cellular communication tower for testing. The other, more important reason, is because when testing the throughput on the TCU, it is in the best interest to remove any element that can prevent the TCU

from reaching its maximum achievable throughput. In other words, it is important to isolate the TCU as much as possible to prevent factors such as signal loss from skewing the test results. To further elaborate on the goal of this project, we're interested in automating throughput testing on the TCU to see how the maximum achievable throughput differs between firmware versions.

3.1.2 TCU

The TCU has already been explained in prior sections of this report. For throughput testing and for testing the TCU in general, the unit is isolated from the rest of the FNV. Instead of connecting to a cellular communication tower, like it will in practical use, it is connected to the CMW using a series of coaxial cables. This ensures that there is virtually zero signal loss and that the maximum throughput can be accurately tested.

3.1.3 Lab Computer

The Lab Computer is basic tower running Windows 7 that serves as the middleman between the CMW500 and the TCU. The automation platform will execute on this computer. The Lab Computer is responsible for many important automation functions including:

- Configuring the TCU and CMW with the parameters given in the test case.
- Executing the throughput test between the two devices.
- Collecting the results from each test; and
- Computing averages with the given results and generating a report

3.2 Throughput Automation Requirements

The previous subsection outlines the lab station as it was assigned because it already implies a set of constraints that needed to be taken into account when constructing the throughput automation platform. With this background knowledge regarding the lab station in place, the requirements of the throughput automation platform can now be outlined. The important features that needed to be included in the platform are as follows:

- Throughput Testing should occur daily after a new firmware release occurs.
- The automation platform should be able to install the new firmware automatically.
- A single test case should be able to run multiple times.
- The automation platform should compute average values of each test case.
- The results of the entire automation session should be pushed a wiki page.
- The Connected Vehicles Team should be notified when automation is complete via email.
- The platform should be built using Python.

- Throughput Tests should be conducted using iPerf3.

On top of these requirements, Table 1, as shown below, shows all of the different test cases that the team decided should be required. Also shown in the table is the theoretical throughput that each test case should be able to obtain.

Table 1. The Required Throughput Test Cases and Theoretical Throughput

Test Case Name	Theoretical Achievable Throughput (Mbits/s)
LTE TCP DL 20MHz Single Band	188
LTE TCP UL 20MHz Single Band	48
LTE TCP BI 20MHz Single Band	188/48
LTE UDP DL 20MHz Single Band	200
LTE UDP UL 20MHz Single Band	60
LTE UDP BI 20MHz Single Band	200/60
LTE TCP DL 20+20MHz CA	288
LTE TCP UL 20+20MHz CA	48
LTE TCP BI 20+20MHz CA	288/48
LTE UDP DL 20+20MHz CA	288
LTE UDP UL 20+20MHz CA	60
LTE UDP BI 20+20MHz CA	288/60
WCDMA UDP DL HSPA	21
WCDMA UDP UL HSPA	7.2
WCDMA UDP BI HSPA	21/7.2
WCDMA TCP DL HSPA	15
WCDMA TCP UL HSPA	4.9
WCDMA TCP BI HSPA	15/4.9
WCDMA UDP DL DC-HSPA	42
WCDMA TCP DL DC-HSPA	42

From this list of test cases, it should made clear that all throughput tests for the TCU can be either LTE tests or WCDMA tests. The reason that these two cellular communication protocols were selected is because they are the only two protocols that you would encounter in today's cellular communication networks. It was decided to omit 2G test cases as 2G is now deprecated and used very rarely.

4 Development of the Test Automation Platform

With the requirements and given lab station in place, consideration needed to be taken into how the throughput test automation platform should be created. In order to become familiar with the

lab setup, as well as the API's needed to control the CMW and TCU, a very incremental development approach was taken to meet the requirements outlined in section 3.2.

4.1 Communicating with the CMW and TCU using Python

Since one of the project requirements was to use the Python programming language to implement the platform, there was clearly a need to be able to communicate with the TCU and CMW using Python.

Communicating with the CMW proved to be a problem, with this problem being described in more detail in section 5.0 of this report. Regardless, through extensive research, it was determined that only way to communicate and configure the CMW remotely was to determine a way to leverage Python to send VISA (Virtual Instrument Software Architecture) commands to the CMW using the GPIB (General Purpose Interface Bus) interface that connected the Lab PC to the CMW. Thankfully, there were two ways that this could be done. The first way was to use the built in pyserial Python library, which allows you to open a configurable Serial connection with any device connected to the Lab PC. This was initially thought to be the solution of interest as it allowed for the use of a built in Python library instead of leveraging any sort of community driven library (Which is preferred by Ford). However, the time and effort required to leverage this solution was far too heavy for this to be a feasible option. Furthermore, a very well established open source library known pyvisa has been known to be the best option for communicating with devices that take VISA commands as input. As a result, this route was taken. Using the pyvisa library, communication with the CMW from the Lab PC was found to be reliable and consistent.

On the other hand, communicating with TCU was a task that required extensive research and development. The way that the TCU connected to the Lab PC was through the use of an Android debugger board. What this means is that from the Lab PC, configuring the TCU occurred using the ADB (Android Debugger Bridge) Command Line. There was no API in existence, so in order to execute commands on the TCU, Python software that wraps around the ADB command line was necessary. What further complicated this communication is that research led to no previous examples of anything like it before. To facilitate this communication, the only solution that proved to be reliable and useable was to leverage the Python library known as subprocess to execute commands on the TCU. The subprocess library allows a user to spawn new processes with the ability to collect return codes. Essentially, the subprocess library was used to open an

ADB shell instance. Commands can then be piped into the shell instance through the subprocess object and return codes can be collected.

With valid communication between the Lab PC and the CMW as well the TCU in place, prototypes were designed that were capable of running each of the tests detailed in Section 1.3.

4.2 Throughput Test Prototypes

As mentioned in section 1.2, there are three fundamental types of throughput types that can be executed. This project mandated that the automation platform is capable of executing all three test types. The initial suggestion was to start building a single Python class that incorporated all three test types, but the foreseeable problem here was the level of complexity that was being added at such an early stage of the development. As a result, it was decided to break each test type into its prototype class. That way, a thorough understanding could be met in regards to how each test type is configured and executed. Furthermore, the prototypes could be used towards developing the final throughput automation platform and having three separate prototypes can help observe commonalities between each test type, which is useful for applying coding principles such as inheritance.

The result was three separate prototypes, one for each of the test types mentioned in Section 1.2. Each of these prototypes were capable of executing either a TCP or UDP throughput test, as well as being capable of collecting the throughput results and returning the average throughput.

4.3 Combining Prototypes

After each prototype was completed, a design meeting was held to decide on the way that the automation platform should be designed. The commonalities recognized from each prototype were collected, and the following was found to be common between each prototype:

- Each prototype had a setup, execution and cleanup function; and
- Each prototype shared similar input parameters.

Furthermore, the differences between each prototype appeared to simply be the code the was executed in each method. Using this information as the premise, the conclusion that was made is that the factory design pattern [2] (As explained in the novel, “Design Patterns: Elements of Reusable Object-Oriented Software”) could be leveraged to create a base test case object that each throughput test type could inherit. That way, a common structure could be enforced to keep the automation platform consistent across all types of throughput tests.

It was also decided that in order to promote consistency, a specific config dictionary object was introduced and enforced. That way, regardless of what type of throughput test you would like to run, the test case input configuration needed to follow a strict format to allow for a common structure to be used.

4.4 Developing the Composite Test Case Class

With the three prototypes in place, a new class needed to be introduced that encapsulated all three prototypes into one. This was one of the initial project requirements, but the issue with developing it at the start was that it would be too complex in nature. The other important feature of this new class is that it needed to be able to execute a given throughput case several times instead of just once. It also needed to be able to collect the results of each individual test run and return an average afterwards. Using the same structure as the three prototypes designed previously, a series of if statements were used to combine all three prototypes into one class. The functionality was also included that allowed for a given test case to have multiple runs and functions for collecting and computing the average throughput were also included.

The result of having this composite test case class is that the three prototypes were no longer needed because all of the functionality that they included was now a part of a larger composite test case class. The only feature that was not added was a test case runner, and that will be described next.

4.5 The Test Session

With the composite test case class now in place, there was now a functional ability to be able to execute any of the test cases outlined in Table 2 found in section 3.2. However, a test case runner needed to be added to allow all of the test cases to be executed one by one. This is where the Test Session class was added in. The Test Session was the most important part of the project as it allowed for automation to occur. The Test Session class is responsible for taking a list of throughput test cases with their parameters as input and then executing each test case one by one and storing the results of each test case in a master dictionary object. At the conclusion, when all the test cases have been executed, the results were passed to a report generating function that formatted the results into a readable table. The Test Session class was ultimately the engine behind the throughput test automation platform so proper error handling needed to exist in order to promote reliable automation. Custom Error Exceptions were created that corresponded to known problems within the automation platform.

4.6 Compiling a Readable Report

As mentioned in the Section 4.5, the Test Session class created a master dictionary that stored all the results from each test case. These results were then compiled into a simple HTML table with each row containing the test name and the average throughput achieved. Colour coding was also added to identify passing and failing test cases. A basic example report is shown in Table 2.

Table 2. A sample output report.

Test Name	Average Throughput Achieved (Mbits/s)
LTE TCP DL 20MHz Single Band	187.7
LTE TCP UL 20MHz Single Band	28
LTE TCP BI 20MHz Single Band	188/48

The simplicity of this report was a design decision. The reasoning for the simplicity came from the idea that it shouldn't take overlooking engineer long to assess the results of the report and he or she should only be drawn towards test results that are appearing in red.

5 Problems Encountered During Development

Since no throughput test automation platform existed prior to this project, there was not a whole lot of software to reference when trying to debug the various problems that were encountered throughout the development of this project. This section details some of the critical problems encountered as well as the solutions that solved them, through the use of engineering analysis and informed decision making.

5.1 Remotely Configuring the CMW

Throughout the project, the CMW500 proved the be the biggest nuisance in regard to developing the platform. Namely, the documentation that was provided with the device was insufficient for a number of things which led to many problems. Of all the problems that required debugging, two of them stood out as being critical to creating a reliable automation platform.

5.1.1 Incompatible Software Drivers

During the initial stages, the software drivers necessary to communicate with the CMW from the Lab PC were not present. Software drivers were then fetched and installed from the manufacturers website. While the problem should have been resolved at this point, the ability to communicate with the CMW from the Lab PC was still broken. At this point, the general idea

was to uninstall any drivers related to GPIB and VISA interfacing including the one installed from the manufacturers website. The reasoning behind this was because it was thought that that there may have been conflicting drivers that were both simultaneously being used to try and interface with the CMW.

After cleaning up the drivers on the Lab PC and then reinstalling the manufacturers software, the ability to send commands to the CMW was still not possible. At this point, it was assured that the software and drivers necessary for communication were present and correctly installed with no other conflicting drivers. To further isolate the problem, all devices other than the CMW were disconnected from the Lab PC. The PC and CMW were also rebooted to try and assure that it was not an Operating System or hardware issue that was causing the problem.

With everything disconnected except the CMW, the communication remained broken. At this point the next thing that was inspected was the GPIB cable that connected the CMW to the Lab PC. Despite the cable being a perfect fit to the connector found on the rear-side of the, it was suspected that the internal wiring of the cable was incorrect. The reason for this is because GPIB connections tend to be unique for devices made by different manufacturers. The GPIB cable that was used was manufactured by Texas Instruments and not Rohde & Shwarz (The CMW manufacturer). Further inspection showed that the GPIB cable being used was in fact invalid for the CMW. In particular, it was observed that the DIO1 (Digital Input/Output 1) and DIO2 pins on the cable were inverted.

Having identified the problem, it was also determined that the original GPIB cable was unavailable. There were two feasible solutions that could have worked. The first solution was to rewire the GPIB cable that was already being used. This solution was thought of but quickly turned down as the tools needed to make this change were not available for immediate use. The other solution, which ended up solving the problem was to connect to the CMW using Ethernet instead of using the GPIB cable. Following this route, a new network card was installed into the Lab PC and the network interface on the PC was statically set to match the same subnet as set on the CMW. The result was a working connection that allowed for seamless communication from the Lab PC to the CMW through the use of sending VISA commands using the pyvisa library.

5.1.2 Validating Successful Remote Commands

When creating automation software, validation must be included to ensure that the desired functionality is being achievable on the devices that are being configured. The proved to be a large problem on the CMW500. Part of the automation requirements was to be able to load

configuration files onto the CMW500. Loading the configuration file was a simple VISA command, but the problem was that there was no command that existed that could query what configuration file was currently on the CMW. Furthermore, there was no return message from the CMW500 after a config file was loaded indicating whether or not the load was successful.

So, a solution was needed to be determined that could somehow determine what file was loaded onto the CMW as well as to determine if a file load was successful or not. In order to solve this issue, the configuration files that were being loaded needed to be further inspected to determine what differentiates them. It was quickly evident that the differentiating factor between configuration files was the frequency at which the CMW500 was set to operate at. As a result, a parsing function was added to the automation platform that loads the list of configuration file names with their respective operating frequency into a dictionary object. Now, when a configuration file is loaded, or if the automation platform is interested in knowing what configuration file is currently loaded, it can query the operating frequency that the CMW500 is currently set at and match it with the corresponding dictionary entry.

5.2 Selecting a Test Case Input Format

When the requirements were given, a list of test cases was also emailed which contained 20 test cases. These 20 test cases needed to be stored in a configuration like file, so it could be fed into the automation platform and executed. Members of the Connected Vehicles Team had initially concluded that a text file was a sufficient configuration file type to store all the test cases. This idea was challenged because although Python has built-in functionality for parsing text files, it was identified that doing so would add an unnecessary level of complexity to the automation platform. As a result, the idea of using a CSV (Comma Separated Values) file was used instead because it solved the following foreseeable problems:

- In adding test cases to the test case configuration file, using a CSV file makes it easier to ensure that the tests that are being added are being added correctly; and
- When parsing csv files, Python has a built-in csv library that can extract parameters from a csv file at ease. Furthermore, it makes parsing and enforcing the configuration file easier. Instead of drawing extensive focus on the actual parsing functionality, more development and thought went towards creating validation functions that ensured each test case had valid parameters.

5.3 Establishing a Connection between the CMW and TCU

The last critical problem that occurred was an issue between the establishing a cellular connection between the CMW and TCU. On many occurrences, despite following the outlined procedures provided by the Connected Vehicles Team, there were times where the CMW would not establish a Packet-Switched State with the TCU. What this means is that despite following the proper procedures to establish a proper data path between the two devices, the data path would never successfully establish. This was a large problem because having a broken data path meant that you couldn't actually execute any throughput test because the two devices could not see each other. Initially, my thought process suggested that the CMW was the culprit as it had given prior issues regarding connectivity. However, rebooting the device proved to be of no solution. The coaxial cables connecting the two devices were also replaced with new ones to test if the physical connection was the reason for the broken pipe. This solution also failed to solve the problem.

As a result, a deeper consideration was made in considering the TCU as being the problem. Several different firmware versions were installed on the TCU to determine if perhaps it was a software issue. What didn't necessarily make sense is that rebooting the TCU should automatically turn the cellular antenna inside the TCU on and it should begin searching for a cellular network to connect to. But it didn't appear to be working or else the Packet Switched State on the CMW would be connected. To solve this problem, an effort was made to learn how to control some of the low-level functionality of the TCU. This functionality could not be controlled using the ADB interface, and so a separate serial communication cable was connected to the TCU's debugger port. Using Qualcomm's QXDM tool, which is a tool that allows you to debug their cellular hardware, the logs determined that a firmware fault was causing the cellular antenna to remain off. Luckily, a simple QXDM command forced the cellular antenna to turn on and finally the CMW Packet Switched State read the correct value. This functionality of turning the TCU's cellular antenna on and off was added into the automation platform as a result of this problem. That way in the future, if it happens again, the automation platform will use this technique when trying to establish a data path with the CMW.

6 Conclusions

From the analysis in the report body, it was concluded that the design decisions made helped realize a successful throughput test automation platform that met the requirements outlined in section 3.0.

Starting with a grasping a good idea of all the fundamental concepts needed to understand how throughput testing works, emphasis was shifted over to the Ford FNV eco-system and why throughput testing was necessary for the TCU. The need for throughput testing on the TCU is to ensure that further developments of the firmware being developed on the TCU is not causing a loss in the throughput performance. Continuous throughput testing, through the use of the automation platform allows for performance issues affecting throughput to be caught far in advance.

The throughput test automation platform that was realized was a result of carefully considering the requirements as determined by the Connected Vehicles Team at Ford. Furthermore, the hardware given served as a constraint, as alternative hardware options could not be considered. These requirements also helped focus on the end goal of producing the automation platform by setting a strict set of guidelines as to what components need to be included in the software design.

From the actual development standpoint of the throughput automation platform, a very incremental design approach was taken. This was to ensure that the software being designed was done so in a way that promoted good code design. Incremental design also helped to ensure that each requirement as indicated by the team was being met. Furthermore, breaking down the software development into smaller parts allowed for code similarities to be identified and later grouped together to create more concise code that was easier to understand. Important use of software design, as well as existing tools such as iPerf3 and QXDM helped realize a working throughput test automation platform.

Although a working throughput test automation platform now exists, the actual creation of the platform did not occur without some problems. Some of these problems were critical issues that required a prompt and effective resolution in order to proceed with development. These problems required careful though and consideration, as was as a fair deal of engineering analysis to solve. As a result, the thought process behind solving these problems has been documented.

7 Recommendations

Based on the analysis and conclusions in this report, it is recommended that the throughput test automation platform that was constructed should be implemented and used immediately. The platform has been evidently designed with a high level of care and it could save a lot of time in evaluating the throughput performance on the TCU. Running each test case that was specified in the test case document has previously taken up to an hour of a team members time each day, so that hour can now be put towards more useful work.

It should be noted that there is still room for improvement, and that the throughput automation platform is still far from perfect. One important feature that should be added that is recommended is a more robust error handler. The error handler that was designed for the current platform only covers the known errors that were presented over the course of the work term. There are certainly hardware handling errors that have been missed that need to be handled in order for reliable operation.

Another important feature that should be added in the future is a proper SMTP (Simple Mail Transfer Protocol) server. The reason for this server is so that email functionality can be included in the platform, which could find itself very useful for notifying the members of the Connected Vehicles Team the results of a successful throughput test session. The feature can also be used to notify team members if an error occurs in the test automation session and provide a valid stack trace with the error that caused a halt in test execution.

Lastly, an effort should be made to include more robust unit tests that have better code coverage than the simple unit tests that were created in the past four months. These unit tests can help prematurely catch bugs when further developing the automation platform. Due to the time constraints of the work term, there was not enough time to put an extensive effort into creating these unit tests, so an effort should be made to add this feature to the platform.

Glossary

CSV: Comma Separated Values.

DIO: Digital Input/Output.

ECU: Electronic Control Unit; an embedded system which coordinates communication between sensors and controls within the Ford vehicle.

FNV: Fully Networked Vehicle.

LTE: Long-Term Evolution; a cellular communication standard.

SMTP: Simple Mail Transfer Protocol.

TCU: Telematic Control Unit; an embedded system found in Ford vehicles that serves as a GPS and cellular modem.

WCDMA: Work-term report; a cellular communication standard.

Bibliography

[1] “Client-Server Model,” *en.wikipedia.org*, 2018. [Online]. Available:

https://en.wikipedia.org/wiki/Client%E2%80%93server_model [Accessed 08 – May – 2018]

[2] E. Gamma, *Design patterns: elements of reusable object-oriented software.* Boston, MA: Addison Wesley, 2008.

[3] “CAN bus,” *en.wikipedia.org*, 2018. [Online]. Available:

https://en.wikipedia.org/wiki/CAN_bus [Accessed 08 – May – 2018]