Report 3: Final Report

MAE 185

Professor Rajit Gadh

Joshua Noll and Mohona Sen

June 07, 2023

Topic of Project

Our project is about route optimization for Amazon's electric delivery vehicles (EDVs).

Overall System Design/Architecture

Our system will incorporate NFC and IOT to address two main goals. Firstly, it will minimize the distance required for an EDV's charging station detour from its delivery route. Secondly, it will minimize the charging time and wait time at the charging station stop.

Questions from Report 1 and Report 2

What is the problem that your project is trying to solve?

As the world transitions from gas to electric cars, shipping/delivery companies including Amazon, UPS, FedEx, USPS, and others must transition their delivery fleets accordingly, and we will specifically assess electric delivery vans, such as the Rivian vans shown in Figure 1. These vehicles require periodic charging, which must be factored into a driver's daily route.

Thus: How can we add a charging station stop to an Amazon electric delivery vehicle shipping route while maximizing energy and time efficiency (route optimization)? Even more specifically, how can we minimize the distance traveled for a charging station detour and minimize charging time and wait time at the station?

Figure 1. Amazon Prime delivery van designed and manufactured by Rivian.

Provide background information and literature review for the problem. Provide citations/references (very important) to your work both in the powerpoint and in the report.

Amazon has committed to having 100,000 electric delivery vehicles (EDVs) on the road

by 2030.[6] These were first rolled out in summer 2022, and 3,000 EDVs are on the road now delivering to over 500 cities already. Amazon's Rivian vans have already delivered 75 million packages within the US.

Amazon delivered about 21 million packages per day globally in 2021.[4] The US accounted for 13 million of these, and these numbers will continue to increase with the demand from online purchases. Customer satisfaction obtained from delivering on-time is an important priority, especially since failing to do so is an extra cost to Amazon; delivery charge refunds must be issued to customers whose packages don't arrive by the set date.[2] Especially with Amazon Prime services, this applies to day-of, one-day, or two-day expectations.

Although it varies and isn't exactly clear to us, drivers seem to currently have up to 8-10 hour shifts and choose when to stop for gas on their own accord.[5] Once they pick the cargo up from an Amazon warehouse, drivers scan a manifest, which has each piece of cargo listed under a certain zone.[3] This is done with Rabbit, Amazon's current route optimization app that is built from mapping technology called MapBox. Rabbit acts as a GPS system and an inventory log, factoring in many parameters: traffic, weather, construction zones, road conditions, GPS data, and time windows for deliveries.

What is your assessment on the way in which RFID/sensors can solve the problem you have selected?

NFC and IOT can be used to compute the most accurate delivery route for electric delivery vehicles. This would be done by using parameters that Rabbit already measures (which we will later mention) and combining those with new data taken from EV chargers and data from the driver's own vehicle to enhance the Rabbit algorithm. This IOT network would allow the Rabbit delivery app to determine the most optimal time and location to charge the EDV.

NFC can be used at the EV chargers themselves to augment and update charging queues for delivery drivers, which would increase the overall time efficiency of the charging process.

What do you plan to do for the problem in this project?

It is important to note that EDVs already have critical information such as energy consumption level, battery life, and driving range, whereas the Rabbit GPS can recognize a vehicle's position relative to a specific destination. The Rabbit app can already plan a route for the day, so based on the battery life at the beginning of each shift, we plan to make this route include a charging station stop based on when it predicts a low battery for the EDV. The Rabbit software would also be able to continuously enhance its prediction in real-time for when the battery level would be low. The driver can make an informed decision later in the day to either

stop at the initially suggested charging station or another nearby charging station with the shortest wait time (queue).

There will also be data that is taken and sent by the charging station. For example, if the charging station were at peak capacity or had a long queue of vehicles awaiting a charger, it would recommend the driver and the vehicle find an alternate charging station; or if there were vacancies at a station, suggest that one. It would also be necessary that the charging station be able to provide GPS positional data to the EDV as well to minimize the distance of the detour from the delivery route. Furthermore, Rabbit could recommend off-peak times or charging when drive times are the longest (e.g. rush hour) so that the driver can essentially sit out traffic. Implementing RFID readers that scan NFC devices could ensure that once drivers arrive at the charger, they can join the charging queue and also receive queue length data in advance. The NFC required is embedded into mobile devices already and can be tapped onto the reader.

What type of RFID or sensor technology (for example, active, passive or semi-active) do you believe would be best suitable for your application?

At the charging station where short range charging port authorization is necessary, active NFC in smartphones and a corresponding RFID reader that reads HF would be optimal. Additionally, electronic sensors such as voltage sensors are used in the car's battery to track battery level and could transmit data back to the Rabbit software.

What type of data does your application generate? Give examples (temperature, power, current, speed, weather, energy consumed, location, etc.).

From passive RFID used for charging authorization at the charging station, the data sent to Rabbit includes:

1. The order of the charging queue. Each NFC device has a unique identification number (UID)[20]. The RFID reader reads each NFC tag. The corresponding UID will be stored and all of the UID will be recorded in chronological order of taps.

2. The length of the charging queue. There will be a central reader for the entire charging station that the driver first visits -- with the driver's NFC tap, they join the queue, so Rabbit will display that the length of the queue is augmented by 1.

3. Location of the charging station. When the reader sends the queue length/order data to Rabbit, it also sends the location of the affiliated charging station back to the spatial database on the Rabbit app.

4. Time stamp of NFC tap during the central RFID reader and the NFC tap at the RFID reader individual charger. This data could be useful for long trend trend analysis, which we will later cover.

Note: Our system does not take into account the remaining charging time at a single charger, because this would require a person to input the time or battery level that they plan to charge to. This is unrealistic for the average non-delivery driver who does not plan their intended charging time.

Data generated within and sent to Rabbit from the EDV includes:

1. The battery level of the van. The battery management system (BMS) of a car already reads voltage sensor data measured via the voltage of the battery pack to determine the battery's state of charge[10]. The voltage sensors are placed across the battery terminals or across individual cells within the battery pack.

2. The car's driving range (also collected similarly with BMS).

3. The remaining miles the car has before the battery dies (also collected similarly with BMS).

As previously mentioned, Rabbit itself already uses certain data[3] to pre-determine the locations of chargers that are en route to delivery destinations. We are building on the model where Rabbit already accounts for:

1. Traffic, weather, construction zones, road conditions.

2. Driver's positional GPS data.

3. Locations of the delivery stops.

4. Time windows for deliveries.

What sort of analysis should you be doing with the data being generated?

Rabbit will analyze the data just mentioned in points 1 through 4 above. At the beginning of a driver's shift, Rabbit will factor in the starting battery level, the driving range, miles left before a dead battery, GPS data, traffic/weather predictions, and time windows between deliveries. Rabbit will thus evaluate a point along the delivery route that battery level would become low, say 20%. Since Rabbit knows the locations of the delivery stops, it will calculate the mileage between delivery stops to predict the best point to stop for charging based on the low-battery point. Thereafter, Rabbit would suggest a charging station en route to a delivery,

which would be an initial prediction at the beginning of the day. This suggestion would be prone to updates throughout the day based on data from the charging stations and any unexpected data from the EDV.

The (later mentioned) middleware will also be sent the charging queue order and length, so it can sum the number of currently stored NFC UIDs in the queue to send and display a queue length on the Rabbit app. The queue before this point is perceived as a list of UIDs recorded in the middleware in chronological order. Rabbit can use this queue length to recommend charging stations (that are likely different from the initial suggestion at the beginning of the day) with shorter queues in real-time throughout the day.

Is the analysis to be done in real time or can it be done asynchronously? Why?

Some data required for accurate charging times needs to be sent in real time and analyzed synchronously while other parts can be analyzed asynchronously. The data that is analyzed synchronously includes information such as traffic times, construction zones, and other data that Rabbit already gathers. This is because on a continual (second-to-second) basis, these factors will change and will affect the optimal charging period. For example, there may be accidents on the road and other delays that were not there the previous day that will affect delivery times and the time taken to reach a charger. Furthermore, the delivery zones to which a driver is assigned differ every day, so the route will also be inherently variable.

In addition to this, Rabbit must also continually receive charging queue lengths to make charging station recommendations to the driver. Although optimal charging locations can be estimated at the start of the day using long-term asynchronous data, these are subject to change. Additionally, the battery level of a vehicle also changes instantaneously and ongoingly throughout the day. If the battery data were only sent at the end of the day after the battery is dead, it would provide no practical use. The total possible range of a car's battery can also decline over time with use and wear.

Lastly, long-term trend analysis can be done asynchronously along with initial route planning. Initial route planning can be performed at the start of the day with address data which has been collected from consumers long before a shipment is ready and with existing charging station location data. Long-term, data collected by the EDV on a daily basis can help predict weekly, monthly, and yearly trends which could provide Amazon with important insights on what times of the week, month, and year are busiest. During these busy times, more frequent charging breaks may be necessary.

How should the overall system be designed/architected for simplicity of consumption and use of the data?

As mentioned, Rabbit is the central platform that will compile and analyze data. The driver will have the opportunity to follow the recommended charging stop during his day or go to an alternate station with a shorter queue or based on when he would like to take a different break.

A central reader that the driver first taps their mobile NFC onto to join the queue will be placed at the station. At the charging station, the Rabbit app will display the driver's position number in the queue. Through middleware, a central reader can transmit the queue onto a digital screen above it that displays a randomized number generated for the user -- like numbers displayed at a restaurant to show people the status of their order. This is necessary since non-delivery drivers don't have access to Rabbit.

There will also be an RFID reader connected to each individual charger so that the first driver in the queue can tap again to claim the open charger. The data for the NFC UID at the top of the queue will be transmitted from the central reader to the middleware. Hence, during full capacity, the next available charger will know what tag to expect next and authorize.

A timer within the middleware can enforce that if the first person in the queue does not claim the open charger within 5 minutes, the queue will assign the charger to the next spot in the queue.

Notes:

1. We are assuming that NFC on smartphones can be adopted as a uniform standard since everyone (including non-delivery drivers) waiting for a charger will need to join the queue.
2. We must account for the fact that non-delivery drivers won't have access to Rabbit at the station. The queue will be displayed on a digital screen above the central RFID reader, in the form of randomly generated numbers associated with the UIDs. The concept would be similar to a screen that displays orders that are ready for pickup at a restaurant. This way, regular people (non-delivery drivers) can see their position in the queue too.
3. We are no longer utilizing the driver's health data (i.e. blood sugar from biometric sensors)[1] in our design for privacy concerns and to avoid overcomplicating our proposal.

Where should the data reside (on the tag/hardware or the software)? Why?

Locally, certain data that is shared between the chargers and the central RFID reader only needs to be momentarily stored in the charger's software. The data needs to be stored from the

time the driver initially scans his or her NFC at the central reader to when they scan it again at the charger itself so that the queue remains intact. However, once the car leaves the charger, this data is no longer important on a local level. To retrieve this data in the short term, caching mechanisms such as in-memory caches in middleware/databases can be used for quick access and retrieval[11]. Software storage was chosen as opposed to hardware because it is more flexible and scalable. Software can be updated and modified easily and is cheaper than scaling hardware. The scalability factor is especially important because in the near future, EDVs and electric vehicles will dominate the road, meaning the demand for available chargers will increase drastically.

Despite the data only being needed at the local level for a short period of time, charging data from the charger is an analytic needed by the Rabbit software continually. This is because this data is used to optimally compute the most efficient charging period and notify drivers of wait times at chargers. Because of this, on a non-local level, the data still needs to be analyzed. Therefore, the most efficient way to store the data for our purpose would be to store it in cloud-based software. By using cloud-based software, the data can be analyzed with cloud computing and shared with all the drivers on the road[12] while also being used by Amazon for long-term trend analysis.

Questions for Report 3

1. What is the current RFID/sensor/IOT technology that is being used in the domain of your problem (frequency, protocol, standard)?

Different charging station companies may use different methods, but we will use ChargePoint technology as an example, considering that it is one of the largest EV charging companies that has over 174,000 stations globally.[17] Their charging stations use NFC on mobile devices (both Apple and Android) that works in coalition with their ChargePoint mobile app for payment. The NFC is used for tap-to-charge against an RFID reader located on the charger[19]. This app also allows users to join a waitlist for nearby chargers if the charging stations are at full capacity,[20] which is essentially like our queue suggestion, but limited to ChargePoint users. It also notifies users when their car is finished charging. This NFC operates at 13.56 MHz (High Frequency), transmits data at a rate of up to 424 kbit/second, and works when the 2 devices are within a 10 centimeter range of each other.[28]

The Tesla's built-in tablet dashboard has a map that displays available charging stations nearby, and display the amount of "available stalls" (open chargers) at each station. It can also add a stop to a Tesla charging station that is on the user's route through the map's navigation. Once the charger is connected to the car, the driver's payment method on file is automatically

billed. The Tesla app notifies the driver once their vehicle is nearly fully charged.[30] However, this process is only available to Tesla owners that have the built-in dashboard navigation, the Tesla app, and thus are integrated into the Tesla ecosystem, so non-Tesla vehicles like a Rivian Amazon van cannot access it. There is no way to join a queue once a user is physically at a Tesla charging station.

Though it has been difficult to obtain consistent findings regarding how or whether Rabbit is still used by Amazon drivers, or whether the Amazon Flex app is the new standard, or whether Rabbit is a part of Amazon Flex, we assume that we can retrofit the Rabbit model based on our original source. As previously explained, Rabbit is the mobile app interface on which an Amazon delivery driver can scan their manifest (which contains information for what delivery zones the packages must be delivered to), and accordingly optimizes a route for the driver, building on a platform and data from MapTech.

As far as standards, two major specifications exist for NFC technology: ISO/IEC 14443 and ISO/IEC 18000-e. The first standard pertains to ID cards used to store information while the latter pertains to RFID communication between NFC devices[35].

2. What hardware did you use including sensing hardware, readers, tags, printers, other devices?

We have shifted our proposal from RFID tags to NFC, and will describe design alterations accordingly. For our IOT network, high frequency RFID readers are mounted on the chargers and compatible high frequency, short range NFC tags are used on mobile devices that the EDV drivers have. NFC is already implemented in modern smartphones[15] and in the United States, over 85% of adults own smartphones[31]. We have also verified that NFC does not rely on an internet connection, so it should still work in remote areas where charging stations may be located.[23]

EDV drivers will already be using their phones anyways for Rabbit and navigation throughout the day. We do recognize that though the use of a mobile device is convenient for most people, this may not be the case for everyone -- for example, people without smartphones or people whose smartphones are low battery or out of battery. Under such considerations, we suggest that Amazon can later choose to allow drivers to opt to have a backup RFID key in the form of a key fob (as we originally proposed) that they can store in their vehicles, and that is compatible with the readers. This proposal initially came about in regards to the tags being portable in the form of key fobs[7] and being easily tapped onto a reader.[8] However, we will not elaborate on this since it is not our main design anymore.

Following the general procedure described in previous reports and presentations, once the driver reaches a charging station, he or she scans his or her phone at the central NFC reader to join the charging queue. Later, once a charger is made available to the next person in the charging queue, the driver will scan their phone's NFC tag a second time at the corresponding charger so that it can confirm that the correct person who holds the first spot in the queue is at the charging station. The driver's position in the queue can still be updated in real time and once the driver is at his or her designated charger, NFC can be used to authorize that the correct person has arrived and even provide real time charging status.

The models of the necessary hardware are described below:

1. The model of the NFC chip shall be whatever is implemented in smartphones. Apple and Android are very private about the specific chips used, so we made later specifications based on the A8 chip.
2. We will not specify a model for the RFID key fob tag as this is optional for Amazon.
3. The model of RFID reader mounted onto EV chargers shall be the IDEAS WAVE ID® Solo Keystroke V2 LEGIC® HF RFID Reader,[16] which operates at 13.56 MHz. This is a suggestion for a reader to mount on a charger assuming that the charger does not already have a reader attached to it, as ChargePoint does.
4. The model of the digital screen to mount on the central RFID reader shall be the P6 Outdoor Full Color 40" x 18" Scrolling LED Display with High Resolution High Brightness[14] that can be purchased on Amazon.

Figure 2. Images of charging station hardware components, left to right: NFC Chip, optional RFID key fob tag, IDEAS WAVE ID® Solo Keystroke V2 LEGIC® HF RFID Reader, P6 Outdoor Full Color 40" x 18" Scrolling LED Display with High Resolution High Brightness.

3. Is there a role for middleware (software) in the application? In the context of the middleware features taught in the class, which specific feature is most relevant to your application and why?

There is a role for middleware in the form of software in this application. For an electric delivery vehicle charging station, our middleware takes the form of software and acts as the platform for the flow of data. It also reduces the amount of data processing that must occur on the mobile device where Rabbit is installed.

The process of data transmission through middleware at the charging station is outlined in the steps below. Figures 3 and 4 can also be used for reference.

1. The driver conducts the first tap authorization with their NFC device on the central RFID reader.
2. The NFC chip UID is sent to the middleware software.
3. The middleware adds the NFC UID to the bottom of the charging queue and augments the length of the queue by one to update the queue length.
4. The middleware can also assign the UID a randomly generated number (i.e. #01 through #99) to display on the digital screen attached to the central RFID reader. This way, the driver's privacy is protected and the middleware can transit the order of random numbers as a queue display on the digital screen, as we described before as a similar process to receiving and monitoring an order number at a restaurant.
5. The middleware sends data to Rabbit on the driver's phone so that their position number in the queue appears. At this point Rabbit will also display the driver's random number on the mobile app. Middleware will therefore integrate Rabbit to the charging station cloud network.
6. The middleware notifies every charger of the UID at the top of the queue so that if any charger opens up, it knows which NFC UID to expect to scan next.
7. When the first vehicle in the queue arrives at the first available charger, it will appropriately be authorized through the NFC tap method again. Any available charger can have its own display screen to show that it is "available," meaning vacant.
8. Middleware sends time stamps of taps and instantaneous queue lengths throughout the day to a database (on the cloud) for long term trend analyses such as the busiest time of the day and other useful statistics.

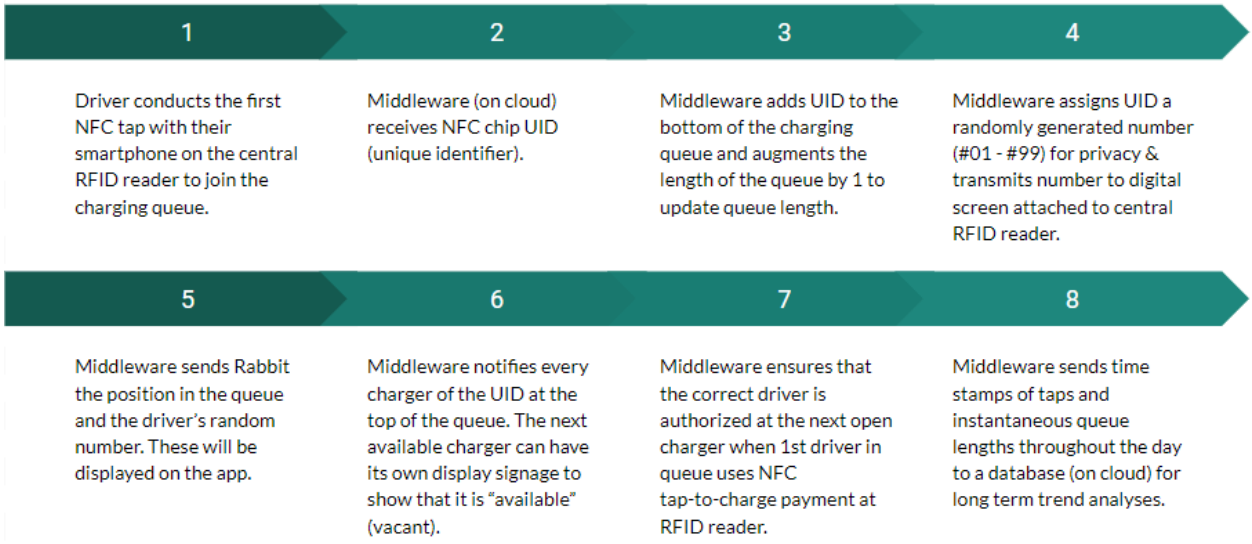

Figure 3. Process of data transfer at the charging station through middleware.

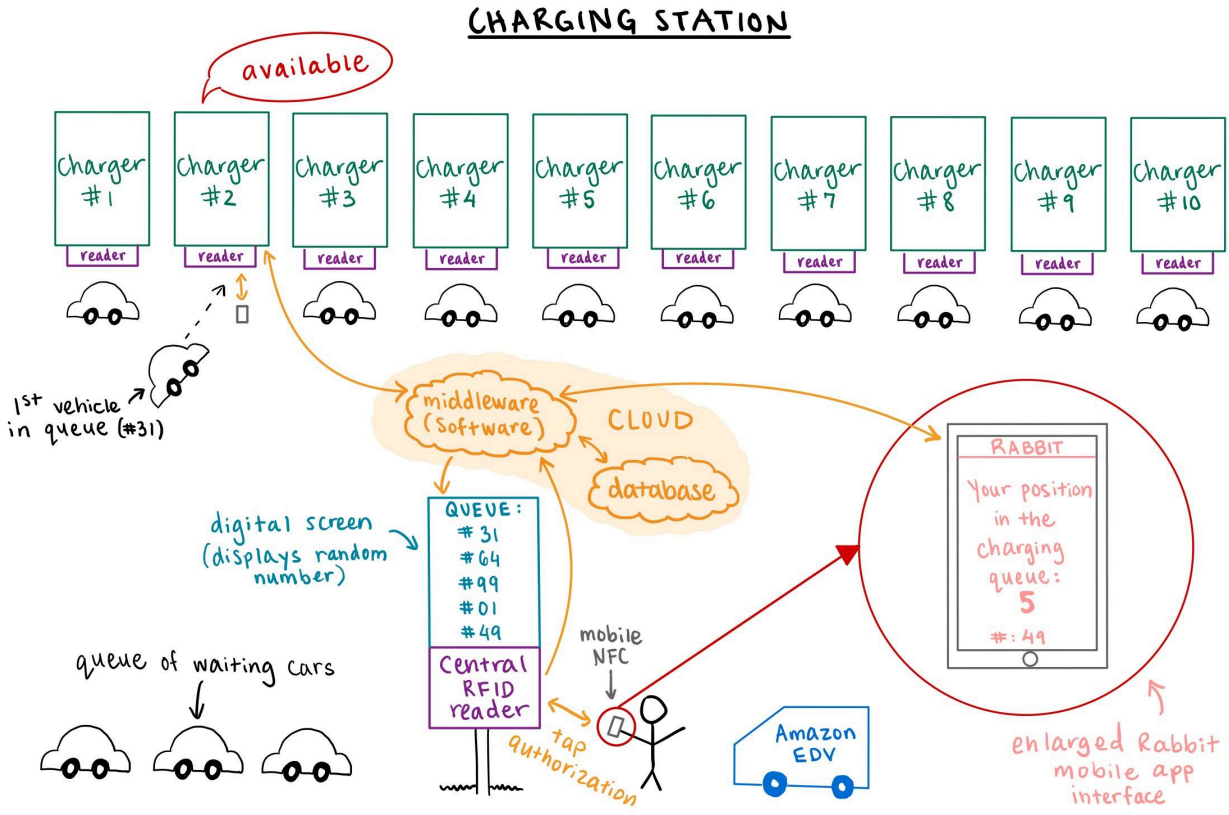

Figure 4. Schematic of charging station design and operation, demonstrating the flow of data through orange arrows.

4. Is the EPC data standard (or any other data standard) relevant for your application? What data format do you recommend for the tag/device memory? Is there enough memory in your selected tags/devices? If not, what do you propose to do and why?

No, EPC is not used in NFC. EPC is used on UHF, not HF, so the EPC data standard is not relevant for our model[27]. The only data standards we need to consider are the previously mentioned NFC data standards such as ISO/IEC 14443 and ISO/IEC 18000-e [18].

The NFC chip within the smartphone does not need to have significant memory -- for example, the TSMC A8 chip that was used in older iPhone 6 and 6 Plus models and had NFC enabled had a 64-bit memory[24]. The chip will simply only be activated in real-time during taps at the charging station, and does not need to store any additional data for future charging purposes. Apple and Android keep their exact NFC chip models quite private, so it is difficult to gauge whether the demand for memory will need to be higher than 64-bit because our research could not find a specific chip model in use today, but whatever technology is already built into iPhones and Androids today will suffice, as tap-to-charge is already in use. When the NFC tag in the smartphone is scanned at the reader at the charging station, the NFC UID is stored within 64 bits of memory (which corresponds to the ISO standard for NFC tags). There is already enough memory for this application because cloud storage on our database and middleware offers a large amount of space for any other data we may need to store that is not on Rabbit.

5. Are there networking or wireless considerations in your problem? How do they impact the performance of the system?

There are wireless considerations throughout the context of our IOT solution. Locally, NFC will still function at the charging station wirelessly and without an internet connection. However, the middleware will likely require an internet connection to communicate with Rabbit and update the driver about their queue status. This can be addressed by setting up a WiFi network at each charging station.

Furthermore, in order for drivers that are still on the road to receive some types of real-time GPS and charging station data (e.g. traffic updates, charging availability, queue times, and proximity to charging station), drivers and the charging station need to have a wireless cellular connection[[24]. This goes back to an earlier component of our solution that updates that Rabbit app in real-time to update the most efficient delivery and charging route. A solution for this is ensuring that drivers have cellular data activated on their phones while they are driving.

6. Does the tag or hardware device need to be secure? Why or why not?

The tag and hardware must be secure, but the inherent design and use of NFC hardware in this context is already secure because the 2 communicating devices (tag and reader) must be within a very close range of 10 centimeters, so it is difficult to obstruct this flow of data at the central NFC reader and at each individual charger's reader.

The individual's NFC authorization may contain other information, such as debit/credit card number, that should not be compromised when the second NFC tap occurs at the EV charger itself, when the driver will likely have to pay for the charging. However, we are confident that today's smartphones already have sufficient security measures and encryption incorporated due to the success of Apple Pay and Google Pay. Also, in case any users care about the privacy of their NFC UIDs, the UIDs are not displayed in public at the charging station -- to reiterate, we randomly generate a number that represents the UID.

7. Provide an estimate for the cost of your system(s)? How did you come up with the estimate?

Our cost calculations are below:

<u>Cost for one charging station</u>
# of readers per station = (1 reader/charger) * 10 chargers + 1 central reader = 11 readers
Up front reader cost[16] = ($237/reader) * (11 readers/station) = $2,607/station
Cost of digital sign attached to central reader[14] = $486
Total hardware cost = $2,607 + $486 = $3,093
Ongoing WiFi cost: $20/month
Cost for one charging station = $3,093 + $20/month

<u>Cost for all charging stations in the U.S.</u>
# of charging stations (assuming 10 chargers/station) = 109,307 chargers * (1 station/10 chargers) = approximately 10,931 stations
Up front cost = $3,093/station * 10,931 stations = $33,809,583
Ongoing WiFi cost = $20/month * 10931 stations = $218,620/month
Total cost for all charging stations = **$33,809,583 + $218,620/month**

<u>Cost for Amazon</u>
Rabbit App upgrades[32] = $500,000
Cost estimate of middleware upgrades[29] (new development): $1,000,000
Total cost for Amazon = $500,000 + $1,000,000 = **$1,500,000**

<u>Cost for all entities involved</u>
Cost for all charging stations + cost for Amazon
= $33,809,583 + $218,620/month + $1,500,000 = **$35,309,583 + $218,620/month**

This cost estimation centers around the implementation of an IOT electric vehicle charging network in the United States which has 109,307 chargers[34]. This estimation takes into consideration upgrading existing charging stations and therefore does not account for the cost of the EV charger itself. If we estimate that there are approximately 10 chargers per charging station, this gives us roughly 10,931 charging stations in the U.S.

Firstly, the software that Amazon delivery drivers currently use, Rabbit, needs to be updated so that it can communicate with charging stations and receive real time data. Rabbit falls under the category of an enterprise app, which is an application that is used by large organizations or enterprises. Between development, quality assurance, project management, UI and UX features, and business analytics, these apps can cost anywhere between $50,000 to $500,000 dollars. For a company like Amazon, which operates on a large-scale in the delivery market, we take a conservative estimate of $500,000 for app upgrades.

Next, existing charging stations need to be outfitted with NFC readers that can interact with drivers' smartphones and the middleware on it. A durable RFID reader mounted on the EV charger stall outdoors, and hence can resist the elements, will cost about $237. However, it is important to remember that these chargers are not owned by the same companies. In fact, Amazon does not own any of these.

In summary, it is expected that Amazon would need to pay $1.5 million for software implementation -- the Rabbit app updates and middleware development. Meanwhile other entities, such as the charging station companies, would have to pay about $33.8 million dollars for hardware implementation, plus an ongoing $218,620/month for WiFi. For all companies involved, the total cost of implementation is $35,309,583 + $218,620/month. This is assuming that charging stations don't currently have WiFi.

8. Perform a high-level cost-benefit analysis for your proposed system. If there is information that you don't have, make reasonable assumptions, but clearly state them.

The overhead (up front) cost for Amazon would be $1,500,000, but in the long-run, Amazon would save money due to the efficiency it gains from this IOT network. By implementing this system, delivery drivers will not have to stop for gas at random times or charge their electric vehicle at non-optimal times.

Because the EV market is only going to increase in the coming decades, it will become increasingly common for Amazon Flex drivers to be driving electric vehicles. If Amazon delivery drivers charged their vehicles for 30 minutes each day at an optimal time and place, each driver would be able to deliver more packages each day rather than wait in long charging lines or go to a charger that is far away from the delivery stop. In 2021, the average LA driver spent 10.2 minutes in traffic per day[26]. Since charging station infrastructure is not complete

(hence many charger stations have no queue), there is no estimate available for the average wait time at charging stations, yet from personal experience and our projections, we can assume that a driver may spend up to 20 minutes waiting for an open charger. Therefore, we project that our proposal can save an EDV driver in a metropolitan area about 10 minutes + 20 minutes = 30 minutes of delivery time, which equates to about 4 additional stops per shift. We extrapolate about 80,000 Amazon Flex drivers working each day[22] since there were 140,000 Amazon Flex drivers in 2021 and we assume a four-day work day rather than a five-day work week because they are contract drivers. In the United States, that would amount to about 320,000 more packages being delivered each day.

To find the profit of each package delivered in the United States, we divided Amazon's annual gross profit (not the same as revenue) in 2021 by deliveries by the total number of packages delivered in the United States. This equates to a profit of $1.22 per package.[33] This means that Amazon has the capability to process $390,720 worth of packages each day. ($1.22/package * 320,000 packages = $390,720).

Thus, in order to recuperate the initial investment not including the hardware, it would take Amazon 3.84 days (Amazon's initial investment divided by the daily profit from implementing IOT network = 3.84 days). This would make the first month ROI of this investment an astounding 7.808. If we do a quick feasibility check, one would presume that this number is too high. However, this does not consider the hardware costs of the system, which is the most expensive portion. It was originally not included because Amazon does not own these EV chargers and the hardware implementation cost would fall on the EV charging companies.

If we take an ROI for all involved entities, and thus consider the hardware costs, we reach a first-month ROI value of 0.33. This number seems more feasible, and within 4 months, the IOT network would be profitable.

9. Is there a need for a database in your system? If so, what type of data would you store there? Explain.

There is a need for a database in our system. Because long-term trend analysis needs to be performed, for example, analyzing the average charging queue length at different times through the day, the best solution would be to store the data in the cloud so that it can be accessed on any device that is connected to the internet. From the middleware, the database would receive data including **tag IDs**, the **length of the queue**, and **time stamps** for when tag IDs are recorded when the NFC tap authorizations occur. This would also provide insight into the average wait time a given vehicle has in the queue before its position in the queue moves up.

Lessons Learned

We have learned how IOT as a whole can function, relating sensors, RFID tags, RFID readers, NFC devices, middleware, databases, and other forms of hardware. Understanding the relationships between these has allowed us to generate a concept design for a system. We have also learned to iterate upon our project proposal, doing away with using biometric data and replacing RFID tags with mobile NFC. We have gained a new understanding of middleware in particular, and we had realized that it was a missing link in how our data outlined in Report 2 was transmitted. Middleware software became our central cloud component facilitating all of our data transfer. It was also interesting to gradually narrow our focus to one main delivery company (Amazon), then one main vehicle type (van similar to Rivian model), and narrowing down the specificity of our problem statement.

It was also interesting tying together some ideas and inspiration from pre-existing technology that companies like Amazon, Tesla, ChargePoint, Apple, and Google already use to create one large IOT system. Though Tesla and ChargePoint both offer charging technologies, we had to draw from their designs to create something more universally accessible to drivers outside of the Tesla and ChargePoint ecosystems -- but rather to drivers within the Amazon Flex/Rabbit ecosystem.

In retrospect, we suppose it was useful to pick a company as large and wealthy as Amazon; although our cost-benefit analysis showed that it remains a bit ambiguous as for whether Amazon will pay for the hardware as well as the software, we think that our project proposal is realistic and implementable for Amazon regardless.

Works Cited

All Reports

[1] Amazon Delivery Drivers Forced to Sign 'Biometric Consent' Form or Lose Job.” *VICE*, 23 Mar. 2021,

https://www.vice.com/en/article/dy8n3j/amazon-delivery-drivers-forced-to-sign-biometric-consent-form-or-lose-job

[2] “Guaranteed Shipping Speeds and Delivery Charges.” *Guaranteed Shipping Speeds and Delivery Charges - AmazonCustomer Service*,

https://www.amazon.in/gp/help/customer/display.html?nodeId=GY6FPCFZJMGBG3LV#:~:text=Amazon%20automatically%20refunds%20delivery%20charges,you%20get%20a%20delivery%20refund

[3] “Home.” *Manufacturing Logistics IT Magazine RSS*,

https://www.logisticsit.com/articles/2020/12/29/how-amazon-manages-its-delivery-routes-(and-how-to-copy-them).

[4] “How Many Amazon Packages Get Delivered Each Year?” *How Many Amazon Packages Get Delivered Each Year? | Supply Chain Transportation and Logistics Center at the University of Washington*, 17 Oct. 2022,

https://depts.washington.edu/sctlctr/news-events/in-the-news/how-many-amazon-packages-get-delivered-each-year

[5] *Questions and Answers about Amazon DSP Working Hours - Indeed*.

https://www.indeed.com/cmp/Amazon-Dsp/faq/working-hours.

[6] Staff, Amazon. “Amazon Now Has over 3,000 Custom Electric Vans Delivering to Customers across the U.S.” *US About Amazon*, US About Amazon, 30 Mar. 2023,

https://www.aboutamazon.com/news/transportation/everything-you-need-to-know-about-amazons-electric-delivery-vans-from-rivian#:~:text=With%20its%20commitment%20to%20have,to%20decarbonize%20its%20transportation%20network

[7] “RFID EV Mobility.” *IDTRONIC Professional RFID*, 20 Sept. 2022,

https://idtronic-rfid.com/en/rfid-applications/rfid-ev-mobility/

[8] Tech, Nexqo. “How's RFID Card Used in Car Charging?” *Nexqo*, 8 Mar. 2022,

https://nexqo.com/2022/03/hows-rfid-card-used-in-car-charging/

[9] Writer, Staff. “RFID Readers for EV Chargers: Six Questions to Ask.” *Kiosk Kiosks*, 10 Feb. 2022,

https://kioskindustry.org/rfid-readers-for-ev-chargers-six-questions-to-ask/

Report 1 and Report 2

[10] Bakshi, S. (2021, May 9). What Is a BMS (Battery Management System) and How Does It Work? MakeUseOf.

https://www.makeuseof.com/what-is-a-bms-battery-management-system-and-how-does-it-work/.

[11] GeeksforGeeks. (2021, June 14). What is the Caching Mechanism? Retrieved from

https://www.geeksforgeeks.org/what-is-the-caching-mechanism/

[12] IBM. (n.d.). Cloud computing. Retrieved May 14, 2023, from

https://www.ibm.com/topics/cloud-computing

[13] Metro State University. (n.d.). Where can I find my student ID number? Retrieved May 14, 2023, from

https://services.metrostate.edu/TDClient/1839/Portal/KB/ArticleDet?ID=47214#:~:text=The%20RFID%20number%20is%20located,called%20out%20in%20red%20numbers.

Report 3

[14] Amazon. (n.d.). Outdoor Scrolling LED Sign, 3'x6', Full Color. Retrieved from

https://www.amazon.com/Outdoor-Scrolling-Resolution-Brightness-Advertising/dp/B0BPP8XXS7?source=ps-sl-shoppingads-lpcontext&ref_=fplfs&psc=1&smid=A3EZ7OPZE6ZS8Q

[15] Apple Inc. (n.d.). NFC. In Human Interface Guidelines. Retrieved from

https://developer.apple.com/design/human-interface-guidelines/nfc

[16] Atlas RFID Store. (n.d.). RF Ideas Wave ID Solo Keystroke V2 LEGIC HF RFID Reader. Retrieved from

https://www.atlasrfidstore.com/rf-ideas-wave-id-solo-keystroke-v2-legic-hf-rfid-reader/?utm_device=c&utm_feeditemid=&utm_term=&utm_source=google&utm_medium=cpc&utm_campaign=03-RFID%20Readers-High&hsa_cam=13144491769&hsa_grp=122618820939&hsa_mt=&hsa_src=g&hsa_ad=522205010796&hsa_acc=4442410237&hsa_net=adwords&hsa_kw=&hsa_tgt=aud-652141730773:pla-425925387354&hsa_ver=3&gclid=CjwKCAjwsvujBhAXEiwA_UXnANGsmofr-PR0d2KgP752Z84jnHl8AUwZthwQj7IRtQgZUzitzwsRmBoCvf4QAvD_BwE

[17] ChargeLab. (n.d.) The 5 biggest electric vehicle charging companies. Retrieved from

https://www.chargelab.co/blog/biggest-electric-vehicle-charging-companies

[18] Charge Northwest. (n.d.). CT4000 Data Sheet (Publication No. 73-001020-01 Rev 2.0). Retrieved from

https://www.chargenw.com/documents/CT4000_Data_Sheet%2073-001020-01_Rev_2.0_.pdf

[19] ChargePoint. (n.d.). Setting and Using Tap Charge. Retrieved from

https://www.chargepoint.com/resources/setting-and-using-tap-charge

[20] ChargePoint. (n.d.). Using Waitlist. Retrieved from

https://www.chargepoint.com/resources/using-waitlist

[21] GoToTags. (n.d.). NFC Chip Features - UID. Retrieved from

https://learn.gototags.com/nfc/chip/features/uid

[22] Federal Trade Commission. (n.d.). Refunds for Amazon Flex Drivers. Retrieved from

https://www.ftc.gov/enforcement/refunds/refunds-amazon-flex-drivers

[23] Nomtek. (n.d.). What are NFC tags? Retrieved from

https://www.nomtek.com/blog/what-are-nfc-tags#:~:text=No%20need%20for%20network%20connectivity,re%20disconnected%20from%20the%20internet.

[24] Notebookcheck. (n.d.). Apple A8 SoC. Retrieved from

https://www.notebookcheck.net/Apple-A8-SoC.127992.0.html

[25] onX Hunt. (n.d.). Does GPS work without data? Retrieved from

https://www.onxmaps.com/hunt/blog/does-gps-work-without-data#:~:text=GPS%20tracking%20on%20your%20phone,even%20when%20you%20are%20offline.

[26] Patch. (n.d.). Angelenos Spend Eye-Popping Number of Hours in Traffic: Study. Retrieved from

https://patch.com/california/los-angeles/angelenos-spend-eye-popping-number-hours-traffic-study

[27] RFID Journal. (n.d.). Does an NFC tag have a unique EPC identifier? Retrieved from

https://www.rfidjournal.com/question/does-an-nfc-tag-have-a-unique-epc-identifier#:~:text=Near%20Field%20Communication%20(NFC)%20tags,when%20the%20tag%20is%20interrogated.

[28] Rohde & Schwarz. (n.d.). NFC White Paper (Publication No. 1MA182_5E). Retrieved from

https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_application/application_notes/1ma182/1MA182_5E_NFC_WHITE_PAPER.pdf

[29] ScienceSoft. (n.d.). Software Development Costs: Overview and Calculation. Retrieved from

https://www.scnsoft.com/software-development/costs

[30] Tesla. (n.d.). Supercharging [Video]. Retrieved from

https://www.tesla.com/support/videos/watch/supercharging

[31] U.S. Census Bureau. (2021, July 1). New Census Bureau report analyzes computer and internet use in the United States. Retrieved September 17, 2021, from

https://www.census.gov/newsroom/press-releases/2021/computer-internet-use.html

[32] Velvatech. (n.d.). How Much Does a Mobile App Cost in 2023? Retrieved from

https://www.velvetech.com/blog/how-much-mobile-app-cost/

[33] Visual Capitalist. (n.d.). How Amazon Makes Its Money. Retrieved from

https://www.visualcapitalist.com/how-amazon-makes-its-money/

[34] EV Adoption. (n.d.). Charging Stations by State. Retrieved June 10, 2023, from https://evadoption.com/ev-charging-stations-statistics/charging-stations-by-state/

[35] NearFieldCommunication.org. (n.d.). Near Field Communication Technology. Retrieved June 10, 2023, from http://nearfieldcommunication.org/technology.html