CISC 322/326 Assignment 1
Conceptual Architecture of Apollo

February 20th, 2022

Group #7: ArchiTiger
Poppy Li 20181706 19xl12@queensu.ca
Xuan Xiong 20147035 18xx15@queensu.ca
Yuen Zhou 20186821 19yz57@queensu.ca
Yingjie Gong 20144264 18yg24@queensu.ca
Zhimu Wang 20190758 19zw28@queensu.ca
Baisheng Zhang 20094496 17bz15@queensu.ca

Table of Contents

1.0 Abstract

This report aims to investigate and define the conceptual architecture for Apollo, an open-source platform for AD cars. This report inspects Apollo's architecture from several aspects, including functionality, concurrency, control, and data flow and subsystems. Through careful analysis and examination of developers' documentation and other public resources, Apollo's high-level conceptual architecture is determined to be Publish-Subscribe Style with a combination of slightly Layered style. We derive this conclusion by exploring the underlying subsystems that work together to enable vehicles to interact with the surroundings and achieve AD. The whole system can be broken down into Perception, Prediction, Planning, Control, Guardian, HD Map, Localization, CANBus, Monitor, and HMI modules. Each module can publish or subscribe to specific topics. The paper provides in-depth descriptions of each module's functionality, their dependencies on each other, and how all these modules function as a whole. Diagrams are created to illustrate the communications among them further.

Apart from the architecture itself, we also present and explain some use cases sequence diagrams to demonstrate how they utilize parts of the architecture. For instance, one essential use case is that the user enters a destination either by typing or by speech recognition. Then the car takes control of everything without any user interference. In addition, we list several external interfaces that show what type of information is transmitted within the system.

AD is a highly complex topic. One of the biggest challenges is to make the car think like humans because driving requires social interactions. Analyzing the current architecture helps us understand what a modern technology-related project may look like. Overall, this paper determines the conceptual architecture of Apollo, and it will be used as the reference for defining Apollo's concrete architecture in the future.

2.0 Introduction and Overview

The automotive industry has experienced a revolution since the 21st century that cars become more and more intelligent, more interactive with drivers, and more electrified. In recent years, with the potential help of 5G, humans have emphasized achieving driver-less and autonomous vehicles. For instance, Tesla develops an Autopilot system that enables autonomous hands-free control. The idea of AD is not innovative at all. Leonardo Da Vinci first proposed and designed in the 1500s ("History of Autonomous Cars", 2022), before the first car was invented. Due to technology limitations and other constraints, it was still not fully realized even after nearly five centuries.

The Apollo project was initiated in 2017 by Baidu, a Chinese technology company specializing in search engine services and artificial intelligence, to help every participant in the game quickly build their autonomous system. Apollo's name was borrowed from NASA's Apollo human spaceflight program since fully accomplishing a self-driving car is as complex as landing on the moon but would reform the world entirely. It is an entirely open-source project that provides software and hardware support for everyone in the industry when implementing their intelligent vehicles ("Baidu Announces Project Apollo", 2017). The existence of Apollo not only accelerates the development process of intelligent vehicles but also builds a collaborative environment that every interested individual could contribute to the project.

Within less than five years, Apollo has gone through 6 essential iterations. Unlike other giants in the industry who have strict control and high security over their technology. Baidu is willing to use its strength in artificial intelligence to work with the general public. From achieving AD at the closed venue in Apollo 1.0 to driver-less driving in Apollo 6.0, it shares selflessly all its source code and detailed documentation on each update and how the user can quickly start using Apollo on their Github page. Baidu has been putting tons of effort into encouraging significant innovations and pushing the automotive industry forward.

This report presents a transparent derivation process of how our team determines the architecture style of the Apollo system based on the knowledge we gained from the course. Architecture analysis is split into six sub-sections, subsystem breakdown, interactions between subsystems, use cases and functionality, evolution, control, data flow, concurrency. We have concluded that Apollo's architecture is Publish-Subscribe Style with a combination of slightly Layered styles. This conclusion will be examined repeatedly in each of the subsections indicated above. Lastly, lessons we learned through the project will be discussed, and a conclusion is put at the end to give an overall summary of our findings.

3.0 Architecture analysis

3.0.1 Subsystem Breakdown

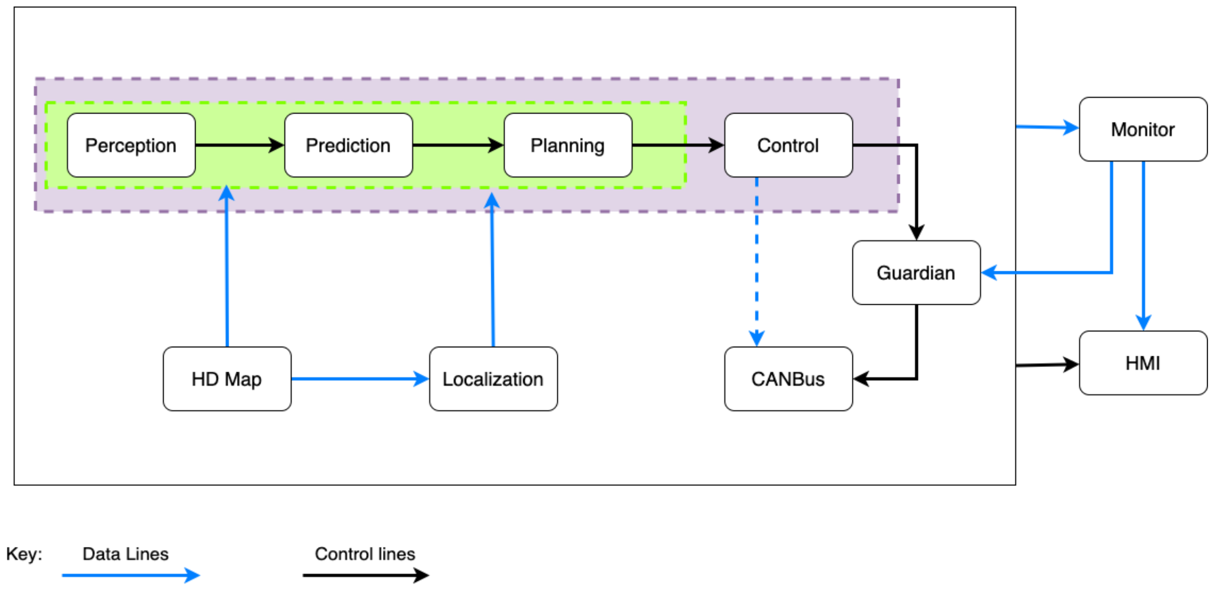

Figure 1. Software Architecture of Apollo System

The Apollo self-driving system can be divided into several modules or subsystems.

Perception: It is used by the system to detect surroundings such as the traffic light, other vehicles, and obstacles on the road. The perception module is one of the most critical parts of the AD system. It is considered to be the "eye" of the system. In other words, no further actions can be activated without this module.

Prediction: As its name implies, the Prediction module predicts perceived obstacles' possible future motion. For instance, it would anticipate if a car on the other lane would switch lanes by observing certain features.

Planning: This module is responsible for making plans for the autonomous vehicle. For instance, overtaking the car in front of the autonomous car,

Control: The Control module performs the plans already set by generating commands to the car, such as accelerating, turning the right turn signal on, and so on.
Guardian: It is a safety module that plays the role of an Action Center, and it should intervene once the Monitor detects a failure.
CANBus: This module is the interface in charge of passing commands generated in the Control module to the vehicle hardware system.
HD-Map: The full name is a High-definition map. This module acts as a library in the system that provides ad-hoc structured information regarding the roads. It allows the AD vehicle to analyze the surrounding in real-time.
Localization: This module utilizes various information sources such as GPS to estimate where the vehicle is.
Monitor: This module monitors all the modules in the vehicle, including hardware. In addition, it also receives data from a bunch of modules and sends the data to HMI for the driver to view vehicle status.
HMI: is a user interface that establishes a connection between the machine and the user. It can view the vehicle status, control other modules, and boost the user's experience.
Routing: When a user enters a destination, the Routing module gives several ways of reaching the destination.

3.0.2 Subsystem Interaction

All the modules indicated above work closely to make the whole system function. The perception module perceives surrounding environments with the help of cameras and radars. It returns information about traffic lights, other traffic participants, and obstacles on the roads. And then, it sends the data to the prediction module, which subscribes to localization, planning, and perception obstacle messages. When the localization module transmits results to the prediction module, it updates the inner stored status and is triggered when it gets obstacle data from the perception module. HD Map module does not subscribe or publish anything. It is in the system to help provide valuable roads information.

The following important module is the planning module, as it nearly interacts with all other modules. It takes the output from the prediction module about the surrounding obstacles. However, the prediction module does not return traffic light information. Thus, the planning module subscribes to the traffic light detection output to get complete road information. After that, it takes routing module output to make reasonable plans. If the current route cannot be fulfilled due to unavoidable reasons, the plan module may request the route to compute for another feasible route. Lastly, it gets the current vehicle location from the localization module.

The control module then takes the output from the planning module and then generates corresponding commands. These commands will then be passed to CANbus, which manages the transmission to the hardware system. A guardian module takes actions based on the module's status between the control module and the CANbus. If all modules are working correctly, data flow remains the same as if the guardian were absent. By contrast, if one of the modules fails, the guardian module prevents commands transmission from the control module to the CANbus and brings the car to a stop.

Last but not the least, the monitor module acts as a supervisor for all the modules and the hardware. It subscribes to all the module's events and then passes received data to HMI. HMI visualizes modules' output, such as planned route, for the user. These two modules have a layered relationship that monitor provides services(software and hardware status) to HMI.

3.0.3 Use cases

Use case 1:

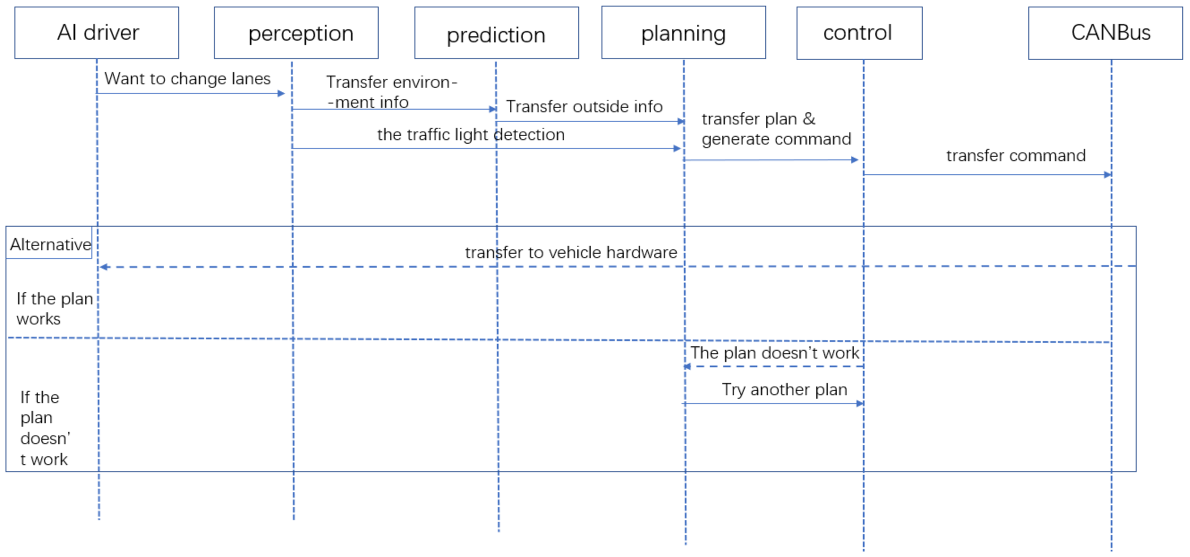

Figure 2. Use Case 1 Switching Lanes

This graph shows a car equipped with the Apollo system. At this time, it requested to change lanes. This request first came to the perception module. The perception module began to use cameras and radar to observe the surrounding environment and transmit information to the next prediction module to predict possible obstacles around in the future. The system then passed to a planning module, which communicates with almost all modules except the traffic light information in the perception module, so we need to add an extra pointer. The processed information is then passed to the next control module, which generates the command and then passes it to the CANBus module, which finally sends it to the vehicle hardware to change lanes.

Use case 2:

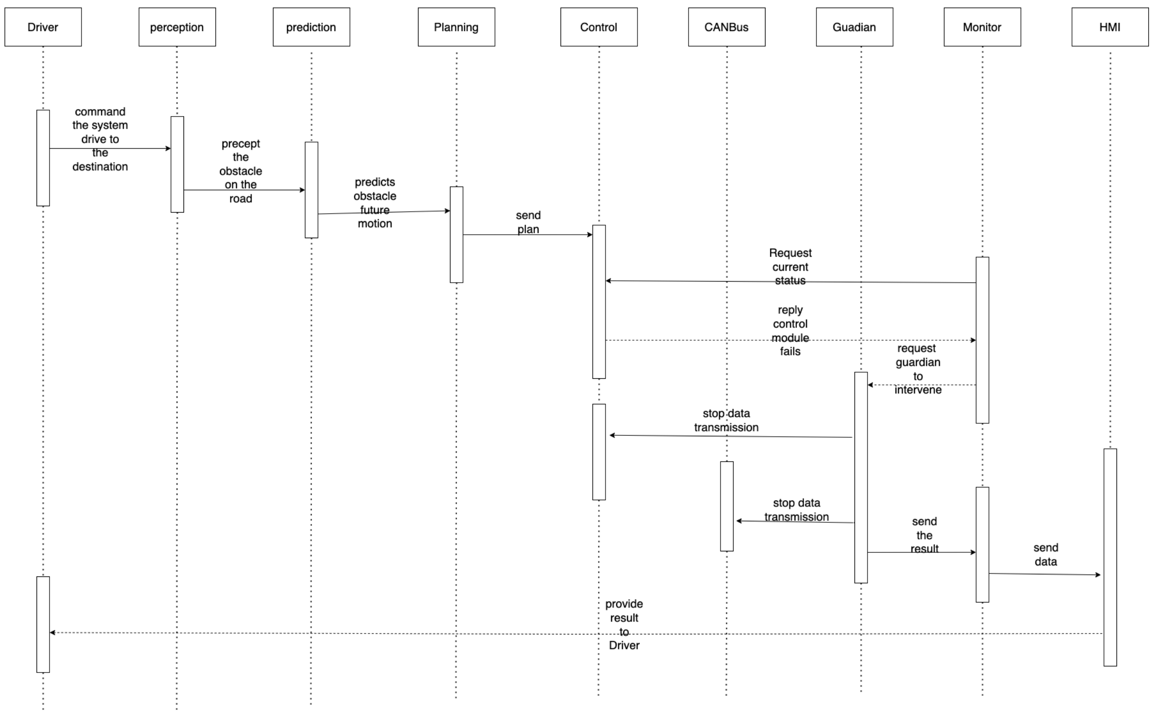

Figure 2. Use Case 2 One Module Fails When System is Running

This graph shows that the module "control" fails in the Apollo system's process. The driver made a request commanding the system to drive to the destination. The prediction module estimates the future motion trajectories for all the perceived obstacles. The planning module plans a safe and collision-free trajectory. The Control takes the planned trajectory as input. However, for some reason, it fails. The Monitor surveil system of all the modules in the vehicle. The Monitor detects one of the module crashes. When the Monitor detects a failure, Guardian will prevent Control signals from reaching CANBus and bring the car to a stop. HMI would provide results to the driver.

3.0.4 Evolution

The Baidu Apollo AD open platform has completed 11 iterations. On December 29, 2021, Apollo ushered in the 7.0 version upgrade, and Baidu's AD open platform had entered the era of instrumentalization. The Apollo 7.0 released in this upgrade has achieved a complete evolution from code to tools, from open-source platforms to tooled platforms. Based on the four open-source cloud service platforms, open-source software, hardware development, and vehicle certification, Apollo 7.0 provides a series of upgrades, including the one-stop practice cloud platform Apollo Studio, industry-leading simulation services, and efficient new models. It is open to full capabilities and can provide a full-stack toolchain for AD. It is easier to use, more advanced, and more efficient to help developers use platform capabilities.

The following paragraph is a brief introduction to the framework of previous versions, which is an overall understanding of Apollo.

Apollo 1.0 works in enclosed venues such as test tracks or car parks. It is also called the Automatic GPS Waypoint Following. At this stage of development, Apollo 1.0 cannot perceive obstacles nearby, drive on public roads, or drive in areas without GPS signals. Building on Apollo 1.0, Apollo 1.5 opens up five additional core capabilities: obstacle perception, planning, cloud simulation, HD maps, and End-to-End deep learning, providing more comprehensive solutions to developers and ecosystem partners to accelerate the deployment of AD. This version of Apollo is suitable for fixed lane cruising. With the expansion of LiDAR, vehicles with this adaptation can presently way better see their environment. They can superior outline their current position and arrange their direction for more secure moving within the path.

Apollo 2.0 supports autonomous vehicles on simple urban roads. Vehicles can navigate the road safely, avoid collisions with obstacles, stop at traffic lights, and change lanes when needed to reach their destination. Another massive update for Apollo 2.0 is allowing Apollo 1.0 users to drive autonomously on some simple city roads. Apollo 2.5 allows vehicles to drive autonomously on obstacle-free highways with cameras for obstacle detection. The vehicle can maintain lane control, drive and avoid collision with the vehicle ahead. Apollo 2.5 offers more efficient development tools, including a visual debugging tool, a data collector for HD maps, and a simulator for cloud-based AD to further enhance developer efficacy. (“Baidu Unveils Apollo 2.5”, 2022)

The main focus of Apollo 3.0 is to provide developers with a platform to build in a closed-off, low-speed environment. The vehicle can maintain lane control, drive and avoid collision with the vehicle ahead. Apollo 3.5 can explore complex driving scenarios such as residential and urban regions. For this reason, it presents modern driving capacities, such as the capacity to complete unprotected turns, which may be a famously challenging move for driverless cars, and oversee speed bumps, clear zones, side passes, contract paths, and parking. (Wiggers, 2019)

Apollo 5.0 is outlined to back the mass generation of geofencing Advertisement. The car presently has 360-degree permeability and has an updated perceptual profound learning show that can handle changes in complex street conditions, making the car more secure and more mindful. Scenario-based arranging has been improved to back other scenarios, such as crossing streets and crossing points.

Apollo 6.0 consolidates unused profound learning models to improve the capabilities for particular Apollo modules, counting Recognition, Expectation, and Arranging. Moreover, this adaptation of Apollo includes different modern administrations: low-speed deterrent expectation demonstrate preparing benefit with semantic outline back; PointPillars-based deterrent location show preparing benefit; Control profiling service, and so on.

3.0.5 Control and Flow of Data

This part is divided into two sections, and the Control and Flow of Data in the Apollo system will be explained in these two sections through the use case diagram and sequence diagram.

In the first section, the use case diagram will play the role of a simple description of what can be done by the system. Usually, our user is available to 4 main functions provided by Apollo AD System. The first is the HD map, and users can check the map through the system on the car boarded screen. At the same time, the map will also show the current position of our user. The function to show the current position is based on the support provided by the GPS localization system, which is the second primary function that users can access. Users can also plan routes by the planning system provided by apollo. This function also includes destination and route selection to help users plan for AD quickly and smoothly. Our user should be able to end AD at any time and restart AD afterwards. (e.g., an emergency happened)

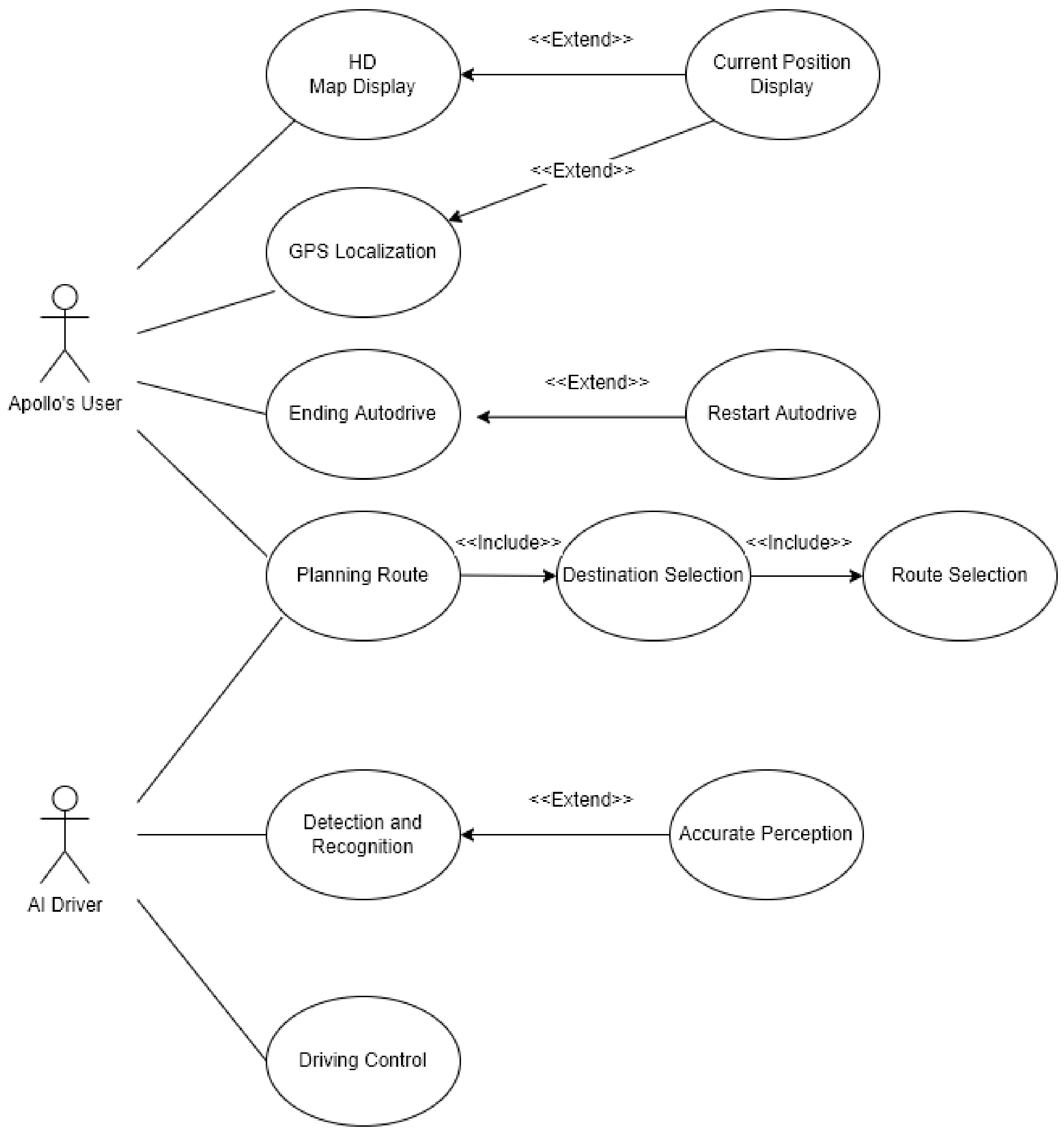

Figure 3. Use Case Diagram

In the second section, with the analysis of the Apollo System, we dug deeper into the cooperation between each component in open-source software parts and made the sequence diagram as below. This sequence diagram shows our understanding of the whole sequence starting from the moment the user gets in the car till the destination is reached.

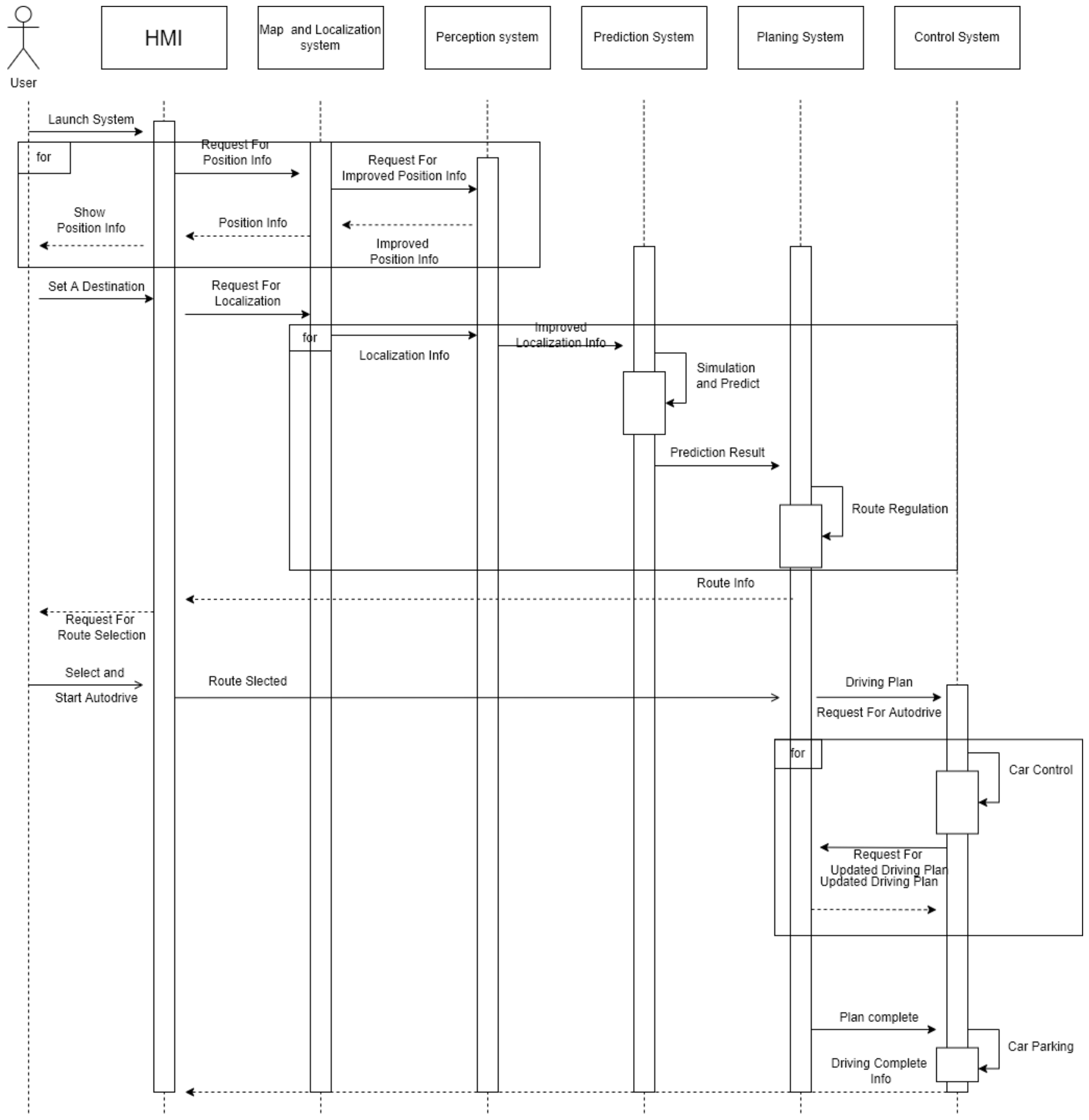

Figure 4. General Sequence Diagram

3.0.6 Concurrency

Apollo has a dedicated scheduling system to satisfy the needs of real-time performance, which allows Apollo to implement concurrency and multiple threads. The essence of this scheduling system is extracting the scheduling logic from the kernel space to the user space and using the coroutine as the primary scheduling unit. Coroutines correlate to Processors in user-space scheduling and user tasks to coroutines.

Apollo currently provides two scheduling strategies, one is the orchestration mode, and the other is the classic mode. Because most AD tasks are run in a certain period, the scheduler can know in advance which tasks should be executed at that moment and will consider the priority. The orchestration mode is based on this feature. This mode can pre-configure which processor each task is executed on and consider the priority relationship. In this case, the assignment of tasks will mainly focus on the initialization phase. When scheduling When completed, the tasks are executed sequentially.

Another scheduling strategy of Apollo is the classic mode. Unlike the orchestration mode, the classic mode tends to the traditional thread pool mode. All processors share the task queue and execute tasks first-in, first-out manner. This strategy can better prioritize tasks. Apollo also allows the grouping of thread pools, which can divide processors into multiple groups, but tasks cannot be transferred between different thread pools.

Concurrency occurs in the Apollo perception module. The Apollo perception module is mainly responsible for checking the obstacles around the car itself. It consists of 5 nodes(LiDAR, Radar, Fusion, Traffic Light preprocess, and Traffic Light process), each of which is a thread, and these threads work for the perception module. There will be connections between threads. For example, "Fusion requires the output data of Lane post-processing, Camera process, and Radar process[3]"("En-Route: On Enabling Resource Usage Testing for Autonomous Driving Frameworks", 2020). Such high concurrency improves the performance of the perception module, allowing Apollo to handle detecting obstacles well.

4.0 Responsibilities among participates

Apollo is an advanced and vast open software platform that provides hardware specifications, in-vehicle software, and cloud services. Any partner can use this platform to develop their driverless system. At the same time, it also provides the world's leading high-precision map service and driverless simulation engine. Apollo's GitHub open-source platform has listed 555 contributors and more than 18,000 releases. The vast amount of data thus accumulated has accelerated the update and development of driverless vehicles worldwide. This speed of development also brings more significant benefits to participants. Therefore, to maintain specific stability and excellent update ability, the Apollo team has established many different mechanisms to maintain the stability and accuracy of the entire project. These mechanisms ensure that each commit is tested correctly, reviewed, and validated to reduce the chance of merge conflicts or other human errors.

(https://github.com/ApolloAuto/apollo/releases)
Apollo has released ten versions, which can be found at the above link. It is a way of indicating the development status of a project to all developers of a software project through the current version. The version number consists of several parts that indicate the different levels of development the project is at. Moreover, it includes the features of each version and the improvement compared to previous versions.

Members of the development team take on different roles as managers of the codebase. These people maintain different aspects of the software development process and facilitate standardized collaborative work.

First of all, coding and design are the most significant basic requirements for every program developer. They should have a profound understanding and programming on the principles of coding. Depending on the specifications, they need to know the correct programming language. Second, program developers should have good management and decision-making skills to ensure that specific design decisions in upfront planning are strictly enforced throughout their life cycle. Third, developers should understand the deployment lifecycle to avoid conflicting design directions for future developers, from initial planning to design, development, testing, deployment to the final support part. The application developer needs to have specific knowledge and ability to drive the stage to completion at each stage. These include event and project management around orchestration and collaboration. Specifically, program developers need to analyze and determine requirements based on data and experience at the beginning of planning and design. Program developers need to continuously test and modify the code according to their needs during the subsequent development and testing phases. Developers must guarantee software quality. After the development and testing steps, developers need to work with other members to detect and ensure that the software can be successfully deployed to ensure that the project is working to completion.

5.0 Lessons learned

In this project, our fundamental goal is to learn and explore the conceptual structure of the AD software Apollo. With the presentation of the conceptual architecture, we study the following aspects: architectural analysis, concurrency, control, data flow, evolution, etc. We have made significant progress, and the teamwork has been rewarding. Here is what we can learn from this project:

1. Development ability
Because Apollo is an open, complete, and secure platform. It can help partners in the automotive industry and AD fields quickly build their AD systems by combining vehicles and hardware systems. This feature of Apollo enables the entire Apollo to allow more people to develop research together, form an integrated and diversified development environment, brainstorm ideas, and quickly solve various problems in the development process.

2. Shared and innovative resources
Apollo's open platform will provide developers with the industry's most cutting-edge technology, wide-coverage, high-accuracy, high-automation map services. It also provides adaption to massive simulation data engines, the world's most data, platform courses from shallow to deep, and the end-to-end autopilot algorithm. The entire Apollo provides a very comfortable platform for developers.

3. User side
Apollo is also user-friendly. Simple address input and the most straightforward command can make users feel the convenience of Apollo automatic driving, the modular design of the overall system, and the maintenance and command response. The accuracy is improved, and the data transmission and storage in the cloud can effectively protect the user's data while transmitting quickly.

4. Technical aspects
During the development process of Apollo, the car will contain a large number of sensors to ensure the safety of operation. Apollo optimizes the communication performance and reduces data copy by sharing memory to improve communication performance. Apollo mainly uses the ROS system, which includes a central node in the ROS system. However, if the central node fails, the entire system will collapse, so in order to ensure the regular operation of the entire system, Apollo optimizes the central node. Apollo uses RTPS The (Real-Time Publish-Subscribe) service discovery protocol implements a complete P2P network topology.

5. Teamwork and Responsibility
"If I have seen further, it is by standing on the shoulders of giants." -Isaac Newton. In this project, we deeply understand the meaning of teamwork. Every team member is being responsible and hardworking. The team leader encourages everyone to search for meaningful information about Apollo and encourages everyone to help each other and share their gains. Because of this sense of responsibility, everyone can work together to complete this project. In the beginning, when we learned that we were going to analyze Apollo's software architecture, everyone was confused because we never knew about AD or Apollo's software. In this case, a team meeting is significant. Through the meetings, everyone understands their tasks and the direction of inquiring data, making the initially complex tasks straightforward. In the next few meetings, the leaders will let everyone share the information they have collected to have a sense of participation. It is precisely because of everyone's enthusiasm that effectively and effieciently.

6.0 Conclusion

Architecture design plays a significant role in the promotion of the Apollo system. The architecture of Apollo's open platform software is not too complex, but each component and their interactions between each other on a working chain shows high technologies applied. As an open-source platform, its quick-to-start architecture cooperates well with core technologies. This provides excellent convenience and flexible scalability for developers in the AD Solutions field to start their work.

Its Publish-Subscribe Style with a combination of slightly Layered Style makes the most of components can be reused in secondary development. Publish-Subscribe Style also eases its system evolution. New versions of them can replace old components without affecting other components in this system. Moreover, with the implementation of these styles, the products derivated from Apollo (e.g., Robotaxi) can have an easy-to-use interface for users without a deep understanding of how the system work.

After analyzing Apollo’s Architecture, we find Apollo can be applied or already applied to many types of scenarios. We will list some of the already deployed ones below.·

- An intelligent transportation system for urban traffic control (Apollo V2X)
- AD taxis for urban roads. (Robotaxi)
- Autonomous parking system. (Valet Parking)

After a discussion, our group members thought the Apollo System could also play an essential role in a barrier-free transportation system specially designed for people with disability in hearing, vision, or some loss in body parts causing difficulty in driving.

However, Apollo’s bright future relies on the development of artificial intelligence. In the current stage, there are still many problems to solve. Among them, the most important one is the security problem. High risk still exists when an emergency happens during AD. For example, the system suddenly shuts down in a remote area, and passengers cannot drive, or the system loses control. Although AD may go beyond traditional manual driving one day, people may choose an experienced human driver rather than an AD machine.

As we mentioned in the ‘what we learned’ part, we are grateful for the chance of learning architecture through this project on Apollo. We still have a lot to research on. Please look forward to the updating on our website.
(https://xuan1030.github.io/CISC322-326-ArchiTiger/)

7.0 Naming Conventions

HD Map: High-Definition Map
AD: Autonomous Driving
HMI: Human-Machine Interface
CANBus: Controller Area Network
GPS: Global Positioning System
ROS: Robot Operating System

8.0 References

Alcon, M., Tabani, H., Abella, J., Kosmidis, L., & Cazorla, F. J. (2020). En-Route: On Enabling Resource Usage Testing for Autonomous Driving Frameworks. New York; Association for Computing Machinery.

Apollo Auto. (2018, July 18). Apollo 3.0: Entering the New Era of autonomous driving. Medium. Retrieved February 21, 2022, from https://medium.com/apollo-auto/apollo-3-0-entering-the-new-era-of-autonomous-driving-d781bc769cef

Baidu USA, LLC. (2018, April 19). Baidu unveils Apollo 2.5, the newest upgrade to its autonomous driving open platform. GlobeNewswire News Room. Retrieved February 21, 2022, from https://www.globenewswire.com/news-release/2018/04/19/1481478/0/en/Baidu-Unveils-Apollo-2-5-the-Newest-Upgrade-to-its-Autonomous-Driving-Open-Platform.html

Baidu (Hong Kong) Limited. (2017, April 19). Baidu Announces Project Apollo, opening up its autonomous driving platform. GlobeNewswire News Room. Retrieved February 21, 2022, from https://www.globenewswire.com/news-release/2017/04/19/1018939/0/en/Baidu-Announces-Project-Apollo-Opening-Up-its-Autonomous-Driving-Platform.html

History of autonomous cars. TOMORROW'S WORLD TODAY®. (2022, January 27). Retrieved February 21, 2022, from https://www.tomorrowsworldtoday.com/2021/08/09/history-of-autonomous-cars/

Wiggers, K. (2019, January 8). Baidu announces Apollo 3.5 and Apollo Enterprise, says it has over 130 partners. VentureBeat. Retrieved February 21, 2022, from https://venturebeat.com/2019/01/08/baidu-announces-apollo-3-5-and-apollo-enterprise-says-it-has-over-130-partners/