# Apollo

https://youtu.be/xryItoceXjs

# Names and Roles

- Group Leader - Ryan Saweczko

  Conceptual architectures and new interactions of enhancement, SAAM analysis

- Presenter 1 - Briggs Fisher

  Report: Introduction, Effects

  Presentation: Introduction, Effects, Lessons Learned, Conclusion

- Presenter 2 - Itay Ben Shabat

  Report/Presentation: Sequence Diagrams

- Tom Lin

  Report/Presentation: New Interactions with the System

- Douglas Chafee

  Report: Proposed Enhancement

  Presentation: Proposed Enhancement, SAAM Analysis

- Karl Dorogy

  Report: Conceptual Architectures, Potential Risks, Abstract

  Presentation: Potential Risks

# Introduction

- From what we have learned about managing large-scale architectures we are now able to add our own addition into Apollo’s conceptual architecture.
- We proposed a communication module that alerts of traffic or obstacles for optimal routing by either communicating with other cars in a peer-to-peer style architecture or by connecting to a server in a client-server architecture
- We will display how our module would fit in the conceptual architecture
- How both implementations interact with other subsystems and how each implementation differs in doing so
- How our module will affect the maintainability, evolvability, performance, testing, and security.
- A use case and sequence diagram for our module in action.
- Finally, we will determine who our stakeholders are and how each implementation delivers their desired nonfunctional requirements, (NFRs).

# Proposed Enhancement

- The current Apollo system is designed so each car runs as an independent entity with no communication or awareness of other Apollo vehicles around it. Operating as an independent system, Apollo vehicles rely on their own perception abilities through camera, Lidar, and RADAR to safely navigate from point A to B.

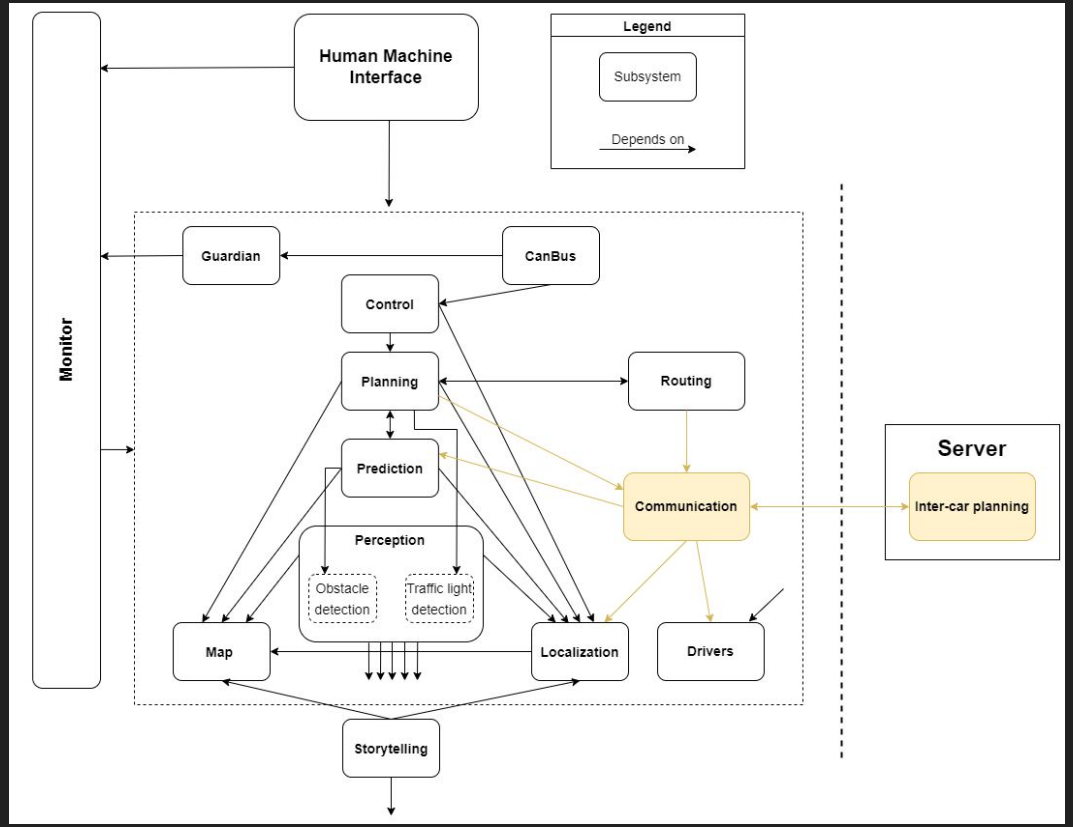
- A possible enhancement to the system is adding a method of communication between vehicles, taking them from independent entities to a network of autonomous cars. This addition would add a level of optimization to routing and reduce travel times in urban environments where congestion is a common problem on roads.

# Conceptual Architectures - Client Server

In the client-server architecture, two new components are required: the communication module, and a new server. The server only depends on the communication module.

Communication module depends on drivers, localization, and prediction.

Planning and routing depend on communication



# Conceptual Architectures - Peer to Peer

In the client-server architecture, one new component is required: the communication module. The communication module depends on routing, drivers, localization, prediction, and planning.

The routing and planning modules will depend on communication.

# New Interactions within the System

From observing the modules within the software architecture and comparing it with our proposed enhancement, we've come up with five modules that will be impacted by our implementation.

- Drivers
- Planning
- Prediction
- Localization
- Routing

# New Interactions within the System - Prediction

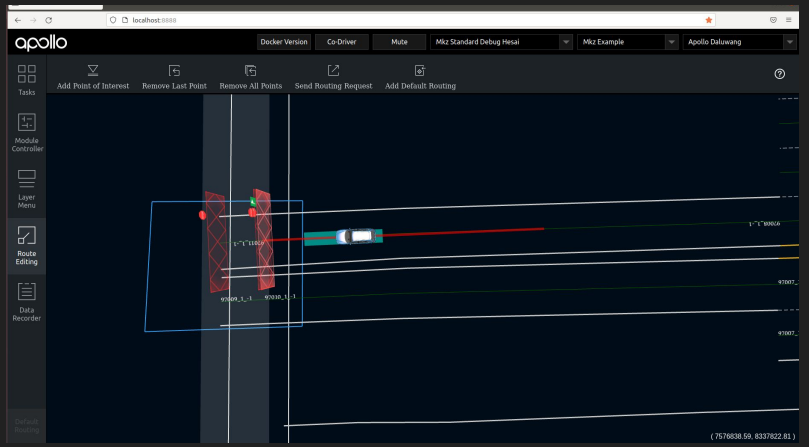
The Prediction module is responsible for determining what the obstacle data is supposed to represent. An obstacle data has four stages it has to go through in order to reach the output; Container, Scenario, Evaluator, and Predictor.
By adding our enhancement,

- The newly added Communications module will use Prediction solely for its Predictor submodule
  - The Predictor submodule is responsible for determining the category/level of the obstacle
  - The output will be the data Communications transmits to all nearby cars
- The specific directory that is impacted is prediction/predictor, with the PredictionComponent::PredictionEndToEndProc being a channel Communications will listen to.

# New Interactions within the System - Planning

The Planning module utilizes information it receives from the Prediction module. It would coordinate the next move the car will take according to the oncoming obstacle data.
With our enhancement,

- The Planning module will account for obstacle datas incoming from other Apollo vehicles
- Specific directories and files that will be impacted / updated are
  - planning_component.cc
  - planning/planner
  - planning/tasks



# New Interactions within the System - Localization

The Localization module provides localization services, such as Real Time Kinetic information and information about the car's current GPS location.
By adding our enhancement,

- The Apollo vehicle will be able to communicate information stored inside Localization (IMU, RTK, GPS location) with other Apollo vehicles
    - This allows for other Apollo cars to know precisely where all the other Apollo cars are
- Specific directories that will be impacted and used are localization/rtk, localization/msf, and localization/ndt

# New Interactions within the System - Drivers

The Drivers module contains required drivers for peripherals the car requires, including, but not limited to, microphone, camera, lidar, and radar.
By adding our enhancement,

- There will be a new driver directory added to the Driver module, Wifi
  - This will allow the Apollo vehicle to have WiFi capabilities and communicate via WiFi
- There will be no specific internal directories impacted upon this addition

# New Interactions within the System - Routing

The Routing module creates a route, or path for the Apollo vehicle after all the obstacle avoidance procedures have passed.
By adding our enhancement,

- Routing will take into account other Apollo vehicles in the area to maximize road efficiency
- Routing will take into account obstacle data from other Apollo vehicles
- Specific directories and files that are impacted are routing.cc, routing_component.cc, and routing/core.

# Effects of the Enhancement

- Extra maintenance will be required for each car’s communication component and to maintain the server if implemented, however the added maintenance is optional as the communication component will not cause issue upon crashing.
- Evolvability simply requires the client (or peer) to update to match the server (or other peer)
- Evolvability issues should be handled with proper backwards compatibility support.
- Performance is bottlenecked by the performance of the network connecting the client to the server or the peer to the peer.
- Client-server performance is more stable than peer-to-peer due to the chance for down, delayed, or hostile peers.

# Testing Impact of the Enhancement

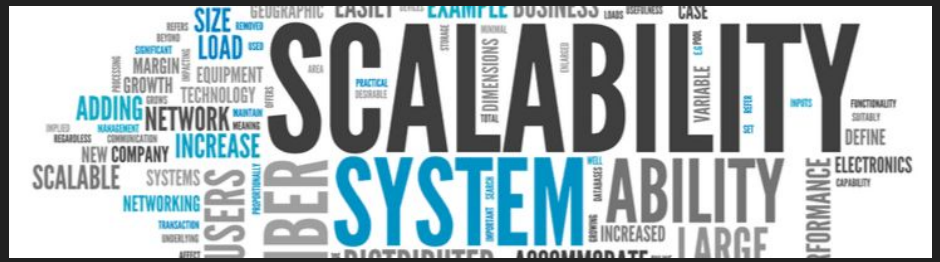
- Testing the enhancement will be done with a mix of unit tests and using the simulation software already in Apollo.
- Unit tests can be used to verify the expected functionality and message sending works
- Using the simulation software while simulating multiple Apollo cars can test practical functionality if peer-to-peer is used, while an additional testing server needs to be constructed to run alongside the simulation to test a client-server implementation.

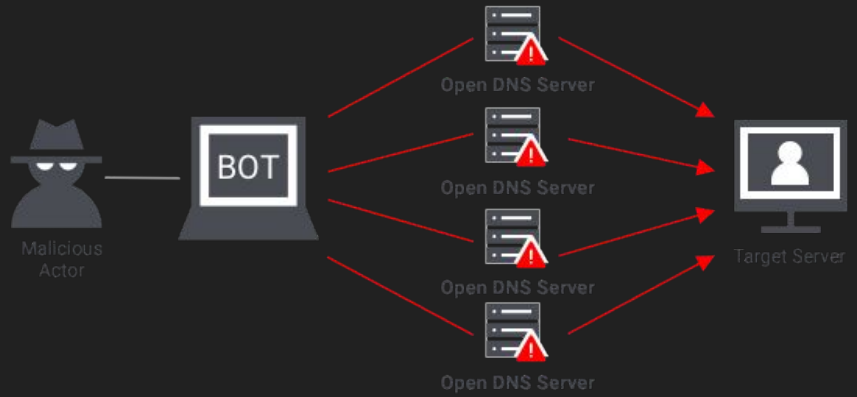
# Potential Risks of the Enhancement

- The inclusion of a single central server in which all Apollo clients can connect to and send their respective information to, creates a significant single point of failure/risk in regards to **availability** within the system.
- A client server connection also makes the quality/**performance** of the Apollo system's routing and planning modules semi-dependent on the overall network's performance at times, potentially risking the response time performance of vital information to the system.
    - For example, a significant lack or delay of information caused by a down server or high traffic network will still severely impact the overall quality of routing between Apollo cars as a whole in terms of car congestion, traffic, and accident avoidance.
- Server **performance** in regards to the **throughput** of information is also a major potential risk in which Apollo's central server should be able to handle and compute all incoming and outgoing information with minimum delay even during peak hours such as national holidays and or around 5:00pm on a week day when most individuals start their commutes.

# Potential Risks of the Enhancement - (Continued)

- Potential **scalability** risk as the number of operational Apollo cars grows eventually and how many clients can be connected to the single central server or network before performance response time and throughput cannot be upheld anymore.
- Generally, across both the client-server and the peer to peer method of implementation, a major potential risk that both implementation methods hold is **security**.
    - Sending of false/malicious routing information/messages to other cars, altering messages in transit to/from cars, and or the interception of private information over a client server or car to car (peer-to-peer) communication
    - In the case for a client-server communication, DDoS attacks are especially apparent in creating an **availability** risk, where hackers could execute distributed denial of service attacks blinding the vehicle from information to the outside world.



# SAAM Analysis: Stakeholders

## Owners/creators of Apollo

- Scalability of the communication module: How many cars can be connected to the network before the response time starts to increase too much.
- Security: If this system gets hacked and malicious instructions are sent to cars due to that, it would cripple the company due to lack of trust from their users.

## Developers of Apollo

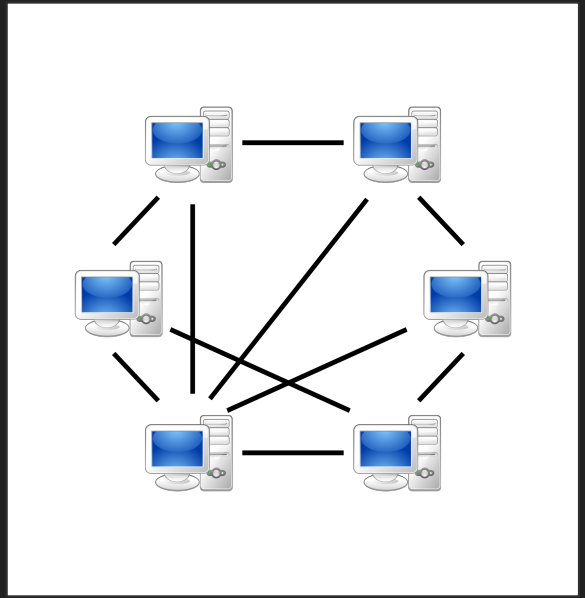
- Maintainability: The developers would want tests that can be run easily and quickly to ensure the new module continues to function properly.
- Modifiability: The new module should be implemented with interactions only to the other modules it is specifically required to interact with, minimizing the dependencies to allow for easier modification in the future.

## Users of Apollo

- Security: Users would want a large amount of security on the communication network, to allow only “valid” Apollo cars to send messages, and filter out messages by malicious entities.
- Performance: Apollo users would want communications to take as little computing power as possible since this feature is meant to optimize travel time.

# SAAM Analysis: Peer to Peer

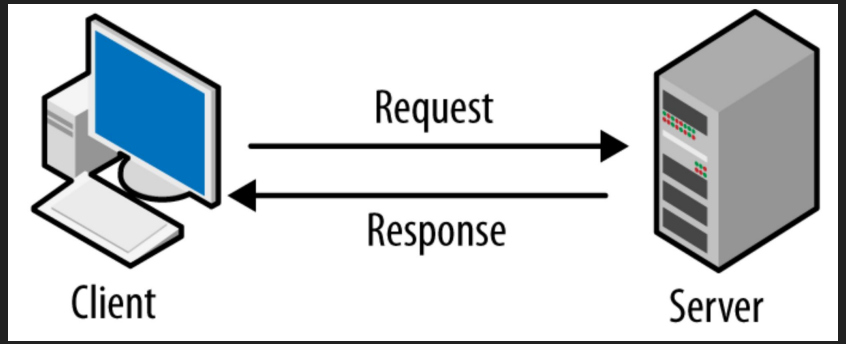
The performance NFR would be slightly more challenging for this implementation. With only a few other Apollo cars near one user's car, the overhead for the communication will be low. However, this overhead would greatly increase as more cars are nearby, and multiple cars all reporting the same events might cause feedback loops and more performance loss. This is different from the scalability since the cars would be able to handle many other Apollo cars nearby, it's just the CPU and bandwidth usage would grow above a reasonable bound, requiring users to get better CPUs before any scalability limit would be reached. This problem can be mitigated to a reasonable degree by reducing the events cars will broadcast to each other if there are more cars nearby, limiting broadcasts to only be large obstructions on the road, or other serious impediments that would strongly affect other cars in the area.



Security would be the hardest impacted with a peer-to-peer implementation. Since Apollo is open source, anyone, including malicious actors, are able to download and use the software. In a peer-to-peer system it would be incredibly hard to police all the cars in the network to stop such actors. We are unsure of what cryptography might exist that would allow cars to be trusted to be sending only valid, accurate information and not faking accidents or other road conditions.

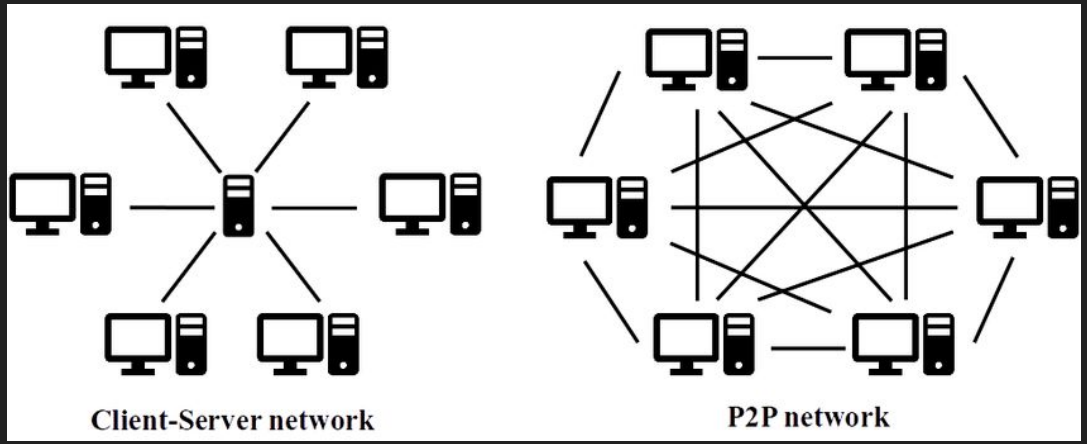
# SAAM Analysis: Client-Server

Scalability, modifiability, and security are all impacted a little harder by this change, and would be a bit harder to reach. Scalability is a problem because instead of only nearby cars communicating in peer-to-peer, now all cars are connecting to a server, increasing the load more. This can be mitigated by expanding out, adding new servers to handle cars in specific cities / countries so the server never gets overloaded. Modifiability is also harder because of the increased complexity having a central server that cars need to connect to will add. Instead of each car being its own ecosystem, which remains mostly true for peer-to-peer, now a setup will require a separate entity to communicate to. Updating the server will need a complete test setup for the server, and more work to integrate the change seamlessly to cars that might be a slightly different version. Finally, security is hard to implement because the Apollo software is open source. However, since there would need to be a central server owned and controlled by Apollo, they would be able to create tokens to give to trusted cars as one solution, or at least the central server would be better equipped to detect malicious information given by one car by seeing if other cars in the same area also report it. This allows the central server some power to determine if a specific car is malicious.



# SAAM Analysis: Comparison

- Each implementation impacts the NFRs in different ways. For the developers, the clear preferred option would be a peer-to-peer implementation. Both implementations would have roughly the same level of maintainability, but peer-to-peer would be easier for them to create and modify.
- From the user's perspective, the client-server implementation is preferable. Having a server to connect to alleviates both of the major concerns of the client: security and performance.
- The owners of Apollo won't have too strong an opinion about which implementation to choose. The peer-to-peer implementation would be easily scalable, since it mostly scales by itself without any necessary changes, while the client-server has better security capacity for them. However, another slight requirement the owners might have that could push them to peer-to-peer is the simpler infrastructure (or lack thereof) required for peer-to-peer.

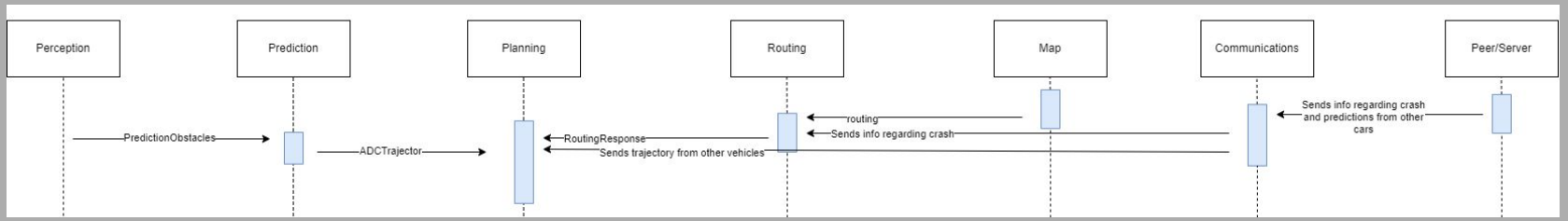


# SAAM Analysis: Conclusion

Given this analysis of the two proposed implementations for a communication module, we believe the best one would be a peer-to-peer style implementation. The only major drawback of this implementation is the security of the system. Creating security in the peer-to-peer system could be achievable through consensus of multiple messages, where if one car alerts with a message, receiving cars won't necessarily follow it unless other cars also in the area report seeing it. Furthermore, the performance penalty that may occur if there are many Apollo cars in the same area can be mitigated through careful control of what messages are sent. If there's a high density of Apollo cars their peer-to-peer messaging scheme can change to only report large amounts of traffic or accidents. This would still allow for improvements in the car's planning capabilities to avoid high-traffic areas, while not hurting performance as much in areas with a high density of Apollo cars.

# Sequence Diagram

A new crash has been reported and the routing adjusts based on this



# Concurrency in enhancement

This implementation will increase the concurrency of the system. With an added communication module, this needs to be running concurrent to all the other modules within the car, talking information from the cyber channels to broadcast and sending received data back into the system. Thanks to the real time operating system being used, with its cyber channels this concurrency should be easy to handle, as it will follow the same style used by all other concurrency within the system.

# Lessons Learned

Through designing and reporting on our additional module, we learned how to insert modules into a large-scale conceptual architecture, as well as the ripple effects that the additional module will cause on the architecture as a whole. We learned how to create two implementations of a module and how to choose the implementation that best suits the nonfunctional requirements our stakeholders desire.

# Limitations

There are a few limitations of having this communication module.
One major one is the performance overhead that would be added to the CPUs when this enhancement is added. Since the module is used for optimization only and won't help critical tasks within driving such as not hitting a person crossing the street, CPU utilization on this is CPU utilization that isn't used in the machine learning modules that do that. So minimizing this modules impact is critical.
Another limitation for a client-server implementation would be the cost of maintaining a central server for cars to connect to. Currently no such central entity exists, everyone using Apollo can do so with no cost to the creator. This simply won't be the case if a server is added, now the owners have some cost they always need to pay to maintain the server's hardware.

# Conclusion

- We decided on the addition of a communication module to report traffic and obstacles either between vehicles or between a server to optimize knowledge for routing.
- We displayed that our module can fit into Apollo’s conceptual architecture and will interact smoothly with the existing subsystems.
- We determined the maintenance increase is minimal, the evolvability is easily scalable, the performance depends on the network connections between peers or between a server, and the risks were greater for the peer-to-peer implementation.
- We provided a use case alongside its sequence diagram for a route being adjusted due to a car crash report.
- Between our two implementations, we determined that our peer-to-peer implementation of our new communication module satisfied the nonfunctional requirements of the most stakeholders, and that the two disadvantages of security and performance for peer-to-peer were mitigatable.