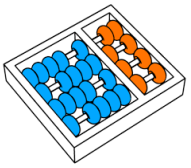
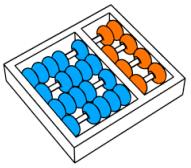
UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO



Projeto Final da disciplina de Sistemas Distribuído (MC714)

Campinas
2024

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO



Projeto Final da disciplina de Sistemas Distribuído (MC714)

Gabriel Gardini - 246289

Paulo Vinícius Pinto - 242863

Campinas
2024

SUMÁRIO

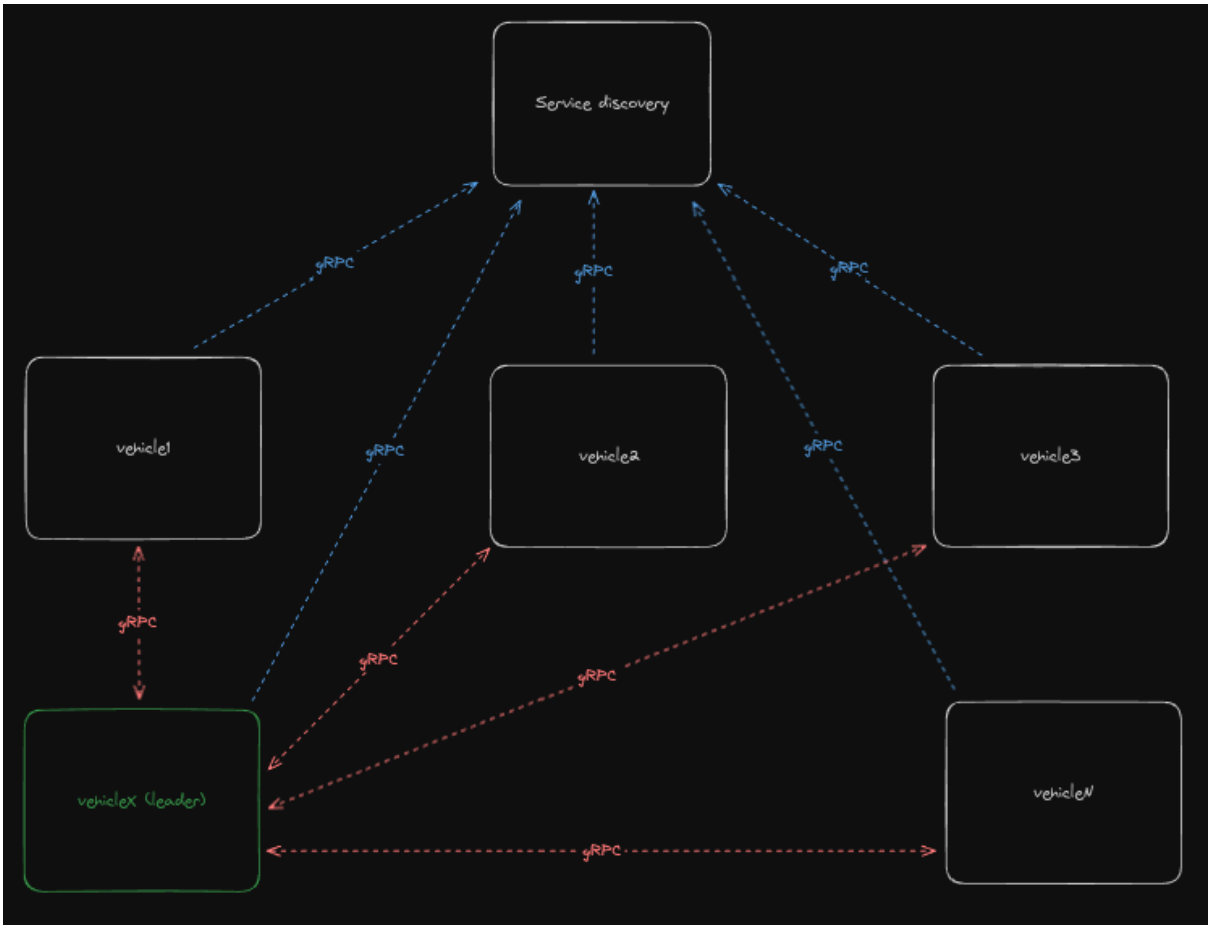
O Problema

Atualmente, com o avanço da tecnologia, muitos carros autônomos estão circulando pelas ruas, tornando o trânsito cada vez mais robotizado. Isso gera a necessidade de sistemas que se comuniquem entre si para trocar informações e evitar acidentes.

Pensando nisso, foi criado um sistema de comunicação de troca de mensagens entre carros autônomos para que estes possam se locomover em cruzamentos sem bater uns nos outros, seguindo as ordens de um carro próximo escolhido como líder.

Implementação

Arquitetura

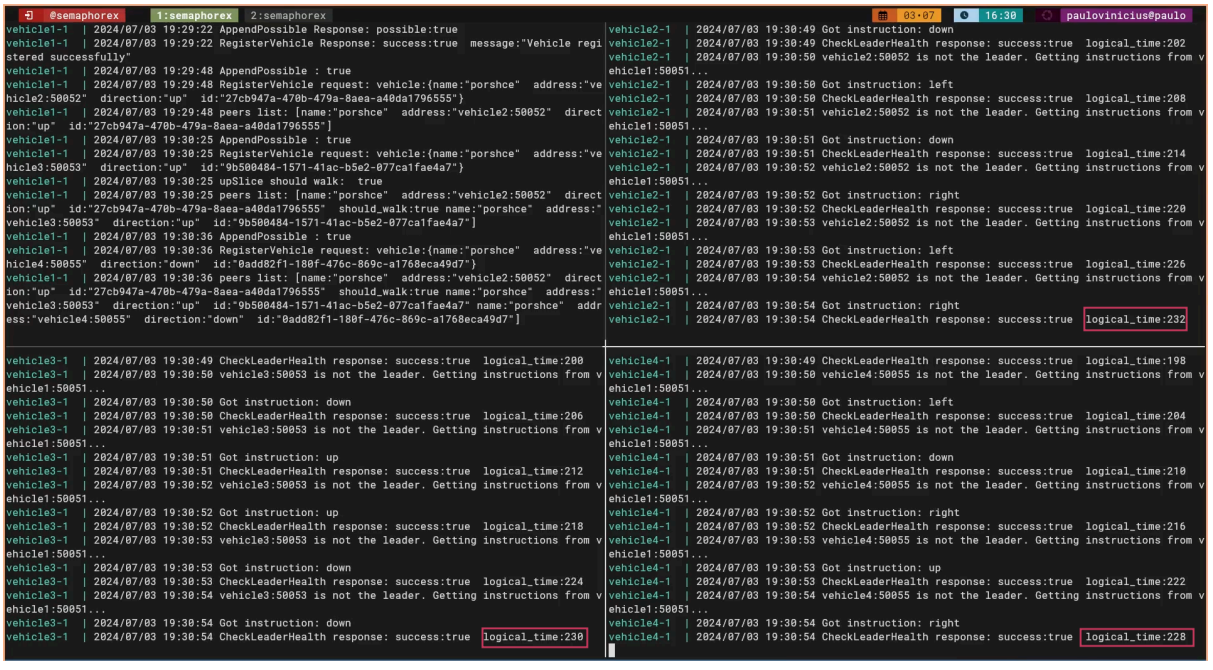


Para resolver o problema sem centralizar rigidamente a coordenação, implementamos uma arquitetura baseada em dois componentes: *Service Discovery* e cliente/servidor de cada veículo. O *Service Discovery* atua como ponto inicial para a conexão dos veículos na região, permitindo que eles descubram outros veículos próximos e, se houver, o endereço do líder responsável pelo pool de veículos. O líder é responsável por enviar comandos como "frente", "trás", "esquerda", "direita" e "parar" para os demais veículos.

Utilizamos o protocolo *gRPC* para a comunicação, devido à sua rapidez e eficiência, minimizando riscos. Além disso, as aplicações foram criadas em Golang e colocada em containers utilizando docker.

Relógio de Lamport

Para garantir a consistência temporal dos eventos no sistema, utilizamos o Relógio de Lamport. Cada mensagem enviada entre os veículos inclui um timestamp lógico, permitindo que as ações sejam ordenadas corretamente e evitando conflitos de informações. Além disso, o relógio é sincronizado entre todos os nós que passam pela pool.



Exclusão mútua

Implementamos a exclusão mútua para garantir que apenas um veículo execute uma ação crítica por vez. O líder gerencia a exclusão mútua, autorizando ou não as movimentações solicitadas pelos veículos, utilizando locks.

Eleição

O algoritmo de Bully é utilizado para a eleição de líderes. Quando um veículo detecta a falha do líder atual, ele inicia uma eleição. Os veículos com maior ID participam da eleição, e aquele com o maior *ID* entre os participantes se torna o novo líder. Este processo garante que sempre haverá um líder coordenando as ações dos veículos.

Os códigos que não foram de autoria própria foram gerados utilizando a ajuda do chatGPT.

Resultados

Inicialmente, tentamos utilizar uma engine de jogos (Ebiten) para visualizar a locomoção dos carros de forma gráfica. No entanto, devido a dificuldades na implementação, optamos por utilizar o terminal. Nos resultados, cada terminal simula um carro autônomo. O carro líder recebe as intenções de direção dos outros carros e decide se libera ou não a movimentação dos mesmos.

Se o carro líder se desconectar, seja por falha ou por se distanciar demais dos outros veículos, uma nova eleição é realizada usando o algoritmo do *Bully*. Nesse processo, o carro com o maior *ID* é escolhido. Se o carro líder se desconectar, seja por falha ou por se distanciar demais dos outros veículos, uma nova eleição é realizada usando o algoritmo do *Bully*. Nesse processo, o carro com o maior ID é escolhido como o novo líder, garantindo a continuidade da coordenação.