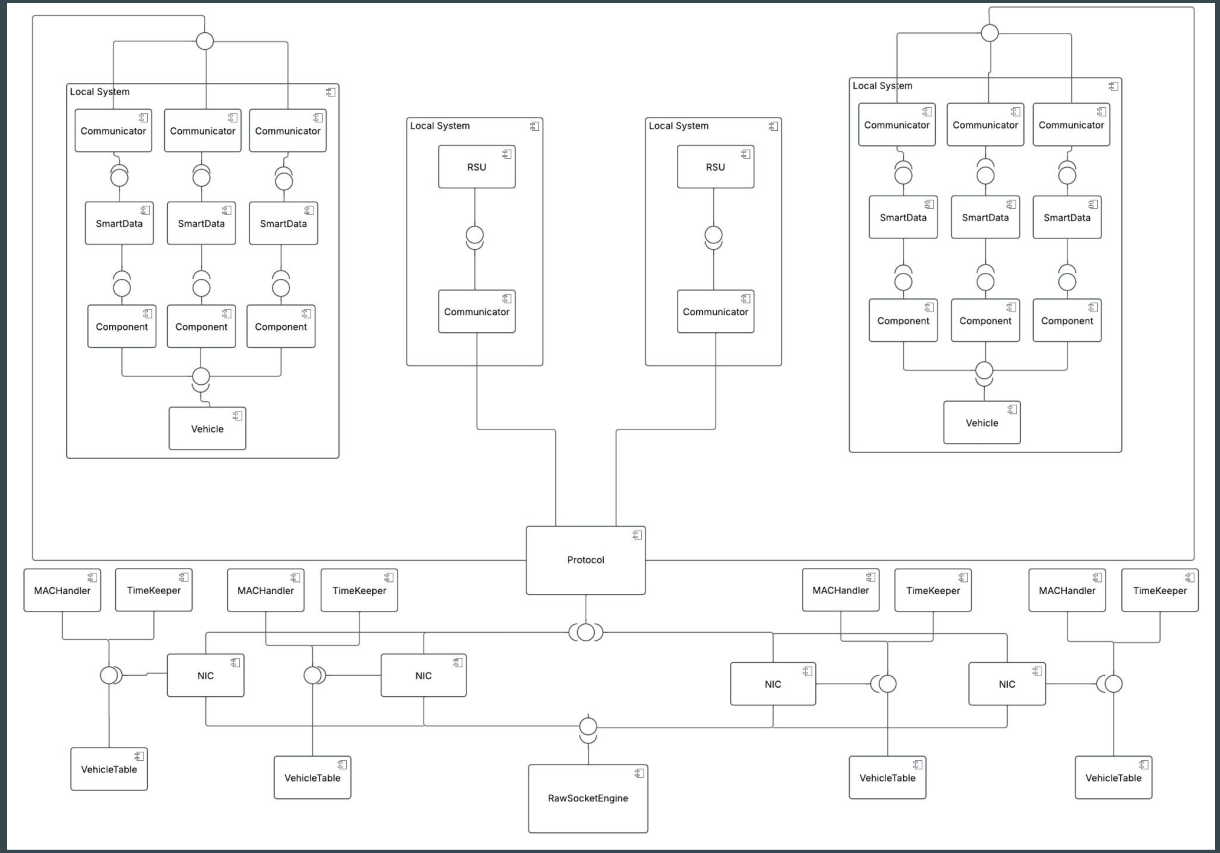
Sistemas Operacionais II - P7

GRUPO C - TARDE

João Vitor dos Santos
Pablo Lopes Teixeira
Fábio Henrique Antunes Coelho
Tiago Oliveira da Luz

Modelagem

Diagrama de Componentes



Classes

- AutonomousAgent
  - Vehicle - Se move pelo espaço, dividido em quadrantes.
    - Possui 5 componentes
  - RSU - Responsável pela sincronização temporal e distribuição de chaves MAC.

Classes

- Component - Estabelece interface padrão para componentes:
  - LIDAR Component
  - GPS Component
  - Controller Component
  - Steering Component
  - Accelerometer Component - Utiliza dados do dataset.

Classes

- Smart Data - Gerencia interesses do componente e resposta a interesses externos:
  - Componente registra interesses e tipo de dado que gera no SmartData
  - Controla recebimento de mensagens, filtrando por interesse do componente.
  - Realiza envio periódico de mensagens resposta (Periodic Thread)

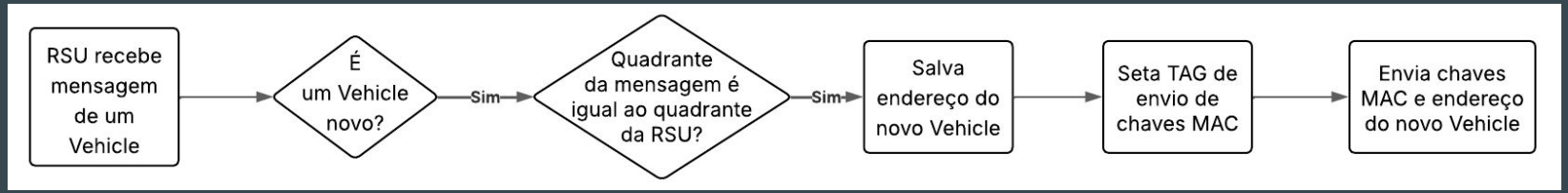
Classes

- NIC
  - Possui comportamentos diferentes para RSU e Vehicle no momento de envio e recebimento de mensagem.
  - Sincronização Temporal (Time Keeper):
    - RSU: Envia mensagem com flag de mensagem de sincronização.
    - Vehicle: A cada duas mensagens de RSU recebidas ressincroniza tempo.
  - Integridade (MAC Handler):
    - RSU: Responde novos veículos com chave MAC do seu quadrante e quadrante vizinhos.
    - Vehicle: Faz verificação de mensagens recebidas de outros veículos e salva chaves recebidas de RSUs.

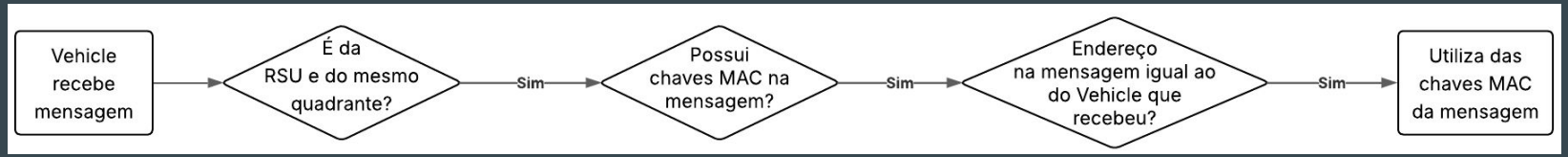
Principais Diferenciais

Unicast na resposta a JOIN

- RSU:



- Vehicle:



Implementação de sincronização e MAC na NIC

- TimeKeeper
  - Verifica se não está sincronizado
    - Checa se o tempo desde a última atualização de timestamp ultrapassa o timeout de sincronização
  - Atualiza o estado de sincronização
  - Atualiza o timestamp
    - Calcula o delay e offset
- MacHandler
  - Cria uma chave MAC
    - Chave MAC de tamanho fixo
  - Gera MACs
    - Xor entre chave e mensagem
  - Verifica MACs
    - Checa se Mac Recebido é igual ao xor entre chave e mensagem recebida

Periodic Threads

Seta chamada periódica da função passada, usando Deadline Task Scheduling:

```
struct sched_attr {
        u32 size;
        u32 sched_policy;
        u64 sched_flags;
        s32 sched_nice;
        u32 sched_priority;
        u64 sched_runtime;
        u64 sched_deadline;
        u64 sched_period;
};
```

```
struct VehicleSched::sched_attr attr;
int ret;
unsigned int flags = 0;

// Set scheduler attributes
attr.size = sizeof(attr);
attr.sched_flags = 0;
attr.sched_nice = 0;
attr.sched_priority = 0;
attr.sched_policy = SCHED_DEADLINE;
attr.sched_runtime = runtime_ns.load();
attr.sched_period = attr.sched_deadline = period_ns.load();

ret = VehicleSched::sched_setattr(0, &attr, flags);
```

Sem mensagens explícitas de JOIN e SYNC

- Se RSU recebe uma mensagem de um Vehicle que está no mesmo quadrante que a RSU e a RSU nunca recebeu mensagem desse Vehicle = JOIN

```
if(_packet_origin == Ethernet::Attributes::PacketOrigin::RSU) {
    if(_quadrant == sender_quadrant) {
        ConsoleLogger::log("RSU: Message with quadrant " + std::to_string(se
        if(!_vehicle_table.check_vehicle(&sender_address)) {
            ConsoleLogger::log("RSU: New vehicle found with address: " + mac
            std::array<unsigned char, ETH_ALEN> sender_address_array;
            memcpy(sender_address_array.data(), &sender_address, ETH_ALEN);
            _vehicle_table.set_vehicle(sender_address_array);
            _send_mac_key = true;
            memcpy(&_unicast_addr, &sender_address, ETH_ALEN);
```

- RSU serve como Master e envia mensagens periódicas com seu timestamp e o Vehicle verifica se está no mesmo quadrante da RSU = SYNC

```
if (attributes.get_packet_origin() == Ethernet::Attributes::PacketOrigin::RSU &&
    ConsoleLogger::log("Received RSU message");
    auto system_timestamp = attributes.get_timestamp();
```

FIM


Shell script
Python
C
What the hell is this?
made with mematic