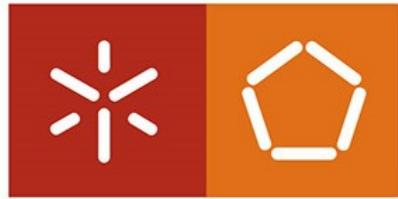


Novos Paradigmas de Redes
2023/2024

Trabalho Prático
Grupo 7

**Sistema de Semáforos Inteligentes suportado por
comunicações V2X**



Universidade do Minho
Escola de Engenharia

André Alves^[pg53651], Renato Gomes^[pg54174], and Afonso Marques^[pg53601]



¹ Universidade do Minho, Braga, Portugal

² www.uminho.pt/PT

Abstract. As redes veiculares permitem a comunicação entre veículos e com a infraestrutura rodoviária, sendo já comuns nos veículos modernos e tendendo a ser tão omnipresentes quanto as redes móveis celulares. Elas melhoram a segurança rodoviária, a eficiência do tráfego e oferecem entretenimento e conforto. Aplicações de segurança reduzem acidentes e fatalidades, enquanto as de eficiência do tráfego, como controle de interseções e semáforos inteligentes, melhoram a qualidade de vida ao reduzir congestionamentos e adaptar-se em tempo real ao trânsito. A comunicação V2X (Vehicle-To-Everything) possibilita a substituição de semáforos físicos por sinais transmitidos por mensagens.

Keywords: VLANS · Redes Celulares · Mosaic · Sumo.

1 Introdução

Neste trabalho, será construído um protótipo funcional de um sistema de semáforos inteligentes suportado por comunicações veiculares. Inicialmente, os veículos enviam mensagens sobre a sua mobilidade para outros veículos e unidades fixas nos cruzamentos, permitindo que os semáforos físicos melhorem a fluidez do tráfego. Esses semáforos podem ser removidos posteriormente, substituídos por sinais transmitidos diretamente aos veículos, permitindo decisões em tempo real. Numa fase avançada, o semáforo virtual pode ser distribuído entre os veículos, que decidirão quando parar ou avançar com base nas mensagens recebidas. O projeto também prevê a coordenação entre semáforos para otimizar o tráfego. O foco principal está na arquitetura do sistema e nos algoritmos de comunicação entre as entidades envolvidas.

O cenário de estudo foi baseado num cruzamento simples da cidade de Espinho, no distrito de Aveiro. A seguinte imagem representa o cenário em questão:



2 Especificação do sistema

2.1 Etapa 1

Na primeira etapa deste trabalho, foram colocados como objetivos principais os seguintes requisitos:

1. a criação da topologia híbrida de rede veicular Ad-Hoc e rede fixa de infraestrutura complementar;
2. definição dos dados que o veículo deve enviar, modo de os obter e formato de envio;
3. criação de uma aplicação para os veículos que implemente uma estratégia de difusão dos dados a um salto de distância;
4. criação de uma aplicação para receção dos dados na RSU, que se limita a recebê-los e enviá-los para o servidor;
5. criação da primeira versão do sistema de semáforos inteligentes que, com base no número de veículos detetados nas várias estradas que vão ter ao cruzamento controlado pelo semáforo, determina qual das estradas recebe o sinal verde e por quanto tempo. Os veículos são detetados através das mensagens recebidas no RSU.

Para a criação da topologia e o cenário em geral, recorremos ao conjunto de ferramentas SUMO e NETEDIT do Eclipse. Juntamente com o aplicativo *scenario-convert* do Eclipse, que tem uma funcionalidade de importação de mapas, criamos um cenário com um cruzamento simples baseado nas ruas em grelha da cidade de Espinho. Este cenário contém duas rotas de sentido único, uma na vertical (r_0) e outra na horizontal (r_1), onde cada uma gera um valor máximo de 100 veículos, verdes na r_0 e amarelos na r_1, todos com as mesmas características a executar as mesmas aplicações.

Existe também uma Road Side Unit, RSU, com latitude igual a 41.004708 e longitude -8.642474. O semáforo está localizado na interseção e contém quatro programas ao seu dispor para executar dependendo das ordens recebidas pela RSU.

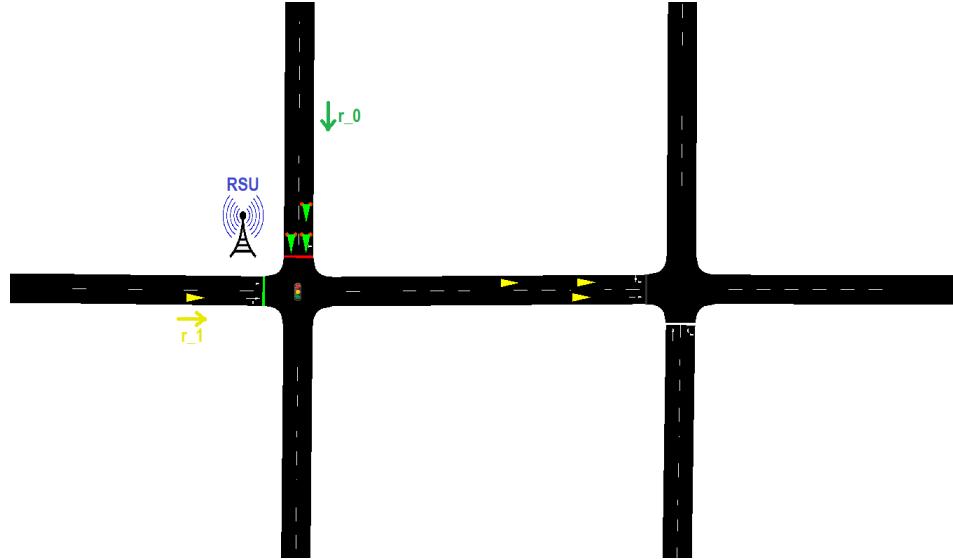


Fig. 1: Cenário de estudo

Os programas do semáforo estão enumerados de 0 a 3. 1 é o programa mais genérico possível, sendo gerado com a criação do semáforo no NETEDIT, e possui um ciclo "normal" das luzes nas duas rotas. O programa 3 é um programa para efeitos de testes que coloca todas as luzes a vermelho nas duas rotas. 0 coloca as luzes a verde na rota r_0 e a vermelho na rota r_1 . O programa 2 faz o oposto do 0. Nesta primeira etapa foram apenas usados os programas 0, 1 e 2, onde 1 foi o programa padrão do semáforo.

```

programID="0":
    <phase duration="62" state="GGGrrr"/>
    <phase duration="10"   state="yyryrr"/>

programID="1":
    <phase duration="32" state="GGGrrr"/>
    <phase duration="10"   state="yyryrr"/>
    <phase duration="32" state="rrrGGG"/>
    <phase duration="10"   state="rrryyy"/>

programID="2":
    <phase duration="62" state="rrrGGG"/>
    <phase duration="10"   state="rrryyy"/>
```

```
programID="3":  
    <phase duration="32" state="rrrrrr"/>
```

A RSU assume um papel central no cenário, tendo como funções principais a receção das mensagens *V2xMessage* de registo enviadas pelos veículos, denominadas GreenWaveMsg, contabilizando durante esse processo o número de carros numa determinada rota num Map temporário. A receção de uma mensagem também desencadeia o processo de verificar os contadores, comparando qual deles é maior, dando a vantagem à rota r_0 nos casos onde os dois contadores sejam iguais. Se uma das rotas no mínimo tiver cinco carros e no total mais do que a outra, então a RSU envia uma mensagem RSUMsg em broadcast, que nesta etapa inicial irá apenas ser lida pelo semáforo. Essa mensagem contem o valor do novo programa que o semáforo vai executar. A aplicação que executa esta lógica é a RSU_Program. No semáforo é executada a TrafficLightApp.

Com recurso à API do Eclipse, conseguimos obter informações sobre os estados dos veículos enquanto decorre a simulação. Nesta primeira etapa, focamo-nos apenas em retirar a informação que consideramos mais importante para que os veículos façam o seu registo na RSU, e esta os possa contabilizar corretamente. Sendo assim, os veículos criam uma mensagem GreenWaveMsg com informação relativa à mensagem secreta, "ABRE!", e à rota onde o veículo circula. Esta mensagem, criada na aplicação MainCarApp, é enviada em broadcast e irá apenas ser captada pela RSU. Os carros também executam duas aplicações adicionais, CarSendsCamAPP e EventProcessingApp, que irão ter mais importância na segunda etapa.

2.2 Etapa 2

A segunda etapa do nosso projeto teve como objetivo melhorar a percepção do semáforo físico relativamente ao ambiente circundante. Para atingir esse objetivo, implementámos um algoritmo de encaminhamento multihop executado pelos veículos. Também se procurou eliminar a enorme quantidade de mensagens enviadas em broadcast e melhorar a noção de vizinho que ficou aquém das expectativas na primeira etapa.

Objetivos

Os principais objetivos desta etapa foram:

1. Especificar e implementar um protocolo de encaminhamento multihop.
2. Desenvolver mecanismos que evitem a sobrecarga da rede (broadcast storms).

Metodologia Protocolo de Encaminhamento Multihop

Nesta segunda etapa, um veículo quando é gerado no cenário, tenta de imediato enviar a sua mensagem de registo GreenWaveMsg. Para isso irá primeiro verificar se está dentro do alcance da RSU, procedendo assim ao cálculo da distância entre ele e ela. Se essa distância for menor que o raio da RSU, definido como um valor inteiro estático igual a 20, então manda o GreenWaveMsg diretamente em unicast para ela, sem saltos intermédios. Caso esteja fora do alcance, utiliza o protocolo de multihop criado pelo grupo.

O protocolo de encaminhamento multihop foi desenvolvido para permitir que um veículo reencaminhe mensagens recebidas de outros veículos sob determinadas condições. A principal condição estabelecida foi que o veículo envia a sua mensagem em unicast para o seu vizinho que se encontra mais próximo da RSU. Desta forma o caminho que o pacote tomará será sempre o menor possível.

Para suportar o novo protocolo, foi necessário implementar mudanças significativas na lógica herdada da primeira etapa. Para começar, a aplicação CarSend-SCamAPP, que antes era praticamente inútil, foi alterada para que apenas realize o envio de mensagens do tipo InterVehicleMsg, que contém informação relativamente ao ‘ID’ do veículo que a envia e a sua posição em coordenadas cartesianas. Estas mensagens são enviadas em broadcast para todos os outros veículos que estejam no seu alcance, passando a ficar registado como vizinho nos veículos que o rodeiam.

O veículo que procura fazer o registo através do multihop, pode então consultar todos os seus vizinhos e, tal como dito anteriormente, determinar qual deles está mais perto da RSU (a posição da RSU é estática e hardcoded) e encaminhar-lhe. O veículo que recebe essa mensagem faz a mesma lógica: se estiver dentro do alcance da RSU, manda para ela diretamente em unicast, caso contrário reencaminha para um vizinho dele.

As mensagens do tipo GreenWaveMsg têm um novo campo que corresponde exatamente ao ‘ID’ do veículo que fez o pedido de registo. Assim, durante os saltos intermédios, conseguimos sempre saber quem foi o veículo original que enviou o pedido.

Prevenção de Broadcast Storms

Para evitar a sobrecarga da rede, reduzimos os tipos das nossas mensagens o máximo que conseguíssemos. Para isso passamos a enviar todas as nossas mensagens em unicast (à exceção das InterVehicleMsg), deste modo limitamos as possíveis *broadcast storms* ao fazer com que eles apenas falem diretamente com quem precisam de entregar a mensagem.

Outra forma que usamos para limitar o número de mensagens foi um uso de *acknowledgments*, ACK, onde todos os carros detêm um campo que indica se eles já comunicaram com a RSU ou não. Para isto, quando a RSU recebe uma mensagem de um certo carro, ela tenta enviar um ACK para esse carro, onde caso este não seja o emissor da mensagem original, este irá procurar no seu alcance pelo emissor original e sucessivamente. Receber um ACK da RSU não impede que receba mensagens de outros veículos, apenas deixa de tentar mandar um pedido em GreenWaveMsg.

2.3 Etapa 3

Na terceira etapa do nosso projeto, procuramos eliminar a necessidade de semáforos físicos, substituindo-os por um sistema de semáforos virtuais. Para tal, é crucial implementar uma comunicação eficiente entre o RSU e os veículos. Esse sistema de comunicação garantirá que as ordens de avançar ou parar sejam transmitidas com precisão aos veículos, consoante o estado do semáforo determinado pelo algoritmo de controlo de tráfego definido na etapa anterior.

Estrutura do Sistema:

Comunicação entre Veículos e RSU: Na segunda etapa, com a comunicação entre os veículos e o RSU melhorada, essa comunicação agora permite que os veículos enviem dados da rota e do seu 'ID' ao RSU, que, por sua vez, utiliza essas informações para tomar decisões sobre o controlo de tráfego.

Comunicação RSU-Veículos: Na terceira etapa, além da comunicação existente, precisamos que o RSU envie mensagens com instruções específicas aos veículos, indicando se devem parar ou avançar. Essa comunicação é fundamental para substituir os semáforos físicos por semáforos virtuais.

Implementação do Semáforo Virtual: Para implementar o semáforo virtual, o grupo pensou em usar o algoritmo na RSU que determina a sequência de verde e vermelho para cada via. Com base nessa sequência, o RSU iria enviar mensagens com as ordens correspondentes aos veículos na proximidade do cruzamento.

Uso do Multihop para Propagação de Mensagens: A ideia que o grupo propõe é a utilização do protocolo multihop, definido na etapa 2, para a propagação das mensagens de 'stop' e avanço entre os veículos. Este protocolo permitiria que a mensagem do RSU fosse distribuída de forma eficiente, mesmo em cenários com veículos fora do alcance direto do RSU.

Mensagens de Paragem no Cruzamento: Quando o RSU determina que uma determinada via deve parar, ele envia uma mensagem de ‘stop’ aos veículos mais próximos do cruzamento. Esses veículos, ao receberem a mensagem, começam a reduzir a velocidade e, eventualmente, param.

Propagação da Mensagem para Veículos Atrás: Os veículos que recebem a mensagem do RSU também têm a responsabilidade de retransmitir essa mensagem para os veículos que estão atrás deles. Essa retransmissão é feita usando o protocolo multihop, garantindo que todos os veículos que se aproximam do cruzamento sejam informados da necessidade de parar.

Infelizmente, devido à falta de tempo, não conseguimos implementar a etapa 3, sendo que a nossa principal dificuldade foi tratar da paragem dos veículos, não tendo conseguido realizar esta etapa mesmo tendo ideias de como o fazer.

3 Testes e resultados

De forma a ter um comprovativo de que de facto o protocolo multihop está a ter efeito na comunicação da simulação, expomos o seguinte exemplo de comunicação em multihop com início no veículo veh_130, que está fora do alcance da RSU, para o seu vizinho veh_124 que encaminha a mensagem para a RSU. Na volta, a RSU vê que quem lhe enviou a mensagem foi o veh_124, então envia o ACK do veh_130 pelo veh_124, que o faz chegar ao veh_130. Ou seja, o veh_130 nunca chega a comunicar diretamente com a RSU.

```

2024-05-31 19:48:20,021 INFO - ----- (at simulation
2024-05-31 19:48:20,021 INFO - Processing event (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - ----- (at simulation
2024-05-31 19:48:20,021 INFO - My Route = r_0 (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - distance to RSU = 23.973965099084882 (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - ----- (at simulation
2024-05-31 19:48:20,021 INFO - Vizinho: veh_124 (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - Vizinho: veh_124 (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - Vizinho: veh_128 (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - Position = CartesianPoint(x=143.24,y=558.56,z=0.00) (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - Position = CartesianPoint(x=140.84,y=558.23,z=0.00) (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - Position = CartesianPoint(x=140.86,y=557.69,z=0.00) (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - Resent to veh_124 - GreenWaveMsg origin in veh_130 (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - ----- (at simulation
2024-05-31 19:48:20,021 INFO - Tried sending message to other car (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - ACK = False (at simulation time 698.000,000,010 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - ----- (at simulation
2024-05-31 19:48:20,021 INFO - ----- (at simulation
2024-05-31 19:48:20,021 INFO - New neighbour is veh_124 (at simulation time 698.000,400,000 s) ----- (at simulation
2024-05-31 19:48:20,021 INFO - ----- (at simulation
2024-05-31 19:48:20,024 INFO - New neighbour is veh_126 (at simulation time 698.001,900,000 s) ----- (at simulation
2024-05-31 19:48:20,024 INFO - ----- (at simulation
2024-05-31 19:48:20,024 INFO - New neighbour is veh_128 (at simulation time 698.001,900,000 s) ----- (at simulation
2024-05-31 19:48:20,024 INFO - ----- (at simulation
2024-05-31 19:48:20,024 INFO - Received ACK with origin at RSU from veh_124 at 698002100000 (at simulation time 698.002,100,000 s) ----- (at simulation
2024-05-31 19:48:20,024 INFO - ----- (at simulation

```

Fig. 2: Logs no veh_130

```

2024-05-31 19:48:20,023 INFO - ----- (at simulation
2024-05-31 19:48:20,023 INFO - Received GreenWaveMsg = 'ABRE! | r_0 | veh_130' from veh_130 (at simulation time 698.001,400,010 s)
2024-05-31 19:48:20,023 INFO - distance to RSU = 12.423115765371737 (at simulation time 698.001,400,010 s)
2024-05-31 19:48:20,023 INFO - ----- (at simulation
2024-05-31 19:48:20,023 INFO - Sent to RSU = 'ABRE! | r_0 | veh_130' (at simulation time 698.001,400,010 s)
2024-05-31 19:48:20,023 INFO - ----- (at simulation
2024-05-31 19:48:20,023 INFO - Resent to RSU - GreenWaveMsg origin in veh_130 (at simulation time 698.001,400,010 s)
2024-05-31 19:48:20,023 INFO - ----- (at simulation
2024-05-31 19:48:20,023 INFO - Resent to veh_130 - ACK origin in RSU (at simulation time 698.001,700,000 s)
2024-05-31 19:48:20,023 INFO - ----- (at simulation

```

Fig. 3: Logs no veh_124

```

2024-05-31 19:48:20,027 INFO - ----- (at simulation
2024-05-31 19:48:20,027 INFO - Received GreenWaveMsg = 'ABRE! | r_0 | veh_130' from veh_124 (at simulation time 698.004,800,000 s)
2024-05-31 19:48:20,027 INFO - Received correct passphrase - ABRE! - from veh_130 (at simulation time 698.004,800,000 s)
2024-05-31 19:48:20,027 INFO - Route r_0 = [veh_130] (at simulation time 698.004,800,000 s)
2024-05-31 19:48:20,027 INFO - Sending ACK to veh_130 (at simulation time 698.004,800,000 s)
2024-05-31 19:48:20,027 INFO - ----- (at simulation
2024-05-31 19:48:20,027 INFO - Sent ACK message to veh_124 with final destiny to veh_130 (at simulation time 698.004,800,000 s)
2024-05-31 19:48:20,027 INFO - ----- (at simulation

```

Fig. 4: Logs na RSU

4 Conclusão

Na primeira etapa, implementamos com sucesso um sistema de semáforos físicos controlados por dados recebidos dos veículos. Os veículos enviavam informações sobre a sua mobilidade, que eram recebidas e processadas pelo RSU. O algoritmo de controlo de tráfego na RSU determinava o estado do semáforo (verde ou vermelho) com base na densidade de veículos em cada rota, melhorando a fluidez do tráfego.

Na segunda etapa, implementamos um protocolo de encaminhamento multi-hop que permitiu aumentar a área de percepção do RSU. Veículos fora do alcance direto do RSU podiam retransmitir mensagens, aumentando a precisão dos dados de tráfego coletados. Esse protocolo foi projetado para minimizar a sobrecarga de rede, garantindo uma comunicação eficiente.

Infelizmente, devido à falta de tempo, não conseguimos realizar a terceira etapa do projeto, que visava substituir o semáforo físico por um semáforo virtual. Esta etapa envolveria a implementação de comunicação bidirecional completa entre o RSU e os veículos, onde o RSU enviaria mensagens de ordens para parar ou avançar diretamente aos veículos, eliminando a necessidade de semáforos físicos. Além disso, a propagação dessas mensagens usando o protocolo multihop não pôde ser implementada.

Apesar de não termos concluído a terceira etapa, as duas primeiras etapas foram realizadas com sucesso modesto, mostrando um avanço significativo na utilização de comunicações V2X para a gestão de tráfego. O sistema desenvolvido

demonstrou ser capaz de melhorar a fluidez do tráfego e reduzir o congestionamento em interseções controladas por semáforos.

O desenvolvimento deste projeto contribuiu para uma melhor compreensão das possibilidades e desafios das redes veiculares e das comunicações V2X. O próximo passo seria completar a implementação do semáforo virtual, explorando ainda mais o potencial das tecnologias veiculares para criar sistemas de transporte mais inteligentes e eficientes. Este trabalho oferece uma base sólida para futuras melhorias e implementações, destacando a importância de soluções tecnológicas avançadas na gestão de tráfego urbano.

References

1. https://eclipse.dev/mosaic/java_docs/allclasses.html
2. <https://sumo.dlr.de/docs/Netedit/index.html>
3. <https://sumo.dlr.de/docs/index.html>
4. <https://eclipse.dev/mosaic/>
5. https://eclipse.dev/mosaic/docs/scenarios/scenario_convert/