Sistema de Semáforos Inteligentes

Trabalho Prático - Novos Paradigmas de Rede - Grupo 13

1 de junho, 2023

Guilherme Sampaio	**Miguel Gomes**	**Rodrigo Pereira**
PG53851	PG54152	PG54198

1. Introdução

O presente relatório é referente ao trabalho prático da Unidade Curricular de Novos Paradigmas de Rede do Perfil de Redes de Nova Geração do Mestrado em Engenharia Informática da Universidade do Minho.

Sucintamente, o projeto consiste em implementar um sistema de semáforos inteligentes suportado apenas por comunicações veiculares (`V2X`), sem recurso a um semáforo físico. A implementação deste projeto recorre às tecnologias lecionadas na Unidade Curricular, entre estas, o `Eclipse Mosaic` e o `SUMO`, que permitem a simulação de redes num contexto rodoviário.

Neste documento, a Seção 2 ilustra e descreve o cenário utilizado durante o projeto e a Seção 3 discute a arquitetura do sistema e as suas diferentes partes em detalhe. Os tópicos relativos à implementação e dificuldades encontradas são discutidos na Seção 4, seguida da análise experimental e principais conclusões do projeto na Seção 5 e Seção 6, respetivamente.

2. Cenário Utilizado

O cenário é um dos componentes mais importantes para o projeto, permite o teste das aplicações desenvolvidas para os veículos e outras unidades de comunicação, assim como a visualização do fluxo de trânsito e respetivas interações com os elementos da rede.

O cenário, chamado "*Fifth Avenue*", representa o cruzamento entre as ruas "*East 16th Street*" e "*5th Avenue*" em Nova Iorque, Estados Unidos. Um excerto do cruzamento foi selecionado e exportado para *Open Street Map (.osm)*, sendo posteriormente convertido num cenário compatível com o `Eclipse Mosaic` através da ferramenta `scenario-convert`.

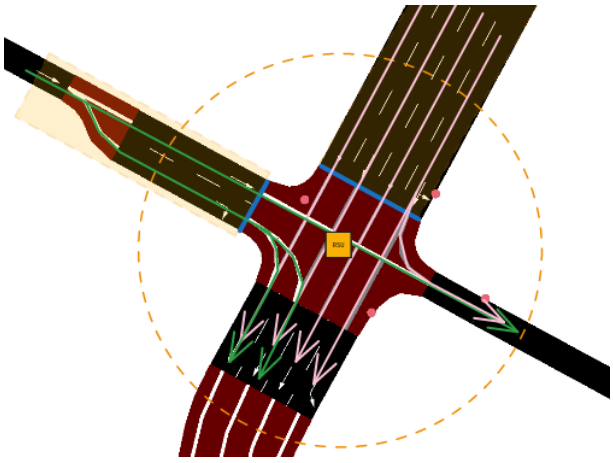

Figura 1: Cenário "*Fifth Avenue*".

A Figura 1 ilustra o cenário assim como indica a localização das unidades de comunicação (RSU), áreas de ação/alcance, localização dos "semáforos" e possíveis rotas para os veículos.

No cenário observamos a cor-de-laranja informações relativas à RSU (*Road-side Unit*). O ícone define a posição da RSU e a circunferência tracejada, centrada no ícone, representa o seu alcance. Veículos dentro desta área são capazes de comunicar diretamente com a unidade de comunicação fixa.

Os retângulos cor-de-laranja representam a área de deteção da RSU, veículos dentro desta área têm a possibilidade de comunicar via *topocast* com outras unidades de comunicação de modo a chegar à RSU. A RSU apenas tem em consideração veículos presentes na área de deteção.

As diferentes linhas sobrepostas às vias de trânsito representam as possíveis rotas[1] que os veículos podem tomar e as linhas azuis perpendiculares às vias indicam o local máximo de paragem dos veículos em caso de tal instrução.

Toda e qualquer configuração relativa ao cenário (número de carros, aplicações utilizadas, velocidades, etc.) pode ser encontrada no repositório do projeto.

3. Arquitetura do Sistema

O enunciado prático descreve o projeto através de etapas incrementais. Na Etapa 1 pretende-se um protótipo simplificado do serviço limitado a um único semáforo físico e ao raio de alcance deste e na Etapa 2 o estabelecimento de encaminhamento *multihop* nos veículos, de modo a aumentar a perceção do semáforo da área circundante. Por fim, a Etapa 3 discute a substituição do semáforo físico por um semáforo virtual.[2]

Nesta secção abordamos a arquitetura do sistema final, incluindo todas as etapas anteriormente descritas.

3.1. Elementos

3.1.1. *Road-side Unit* (RSU)

A *Road-side Unit* é uma unidade de comunicação fixa instalada no cruzamento (visível na Figura 1), que recebe e processa mensagens dos veículos equipados com unidades de comunicação (*On-board Units*).

Neste projeto a RSU lida dois tipos de mensagem:
1. Mensagens de Deteção – mensagens responsáveis pela deteção de veículos dentro da área de deteção.
2. Mensagens de Estado – mensagens responsáveis pela transmissão do estado do semáforo para os veículos.
3. Mensagens de Controlo – mensagens localizadas para controlar veículos de uma determinada rota e via de trânsito.

Ambas mensagens permitem a RSU ter a noção da quantidade de carros na sua área de deteção, de modo a permitir a execução do algoritmo responsável pela mudança de estados, efetivamente, eliminando a necessidade de um semáforo físico.

3.1.2. Veículos e *On-board Units* (OBU)

Todos os veículos possuem uma unidade comunicacional, *On-board Unit*, que permite, à semelhança da *Road-side Unit*, o envio e receção de mensagens. Além das mensagens descritas na Seção 3.1.1, os veículos, periodicamente, enviam *Cooperative Awareness Messages* (`CAM`).

As `CAM`'s são mensagens extremamente importantes no envio de informação entre veículos, neste projeto são utilizadas para a definição dos vizinhos de cada veículo, dado o seu alcance de comunicação.

[1] Todas as rotas foram geradas automaticamente pelo Open Street Map.

[2] Existe uma Etapa 4 de natureza opcional que não foi pensada nem implementada neste projeto.

Todas as mensagens referidas ao longo desta secção podem ser analisadas detalhadamente na Seção 4.

3.2. Comunicação

A comunicação utilizada neste projeto pode ser classificada em dois tipos: comunicação inter-veicular e comunicação entre veículo e *Road-side Unit*.

3.2.1. *Cooperative Awareness Messages*

As mensagens *Cooperative Awareness Messages* são um tipo de mensagem inter-veicular que permite o conhecimento da vizinhança de um nó. Mais especificamente, estas mensagens são, periodicamente, *broadcasted* por todos os veículos a um salto de distância tendo em conta o alcance do nó de modo a obter uma tabela de vizinhança constantemente atualizada.

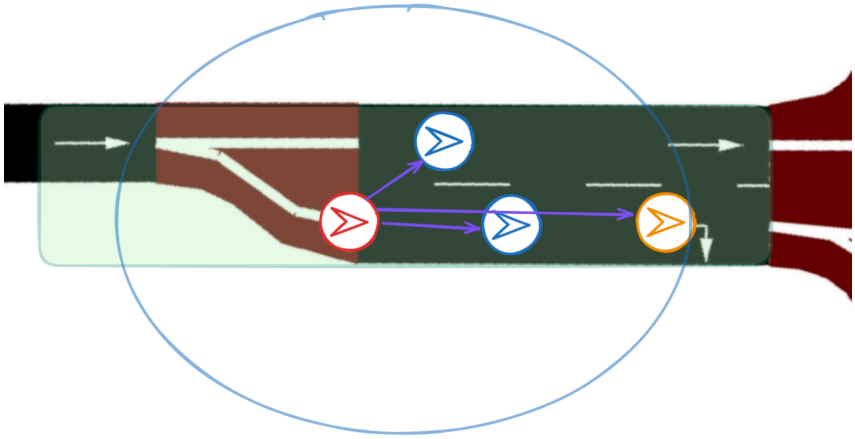

Figura 2: Transmissão de mensagens de controlo.

A Figura 2 ilustra o envio de mensagens CAM, *broadcasted* a cada meio segundo ($500ms$), e o comportamento na receção de uma `CAM` pode ser observado no algoritmo definido na Figura 3.

```
OnReceiveCAM(Message):
1  Neighbours: String ⟶ CAM = ∅
2  if Message.position() ≤ MAX_RANGE:
3      Neighbours[Message.id()] ⟵ Message
4  else:
5      Neighbours[Message.id()] ⟵ ∅
```

Figura 3: Receção de mensagens `CAM`.

A adição e remoção de veículos na vizinhança tem de ser feita do modo descrito acima, uma vez que no `SUMO` qualquer mensagem enviada, independentemente do alcance dos recetores, é recebida por todos os nós recetores. Cabe ao nó decidir se vai ignorar ou não a mensagem recebida tendo em conta vários parâmetros, como o alcance máximo do nó, neste caso.

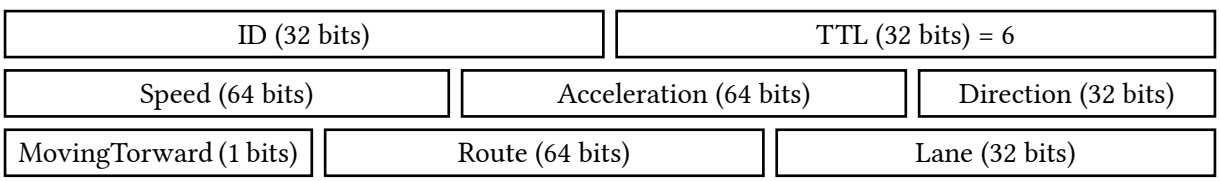

Figura 4: Estrutura de uma mensagem `CAM`.

A Figura 4 ilustra a composição de uma mensagem `CAM`. Todas as mensagens utilizadas neste projeto são serializadas automaticamente pelo `Java`, o que facilita o desenvolvimento. Porém, o uso da *interface* `Serializable` leva a um desperdício significativo de memória. Uma possível solução seria o *encoding* manual das mensagens em binário antes da sua transmissão.

As mensagens `CAM` também são recebidas pela *Road-side Unit*, com o mesmo objetivo de descoberta da vizinhança. Nas RSU's, a noção de vizinho indica o veículo que está na área de alcance da RSU, capaz de receber mensagens de estado. Já nos veículos os vizinhos são úteis da transmissão de mensagens destinadas à RSU quando ainda este não se encontre no seu alcance.

3.2.2. Mensagens de Estado

As mensagens de estado, são ambas inter-veículares como entre RSU e veículos, são mensagens V2X. O seu objetivo é informar os veículos na área de deteção do estado do semáforo virtual (mudança para vermelho, laranja ou verde).

É na receção de uma mensagem de estado que um veículo decide como agir. Se a mensagem exibir o estado `GREEN` então o carro prossegue a marcha, se exibir o estado `RED` então o carro para imediatamente e o mais próximo possível do limite máximo de travagem (definido na Figura 1). Caso a mensagem exiba o estado `YELLOW` ou `BLINKING` os veículos são forçados a reduzir a sua velocidade.

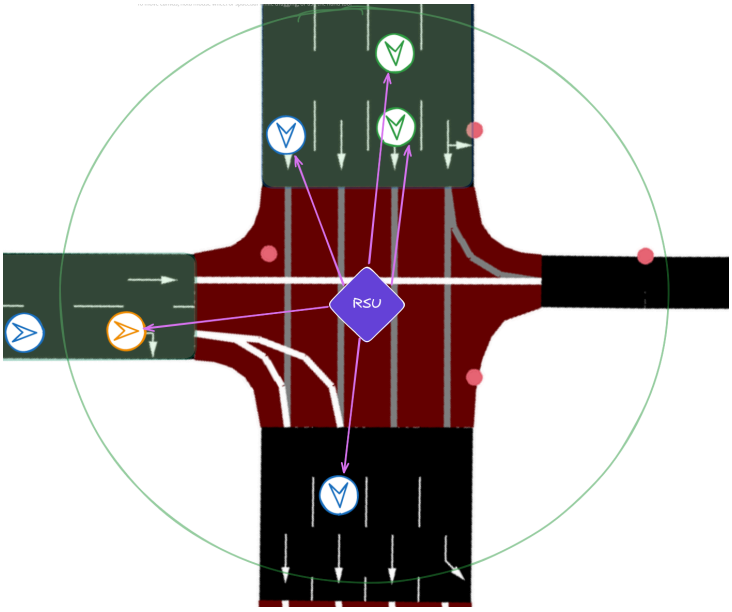

Figura 5: Transmissão de mensagens de controlo.

A Figura 5 ilustra a transmissão de uma mensagem de estado. Este tipo de mensagem é transmitida pela RSU, enviando periodicamente ($\Delta = 500ms$) a todos os veículos no seu alcance. Não existe a necessidade de encaminhar este tipo de mensagem aos veículos que estão na área de deteção, mas não no alcance da RSU, pois estes têm noção das ações realizadas pelos veículos na sua frente. Ou seja, se um veículo para, então todos os veículos à sua retaguarda também o vão fazer.

Assim garantimos que mesmo se os veículos não estiverem no alcance da RSU, conseguem agir conforme o estado do semáforo virtual.

ID (32 bits)	TTL (32 bits) = 6
IsOn (1 bit)	State (? bits)

Figura 6: Estrutura de uma mensagem de estado.

A Figura 4 representa a estrutura das mensagens de estado. É de notar que a gestão de estados é feita por rota e respetivas via de trânsito, portanto, o campo "*State*" da mensagem representa um mapa cuja chave é o nome da rota, para um mapa de vias de trânsito para estados, isto é, String → (String → Status), onde "*Status*" é um `Enum` cujos valores são `GREEN, RED, YELLOW, BLINKING`.

O campo "*IsOn*" é um booleano e indica se o sistema de "semáforo" está ligado ou não.

O grupo considera que esta mensagem é um ponto de otimização óbvio, dado que são envidados dados desnecessários, podendo afetar o desempenho do sistema. Uma solução seria o envio do estado referente apenas à rota dos veículos destino.

Nota: Este tipo de mensagem apenas foi utilizado durante a segunda etapa onde ainda existia um semáforo físico. A partir da Etapa 3 considera-se substituida pela mensagem de controlo, discutida na Seção 3.2.4.

3.2.3. Mensagem de Deteção

As mensagens de deteção são mensagens exclusivamente transmitidas entre os veículos e a RSU, quando o veículo se encontra na área de deteção da RSU, mas não no seu alcance. Este tipo de mensagem tem o intuito de alertar a *Road-side Unit* da presença de veículos na zona de deteção de uma determinada rota, de modo a quantificar os veículos e facilitar a mudança de estado e decisões do semáforo virtual.

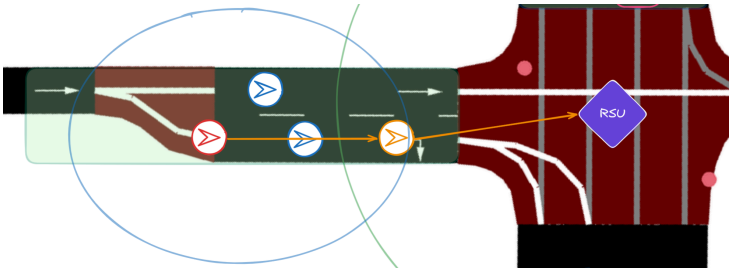

Figura 7: Transmissão de mensagens de controlo.

Os veículos que geram mensagens de deteção não estão no alcance da RSU, por isso, de modo a comunicar a mensagem, utilizam o método de transmissão *Topological Unicast*, como ilustrado na Figura 7. Este método consiste no sucessivo encaminhamento da mensagem para o veículo mais próximo da RSU, que esteja no alcance do veículo originador da mensagem, até que a mensagem chegue a um veículo que esteja no alcance da RSU e seja capaz de lhe transmitir diretamente.

```
OnReceiveDetection(Message):
1   if Self in RSU_RANGE:
2       SendToRSU(Message)
3   else:
4       ClosestToRSU ← GetClosestToRSU(Neighbours)
5       SendToVeichle(ClosestToRSU)
```

Figura 8: Receção de mensagens de deteção.

A Figura 8 descreve o algoritmo utilizado de maneira mais clara na receção de mensagens de deteção por parte de outros veículos.

ID (32 bits)	TTL (32 bits) = 6
Message (? bits)	
Route (32 bits)	Lane (32 bits)	Position (64 bits)

Figura 9: Estrutura de uma mensagem de deteção.

A Figura 9 define a estrutura de uma mensagem de deteção. A campo *Message* apenas existe por motivos de *debug*, podendo ser omitido nas mensagens. Os campos *Route*, *Lane* e *Position* contém informações relevantes para a tomada de decisões por parte da RSU, relativamente às mudanças de estado do semáforo virtual.

3.2.4. Mensagens de Controlo

As mensagens de controlo, como o nome dá a entender, controlam as ações tomadas pelos veículos. O seu objetivo, analogamente às mensagens de estado, é obrigar a uma reação por parte dos veículos aquando mudanças de estado no semáforo virtual. As mensagens de controlo transmitem uma ação que pode ser de entre `STOP`, `GO` e `SLOW_DOWN`.

São mensagens localizadas, pelo que, apenas contém ordens relativas a uma determinada via de trânsito numa rota qualquer. Estas mensagens são enviadas sempre que há uma mudança de estado no semáforo e são transmitidas a todos os veículos da na área de deteção da RSU, para a determinada rota e via.

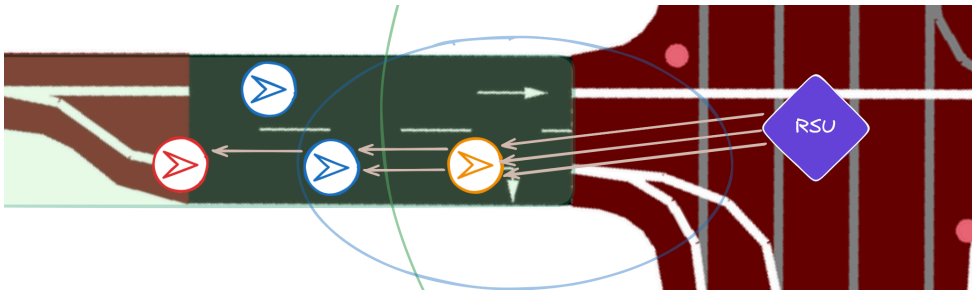

Figura 10: Transmissão de mensagens de controlo.

A Figura 10 ilustra a transmissão de uma mensagem de controlo.

Como a transmissão é feita para todos os veículos incluídos na área de deteção, proém, vão existir veículos que necessitem da mensagem, mas que não estão no alcance da RSU. Para combater este problema, a mensagem é inicialmente transmitida por *unicast* pela RSU a um dos veículos no seu alcance. Este veículo fica responsável por re-transmitir a mensagem recebida aos restantes veículos na via de trânsito afetada. Esta retransmissão é realizada mediante *topocast*, o mesmo método utilizado nas mensagens de deteção.

Devido ao comportamento dos veículos perante os restantes à sua frente, este tipo de mensagem torna-se redundante quando utilizado em conjunção com as mensagens de estado, daí a remoção das mensagens de estado da Etapa 2 para a Etapa 3. Com esta alteração não só aumentamos o desempenho do sistema ao enviar apenas a informação necessária a cada veículo como obtemos um controlo mais intrínseco sobre cada rota.

TTL (32 bits) = 6	Rule (32 bits)
ToWho (? bits)	GeneralLocation (64 bits)

Figura 11: Estrutura de uma mensagem de controlo.

A Figura 11 representa a estrutura de uma mensagem de controlo. O campo *Rule* é um valor inteiro (`Enum`) com as possíveis ordens atribuídas aos veículos: `STOP`, `GO` ou `SLOW_DOWN`. Já o campo *ToWho* identifica a via de trânsito e respetiva rota afetada pela regra contida na mensagem.

Finalmente, o campo *GeneralLocation* indica uma posição aproximada do destino da mensagem. Este valor é apenas utilizado quando o nó de retransmissão não possui na sua lista de vizinhos o destino. Sendo usado para calcular as distâncias de cada um dos vizinhos de modo a escolher o melhor candidato.

O *topocast* é um método de transmissão *single-hop* que serve como base para o protocolo de *greedy forwarding* com *multihop* implementado, logo este necessita sempre de um destino para encaminhar a mensagem recebida. Inicialmente, quando o destino não se encontrava na sua vizinhança, para reencaminhar uma dada mensagem, o veículo fazia um *broadcast* da mensagem de modo a chegar a alguém, porém este método não só quebra a regra do *greedy-forwarding* com *multihop* como prejudica imensamente o desempenho do sistema na totalidade devido à retransmissão da mensagem por toda a rede até o seu *TTL* expirar.

Agora, recorrendo a este campo, sabemos uma posição aproximada do veículo destino da mensagem, e, ao invés, de um *broadcast* é enviada a mensagem para o nó mais perto dessa área geográfica.

4. Implementação

Nesta secção discutimos as principais dificuldades encontradas e fazemos uma análise ao algoritmo de troca de estado do semáforo virtual implementado.

Relativamente às dificuldades encontradas, a principal e mais limitante foi a qualidade da documentação da API de `Java` do `Eclipse Mosaic`. O grupo considera que a tarefa mais temporalmente dispendiosa foi a aprendizagem de como programar neste contexto de redes veiculares, por exemplo, que métodos estão expostos ao programador e como os cenários e aplicações são internamente organizadas.

Além disso, o método de compilação e "*linking*" de bibliotecas do `Eclipse Mosaic` por vezes originava problemas, tendo de ser realizado um *script* de compilação para poupar tempo e trabalho.

4.1. Troca de Estado

O algoritmo de troca de estado do semáforo virtual é o componente mais importante de todo o sistema, sem ele não seria possível a manutenção e gestão do tráfego. Assimilando todas as mensagens descritas na Seção 3, a RSU possui, constantemente, informação relativa à quantidade, direção, velocidade, etc. (campos enunciados na Seção 3.2.1) dos veículos presentes nas áreas de deteção de cada rota.

Com isto o desenvolvimento do algoritmo depende da imaginação do programador. Neste projeto optamos por criar um algoritmo baseado em tempo e no número de veículos à espera/a chegar em cada rota.

```
UpdateState():
Periodically (60s) :
    RouteExceedingThreshold ← null
    for Route in RouteStates:
        if Route.waiting() ≥ THRESHOLD:
            RouteExceeding ← Route break
    if RouteExceedingThreshold ≠ null:
        RouteStates.setGreen(Route)
    else:
        RouteStates.invertStates()
    SendControlMessages(Neighbours)
```

Figura 12: Algoritmo de troca de estado.

A Figura 12 descreve o algoritmo utilizado. Geralmente, a cada minuto a cor dos sinais do semáforo é invertida, porém, caso uma das rotas exceda um *threshold* fixo em número de veículos à espera, o estado desta rota é mudado para verde. Apesar de não ser a maneira ideal de gerir tráfego, é um método simples e relativamente eficiente que aproveita toda a informação transmitida pelos veículos e RSU.

5. Análise Experimental

Para validar a implementação dos algoritmos de encaminhamento da rede veicular, o funcionamento do sistema de semáforos inteligentes e a comunicação entre dispositivos, utilizamos o `SUMO/Mosaic` para simular as redes veiculares. O cenário "*Fifth Avenue*", descrito na Seção 2, que representa um cruzamento em Nova Iorque, foi a base para as simulações. Durante os testes, configuramos vários parâmetros, incluindo o número de veículos, as suas velocidades máximas e as rotas que devem seguir.

A análise dos dados foi fundamental para a validação dos resultados. Utilizamos os *logs* criados durante as simulações para confirmar que os algoritmos de encaminhamento funcionavam conforme o esperado. Especificamente, analisamos os *logs* das viaturas, pois estes contêm um *log* de "*Forwarding X to Y*", que indica quando e para onde as do tipo *X* estão a ser reencaminhadas. Este tipo de *log* mostrou-se crucial para verificar o comportamento dos algoritmos de encaminhamento em tempo real, garantindo que as mensagens são transmitidas corretamente entre os veículos e as unidades de comunicação (RSU).

Os resultados mostraram que os veículos dentro do alcance da RSU conseguiam receber as mensagens de estado do semáforo virtual e agir de acordo com essas informações. Observamos também que os veículos fora do alcance da RSU, mas dentro da área de deteção, conseguiam enviar e receber mensagens através do método de transmissão *topocast*, confirmando o bom funcionamento do algoritmo de encaminhamento *multihop*.

Provar a correção de um sistema como o apresentado neste relatório através de palavras e imagens não é viável. Assim, o grupo produziu um vídeo exemplo com a demonstração do projeto a funcionar no sistema desenvolvido. O vídeo está disponível no YouTube através do endereço: https://youtu.be/OY2H8n9o-L0.

6. Conclusão

Ao longo deste trabalho, desenvolveu-se um sistema de semáforos inteligentes, sem qualquer tipo de recurso a semáforos físicos, utilizando apenas comunicação inter-veícular e entre unidades de comunicação estáticas (como RSU's). Com isto, o grupo considera ter cumprido com todos os objetivos acor-

dados, tendo este sido um projeto relativamente bem sucedido apesar das dificuldades encontradas, nomeadamente a utilização da `API` do `Eclipse Mosaic`.

No futuro, poder-se-ia fazer uma análise experimental mais detalhada, includindo análise de desempenho e escalabilidade do sistema, assim como algumas otimizações através da introdução de algoritmos de encaminahmento mais complexos, mediante os resultados obtidos da análise. Adicionalmente, gostariamos de ter implementado a Etapa 4, relativa à descentralização total do sistema e comunicação inter-semáforo, podendo correlacionar diferentes semáforos em diferentes áreas, de modo a obter uma melhor gestão do tráfego.