FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Implementação de manobras de reorganização de veículos em pelotão usando comunicações inter-veículares

Renato Alexandre da Silva Ferreira

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Orientador: Professor Rui Esteves Araújo

Co-Orientador: Engenheiro António Manuel Figueiredo Lopes

07 de Outubro de 2020

Abstract

Through the passage of years, automation of vehicles towards diverse scenarios has been increasing. Nowadays, projects study the vehicles' performance as a part of a platoon formation, with main focus on their security, stability and consumption. This wave of evolution brought new communication technologies throughout the last 30 years as a response for the communicative issues present in platoon formations. In order to solve some of those problems, the concept of IVC (Inter-Vehicle Communications) was created and enabled the implementation and execution of several maneuvers with the exchange of messages between vehicles of the same formation and, if necessary, with vehicles external to it. The maneuver implemented in this project uses IVC, more specifically DSRC (Direct Short Range Communication), based on IEEE 802.11p standard, with the goal of maintaining the vehicles' stability and security while it reorganizes the platoon formation upon the leader exit.

Throughout the document, it will be possible to evaluate the impact this maneuver may have in platooning world, avoiding the vehicles separation and consequent congestion of the road. Besides this, diverse factors that may have influence over the maneuver's execution will be analysed, such as the impact of the CACC (Cooperative Adaptative Cruise Control) parameters variation on the vehicles' stability, velocity, acceleration and distance between them; The loss of data during the communications and its impact on the maneuver failure rate; The bitrate variation's impact on cases where there's congestion and cases without congestion or interfering communications, using the values supported by the IEEE 802.11p standard and choosing the rate that benefits the maneuver.

All these analysis will help realize that this maneuver may be capable of responding with quality and a good performance, showing that it can be a great asset to the platooning world.

Resumo

Ao longo destes recentes anos, a autonomização dos veículos perante diversos cenários tem vindo a ser cada vez maior. Hoje em dia, há projetos que estudam o desempenho dos veículos quando inseridos em pelotões, tendo como principal foco a segurança, estabilidade e consumo dos mesmos. Com estas evoluções foram criadas novas tecnologias de comunicação ao longo dos últimos 30 anos face às necessidades comunicativas dentro dos pelotões. Para tal, foi criado o conceito de IVC (*Inter-Vehicle Communications*) através do qual se tornou possível implementar e executar todo o tipo de manobras graças à troca de mensagens entre veículos do mesmo pelotão e, se necessário, com veículos externos ao mesmo. A manobra implementada neste projeto recorre a IVC, mais concretamente DSRC (*Direct Short Range Communications*), baseada na norma IEEE 802.11p, para manter a estabilidade e segurança dos veículos em pelotão enquanto reorganiza a formação após saída do líder da mesma.

Será possível ver o impacto que a manobra poderá ter no mundo do *platooning*, impedindo separação dos veículos e consequente congestionamento das vias. Serão estudados diversos fatores determinantes e que podem ter influência na forma como a manobra é executada, como a variação dos parâmetros passados aos controladores CACC (*Cooperative Adaptative Cruise Control*) que têm impacto na estabilidade, velocidade, aceleração e distância entre os veículos do pelotão; A perda de dados nas comunicações e a taxa de falha de execução da manobra perante essas perdas; A melhor taxa de transmissão de dados para um cenário em que há congestionamento e outro em que não há, permitindo perceber qual o valor do *bitrate* entre os valores suportados pelo IEEE 802.11p, que beneficia a execução da manobra dependendo das condições em que esta é executada.

Todas estas análises permitirão chegar a uma conclusão que assenta na boa capacidade de resposta e desempenho desta manobra, podendo ser uma mais valia para o mundo do *platooning*.

Agradecimentos

As primeiras palavras de agradecimento são dirigidas aos meus pais que, apesar das adversidades ao longo destes 6 anos, sempre me deram o maior apoio.

Ao meu irmão, o meu maior pilar e exemplo, quero deixar um especial agradecimento pelo que fez por mim e por me ajudar em muitos aspetos académicos e pessoais.

Quero deixar o meu obrigado aos restantes familiares por acreditarem em mim e por me ajudarem a ser quem sou hoje.

Aos meus amigos mais próximos quero agradecer pela companhia, conselhos e momentos de descontração que proporcionaram quando mais precisei.

Aos amigos que fiz ao longo destes anos quero deixar um agradecimento pelos ensinamentos, pelas partilhas e por terem feito parte do meu percurso como estudante.

Quero agradecer ao Professor Doutor Michele Segata, criador do Plexe-Veins, por me ter ajudado a compreender melhor o simulador e por se mostrar sempre disponível para esclarecer as minhas dúvidas.

Por último, mas não menos importante, quero agradecer aos meus orientadores, Professor Rui Esteves Araújo e Engenheiro António Manuel Figueiredo Lopes, por me guiarem ao longo deste semestre, pela partilha de ideias e conhecimentos.

Muito obrigado!

Renato Ferreira

Dedico este trabalho aos meus pais e irmão, os meus pilares.

Renato Ferreira

Conteúdo

1 Introdução **1**
1.1 Veículos autónomos . . . 1
1.1.1 Obstáculos à total autonomização . . . 2
1.1.2 Outros desafios e formas de resolução . . . 3
1.1.3 *Platooning* . . . 5
1.2 Objetivos e estrutura da dissertação . . . 6

2 Estado da Arte **7**
2.1 Diferentes abordagens para condução autónoma . . . 7
2.2 Tecnologias de comunicação . . . 9
2.2.1 DSRC com base em IEEE 802.11p . . . 10
2.2.2 Cellular-V2X . . . 12
2.2.3 DSRC baseada em IEEE 802.11p vs C-V2X Release 14/15 . . . 13
2.2.4 5G . . . 15
2.3 Projetos Europeus . . . 16
2.4 Veículos com condução semi-autónoma . . . 22

3 Visão geral do sistema **27**
3.1 Formulação do problema . . . 27
3.2 Solução proposta . . . 28
3.3 Arquitetura do simulador . . . 29
3.4 Partilha de informação para o CACC . . . 31
3.5 Organização hierárquica do pelotão . . . 34
3.6 Modo de funcionamento . . . 35
3.7 Atuação dos intervenientes . . . 36

4 Simulação e análise de resultados **41**
4.1 Simulação sem perdas . . . 41
4.2 Simulação com perdas . . . 54
4.3 Impacto da variação do *bitrate* no desempenho da manobra . . . 56
4.3.1 Cenário descongestionado . . . 57
4.3.2 Cenário congestionado . . . 58
4.4 Comparação com outras abordagens . . . 59

5 Conclusão **61**
5.1 Conclusões principais . . . 61
5.2 Trabalhos futuros . . . 62

Referências **65**

A **71**

B **75**

B.1 LeaveManeuver.cc 75
B.2 LeaveManeuver.h 81
B.3 Leader_Leave.cc 89
B.4 Leader_Leave.h 97
B.5 LeaveManeuverScenario.cc 105
B.6 LeaveManeuverScenario.h 107
B.7 omnetpp.ini 109

Lista de Figuras

1.1 J3016^{TM} "*Levels of Driving Automation*" - Gráfico demonstrativo dos níveis de automação e respetivas características. 2
1.2 Importância da introdução do V2X para a obtenção dos níveis 3-5 de automação [12]. 5

2.1 Esquema temporal dos intervalos de acesso aos canais SCH e CCH [27]. 11
2.2 Interfaces de comunicação para comunicações diretas e comunicações com a rede [22]. 12
2.3 Tipos de comunicação suportada no projeto 3GPP [28] 12
2.4 Transmissão de mensagens utilizando modelo híbrido de PC5 e Uu [28]. 13
2.5 Semelhanças e diferenças entre uma tecnologia DSRC e C-V2X 13
2.6 Comparação em termos de tempo de reação e alcance entre C-V2X e 802.11p DSRC [22]. 14
2.7 Diferenças de comprimento de onda/frequência entre as principais bandas aplicáveis a uma rede celular [34]. 16
2.8 Arquitetura de alto nível do projeto ICT4CART [37]. 19
2.9 Arquitetura de alto nível do projeto ENSEMBLE [37]. 21

3.1 Principais manobras de pelotões estudadas [14]. 27
3.2 Comunicação intra pelotão através de protocolo de *beaconing* [41]. 28
3.3 Arquitetura do Veins e respetivas interfaces com OMNeT++ e SUMO [47]. . . . 29
3.4 Representação das camadas de comunicação do Veins [47]. 30
3.5 Arquitetura do Plexe-Veins e respetivas interfaces [42]. 31
3.6 Diagrama de estado para o *Leader* associado ao cenário 1. 37
3.7 Diagrama de estado para o *Leader* associado ao cenário 2. 37
3.8 Diagrama de estado para o *Leader* associado ao cenário 3. 38
3.9 Diagrama de estado para o *Leader* associado ao cenário 4. 38
3.10 Diagrama de estado para o *Available* associado ao cenário 1. 39
3.11 Diagrama de estado para o *Available* associado ao cenário 2. 39
3.12 Diagrama de estado para o *Available* associado ao cenário 3. 39
3.13 Diagrama de estado para o *Follower*. 40

4.1 Variações nas acelerações, velocidades e distâncias entre os veículos para o primeiro cenário. 43
4.2 Variações nas acelerações, velocidades e distâncias entre os veículos para o primeiro cenário. 44
4.3 *Frames* da manobra no cenário 1. 45

4.4 Variações nas acelerações, velocidades e distâncias entre os veículos para o segundo cenário. 46
4.5 Variações nas acelerações, velocidades e distâncias entre os veículos para o segundo cenário. 48
4.6 *Frames* da manobra no cenário 2. 49
4.7 Variações nas acelerações, velocidades e distâncias entre os veículos para o terceiro cenário. 50
4.8 Variações nas acelerações, velocidades e distâncias entre os veículos para o segundo cenário. 52
4.9 *Frames* da manobra no cenário 3. 53
4.10 *Frames* da manobra no cenário 4. 54
4.11 Impacto da perda de pacotes no desempenho da manobra. 55
4.12 Número médio de pacotes entregues durante a execução da manobra. 56
4.13 Percentagem de falha da manobra perante falha de pacotes para diferentes valores de *bitrate* num cenário sem congestiomento de trânsito. 57
4.14 Percentagem de falha da manobra perante falha de pacotes para diferentes valores de *bitrate* num cenário com congestiomento de trânsito. 58

Lista de Tabelas

1.1 Cenários de perigo e empresas responsáveis pelo desenvolvimento de uma solução para o respetivo problema. 4

3.1 Terminologia das mensagens e respetivos dados adicionais. 34

4.1 Especificações das simulações. 41
4.2 Especificações referentes aos veículos externos das simulações. 55

Acrónimos

ACC	*Adaptive Cruise Control*
C-V2X	*Cellular Vehicle to Everything*
CACC	*Cooperative Adaptive Cruise Control*
CC	*Cruise Control*
DSRC	*Dedicated Short Range Communications*
IVC	*Inter-Vehicle Communications*
LiDAR	*Light Detection And Ranging*
LTE	*Long Term Evolution*
RaDAR	*Radio Detection And Ranging*
SINR	*Signal-to-Interference + Noise Ratio*
SNR	*Signal-to-Noise Ratio*
VANET	*Vehicular Ad-hoc Network*
V2I	*Vehicle-to-infrastructure*
V2N	*Vehicle-to-network*
V2R	*Vehicle-to-roadside*
V2V	*Vehicle-to-vehicle*
V2X	*Vehicle-to-everything*

Capítulo 1

Introdução

Ao longo dos últimos 30 anos, a ideia de condução autónoma tem vindo a ganhar força. O que nos anos 90 era matéria de ficção científica começa, hoje em dia, a tornar-se uma realidade cada vez mais eminente. Grandes empresas automóveis como a BMW [1] [2], Mercedes [3], Continental [4] [5], entre outras [1], têm vindo a tomar uma posição preponderante no que toca ao estudo de iniciativas focadas no desenvolvimento desta tecnologia.

Para além da enorme margem de inovação do mundo automóvel, toda esta ideologia traz também vantagens no que diz respeito a segurança rodoviária. Sabe-se que, em Portugal, segundo o sítio da PORDATA [6], o número de acidentes de viação, consequentes feridos e mortos têm assumido uma tendência crescente. Com a evolução desta tecnologia que eleva, cada vez mais, a capacidade de autonomizar os veículos, estes problemas causados principalmente pelo ser humano podem vir a ter impactos atenuados. Tal acontece porque o ser humano e as suas ações (condução negligente, excesso de velocidade, condução sob o efeito de álcool, uso de telefone durante a condução, etc.) são retirados da equação.

1.1 Veículos autónomos

Existem diversos aspetos a ter em conta quando se fala de veículos totalmente autónomos tais como os seus níveis de automação, os principais obstáculos para que se possa alcançar o nível máximo de automação e soluções para ultrapassar esses mesmos obstáculos.

Para se perceber os que são os níveis de automação, a SAE (*Society of Automotive Engineers* [7]) lançou o gráfico J3016^{TM} "*Levels of Driving Automation*" - figura 1.1 -, no qual se pode ver que há seis níveis de automação. Os primeiros três níveis apresentam funcionalidades do sistema para automação com características orientadas para o suporte do condutor. Nos restantes três níveis

[1] Muitas das principais companhias automóveis estabelecem parcerias com empresas mais focadas no desenvolvimento da tecnologia necessária para a autonomização dos veículos.

o sistema autónomo passa a ser a figura principal, sendo responsável pelo controlo do veículo dentro de certas condições ou, em caso de automação total, sem qualquer tipo de limitações.

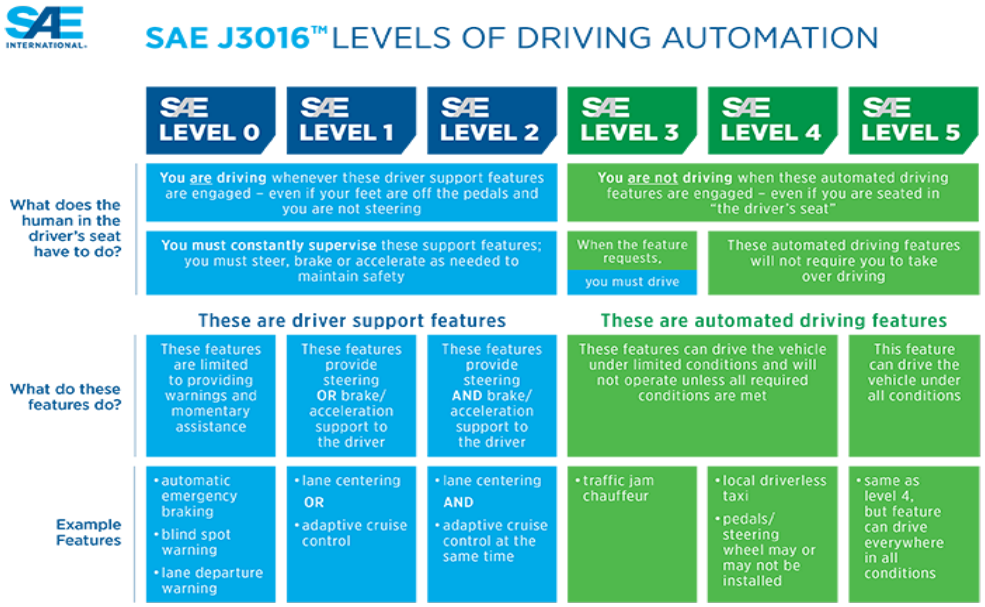


Figura 1.1: J3016^{TM} "*Levels of Driving Automation*" - Gráfico demonstrativo dos níveis de automação e respetivas características.

1.1.1 Obstáculos à total autonomização

Tal como se aprofundará no Capítulo 2, secção 2.4, existem imensos modelos de diferentes marcas que já apresentam uma face semi-autónoma capaz de auxiliar e, em alguns casos, substituir o condutor. Em prol do desenvolvimento de veículos totalmente autónomos, múltiplos esforços têm sido feitos, mas, de acordo com o sítio *MIT Technology Review* [8], há três problemas a atrasar o processo de total autonomização dos veículos:

- Segurança do veículo - Este talvez seja o maior desafio tendo em conta que a população mundial ainda tem alguma dificuldade em aceitar e confiar num veículo autónomo. Em Portugal, segundo um artigo lançado pelo sítio PPLWARE [9], 71% das pessoas nascidas entre 1950 e 1960, 68% das pessoas nascidas entre 1961 e 1981 e 64% dos *Millennials* não confiam na tecnologia. Portanto, seria importante conceber um veículo capaz de tomar decisões ao nível de um condutor experiente e exemplar para aumentar os índices de confiança da população;

- Capacidade de decisão do veículo - Saber quando moderar a velocidade ou acelerar, quando ultrapassar, quando parar, entre outras ações, é um dos desafios a ultrapassar, havendo ainda muito por onde evoluir. Tem de ser feito um vasto estudo sobre tomada de decisão de modo a criar limites para o que é uma decisão aceitável ou não;

- Preço do veículo - O custo acaba por ser o último dos desafios a superar tendo em conta que só o sistema autónomo poderá custar dezenas de milhares de euros/dólares, sendo necessário avaliar a viabilidade comercial e financeira da disponibilização desta tecnologia.

Para além destes três problemas, é necessário ter em conta ainda outros fatores determinantes para que a total automação do veícuo seja possível:

- Limitações técnicas - A capacidade de processamento dos sistemas autónomos, as infraestruturas necessárias para o lançamento de veículos autónomos na estrada, bem como as motivações dos principais OEMs (*Original Equipment Manufacturer*) para o desenvolvimento dos equipamentos necessários para alcançar esse fim devem ser balanceados de modo a beneficiar todas as frentes envolvidas;

- Confiabilidade do sistema - Tem de haver um vasto estudo para garantir que os sistemas autónomos implementados respeitam os requisitos de segurança e de correto funcionamento ao longo dos seus ciclos de vida;

- Regulamentação - Sendo que diferentes países ou até mesmo diferentes zonas de um país podem ser reguladas por diferentes entidades ou até usufruir de serviços com diferentes características, é necessário que os sistemas se consigam adaptar a estas mudanças notadas, por exemplo, em travessias de fronteiras.

1.1.2 Outros desafios e formas de resolução

Para além dos desafios mencionados na secção 1.1, existe também o problema que advém da inserção destes veículos em meios de realidades diferentes. Tal como foi mencionado em [10], o lançamento destes veícuos deverá ser feito em espaços controlados ou meios urbanos desenvolvidos. Por outro lado, será mais difícil solucionar o problema num meio rural devido às dificuldades a nível de mapeamento das vias e condicionamento das mesmas.

Toda esta inclusão dos veículos autónomos no meio urbano requer um estudo e análise profundos do comportamento de um sistema de características autónomas em cenários concretos e comuns que acontecem todos os dias nas estradas. A seguinte tabela, adaptada de [11], mostra os principais problemas existentes na via pública e as empresas que, numa fase mais precoce, tiveram maior influência na procura e descoberta de soluções para esse mesmos problemas[2]:

[2] A primeira empresa de cada linha da tabela corresponde ao líder de aplicação

Cenário	Aplicação	Líder de aplicação e participantes
Colisão lateral	Segurança em interseções	CAS, TNO, BOSCH, RENAULT
	Mudança de faixa de rodagem	PIAGGIO, VOLVO, RENAULT, BOSCH, CAS
	Segurança na ultrapassagem	PIAGGIO, CRF, BOSCH, TNO, CAS, MMSE, CA
Colisão longitudinal	Colisão frontal	VOLVO, CRF, BOSCH, TNO, CAS, MMSE
	Colisão traseira	CRF, BOSCH, MMSE
	Limites de velocidade e distância de segurança	MMSE, VOLVO, RENAULT, BOSCH, TNO
Condições da estrada	Estrada escorregadia	VOLVO, CRF, RENAULT, BOSCH, TNO, PIAGGIO, VTT
	Aviso de curva	CRF, BOSCH, PIAGGIO, MMSE
Usuários da via vulneráveis	Deteção de usuários vulneráveis	VOLVO, RENAULT, BOSCH, PIAGGIO, VTT

Tabela 1.1: Cenários de perigo e empresas responsáveis pelo desenvolvimento de uma solução para o respetivo problema.

Para resolver estes problemas, os desenvolvedores acima mencionados poderiam adotar uma abordagem sensorial e/ou uma abordagem comunicativa. Apesar de um sistema sensorial conseguir desempenhar as funções autónomas de um veículo por si só, a introdução de comunicações V2X (*Vehicle-to-everything*) revela-se bastante pertinente, pois permite que haja um maior controlo do meio envolvente e das ações dos corpos que circulam na via graças à partilha de informação sobre cada utilizador da via através destas comunicações.

Apesar de haver diferenciação nas abordagens a adotar de modo a obter um sistema de condução autónomo, a fusão do espectro comunicativo e sensorial surge como o paradigma mais viável. Segundo uma apresentação levada a cabo por Ravi Puvvala, CEO da Savari[12], uma das maiores empresas no ramo de tecnologia V2X, no âmbito de um *workshop* inserido no programa do IWPC (*International Wireless Packaging Consortium*) de 2018, os veículos de hoje em dia apresentam o nível 2 de automação. De modo a entrar na gama dos veículos que possuem capacidade de desempenhar uma condução sem necessidade de um condutor, a introdução do V2X é vista como um bem essencial, tal como se pode ver pela figura 1.2.

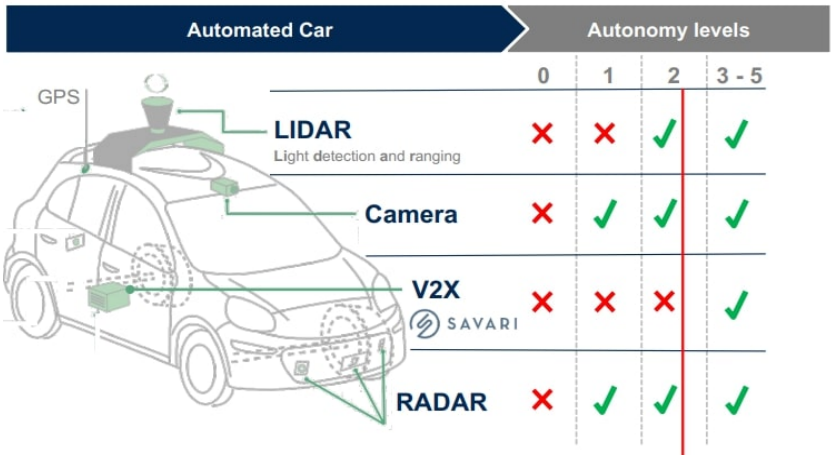


Figura 1.2: Importância da introdução do V2X para a obtenção dos níveis 3-5 de automação [12].

Este casamento entre a componente sensorial e componente comunicativa constitui um enorme passo rumo à aquisição de total autonomia dos veículos. Para além disso, constitui um enorme avanço no que diz respeito à possibilidade de *platooning*.

1.1.3 *Platooning*

Platooning consiste em alinhar diversos veículos numa das faixas de rodagem mantendo entre eles um espaçamento específico através de sensores e mensagens do tipo *beacon*. Perante esta definição, há imensos aspetos que podem ser estudados e projetos que podem ser desenvolvidos como forma de melhorar o desempenho, segurança e organização dos veículos na via pública.

Através da comunicação entre os veículos constituintes do pelotão, é possível executar certas ações em conjunto que facilitam a circulação na via como a ultrapassagem, mudança para um faixa mais vantajosa, mudanças de velocidade, entre outras. Em termos de consumo dos veículos, pode também ter um impacto positivo em questões relacionadas com diminuição do consumo de combustível, tal como se pode verificar no artigo [13].

No entanto, para que haja a formação de um pelotão é necessária a conjugação de alguns requisitos - a formação deve ter um líder que é responsável pela organização do pelotão e por orientar os seus seguidores para que estes ajam em conformidade com ele; a formação deve ter um tamanho superior a 1, senão não teria sentido haver a criação de um pelotão pelo que o veículo circularia sozinho. Para que este último requisito seja possível, é necessária a criação de uma manobra que possibilite a junção de um veículo num determinado pelotão. Assim como há uma manobra para entrada de veículos também se torna necessária a criação de uma outra manobra responsável pela saída de veículos da formação de veículos. É aqui que surge um problema consequente à saída do membro mais importante do pelotão - o líder. A saída do líder tem duas soluções que consistem na eleição de um novo líder ou na dissolução do pelotão. A segunda solução seria aquela mais fácil de realizar, no entanto poderia trazer alguma desorganização e congestionamento na via pública,

pelo que seria benéfico manter de alguma forma a formação intacta. Para tal, aplica-se a primeira solução.

Após a eleição do novo líder, é preciso que este assuma a posição primária do pelotão independentemente da sua posição original. Para este efeito, não existe propriamente uma manobra que execute esta reorganização dentro do grupo após a saída do líder original apesar de em [14] ser apresentada uma proposta para tal. Contudo, neste estudo não há a execução das trocas posicionais e hierárquicas dentro do pelotão, assumindo que o novo líder é o primeiro seguidor, pelo que seria interessante a criação de uma manobra que efetivamente procedesse à reorganização no espaço da formação, algo a ser falado com mais detalhe a partir do capítulo 3.

1.2 Objetivos e estrutura da dissertação

Este documento referente ao projeto de dissertação tem como função explicar alguns aspetos básicos sobre condução autónoma, convergindo na implementação de uma manobra capaz de reorganizar um pelotão cujo líder deseje sair através de IVC. Os objetivos do projeto passam por:

- Exploração de trabalhos relacionados de forma a definir o problema a ser tratado;
- Estudo de simuladores capazes de fornecer ferramentas que permitam estudo de pelotões;
- Implementação de uma manobra capaz de lidar com os problemas apresentados de forma eficaz;
- Teste e estudo dos resultados para garantir a fiabilidade da utilização do mesmo.

No que diz respeito a este documento, neste capítulo inicial foi dada uma breve introdução acerca do que representa a condução autónoma em termos de níveis de autonomização e quais os principais obstáculos a ultrapassar para chegar a esse fim - condução totalmente autónoma. Para encerrar o capítulo introdutório, foi também abordado o problema a ser tratado neste projeto. No segundo capítulo será falado o estado da arte referindo as diferentes abordagens e suas tecnologias. No terceiro capítulo será avaliado e aprofundado o problema a ser tratado, serão explicadas as principais tecnologias e métodos utilizados para chegar à solução proposta. No capítulo seguinte, esta solução será esmiuçada ao ponto de se poder perceber toda a lógica e interação dos participantes na manobra. Seguidamente, será apresentado um capítulo destinado ao estudo dos resultados obtidos nos testes propostos e realizados, sendo o documento finalizado com o capítulo conclusivo onde dar-se-á o parecer global do projeto.

Capítulo 2

Estado da Arte

Este capítulo tem como principal finalidade apresentar a pesquisa feita em relação a abordagens e tecnologias para efeitos de condução autónoma, projetos existentes e modelos de veículos com funcionalidades autónomas.

2.1 Diferentes abordagens para condução autónoma

No que diz respeito às diferentes abordagens, como foi explicado na secção 1.1.2 do Capítulo 1, pode-se recorrer a métodos mais relacionados com a parte sensorial e a outros com uma vertente mais direcionada para as comunicações entre veiculos diferentes e/ou entre veículos e infraestruras pertencentes ao mundo rodoviário.

Abordagem sensorial

Partindo da perspetiva sensorial do problema, os dados sobre o ambiente em que o veículo se insere são obtidos através de sensores ou outros dispositivos que ajudem a captar e fornecer informação sobre determinados acontecimentos que requeiram uma intervenção do sistema autónomo.

De acordo com o sítio *Level Five Supplies* [15] existem duas categorias de sensores:

- **Sensores ativos** que são capazes de emitir uma onda com determinadas propriedades e analisar as reflexões destas mesmas ondas de modo a analisar o espaço exterior. Os principais exemplos são [1]:
  - **RaDAR** (*Radio Detection And Ranging*) é um dispositivo que permite detetar objetos e calcular as suas respetivas distâncias, neste caso, ao veículo em que estiver integrado.

[1] Os exemplares de sensores escolhidos partem da escolha de uma das empresas presentes na última linha da tabela apresentada no blog *UnitedLex* [16].

Porém, apresenta uma fraca capacidade para identificar os objetos detetados. Em termos de frequência, por norma, as ondas inserem-se na gama dos 77GHz para uso em longas distâncias e 24GHz para curtas distâncias.

  * A BOSCH, entre outras empresas, tem investido no desenvolvimento de tecnologia sensorial. Um dos seus mais conhecidos dispositivos que implementa um RADAR é o LRR4 (*Long-Range RADAR 4th-Generation*) [17] que se trata de uma RADAR monoestático multimodal com seis antenas. Opera com um espectro de frequências entre os 76 e 77 GHz, com um alcance entre 36 centímetros e 250 metros.

- **LiDAR** (*Light Detection and Ranging*) é um dispositivo usado desde os tempos mais precoces dos veículos autónomos. O seu funcionamento baseia-se na emissão de feixes de laser que intercetam os objetos e a sua reflexão permite recriar o espaço em três dimensões. Ao contrário do RaDAR, esta tecnologia permite identificar os objetos detetados.

  * O *Velodyne Alpha Prime* ^{TM} [18] concebido pela *Velodyne Lidar* é um dos melhores dispositivos existentes no momento, com alcance de 360º na horizontal e 40º na vertical, apresentando ainda uma resolução de 0.2ºx0.1º. Em termos de distância, pode alcançar os 220 metros, utilizando um laser de 903 nanómetros de comprimento de onda.

- **Sensor ultrasónico** é um pequeno dispositivo já incorporado em veículos desde a introdução de sensores de estacionamento. Funciona como um complemento aos restantes sensores, permitindo apurar a sensibilidade em situações em que o veículo circule a uma velocidade muito baixa, principalmente em manobras de estacionamento.

  * Mais uma vez, a BOSCH apresenta uma contribuição no desenvolvimento destes sensores. Neste caso em concreto, os USS6 (*UltraSonic Sensors 6th Generation*) [19] vêm aumentar em cerca de 20% o alcance de deteção em relação à geração anterior de sensores ultrasónicos lançados pela BOSCH. Este dispositivo apresenta ainda um alcance máximo de distância até 5.5 metros e mínimo de 15 centímetros. Tem uma alcance horizontal de 70º e vertical de 35º.

- **Sensores passivos** que não apresentam a mesma forma de análise do que os ativos, sendo que apenas recebem informação do exterior sem emitirem qualquer tipo de onda:

  - **Câmera** é um dispositivo comum a muitas funcionalidades do dia-a-dia. É uma tecnologia à qual a comunidade já se adaptou e o seu uso em veículos autónomos traz bastantes benefícios. Através da câmera, será possível detetar e distinguir as marcas rodoviárias existentes nas estradas, assim como classificar objetos de uma forma precisa. Existem empresas que preferem incluir múltiplas câmeras estrategicamente posicionadas para o reconhecimento de imagem. No entanto, a ideia de uma única câmera com visão panorâmica de 360º tem vindo a mostrar-se uma alternativa válida.

* A MPC3 (*Multi Purpose Camera*) [20], desenvolvida pela BOSCH, é constituída por um SoC (*System-on-Chip*) com um microprocessador, também concebido pela mesma empresa, capaz de fazer processamento de imagem usando métodos de inteligência artificial de modo a permitir uma melhor interpretação do que está a ser analisado e distinguir os objetos detetados.

Como forma de aumentar a capacidade sensitiva e melhorar a resposta do veículo, a conjugação de diferentes tipos de sensores é vista como a solução mais viável e segura.

Abordagem Comunicativa

Por outro lado, partindo do espectro das comunicações, surge uma panóplia de alternativas como solução do problema da condução autónoma por sensorização. Grandes empresas da indústria automóvel têm vindo a aprofundar os seus conhecimentos no que toca a comunicações, neste caso, do veículo. Apesar de o uso de comunicações para melhorar o desempenho do sistema autónomo ser visto como necessário, é importante referir que a abordagem comunicativa por si só não garante a autonomização do veículo, precisando sempre de ser complementada pelo lado sensorial do sistema.

Todas as formas de interação do veículo com o meio que o rodeia inserem-se na categoria de comunicações V2X. De acordo com um artigo publicado pelo IEEE (*Institute of Electrical and Electronics Engineers*) sobre um estudo levado a cabo por Weijie Liu, Shintaro Muramatsu e Yoshiyuki Okubo [21], estas comunicações dividem-se em sub-categorias como V2V (*Vehicle-to-vehicle*) e V2I (*Vehicle-to-infrastructure*), V2P (*Vehicle-to-Pedestrian*) e V2N (*Vehicle-to-Network*). Como já se mencionou, V2X engloba todos os tipos de comunicações, seja com outros veículos ou infraestruturas. V2V, por sua vez, é a comunicação feita entre veículos, enquanto que V2I representa a comunicação feita entre um veículo e uma infraestrutura. Por fim, V2P são as comunicações feitas entre veículos e pedestres e V2N são as comunicações entre os veículos e o servidor de rede.

2.2 Tecnologias de comunicação

Em relação às tecnologias usadas nestas comunicações, tem vindo a ser cada vez maior a discussão entre utilização de C-V2X (*Cellular Vehicle-to-everything*) ou DSRC (*Direct Short Range communication*). Partindo do pressuposto de que ambas operam na gama dos 5.9 GHz (apesar de C-V2X apresentar um leque enorme de frequências, sendo 3.5 GHz a mais comum), terão modos de operação diferentes uma da outra, visto que se tratam de tecnologias que adotam métodos comunicativos diferentes.

De acordo com [22], a norma 802.11p representa a base para a comunicação V2X, surgindo então, através dessa mesma norma, a DSRC. Derivada do Wi-Fi, foi otimizada a nível de custos e, apesar da sua simplicidade, é uma tecnologia muito eficaz a trabalhar com operações distribuídas, segundo [23].

Mais recentemente, numa tentativa de melhorar o alcance, eficácia da comunicação e o leque de casos de uso e de informação sobre os acontecimentos na estrada, surgiu uma tecnologia derivada de LTE, da 3GPP (*3rd Generation Partnership Project*) - o C-V2X. Esta tecnologia introduz mecanismos capazes de melhorar o desempenho entre operações distribuídas, permitindo também comunicações com a rede, aumentando as possibilidades de serviços e aplicações.

2.2.1 DSRC com base em IEEE 802.11p

Tal como é referido em [24], em 1999, o espectro de frequências destinado a DSRC, 75 MHz, foi alocado na faixa dos 5.9GHz pela *Federal Communications Comission* nos EUA, com o intuito de ser usado para fins de comunicação inter-veícular e entre veículos e estruturas. Já a Europa, através do *European Telecommunications Standards Institute*, alocou 30 MHz na mesma faixa, em 2008.

Como é explicado em [25], a tecnologia DSRC é baseada na norma IEEE 802.11a, mas com ajustes para operações com baixo *overhead* dentro do espectro a si destinado. Passou, então, a ser designada por IEEE 802.11p.

O espectro do DSRC é dividido em sete diferentes canais, tendo:

- Um canal central de controlo apenas para comunicações relacionadas com segurança;
- Dois canais nas extremidades para tarefas relacionadas com prevenção de colisões;
- Os restantes canais são utilizados para tarefas diversificadas.

Ainda partindo do mesmo artigo, sabe-se que a camada de ligação das comunicações de segurança dentro do canal de controlo possui as seguintes características:

- ***Single-hop* -** Grande parte das aplicações de segurança baseiam-se em comunicações diretas entre veículos;
- **Descoordenado -** Comunicações são de cariz distribuído;
- ***Broadcast* -** Comunicações têm como alvo veículos tendo em conta a sua localização e não identidade;
- **Mensagens -** Mensagens curtas de dados com informação pertinente para aplicação de segurança.

Dentro de um sistema:

- **Sem limites -** A comunicação não tem limites, podendo a informação ser propagada sem haver preocupações com limites;
- **Com veículos igualmente equipados -** Todos os veículos estão equipados com DSRC, permitindo uma frequente comunicação entre eles;

- **Com diferenciação de canais -** Todas as mensagens de segurança são trocadas no canal central, trocando outro tipo de mensagens apenas em casos excecionais.

Como se verá posteriormente, na secção 3.2, mais especificamente na subsecção 3.3, a norma IEEE 802.11p é combinada com a norma IEEE 1609.4 na camada de comunicações MAC para operações multi-canal. Esta norma revela uma importância acrescida dado que é importante manter o canal central livre para as mensagens de segurança que apresentam maior prioridade.

IEEE 1609.4

Para que haja uma correta utilização dos diferentes canais destinados a DSRC, é necessária a correta sincronização entre os intervalos em que o acesso aos canais de serviço (SCH) e ao canal central de controlo (CCH) é permitido. De acordo com o estudo [26], esta sincronização e utilização multi-canal é possibilitada pelo protocolo IEEE 1609.4 facilitando a comunicação entre dispositivos que suportam comunicações com base na norma IEEE 802.11p.

A figura 2.1 demonstra um esquema que permite perceber como se organiza temporalmente o acesso aos canais SCH e CCH.

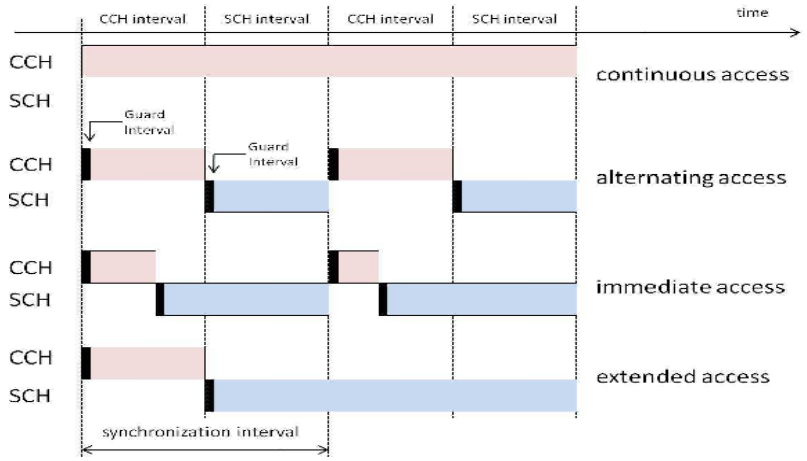


Figura 2.1: Esquema temporal dos intervalos de acesso aos canais SCH e CCH [27].

Para apoiar a explicação deste esquema temporal, é importante mencionar que um par de intervalos destinados ao canal CCH e SCH representa um intervalo de sincronização, como é referido em [26] e [27]. Como cada intervalo de canal é de 50 ms para o acesso alternado, o intervalo de sincronização terá um valor máximo de 100 ms, permitindo cumprir o requisito de segurança que define que aplicações de segurança não toleram latências superiores a esse mesmo valor. Para acesso contínuo, usado na manobra estudada neste projeto, o CCH tem acesso ilimitado a nível temporal, não tendo de se preocupar com um possível acesso aos canais de serviço. No acesso imediato, os canais SCH podem proceder à comunicação sem esperar pelo final do intervalo atribuído ao CCH, permitindo evitar atrasos residuais provocados pelas comunicações deste último. Por fim, o acesso extendido permite que os canais SCH comuniquem sem ter de parar para o intervalo do CCH.

Para a manobra a ser implementada neste projeto, será usado apenas o canal de controlo central. O impacto causado pela utilização desse único canal não será significativo visto que apenas serão trocadas mensagens de cariz crítico para o bom desempenho do mesmo, representando assim tarefas de prioridade e necessidade de segurança máximas, como se verá nas secções referentes à estrutura e implementação da manobra (3) e simulação (4). No entanto, é importante perceber como se desenrola este processo de comunicação multi-canal.

2.2.2 Cellular-V2X

Tal como é explicado através da figura 2.2, existem dois modos de transmissão.



Figura 2.2: Interfaces de comunicação para comunicações diretas e comunicações com a rede [22].

As comunicações diretas, por exemplo, entre veículos, usam uma **interface PC5** construída sobre LTE (*Long Term Evolution*) *Direct Device-to-device*, *Release 14*, com melhorias para altas velocidades e em termos de sincronização e baixa latência.

Por outro lado, as comunicações com a rede, desenvolvida sobre LTE *Broadcast* para fazer *broadcast* de mensagens entre servidores V2X para veículos ou outras infraestruturas, usam uma **interface Uu**.

A figura 2.3 permite perceber a subcategorização do V2X para o projeto 3GPP.

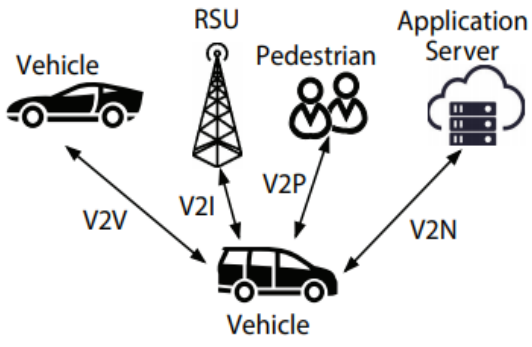


Figura 2.3: Tipos de comunicação suportada no projeto 3GPP [28]

As comunicações V2V, V2I e V2P utilizam a interface PC5 sempre que o destinatário se encontrar na zona de alcance dessa mesma interface. Caso contrário, tem de se recorrer ao modelo híbrido que usa as duas interfaces, como se pode ver pela figura 2.4.

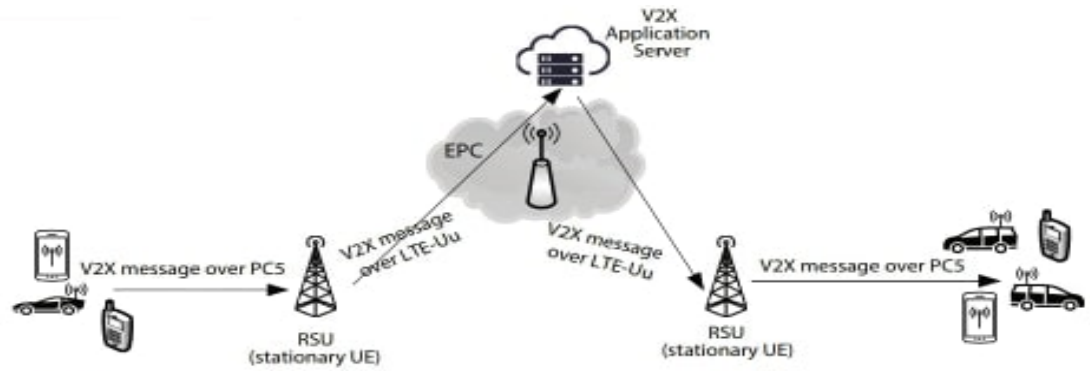


Figura 2.4: Transmissão de mensagens utilizando modelo híbrido de PC5 e Uu [28].

Quando um veículo pretender enviar uma mensagem para outro veículo ou infraestrutura que se encontre dentro do alcande de PC5, então a comunicação pode acontecer diretamente. No entanto, caso o veículo ou infraestrutura destinatária esteja fora do alcance de PC5, é feita a comunicação com a rede, através de uma interface Uu, que faz *broadcast* da mensagem original para uma infraestrutura dentro do alcance de PC5 do dispositivo destinatário, de modo a que a mensagem possa ser entregue diretamente a partir desse ponto, tal como é explicado em [28]. Importante mencionar que comunicações V2N e I2N (*Infrastruture-to-Network*) usam interface Uu.

2.2.3 DSRC baseada em IEEE 802.11p vs C-V2X Release 14/15

A figura 2.5 trata-se de uma tabela elaborada pela *Autotalks*[23] e demonstra as principais diferenças e parecenças técnicas entre as duas tecnologias.

Topic	DSRC 802.11p	C-V2X Rel. 14 / 15
Origin	WiFi	LTE uplink
Modulation	OFDM	SC-FDM
Transmission time	Varying according to packet length. Typically 0.4mS Capable of copying occasional long packets	1mS Increasing energy per bit for long communication range
Concurrent transmissions	No	Yes Decreasing range due to “half-duplex” and “near-far” problems
Symbol duration	8uS Fast channel tracking	71uS
Line coding	Convolution code	Turbo code High processing gain for long communication range
Transmission scheduling	CSMA: Transmit when no ongoing reception No pre-determined transmission slots fitting facilities layer per-cycle decision whether to transmit	Semi-persistent sensing of least occupied resource Collisions aren’t sensed. Slow response to changing environment
Retransmission	None	Yes. Typically activated in high-speed. Overcoming network collisions and increasing communication range
Time synchronization	Loose Asynchronous	Very tight Synchronous requirements
Source: Autotalks		

Figura 2.5: Semelhanças e diferenças entre uma tecnologia DSRC e C-V2X

Em termos de tecnologia usada para comunicação, a DSRC usa tecnologia OFDM (*Orthogonal Frequency-Division Multiplexing*) com CSMA (*Carrier Sense Multiple Access*), fornecendo uma comunicação robusta e independente do sinal de GPS (*Global Positioning System*) enquanto que a C-V2X usa SC-FDM (*Single Carrier Frequency Division Multiplex*) ou OFDM (dependendo da sua implementação), permitindo uma capacidade maior para comunicações de maior alcance. Na rede 5G, por exemplo, que será abordada de seguida, utiliza-se tecnologia OFDM.

Em termos de segurança nas comunicações, ambos os sistemas usam um método de criptografia de chave pública. Umas das principais lacunas da comunicação V2X parte da falta de robustez e falta de capacidade de se manter imune a ciberataques.

Como apresentado em [22], em relação ao 802.11p, a tecnologia C-V2X possui:

- Um alcance 2x maior dependendo da frequência;
- Tempo de reação do sistema autónomo 300% maior;
- *Link budget* maior em 8dB.

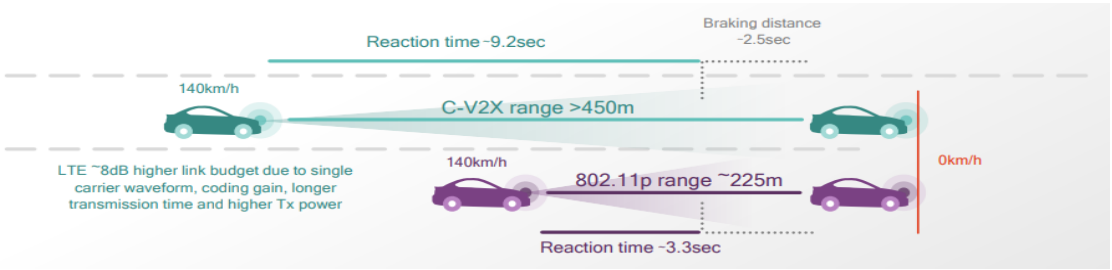


Figura 2.6: Comparação em termos de tempo de reação e alcance entre C-V2X e 802.11p DSRC [22].

No entanto, a Autotalks [29] apresentou uma solução capaz de juntar o melhor dos dois mundos. Assim surgiu a ideia de *Global V2X* e esta empresa tem já *chipsets* equipados com C-V2X, para além da tecnologia nativa de 802.11p DSRC. Um facto interessante desta inovadora tecnologia é o de não necessitar de estar connectado a uma rede celular. Desta forma, os fabricantes terão menores custos sendo que podem desacoplar o módulo NAD (*Network Access Device*), tendo assim mais liberdade na escolha dos mesmos para otimização de custos. Esta medida de desacoplação reduz riscos de interferências e ciberataques sendo que o módulo de *Global V2X* encontra-se isolado da NAD.

Exemplos de *chipsets* com esta tecnologia são o CRATON2 [30] e o SECTON [31] que possuem bastantes semelhanças. O primeiro apresenta especificações como:

- Modems de canal duplo para IEEE 802.11p e C-V2X (modo PC5) *Realease 14/15*;
- IEEE 802.11a/b/g/n/ac operacional a 2.4GHz/5GHz, até 433 Mbps;
- Conectividade WLAN;

- Múltiplas possibilidades de interface, como USB 2.0, *Ethernet* 10/1000/1000 *Audio Video Bridging*;
- Entre outras.

O segundo trata-se de um *add-on* que tem como principais especificações:

- Modems de canal duplo para IEEE 802.11p e C-V2X (modo PC5) *Realease 14/15*;
- IEEE 802.11a/b/g/n/ac operacional a 2.4GHz/5GHz, até 433 Mbps;
- Conectividade WLAN;
- Interface USB 2.0;
- *Host drivers* para Linux ou QNX;
- Entre outras.

2.2.4 5G

O 5G representa o futuro das redes de comunicação sem fios e traz imensas vantagens em relação às tecnologias de comunicação móvel anteriores, tal como menciona a Qualcomm em [22] e [32]:

- Aumento substancial das larguras de banda e taxas transferência até vários Gbps;
- Maior variedade de formas de conexão e maior número de antenas que permite o aumento de cobertura e capacidade;
- Alta tolerância à falha devido à multi-conectividade;
- Latência muito menor do que 3G e 4G graças a uma estrutura flexível de organização de tramas.

Através do artigo [33], é possível perceber como funciona o 5G. Basicamente, cada célula da rede celular é subdividida em micro e pico células e os dados são transmitidos por ondas rádio com frequências iguais às usadas numa rede Wi-Fi (mmWAVE - figura 2.7), permitindo uma conexão muito mais rápida e eficiente do que, por exemplo, o 4G. É importante referir que o mmWAVE é apenas uma pequena parte do vasto mundo do 5G, ainda por explorar, sendo ainda algo para um futuro de médio longo prazo, havendo outras tecnologias 5G perto de serem lançadas no mundo das comunicações veículares.

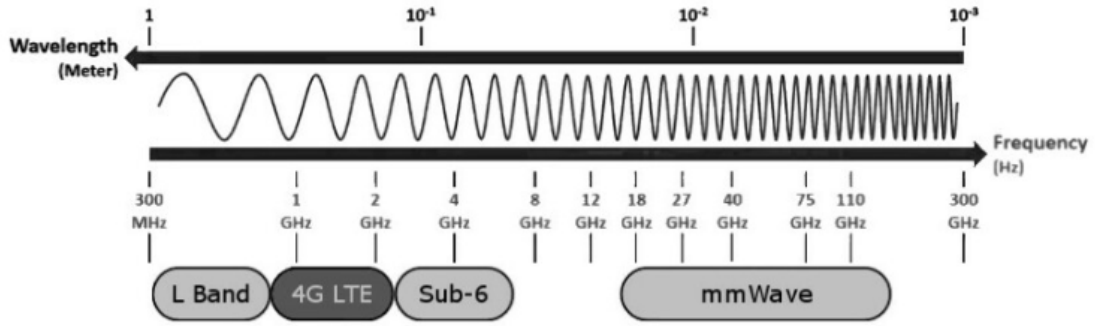


Figura 2.7: Diferenças de comprimento de onda/frequência entre as principais bandas aplicáveis a uma rede celular [34].

Uma das principais aplicações desta tecnologia de 5ª geração acaba por ser a IoT (*Internet of Things*) [35], que permitirá que a ideia de *Smart Cities* seja realizável, sendo que há uma largura de banda muito maior, facilitando a integração dos veículos de condução autónoma cujo sistema de comunicação terá um desempenho e eficiência muito melhores com C-V2X.

A Qualcomm apresenta-se como a principal fabricante de modems de rede 5G, tendo lançado, em 2019, o Snapdragon X55 [36]. Este dispositivo providencia:

- Apoio para 5G NR TDD e FDD (*New Radio Time Division Duplex* e *Frequency Division Duplexing*);
- Espectro de frequências mmWave;
- Partilha de espectros 5G/4G fornecendo uma maior flexibilidade na utilização das frequências;
- Tecnologia MIMO (*Multiple Inputs Multiple Outputs*) que permite que os operadores utilizem o espectro de frequências de forma mais eficiente (aproveitamento a nível de azimute e elevação);
- Até 7.5 Gbps e 3 Gbps de velocidade de *download* e *upload*, respetivamente.

2.3 Projetos Europeus

O EUCAR (*European Council for Automotive R&D*) vai em busca de inovação e investigação do avanço tecnológico dentro do mundo da automação através de parcerias entre grupos automóveis que ambicionam alargar os horizontes dos veículos autónomos. Em [37], podemos ver os principais projetos nos quais o EUCAR tem, ou teve, uma participação ativa, tanto a nível de tecnologia como a nível de financiamento[2].

- **DENSE - Junho 2016**

[2]As datas associadas a cada projeto correspondem ao seu início.

Os principais objetivos deste projeto passam pela deteção e monitorização do espaço em torno do veículo pegando no factor climatérico como o principal problema para os sensores existentes.

Para resolver este problema, recorre-se à fusão de três tecnologias diferentes para chegar a um novo conjunto de sensores mais eficientes:

– RaDAR de alta resolução (MIMO RaDAR);

– Uma câmera SWIR (*Short Wave InfraRed*);

– Um LiDAR SWIR;

Cada um destes três sensores não resulta individualmente, tornando-se necessário haver uma fusão entre eles para atingir os requisitos de segurança e visibilidade.

Duração - 36 meses.

- **L3Pilot - Setembro 2017**

Este projeto tem como finalidade testar a viabilidade da condução autónoma em termos de transportação, partindo das funções características do nível 3 de automação, segundo a SAE (figura 1.1), e tentando avaliar a introdução de funcionalidades de nível 4.

Para tal, os grupos responsáveis por este projeto preocupam-se em criar um ambiente apropriado para a condução autónoma para toda a Europa, correndo testes por todo o continente e recolhendo dados para avaliação das funções autónomas.

Duração - 48 meses.

- **ADAS&ME - Setembro 2016**

ADAS significa *Advanced Driver Assistance Systems* e é a finalidade deste projeto, ou seja, incorporar o estado do condutor, ambiente envolvente e interação entre condutor e veículo para fins de segurança e mais eficiente uso da estrada.

Com este projeto é possível recolher dados sobre a interface entre condutor e veículo, possibilitando a deteção de casos como sonolência, distração, fadiga, entre outros fatores que podem comprometer a integridade do condutor, veículo e/ou outros constituintes da via pública.

Duração - 42 meses.

- **ENABLE-S3 - Maio 2016**

A principal meta do ENABLE-S3 é aumentar os níveis de confiabilidade dos sistemas autónomos, mantendo o processo de testes eficiente e a um custo moderado.

Para alcançar esta meta, enquanto o projeto estiver ativo, serão desenvolvidos métodos de verificação e validação para diversos domínios que, mais tarde, poderão ser relacionados a fim de partilhar dados pertinentes para os sistemas autónomos. Domínios estes que são: automóvel, aeroespacial, marítimo, ferroviário, saúde e agricultura.

Para desenvolver esta nova tecnologia de verificação e validação, são testados diversos cenários virtuais, semi-virtuais e reais com modelos sensoriais e métodos de os estimular, com o objetivo de variar as condições do meio envolvente e analisar o comportamento do sistema.

Duração - 36 meses.

- **ARCADE - Outubro 2018**

A principal motivação dos intervenientes deste projeto é implementar uma abordagem de desenvolvimento, teste, validação e venda de sistemas CAD (*Connected, Cooperative and Automated Driving*).

Para chegar a esse fim, há uma necessidade de cooperação entre os grupos de interesse dos diferentes setores: automóvel e insfraestutura, principalmente. Em termos práticos, é impreterível a recriação de cenários de teste e estudo das necessidades relacionadas com veículos autónomos, clarificando as falhas e soluções de melhor interesse para a indústria, equipas de investigação e restantes participantes do projeto.

Duração - 36 meses.

- **HEADSTART - Janeiro 2019**

Este projeto tem como meta definir procedimentos de teste e validação para funções CAD - comunicação, cibersegurança e posicionamento - integrando-as em testes e correlacionando os resultados obtidos nas simulações e em cenários reais. Este projeto terá impacto a nível de:

– Teste e validação - Potenciação de estratégias de desenvolvimento para redução de tempo e custos;

– Avaliação - Criação de protocolos de avaliação de segurança;

– Certificados - Regulamentos que garantam a introdução segura de sistemas CAD no mercado.

Duração - 36 meses.

- **ICT4CART - Setembro 2018**

O principal objetivo do ICT4CART é desenhar, implementar e testar uma infraestrutura de informação e comunicação que permita a transição para níveis de automação superiores (até nível 4) pegando em problemas relacionados com conetividade híbrida, gestão de dados, cibersegurança, privacidade e localização precisa. Todos os desenvolvimentos serão testados em espaço e tempo real na zona central e sul da Europa.

Em termos técnicos, este projeto inclui:

– Comunicação híbrida através de ITS G5 (*Intelligent Transport System Generation 5*) e LTE-V (*LTE-Vehicle*);

– Um ambiente distribuído para recolha de dados e análise dos mesmos;

– Técnicas de cibersegurança e privacidade de dados.

Em termos de arquitetura de alto nível, a figura 2.8 demonstra como estará organizado este projeto:

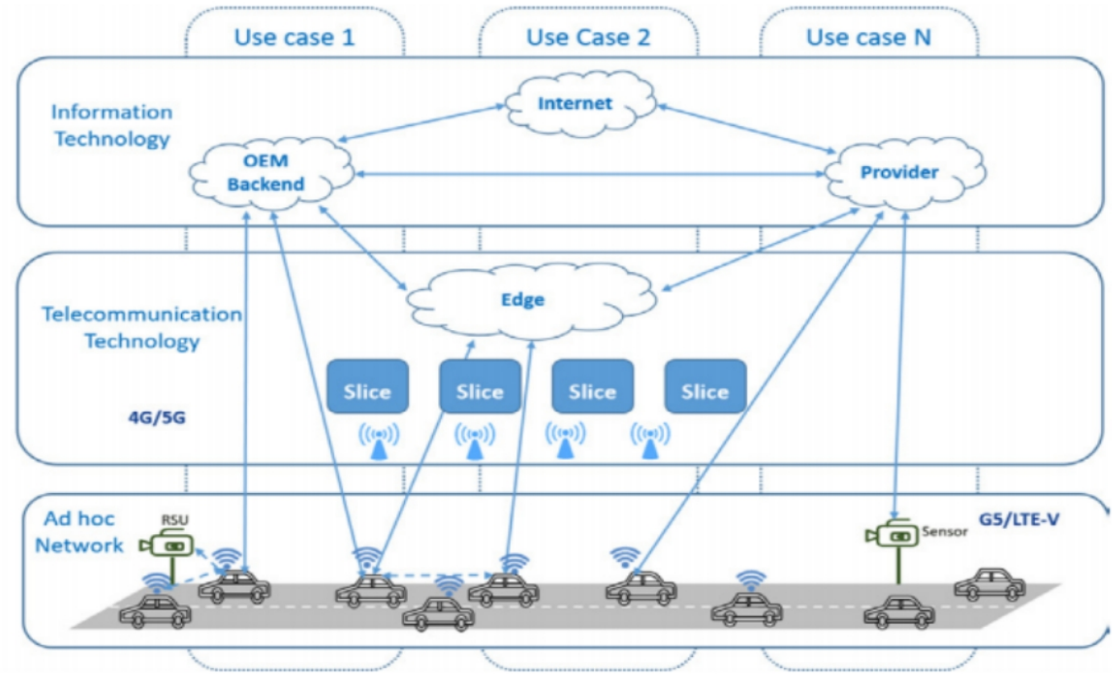


Figura 2.8: Arquitetura de alto nível do projeto ICT4CART [37].

Duração - 36 meses

- **5G-CARMEN - Novembro 2018**

Neste projeto, os participantes pretendem providenciar MVNOs (*Mobile Virtual Network Operators*), tecnologia OTT (*Over-the-Top*) e serviços multi-utilizadores através da tecnologia 5G com um fim de disponibilizar soluções mais seguras, inteligentes, ecológicas e económicas para veículos autónomos.

Os quatro cenários de maior relevância que representam casos de investigação são:

– Determinação de manobras entre veículos conectados;

– Determinação de manobras entre veículos conectados e com nível 3 ou 4 de automação;

– Qualidade de experiência adaptável e priorizada;

– Controlo de emissão de informação.

Para alcançar resultados relevantes neste campo, no que diz respeito à componente técnica, os participantes no projeto desenvolvem o seu trabalho em torno de uma rede híbrida combinando

comunicações V2V e V2I, para curtas distâncias, com comunicações V2N, para longas distâncias. Focam-se ainda em soluções fornecedoras de serviços que combinam redes celulares e em malha, múltiplos domínios e gestão de protocolos em fronteiras, algo que será bastante útil para a tecnologia C-V2X.

Duração - 36 meses.

- **5GCroCo - Novemcro 2018**

O foco deste projeto passa por disponibilizar serviços CCAM (*Connected, Cooperative and Autonomous Mobility*) a diversos países para facilitar a passagem por fronteiras. Esta preocupação advém do facto de diferentes países apresentarem tecnologias e protocolos de rede diferentes.

A validação deste projeto partirá dos seguintes casos de uso:

– Condução através de controlo remoto;

– Mapas de alta precisão;

– ACCA (*Anticipated Cooperative Collision Avoidance*).

E utilizará:

– 5G NR;

– Computação distribuída MEC (*Multiple-access Edge Computing*);

– Qualidade de Serviço preditiva;

– *Network Slicing*;

– Sistemas de posicionamento melhorados.

Duração - 36 meses.

- **ENSEMBLE - Junho 2018**

Este projeto, direcionado para veículos pesados, tem como objetivo implementar a técnica de *platooning* em diferentes marcas de camiões para melhorar consumo de combustível, segurança e fluxo do trânsito. A fase de demonstração será feita recorrendo a sete camiões de marcas diferentes organizados em pelotão a circular em condições reais.

Olhando para a parte técnica da questão, este projecto subdivide-se em três tarefas de duração anual:

– No primeiro ano, os parceiros participantes irão desenvolver a base da ideia com as devidas especificações necessárias ao funcionamento do sistema de *platooning*;

– No segundo ano, essas especificações são entregues aos fabricantes e fornecedores para se passar à fase de construção dos camiões;

– No terceiro e último ano, o projeto é testado em pistas apropriadas e em algumas estradas públicas. Serão tidos em consideração os resultados obtidos e a sua relação com os requisitos iniciais.

Em termos de arquitetura de alto nível, a figura 2.9 demonstra como estará organizado este projeto:

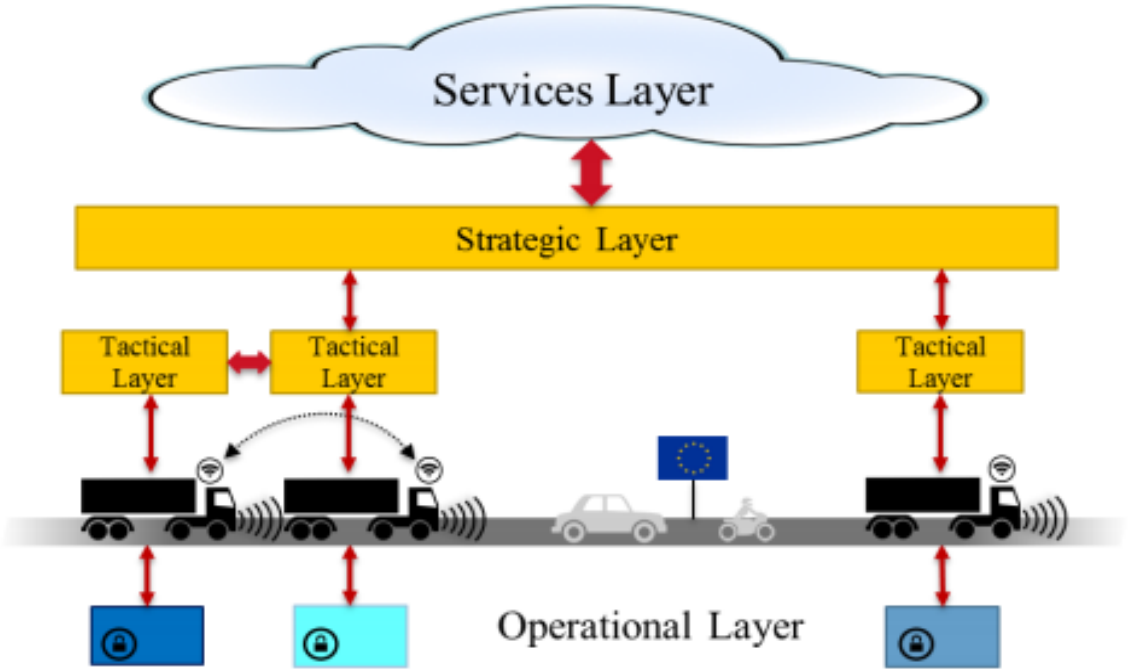


Figura 2.9: Arquitetura de alto nível do projeto ENSEMBLE [37].

Duração - 36 meses.

Outros projetos que também devem ser tidos em consideração são:

- **5G-MOBIX - Novembro 2018 [38]**

Este projeto tem como objetivo estudar sistemas de CCAM em fronteiras e cenários urbanos recorrendo a tecnologia 5G, podendo avaliar as infraestruturas de 5G que melhor beneficiam o conceito de CCAM. O objetivo máximo passa por inovar em termos de aplicações de condução autónoma em níveis de automação altos. Estes estudos permitirão definir melhor o rácio entre custo e benefícios do 5G. Os diversos benefícios e casos de estudo são:

- Ultrapassagem cooperativa;
- Fusão numa das faixas de rodagem;
- Pelotões de camiões;

- Condução em cenários urbanos;
- Controlo remoto de veículos;
- Atualização HD dos mapas;
- entre outros.

A mencionar que Portugal tem diversas empresas envolvidas neste projeto.

Duração - 36 meses.

- **C-Roads - Fevereiro 2017 [39]**

O C-Roads é um projeto que tem como objetivo lançar C-ITS (*Cooperative Intelligent Transport Systems*) que permita aplicações com fins de troca de dados através de comunicações sem fios. Dados estes que podem estar relacionados com o tráfico, situações de perigo na estrada, velocidade ótima para melhorar o congestionamento nas vias, entre outros. É também importante avaliar os diferentes casos estudo relacionados com a travessia de fronteiras de modo a aumentar a interoperabilidade entre fronteiras.

Portugal tem um papel importante neste projeto, sendo um dos membros deste projeto. O objetivo máximo passa por avaliar e estudar as melhores soluções, neste caso para Portugal, para a disponibilização desta tecnologia. Assim que estiver lançada nos diversos países membros, problemas com passagens de fronteiras e as mudanças nas diferentes redes deverão estar ultrapassadas.

Duração - 46 meses.

2.4 Veículos com condução semi-autónoma

Grandes companhias do ramo automóvel têm o seu foco direcionado para o futuro no que diz respeito à condução autónoma. Várias delas apresentam projetos para um futuro próximo, numa tentativa de se afirmarem nesta vertente de enorme peso. Em termos de veículos totalmente autónomos, ainda não existe uma solução operacional, mas existem vários veículos que já têm um nível de automação considerável (níveis de automação mencionados no Capítulo 1 na secção 1.1), de acordo com o sítio *US News* [3] [40].

- **2019 Toyota RAV4**

A Toyota tem vindo a trabalhar num pacote de assistência ao condutor, tendo lançado, em 2019, uma nova versão do pacote - *Safety Sense 2.0*.

O pacote *Safety Sense 2.0* contém sistemas como:

[3] A lista está organizada cronologicamente em relação ao ano de lançamento.

- **Sistema de pré-colisão** que utiliza uma câmera e sensores laser (LiDAR, por exemplo) que permitem detetar objetos na faixa de rodagem, alertando o condutor para o possível perigo. Em caso de não haver resposta humana, o sistema comanda os travões e efetua a travagem do veículo;
- **Sistema de aviso de saída da faixa de rodagem** que utiliza sinais sinoros e visuais que alertam o condutor quando o veículo se desvia da faixa de rodagem em que segue sem usar o sinal luminoso de mudança de direção. As marcas rodoviárias são detetadas através de uma câmera;
- **Sistema de luzes máximas automáticas** que utiliza uma câmera que analisa as luzes dos veículos adjacentes de modo a otimizar a iluminação e, consequentemente, a visibilidade do condutor;
- **Sistema de reconhecimento dos sinais de trânsito** que faz uma análise dos sinais de trânsito, fornecendo informação sobre os mesmos ao condutor. Caso o condutor se encontre em estado de incumprimento das regras estabelecidas pelos mesmos sinais, avisos sonoros e visuais são gerados.

- **2019 Nissan Leaf**

O Nissan Leaf trata-se de um modelo elétrico que tem vindo a ser uma das principais criações da marca no que diz respeito a tecnologia de condução autónoma. As principais funcionalidades de cariz autónomo deste modelo são:

- ***ProPilot Assist*** que se trata de um sistema de auxílio ao condutor no que diz respeito a mudanças de direção, aceleração/desaceleração do veículo. Trata-se de um sistema que, ainda assim, requer a participação e monitorização do condutor;
- **Travão de emergência automático**;
- **Sistema de monitorização de luminusidade**;
- **Monitorização e deteção de sonoloência por parte do condutor** e do **ângulo morto**;
- **Deteção de pedestres**;
- **Câmera de 360º** para auxílio da manobra de estacionamento.

- **2019 Tesla Model 3**

A Tesla é uma das principais marcas automóveis no que diz respeito a veículos elétricos e semi-autónomos. Depois do Tesla Model S, a marca decidiu criar o Model 3, mais económico e com características como:

- ***Enhanced Autopilot*** com capacidade de efetuar mudanças de direção, acelerar/desacelerar e mudar de faixa de rodagem [4];
- **Travão de emergência automático**;
- **Aviso de colisão fronteira e lateral**;

- **2019 BMW 5 Series**

O Série 5 da BMW disponibiliza ao condutor vários equipamentos com funcionalidades autónomas, incluindo:

- ***Driving Assistance Plus Package*** com capacidade de auxiliar o condutor em condições de pára-arranca, permitindo manter a distância para os veículo da frente e mantendo o veículo centrado na faixa de rodagem;
- ***Parking Assistance Package*** que permite o uso da chave para controlo do veículo em lugares de estacionamento difíceis, enquanto o condutor se encontra ao lado do mesmo;
- ***Active Protection System*** que prepara o veículo sempre que for detetado um perigo eminente de colisão;
- **Deteção de pedestres**.

- **2019 Cadillac CT6**

A Cadillac, uma marca conhecida pela sua gama de veículos luxuosos, apresentou um investimento num sistema semi-autónomo integrado no Cadillac CT6. Tal comos os veículos anteriormente mencionados, o CT6 apresenta as seguintes funcionalidades:

- **Controlo de aceleração e mudanças de direção**;
- ***Super Cruise***, um sistema semi-autónomo que mapeia as autoestradas através de dados provenientes do LiDAR, permitindo que o veículo se mantenha na sua faixa e permitindo uma condução autónoma. Mais uma vez, o condutor deve permanecer atento para poder intervir em caso de perigo.

- **2019 Tesla Model X**

A Tesla tem vindo a mostrar-se uma das companhias com maior interesse na atuonomização dos seus veículos. O Model X, um SUV elétrico, apresenta:

[4] A Tesla disponibiliza um pacote com funcionalidades adicionais como auto-estacionamento e possibilidade de chamar o veículo até à atual posição do condutor

– **Sistema de quatro câmeras e uma rede de doze sensores** capaz de fornecer ao sistema *Autopilot* todo o tipo de informação sobre o meio que envolve o veículo, permitindo evitar colisões frontais, laterais e traseiras;

– ***Enhanced Autopilot***, um sistema otimizado e com mais funcionalidades em relação ao *Autopilot*, capaz de manter a velocidade do veículo no trânsito, estacionar sozinho e capaz de se autoconduzir até ao condutor na saída de um parque de estacionamento, por exemplo.

- **2019 Mercedes-Benz S-Class**

Este modelo lançado pela Mercedes, apresenta uma das maiores evoluções tendo em conta a lista apresentada pelo sítio *US News* [40]. Para além das funcionalidades comuns a todos os modelos anteriormente mencionados, o Mercedes-Benz S-Class apresenta um sistema V2X que o permite comunicar com todos os veículos e estruturas equipados com sistemas iguais ou idênticos capazes de responder ao veículo. Para além deste sistema de comunicação, este modelo da Mercedes traz uma funcionalidade de condução inteligente capaz de monitorizar todos os acontecimentos alheios ao funcionamento do veículo, podendo intervir e tomar decisões caso o condutor não reaja quando solicitado.

- **2019 Audi A8**

Não querendo ficar para trás, a Audi, em 2019, apresentou o modelo semi-autónomo - A8. Tal como estes modelos falados anteriormente, este veículo é capaz de se auto-conduzir em trânsito de baixa velocidade, mantendo-se na sua faixa de rodagem. Para além disso, o A8 tem assistente para quando passar em cruzamentos ou entroncamentos, assitente de visão noturna, deteção de pedestres, entre outros.

- **2020 Volvo XC60**

A Volvo, desde há alguns anos para cá, tem apostado em tecnologias capazes de aumentar a segurança durante a condução. Através deste investimento, conseguiu integrar no Volvo XC60 algumas funcionalidades características de um veículo autónomo como, por exemplo:

– **Detetar riscos de colisão** tanto com outros veículos como com outro tipo de elemento que possa circular na via pública, providenciando ajuda ao travar ou desviar o veículo do corpo em questão;

– ***Pilot Assist*** que é capaz de executar mudanças de direção e variar a aceleração, apesar de ser necessária a monitorização ativa por parte do condutor.

- **2020 Lexus LS**

Em 2020, a Lexus acrescentou ao seu espectro automóvel, um modelo semi-autónomos - Lexus LS. Este veículo de luxo tem:

- ***Lexus Safety System+ A*** que inclui auxílio para se manter na faixa de rodagem, radar dinâmico para *cruise control* e assistência na mudança de direção;
- ***Lexus Safety System+ 2.0*** que engloba funcionalidades autónomas como reconhecimento de sinais de trânsito, monitorização e controlo da luminusidade, travão de emergência automático, entre outros.

Capítulo 3

Visão geral do sistema

Neste capítulo abordar-se-á o problema a ser tratado mais profundamente, propondo-se uma solução para o mesmo. Também será explicada de forma mais detalhada a organização da manobra, tanto a nível de hierarquia de pelotão como a nível de ações de controlo ativadas pelos diversos tipos de mensagens.

3.1 Formulação do problema

Existem diversos estudos no que diz respeito a manobras que envolvam pelotões ou apenas um veículo específico. Em [14], projeto que se apoiou nas ferramentas de simulação providenciadas pelo VENTOS [41], verifica-se que o sistema estudado e implementado, recorrendo a veículos com sistema CACC (*Cooperative Adaptative Cruise Control*) e sistema de comunicações VANET (*Vehicular Ad Hoc Networks*), cobre as manobras fundamentais tanto dentro de um pelotão como entre pelotões diferentes. Tais manobras são organizadas na figura 3.1.

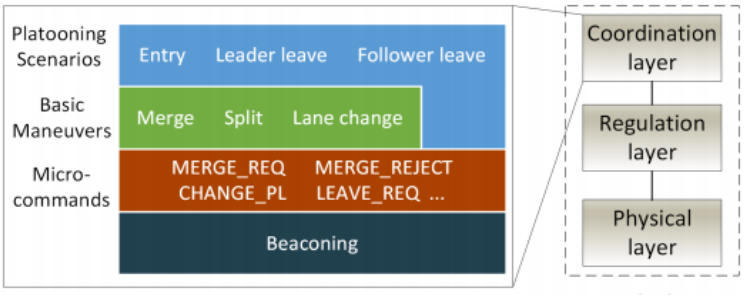


Figura 3.1: Principais manobras de pelotões estudadas [14].

Como se pode ver na figura 3.1, as principais manobras são a de fusão (*merge*), separação (*split*) e mudança de faixa de rodagem (*lane change*). No entanto, dentro destas manobras existem cenários que especificam as diferentes manobras como é o caso da entrada e saída de veículos no

pelotão. Ainda dentro da saída de veículos do pelotão, há a diferenciação entre a saída do líder e saída de um membro que não seja líder.

No caso de saída do líder, esta manobra estudada em [14] cria uma mensagem que alerta todos os membros do pelotão para escolha do novo líder. No entanto, para fins de simplicidade de estudo, assume-se que o membro imediatamente atrás do atual líder se tornará o novo líder. Posto isto, seria interessante conceber uma manobra de reorganização do pelotão após saída do líder que aglomerasse os diversos comandos necessários para a mudança da liderança do pelotão, independentemente da posição em que se encontrasse o futuro líder dentro do próprio pelotão. Desta forma, pretende-se combater a dissolução de pelotões e, com tal, evitar congestionamento e perigo inerentes a veículos que não se encontrem coordenados entre si.

3.2 Solução proposta

A solução proposta passa pela implementação de uma manobra em que se verifique se há veículos disponíveis para desempenhar o papel de líder e que proceda à troca das posições dentro dos pelotões. De grosso modo, o líder deteta um futuro líder e monitoriza a execução das ações necessárias para o colocar numa posição que o permita tomar controlo do pelotão.

Para tal, recorrer-se-á ao Plexe-Veins [42], uma extensão *open-source* para estudo de pelotões para Veins [43]. O Veins, por sua vez, baseia-se em ferramentas como o OMNeT++ 5.5.1 [44] e o SUMO [45] que permitem a recriação de comunicação veícular e visualização das ações dos veículos, respetivamente.

O Plexe utiliza, tal como o VENTOS, um protocolo de *beaconing* - implementado no Veins -, que permite a transmissão de mensagens entre os veículos da forma como ilustra a figura 3.2.

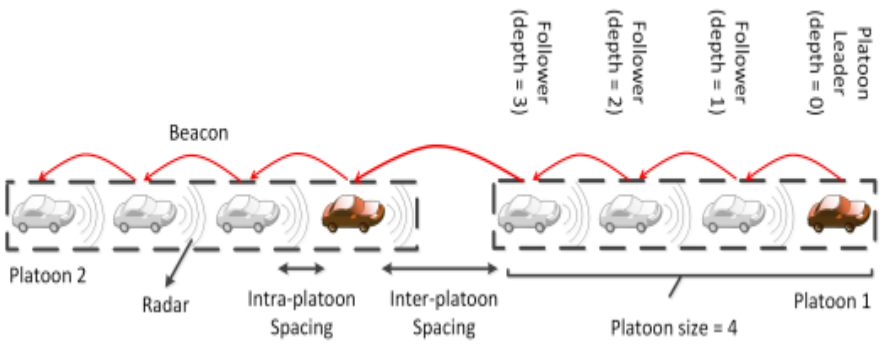


Figura 3.2: Comunicação intra pelotão através de protocolo de *beaconing* [41].

Este protocolo de *beaconing* é necessário para a troca de informação e parâmetros essenciais ao bom funcionamento do sistema CACC. Estas mensagens, os *beacons*, são transmitidas em modo *single-hop* por cada veículo presente no sistema VANET. Cada mensagem poderá conter

informação sobre posição atual do veículo, a sua velocidade, o pelotão a que pertence, entre outros dados que sejam pertinentes num determinado cenário. É sugerida a criação de mensagens específicas para desencadear as diversas ações, dependendo do seu destinatário, emissor e tipo.

O trabalho efetuado no âmbito da criação desta solução pode ser visto em [46], sendo que o código mais relevante se encontra no anexo B.

3.3 Arquitetura do simulador

Para ser possível o estudo desta manobra de reorganização do pelotão, foi necessária uma recolha de informação sobre ferramentas e *software* capazes de fornecer funcionalidades e características que correspondessem à atualidade dos veículos autónomos principalmente a nível de comunicações e que permitissem o estudo de ações em pelotão. Para o efeito, foi escolhido o Veins e a sua extensão - o Plexe-Veins.

O Veins, tal como já foi mencionado anteriormente, fornece módulos que podem ser combinados dentro de outros módulos fornecidos pelo OMNeT++ e que permitem recriar e simular uma rede de comunicação veícular. O OMNeT++ será responsável pela criação de uma rede na qual será simulado todo o processo de comunicação entre os veículos.

No entanto, o Veins não inclui módulos de controlo de mobilidade de veículos, mas providencia uma interface que interopera com o *software* SUMO, tal como se pode verificar na figura 3.3.

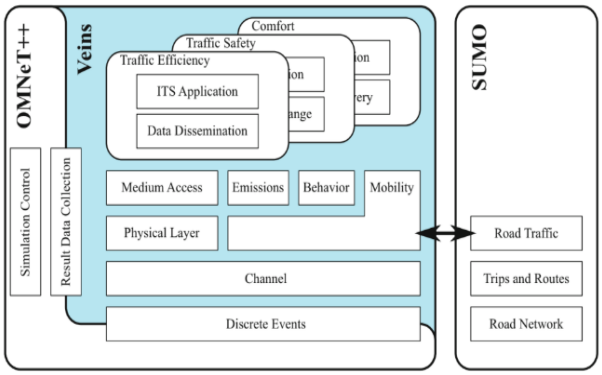


Figura 3.3: Arquitetura do Veins e respetivas interfaces com OMNeT++ e SUMO [47].

Em termos de camadas de comunicação e respetivos protocolos, o Veins fornece módulos que replicam detalhadamente as camadas mais baixas usadas em comunicação inter-veícular [47]. Tal como foi mencionado em 2.2.1, foi criado um protocolo específico que permitisse combater as falhas do IEEE 802.11a - o IEEE 802.11p. É sobre este mesmo protocolo que assentam as camadas mais baixas, tanto a física como a de MAC. Outro aspeto relevante mencionado passava pela divisão do espectro de frequências destinado a DSRC em diferentes canais. A monitorização desses canais é feita através da norma IEEE 1609.4, demonstrando assim, em combinação com o IEEE 802.11p, um papel fulcral no bom funcionamento destes sistemas de comunicação entre

veículos. A figura 3.4 ilustra a organização por camadas de comunicação do *software* Veins, segundo o modelo OSI (*Open System Interconnection)*, onde se pode verificar isso mesmo.

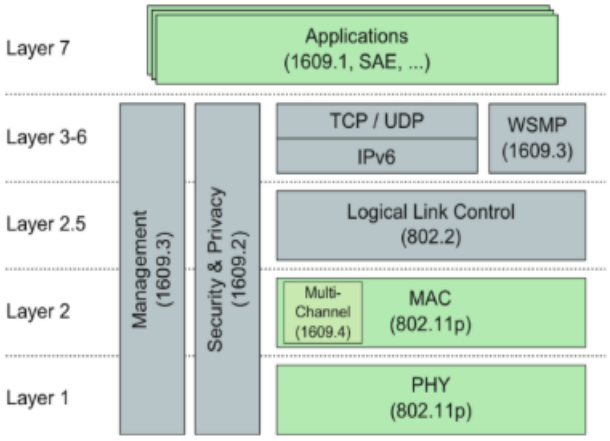


Figura 3.4: Representação das camadas de comunicação do Veins [47].

No entanto, o Veins por si só não contém funcionalidades que permitam estudar o comportamento dos veículos quando estes estão inseridos em pelotões. Aqui entra o Plexe-Veins que extende o Veins, ou seja, todas as funcionalidades que este último disponibiliza podem ser usadas pelo Plexe. Este, por sua vez, contém módulos capazes de simular a comunicação veícular necessária para transmitir informação a ser usada pelos sistemas CACC - usando *beaconing* -, permitindo uma coordenação eficaz entre os membros de uma determinada formação. Contém ainda outros modelos cooperativos, para além do CACC: IDM (*Intelligent Driver Module*), Krauss, CC/ACC (*Cruise Control/Adaptative Cruise Control*). A figura 3.5 permite perceber melhor com que interface se faz a ligação entre a *framework* do Veins e OMNeT++ com o SUMO, que, com o Plexe-Veins, já contém os modelos cooperativos. A interface TraCI (*Traffic Control Interface*) [48], permite obter os dados dos objetos (estradas, semáforos e, o mais o importante, veículos) de forma a poder haver uma correta monitorização e manipulação dos mesmos durante a simulação. TraCI usa arquitetura de comunicação tipo TCP cliente/servidor para ter acesso aos dados da simulação do SUMO, funcionando este último como um servidor e a interface como cliente. Para o Plexe-Veins, foi implementada também uma extensão desta interface [42] que se revela extremamente importante, permitindo:

- Obter informações de posicionamento, aceleração e velocidade dos veículos, permitindo ainda estabelecer estes valores;
- Estabelecer o controlador desejado;
- Estabelecer o espaçamento entre os veículos;
- Mudar de faixa de rodagem com ou sem método de segurança;
- Armazenar informação sobre o líder ou veículo frontal para a utilização de um *FakedCACC* que se revela bastante útil para um veículo ter informações de como o pelotão se está a

deslocar, mesmo não estando nele. Esta função será usada quando um veículo se tornar temporariamente externo, como se verá na secção 3.6;

- Outras ações.

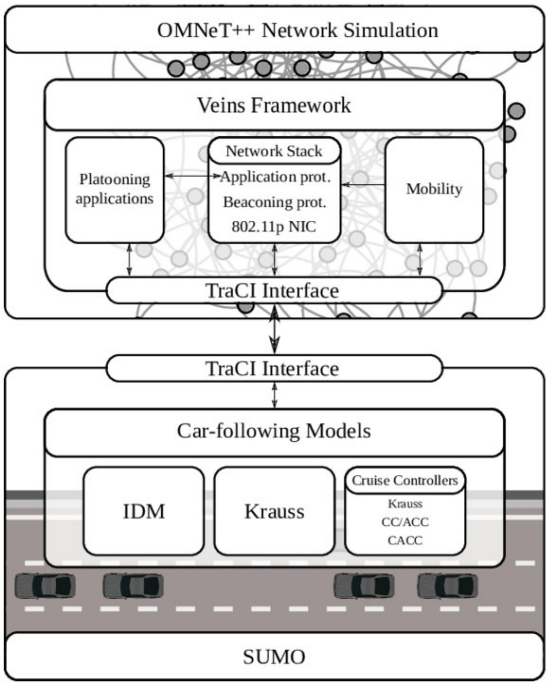


Figura 3.5: Arquitetura do Plexe-Veins e respetivas interfaces [42].

Além desta interface, há ainda outras funcionalidades cruciais para o bom funcionamento do simulador fornecidas pela classe *BasePositionHelper*. Tal como é explicado no sítio do Plexe-Veins, secção da documentação [49], há muitas outras classes que necessitam de saber que veículo estão a monitorizar, a que pelotão esse veículo pertence e respetivo líder, etc. Para tal, foi criada a classe *BasePositionHelper* que disponibiliza essa informação, assim como métodos para que esses dados possam ser alterados durante a simulação.

3.4 Partilha de informação para o CACC

Explicando um pouco em que consiste este modelo cooperativo que se revela crucial para manter a estabilidade da formação de veículos, convém ter em conta algumas noções sobre o que são os modelos CC e ACC através do que nos é dito em [42] com informações retiradas de [50](capítulo 5 para CC, capítulo 6 para ACC e capítulo 7 para CACC).

O modelo CC é já bastante comum em diversos modelos automóveis e permite que o veículo mantenha uma velocidade constante sem que o condutor tenha de se preocupar com o controlo da velocidade, adquirindo um conforto extra durante a condução. A lei de controlo para este modelo é definida pela equação

$$\ddot{x}_{des} = -k_p(\dot{x} - \dot{x}_{des}) + \eta \tag{3.1}$$

em que $\dot{x}$ representa a velocidade atual, $\dot{x}_{des}$ representa a velocidade desejada, k_p é o ganho do controlador proporcional (valor padrão de 1) e η são disturbios aleatórios causados pelas imprecisões no atuador e medição de velocidade (valor padrão de 0).

Sendo que o modelo CC apenas lida com as velocidades real e desejada, é necessária a intervenção do condutor assim que aparece um obstáculo ou outro veículo em frente. Para manter o conforto do condutor, permitindo que este se mantenha livre da necessidade de controlar velocidade, foi implementado o modelo ACC cuja lei de controlo é definida por

$$\ddot{x}_{i_des} = -\frac{1}{T}(\dot{\varepsilon}_i + \lambda \delta_i) \tag{3.2}$$

$$\delta_i = x_i - x_{i-1} + \iota_{i-1} + T\dot{x}_i \tag{3.3}$$

$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \tag{3.4}$$

em que i é referente ao veículo em questão, enquanto que i - 1 o veículo à sua frente. T representa o tempo de avanço em segundos, $\dot{\varepsilon}_i$ representa a velocidade relativa entre os dois veículos, ι_{i-1} indica o comprimento do veículo em frente e δ_i é o erro da distância.

Os controladores anteriores são bons controladores para o veículo que lidera o pelotão. No entanto, para os veículos que o seguem, já não são eficazes tendo em conta que numa formação de pelotão é necessário ter em consideração a distância entre veículos e a *string stability*. Um controlador providencia esta estabilidade de fio quando se mostra capaz de atenuar disturbios relacionados com o movimento dos veículos por todo o pelotão. Os controladores com capacidade para o fazer são os CACC que partilham informação entre si de modo a manter a estabilidade dentro do próprio grupo. A lei de controlo para o i-nésimo veículo do pelotão é definida por

$$\ddot{x}_{i_des} = \alpha_1 \ddot{x}_{i-1} + \alpha_2 \ddot{x}_0 + \alpha_3 \dot{\varepsilon}_i + \alpha_4(\dot{x}_i - \dot{x}_0) + \alpha_5 \varepsilon_i \tag{3.5}$$

$$\varepsilon_i = x_i - x_{i-1} + \iota_{i-1} + gap_{des} \tag{3.6}$$

$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \tag{3.7}$$

onde $\ddot{x}_0$ e $\dot{x}_0$ são a aceleração e velocidade do líder do pelotão e $\ddot{x}_{i-1}$ representa a aceleração do veículo à frente do i-nésimo veículo. ι_{i-1} representa o cumprimento do veículo da frente. Neste caso, o cálculo do erro da distância já tem em consideração a distância de segurança entre veículos que, neste caso, será de 5 metros.

Por sua vez, os parâmetros α da equação 3.5 são determinados por

$$\alpha_1 = 1 - C_1; \quad \alpha_2 = C_1; \quad \alpha_5 = -\omega_n^2 \tag{3.8}$$

$$\alpha_3 = -(2\xi - C_1(\xi + \sqrt{\xi^2 - 1}))\omega_n \tag{3.9}$$

$$\alpha_4 = -C_1(\xi + \sqrt{\xi^2 - 1})\omega_n \tag{3.10}$$

nas quais C_1 é um fator entre as acelerações do líder e do veículo precedente. ξ é o fator de amortecimento e, por fim, ω_n é a largura de banda do controlador. Mais à frente, neste documento, será visto como as mudanças nestes dois últimos parâmetros (ξ e ω_n) terão consequências no desempenho geral da manobra.

Para haver o correto funcionamento e melhor desempenho do controlador CACC é necessária a partilha de informações referentes a cada veículo do pelotão. Para tal, foi utilizado um modelo de pacotes em que cada mensagem possui os dados requeridos pelo controlador para manter a estabilidade e segurança da formação durante o seu tempo de execução.

Para haver uma correta diferenciação das fases que constituem a manobra, cada ação será despoletada por um determinado tipo de mensagem. O Plexe-Veins disponibiliza uma mensagem de manobra universal a todas as manobras que são implementadas dentro deste *software* - ***ManeuverMessage***. Esta mensagem caracteriza-se por incluir dados como ***vehicleId***, ***externalId*** [1], ***platoonId*** e ***destinationId***. Todas os restantes tipos de mensagens foram criados extendendo esta mensagem base, ou seja, estes dados farão parte da informação a ser transportada por cada mensagem, seja qual for a sua topologia. No entanto, para haver a diferenciação necessária entre as mensagens e para que haja uma correta ativação das ações dos veículos, cada topologia contém dados adicionais e específicos.

Cada uma das mensagens ativará uma ação que será efetuada pelo seu recetor. Todas as mensagens *Ack* têm como destinatário o líder do pelotão; as mensagens *Gap* e *Close* têm como destinatário o veículo imediatamente atrás do futuro líder, seja na formação inicial ou final; as restantes mensagens têm como destinatário o veículo disponível para assumir o papel de novo líder; a mensagen *Leave* é enviada pelo líder para todos os elementos do pelotão, mas apenas os veículos aptos para serem líderes podem responder. Caso nenhum dos veículos esteja apto para responder, o último veículo do pelotão notifica o líder através da mensagem *Abort*. O processo de dissolução da formação começa então através das mensagens do tipo *Dissolve* e *DissolveAck*. A tabela 3.1 organiza as mensagens por terminologia e os dados que serão transmitidos através delas.

[1] Identificação do veículo no SUMO, daí ser externo.

Mensagem	Dados adicionais
Leave	platoonSpeed, platoonLane, newPlatoonFormation[], platoonLeaderId
LeaveAck	currentLaneIndex, frontId
Abort	abortFlag
Dissolve	-
DissolveAck	-
Gap	speed, distance, originalPlatoonData[], reJoinerId
GapAck	platoonSpeed, platoonLane
Lane	newLane
LaneAck	laneAfterAck
Close	speed
CloseAck	sequenceNumber
MoveToPosition	platoonSpeed, platoonLane, newPlatoonFormation[]
MoveToPositionAck	platoonSpeed, platoonLane, newPlatoonFormation[]
JoinFormation	platoonSpeed, platoonLane, newPlatoonFormation[]
JoinFormationAck	platoonSpeed, platoonLane, newPlatoonFormation[]
UpdatePlatoonFormation	platoonSpeed, platoonLane, platoonFormation[]
UpdatePlatoonFormationAck	platoonSpeed, platoonLane, platoonFormation[]

Tabela 3.1: Terminologia das mensagens e respetivos dados adicionais[2].

3.5 Organização hierárquica do pelotão

No que diz respeito à organização hierárquica de um pelotão, é essencial definir cargos e respetivas funções com o foco na melhor resposta possível por parte dos participantes da manobra a nível individual e em formação.

A hierarquia pensada para a implementação desta manobra assenta em três papéis que os veículos podem desempenhar[3]:

- ***Leader*** **-** Como o próprio nome indica, é o veículo responsável por monitorizar e controlar as ações dos veículos que o seguem. É o veículo que ocupa a posição frontal do pelotão e é indispensável para o mesmo;

- ***Follower*** **-** Todo o veículo que tiver este título apenas segue o pelotão, respondendo às indicações dadas pelo líder. Estes veículos também terão um papel importante na propagação de informação aos veículos adjacentes, através do protocolo de *beaconing*;

[2]Os nomes dos dados são apresentados em inglês para manter a terminologia usada no Plexe-Veins durante a implementação da manobra.

[3]Mais uma vez, os termos serão em inglês para acompanhar a terminologia usada na implementação.

- ***Available*** **-** Todo o veículo que tiver este título estará apto a desempenhar o papel de líder, caso este decida que quer abandonar o pelotão. Pode ser encarado como um dado que o próprio condutor pode disponibilizar no início da condução ou assim que entra num pelotão novo. Pode ocupar todas as posições exceto a posição frontal da formação.

Cada elemento do pelotão tem a sua devida importância, apesar de apresentarem comportamentos e formas de reagir diferentes umas das outras. No entanto, a combinação das capacidades de cada um demonstra ser essencial para o bom funcionamento e aproveitamento desta manobra de reorganização.

Para se perceber melhor visualmente a função de cara veículo durante a manobra, será usado o seguinte padrão de cores:

- ***Leader*** $\longrightarrow$ azul;
- ***Available*** $\longrightarrow$ verde;
- ***Follower*** $\longrightarrow$ branco;
- **veículo externo** $\longrightarrow$ vermelho.

3.6 Modo de funcionamento

Antes de partir para uma explicação mais específica do modo de funcionamento de cada interveniente da manobra, torna-se mais fácil entender a forma como esta funciona no seu cômputo geral.

Assumindo que existe um pelotão constituído por cinco veículos, sendo o *Leader*, o *Available* e o *Follower* representados por L, A e F, respetivamente, esta é a sugestão para a implementação da manobra de reorganização do pelotão:

1) L envia uma mensagem a todos os membros do pelotão informando-os da sua intenção de deixar o pelotão;

2) Apenas veículos do tipo A podem responder de volta a esta mensagem. Caso não haja um veículo disponível para ser o novo líder, passa-se ao processo de dissolução da formação;

3) Após receber a informação sobre o seu substituto, L informa o veículo do tipo F imediatamente atrás de A para abrir espaço de segurança para que este possa executar a próxima ação sem causar problemas na via. Para tal, F deve tornar-se líder temporário do pelotão formado por ele próprio e pelos veículos que se encontram atrás dele. Esta ação é fulcral para que esses mesmos veículos não percam uma referência assim que se retirar um veículo do meio do pelotão;

4) Assim que as condições estiverem reunidas, A muda de faixa de rodagem e abandona temporariamente o pelotão;

5) L ordena que o espaço que foi aberto anteriormente seja fechado e que A se coloque na posição desejada para se juntar de novo à formação;

6) Assim que A chega à sua nova posição, notifica L;

7) L ordena a nova criação de espaçamento para que A mude para a faixa em que o pelotão circula, ocupando a segunda posição da formação;

8) Depois de se fechar de novo o espaço criado, A torna-se o novo líder do pelotão;

9) L torna-se num veículo externo e abandona o pelotão.

Importante referir que sempre que um veículo se torna externo ao pelotão, ainda que temporariamente, o seu controlador torna-se do tipo ACC. Isto deve-se ao facto de o veículo precisar de uma forma de se controlar a si mesmo mantendo alguma autonomia. Este controlo próprio permitirá evitar possíveis situações de colisão com outros veículos presentes na via de circulação, por exemplo. Sempre que se junta a um pelotão, o controlador muda para CACC que é o controlador usado para controlo cooperativo da formação.

Outra abordagem poderia ser retirar imediatamente o líder do pelotão e fazer uma junção do veículo A pela frente do pelotão. No entanto, a segurança e estabilidade consequentes da manutenção do mesmo líder durante toda a manobra pendiam a balança para essa mesma abordagem. Tal acontece porque, na abordagem escolhida, o veículo A toma proveito da informação do veículo que estará à sua frente após a entrada final no pelotão para definir a própria posição necessária para voltar a entrar na formação. Na segunda abordagem, isto não seria possível, podendo ter consequências indesejáveis e não tão eficientes na manobra.

3.7 Atuação dos intervenientes

Partindo para um panorama mais individualista dos participantes na manobra, torna-se necessário preparar cada passo de cada veículo de forma a manter todos os membros do pelotão em sintonia e com um objetivo comum.

O líder assume toda a responsabilidade e protagonismo ao tomar uma posição de regulador da manobra, tal como um líder deve tomar. Pode-se dizer que, olhando para os três tipos de veículo, a comunicação é feita de forma centralizada, sendo o líder como o centro das atenções. Ou seja, os veículos do tipo *Follower* e *Available* comunicam apenas com o líder e nunca entre si. Isto permite evitar confusões na forma como todo o sistema funciona e manter uma linha de pensamento e implementação mais limpa e organizada.

Começando pelo *Leader*, os seguintes diagramas de estados permitem ter uma noção de como este interage com os restantes membros do pelotão[4]. Foi feita a divisão dos diagramas por cenários para se tornar mais simples perceber o que realmente acontece em cada caso. Os diagramas

[4]Tanto nestes diagramas como nos diagramas associados ao *Available*, o veículo do tipo *Available* assume a terminologia de *Rejoiner* dado que abandona o pelotão temporariamente, voltando a juntar-se perto do final da execução da manobra.

encontram-se em inglês para seguir a terminologia usada no código dos ficheiros desenvolvidos, tornando mais fácil a compreensão da lógica da manobra ao analisar o que foi implementado.

Tendo em conta que a localização do veículo *Available* não é fixa, podendo estar em qualquer posição dentro do pelotão, tona-se necessária a subdividir a manobra tendo em conta diferentes cenários. Para haver distinção entre os diferentes casos foi atribuído um número a cada cenário:

- **Cenário 1 -** O veículo *Available* encontra-se imediatamente atrás do líder inicial do pelotão;
- **Cenário 2 -** O veículo *Available* encontra-se no meio do pelotão, ou seja, não estando em último ou em segundo na formação inicial;
- **Cenário 3 -** O veículo *Available* encontra-se na última posição da formação inicial;
- **Cenário 4 -** Não existe veículo *Available*, sendo necessária a dissolução do pelotão.

Começando pelo cenário 1:

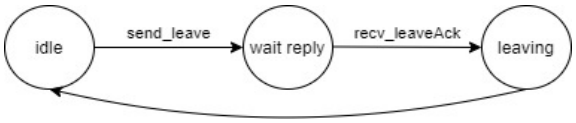


Figura 3.6: Diagrama de estado para o *Leader* associado ao cenário 1.

Neste cenário, como o substituto se encontra imediatamente atrás do líder inicial, basta este último notificar o veículo atrás de si e separar-se após receber o *Ack*.

Seguindo para o cenário 2:

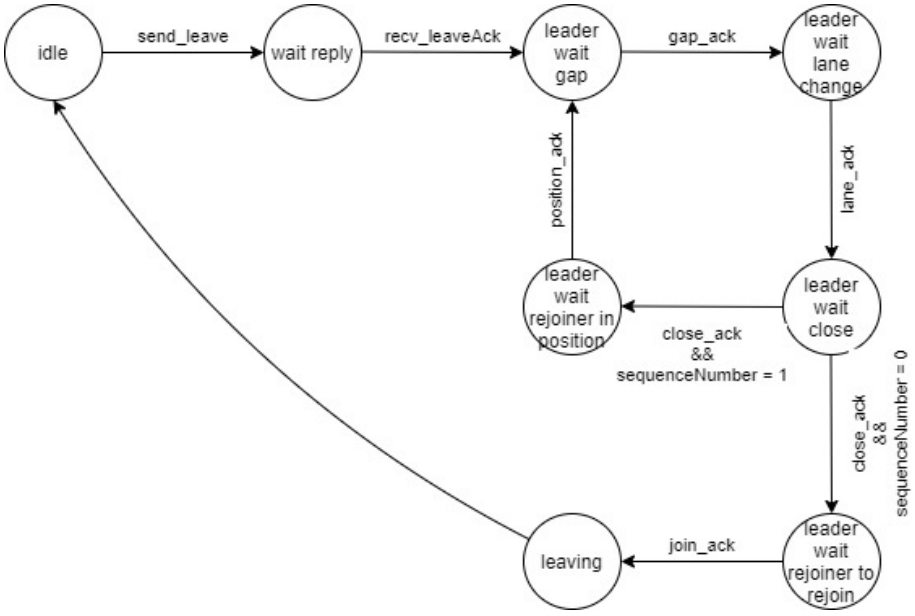


Figura 3.7: Diagrama de estado para o *Leader* associado ao cenário 2.

Neste caso já se pode ver como o líder organiza a manobra tendo em conta as necessidades de abertura de espaços de segurança para mudanças de faixa por parte do veículo *Available*, bem

como o fecho dessas mesmas aberturas. De notar que, neste caso, o ***sequenceNumber*** corresponde à faixa em que o veículo substituto do líder se encontra após o envio da mensagem *CloseAck*[5].

Passando ao cenário seguinte:

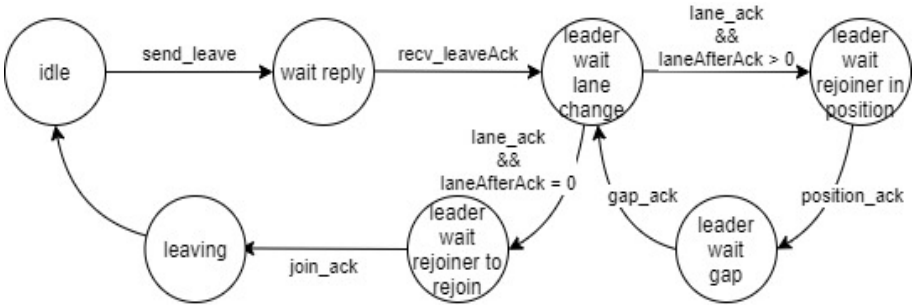


Figura 3.8: Diagrama de estado para o *Leader* associado ao cenário 3.

A diferença deste caso para o cenário anterior é o facto de o líder esperar por uma mudança de faixa de rodagem em vez de uma abertura de espaço que, no cenário 2, antecede essa mudança de faixa, após a receção de uma mensagem do tipo *LeaveAck*. Isto deve-se ao facto de o futuro líder se encontrar na última posição do pelotão. De referir que, ***laneAfterAck*** representa o mesmo valor que o ***sequenceNumber*** do diagrama de estados do cenário anterior. São terminologias diferentes por fazerem parte de tipos de pacotes diferentes.

Por último, o cenário 4:

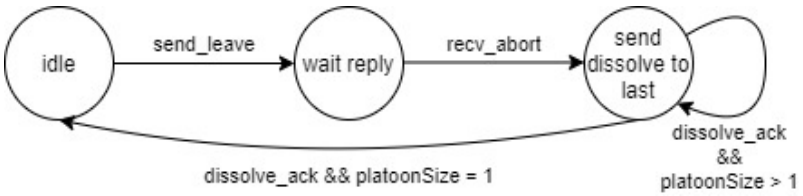


Figura 3.9: Diagrama de estado para o *Leader* associado ao cenário 4.

Neste cenário, não havendo veículo disponível para assumir o papel de líder, tem de haver a dissolução do pelotão. Para tal, o líder envia uma mensagem com esse efeito para o último veículo da formação até não restar mais nenhum membro a não ser ele mesmo.

Através dos diagramas das figuras 3.6, 3.7, 3.8 e 3.9, confirma-se o que foi anteriormente mencionado na secção 3.4 em relação ao tipo de mensagens recebidas pelo líder do pelotão. Este apenas recebe e reage a mensagens do tipo *Ack*, recolhendo desta forma informação importante sobre os veículos que o seguem.

Seguem-se os diagramas de estados referentes ao papel de *Available*, também divididos por cenários. Começando pelo primeiro:

[5]O pelotão começa sempre na faixa mais à direita - faixa 0

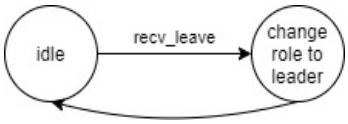


Figura 3.10: Diagrama de estado para o *Available* associado ao cenário 1.

Neste primeiro cenário, tal como já se explicou para o caso do líder, o veículo substituto apenas tem de mudar o seu papel para assumir a liderança do grupo, notificando o seu antecessor da mudança para que este siga o seu caminho de forma individual.

Relacionado com o cenário 2, tem-se o seguinte:

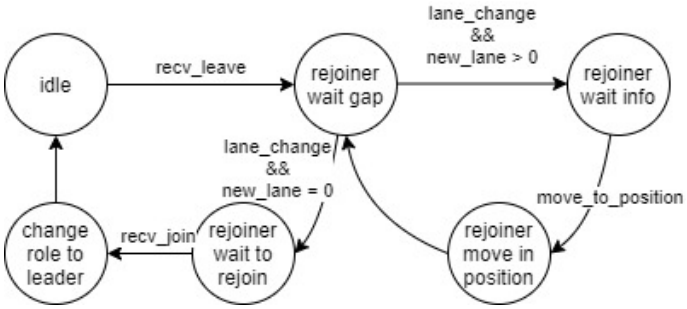


Figura 3.11: Diagrama de estado para o *Available* associado ao cenário 2.

Neste cenário, pode ser constatado pelo diagrama de estados que existem dois processos diferentes de criação de espaço de segurança para mudança de faixa, indo ao encontro do diagrama para o líder correspondente ao mesmo cenário. Para diferenciar os dois momentos, recorre-se ao número da faixa para a qual o *Available* irá mudar.

No cenário 3, pode-se ver o seguinte:

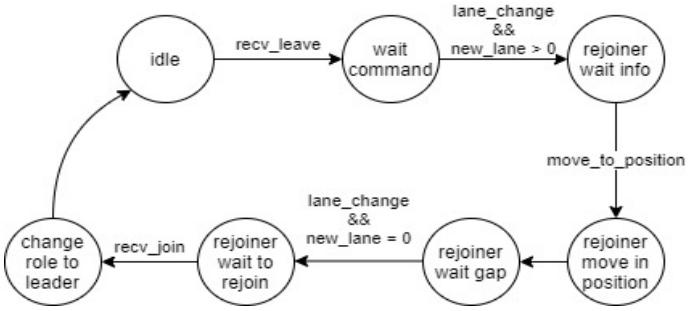


Figura 3.12: Diagrama de estado para o *Available* associado ao cenário 3.

Novamente, pode-se verificar que para este cenário apenas se procede ao espaçamento de veículos uma única vez, sendo este espaçamento para nova entrada do futuro líder na formação.

Para o cenário 4 não existe diagrama de estados para o *Available* visto que este cenário caracteriza-se pelo facto de não haver veículos desse tipo.

Para finalizar, tem-se o diagrama referente à lógica responsável pelo funcionamento dos membros da formação que são do tipo *Follower*:

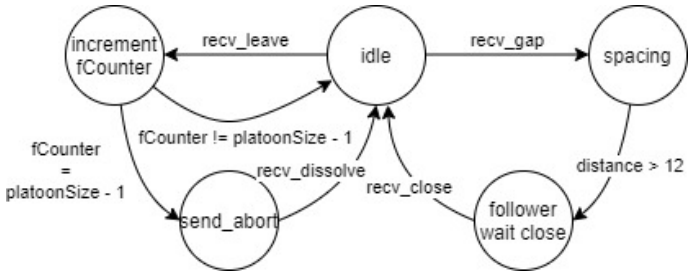


Figura 3.13: Diagrama de estado para o *Follower*.

O lado direito deste diagrama de estados representa a forma como cada *Follower* deve agir quando tem de abrir espaço para o veículo *Available* se movimentar. O lado esquerdo representa um contador que é global a todo o sistema e é utilizado como ferramenta de deteção de ausência de veículos disponíveis para assumir o papel de novo líder. Basicamente, todos os veículos recebem a mensagem do veículo que lidera o pelotão com a informação de que este quer separar-se dos restantes. Sempre que essa mensagem passa por um veículo que não está apto a assumir o controlo do pelotão, o contador ***fCounter*** é incrementado em 1. Como a forma de propagação de mensagem através de *beacons* acaba por tomar uma postura iterativa, ao chegar ao último veículo do pelotão, se o contador igualar o ***platoonSize* - 1**, o último veículo informa o líder de que ninguém está disposto a assumir o seu cargo, sendo necessário abortar e dissolver o pelotão.

Capítulo 4

Simulação e análise de resultados

Neste capítulo estão presentes os resultados das simulações para cada cenário, bem como uma análise dos mesmos. Todos os gráficos referentes às acelerações, velocidades e distâncias foram gerados através de um *script* em R que o simulador disponibiliza. Os gráficos referentes à taxa de falha da manobra com valores padrão e os gráficos criados para o estudo do impacto da variação do *bitrate* na execução da manobra foram conseguidos através do RStudio, após serem registados os testes falhados - secções 4.2 e 4.3.

A tabela 4.1 mostra as principais especificações usadas em todas as simulações de todos os cenários, excluindo, para já, os simulações para fins de estudo de variações comportamentais da manobra perante variações do *bitrate* e da taxa de perda de dados.

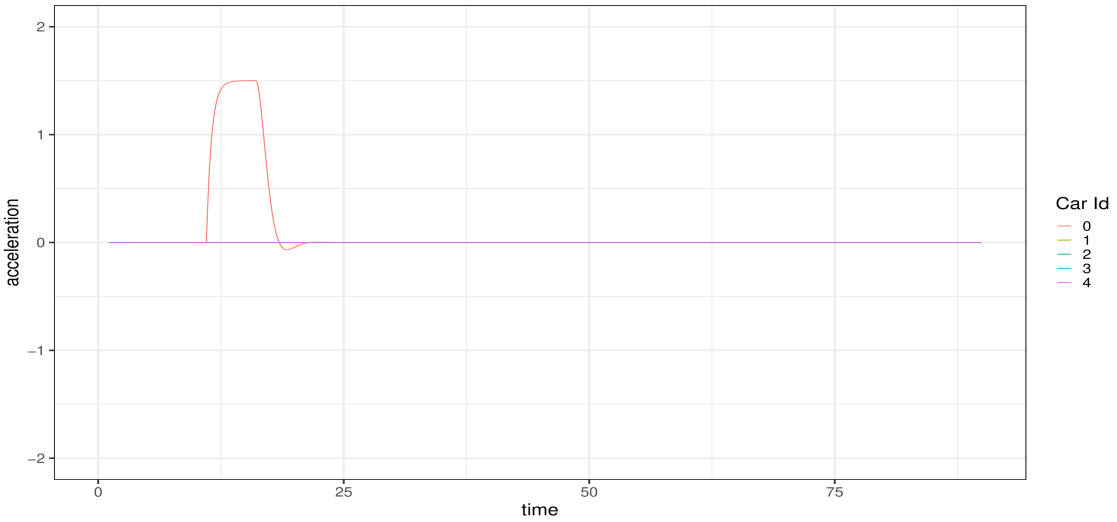
Especificações	
Tamanho do pacote	200 *Bytes*
Taxa de perda de pacotes	0%
Modelo camadas PHY/MAC	IEEE 802.11p/1609.4 canal único (CCH)
Frequência	5.89GHz
Bitrate	6Mbit/s
Potência de transmissão	100 mW
Velocidade de circulação	100 Km/h
Número de veículos utilizados	5
Número máximo de veículos possível	8

Tabela 4.1: Especificações das simulações.

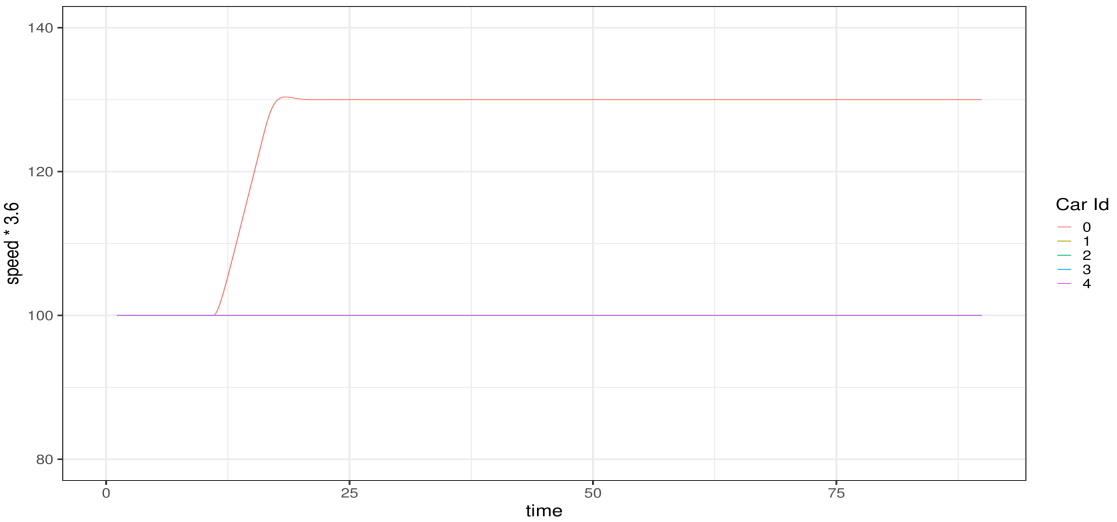
4.1 Simulação sem perdas

Nesta primeira fase da simulação, apenas se tentou perceber a forma como a manobra se comportava em cada cenário. Para tal, armazenando diversos valores referentes aos dados partilhados entre os veículos intervenientes na manobra, foram determinados gráficos que demonstram a forma como as variáveis de aceleração, velocidade e distância para o veículo da frente se alteram durante a execução da manobra de reorganização da formação.

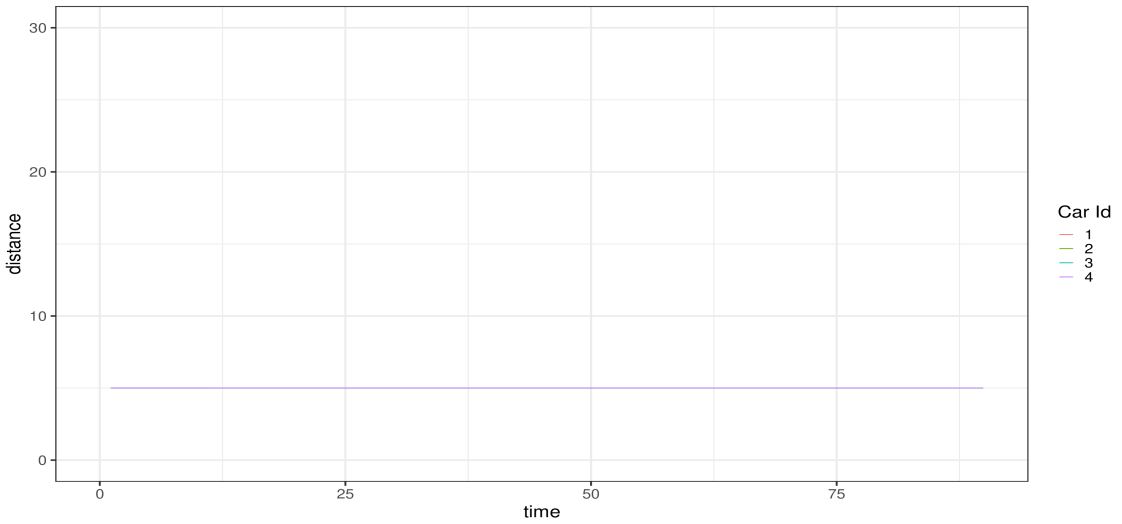
Para cada caso de estudo foram feitas duas execuções, mas cada uma com parâmetros diferentes para o controlador CACC. Na primeira, os parâmetros são ξ=1 e ω_n=0.2, e na segunda, os mesmos parâmetros serão ξ=2 e ω_n=1. Começando pelo primeiro cenário, utilizando o primeiro par de valores para o controlador CACC, foram obtidos os gráficos apresentados na figura 4.1.



(a) Aceleração de cada veículo.



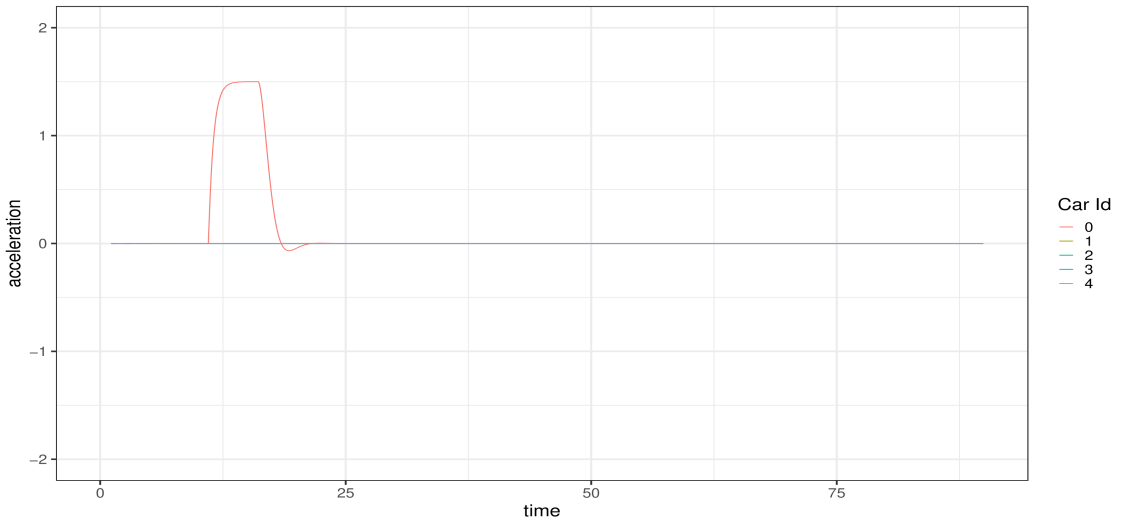
(b) Velocidade de cada veículo.



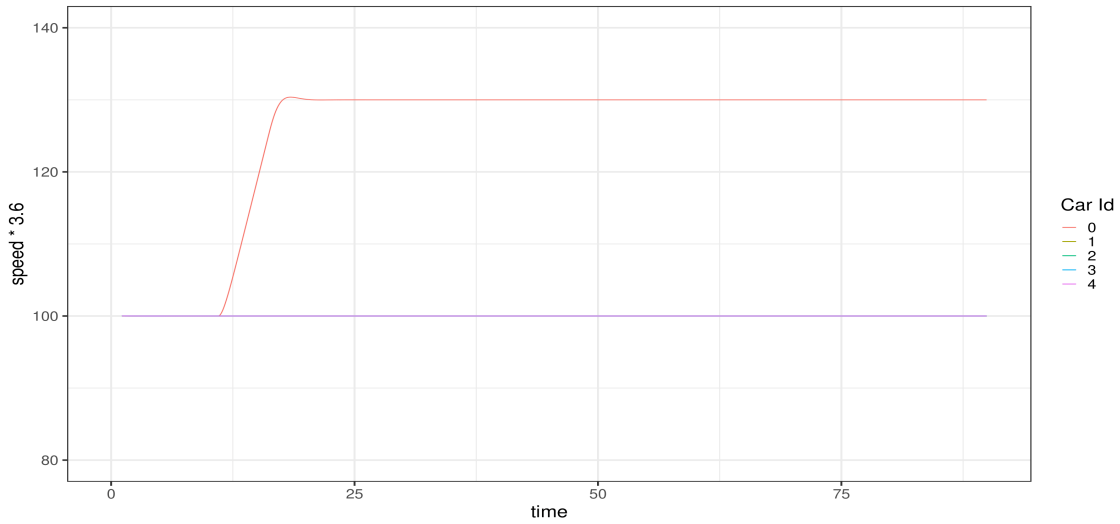
(c) Distância de cada veículo para o da frente.

Figura 4.1: Variações nas acelerações, velocidades e distâncias entre os veículos para o primeiro cenário.

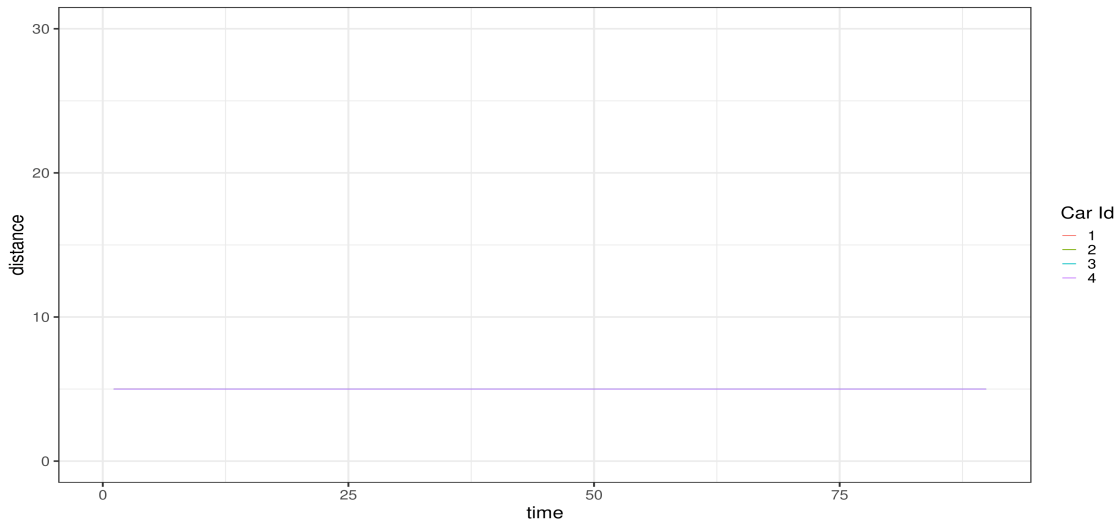
Na figura 4.2 é possível verificar o mesmo cenário, mas desta feita com o segundo par de valores para o CACC ($\xi = 1$ e $w_n = 0.2$).



(a) Aceleração de cada veículo.



(b) Velocidade de cada veículo.



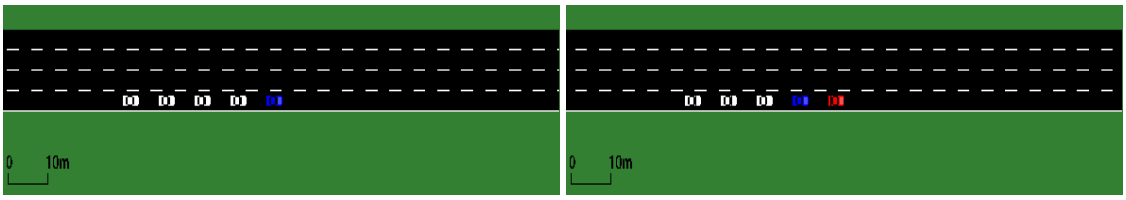
(c) Distância de cada veículo para o da frente.

Figura 4.2: Variações nas acelerações, velocidades e distâncias entre os veículos para o primeiro cenário.

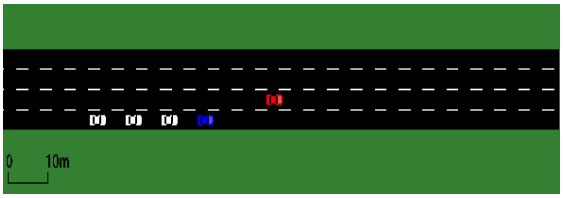
Era expectável que a execução em que foi utilizado o par de valores mais altos para o CACC fosse mais rápida. No entanto, no cenário 1, os tempos eram iguais, como se pode verificar pelas figuras 4.1 e 4.2. Isto deve-se ao facto de o simulador estar programado para começar a manobra aos 10 segundos e como, neste caso, apenas se espera que o líder inicial mude de faixa e se separe do pelotão, o comportamento da manobra nas duas execuções é exatamente igual.

Também se pode verificar que a única mudança no pelotão está dependente do veículo que vai sair do mesmo. O líder inicial é o único que tem um pico de aceleração e um aumento de velocidade para se distanciar do grupo, como se pode ver pelas subfiguras 4.1a, 4.1b, 4.2a e 4.2b. Os restantes veículos mantêm a sua distância entre eles, tal como se pode constatar pelas subfiguras 4.1c e 4.2c.

Em seguida, pode-se ver como se desenrola a execução da manobra para o cenário 1.



(a) *Frame* 1. (b) *Frame* 2.

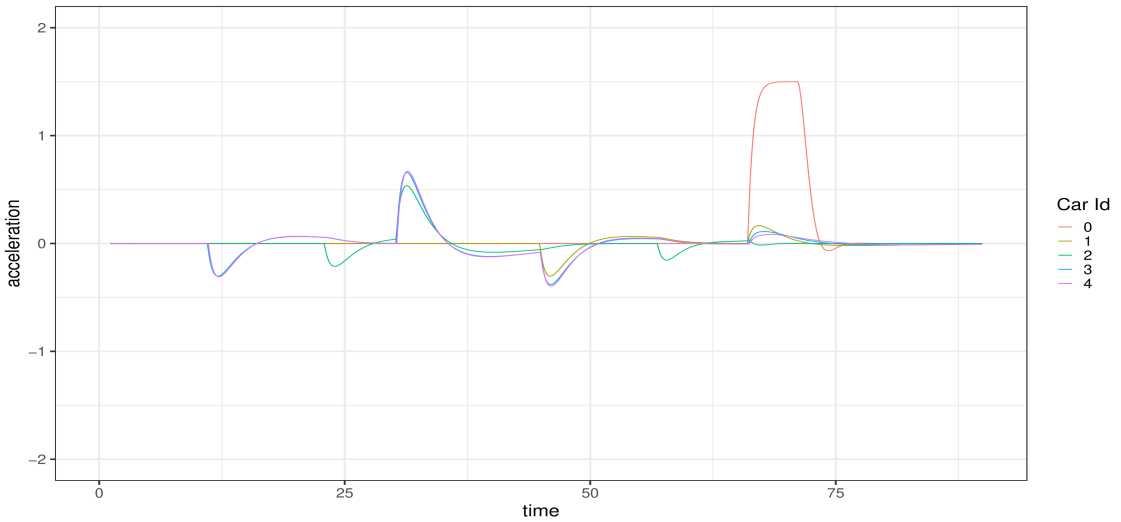


(c) *Frame* 3.

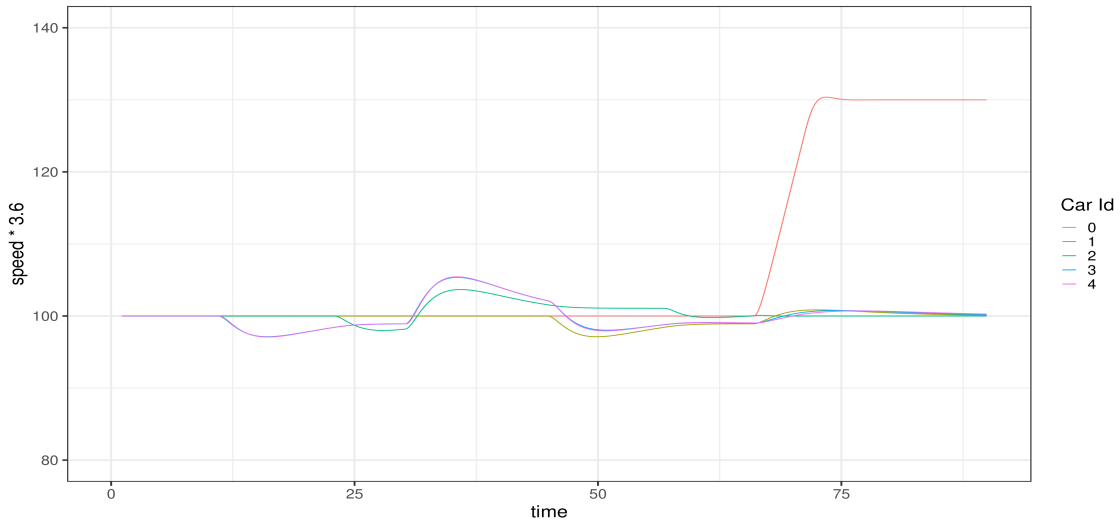
Figura 4.3: *Frames* da manobra no cenário 1.

Como o futuro líder se encontra imediatamente atrás do líder inicial, basta este último informar o pelotão da sua saída e dar o lugar ao seu sucessor na liderança.

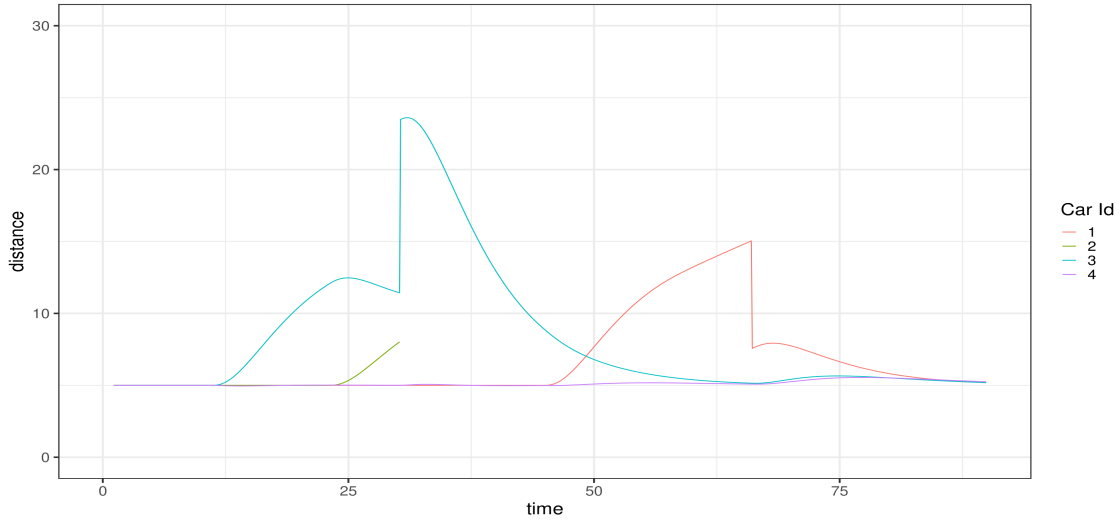
Pegando no cenário 2, utilizando o par de valores mais baixos, verificou-se a diferença temporal, sendo o tempo médio da manobra para a primeira execução de 75 segundos, enquanto que na segunda foi de 40 segundos, valores aproximados. Neste caso o veículo A está na posição 2 do pelotão.



(a) Aceleração de cada veículo.



(b) Velocidade de cada veículo.

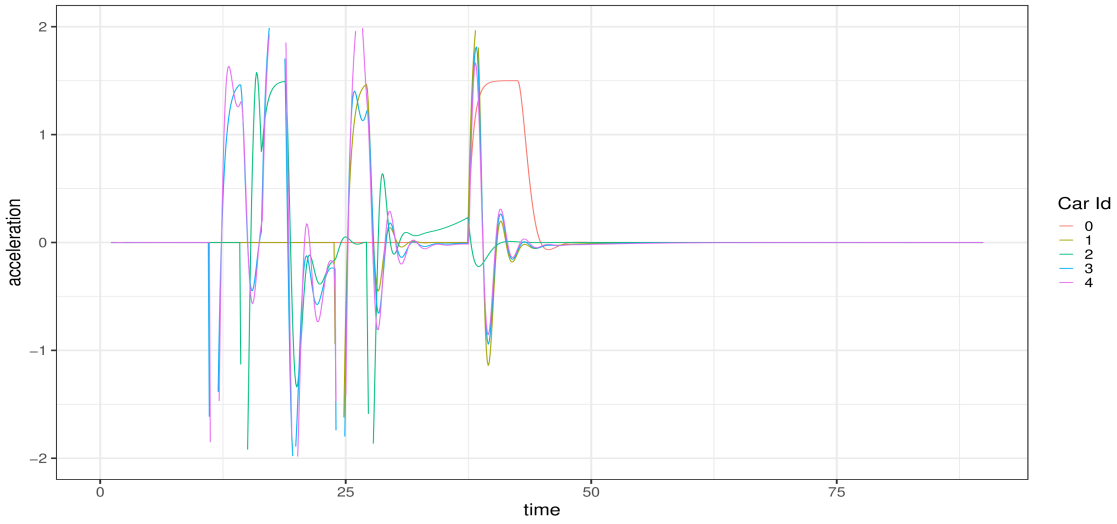


(c) Distância de cada veículo para o da frente.

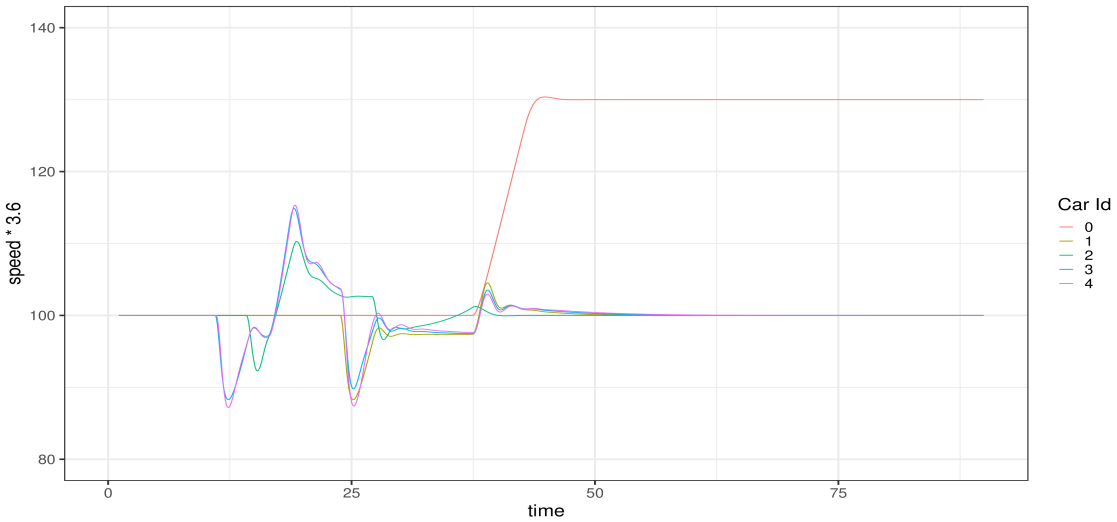
Figura 4.4: Variações nas acelerações, velocidades e distâncias entre os veículos para o segundo cenário.

Como é possível ver pelas subfiguras 4.4a e 4.4b, quando o líder começa a manobra, os veículos 3 e 4 desaceleram para se criar o espaço necessário para a saída do veículo A, como se pode ver pela linha do veículo 3 na subfigura 4.4c. Nota-se ainda, que uns segundos depois, o veículo A desacelera para também criar uma distância de segurança antes de sair. Em seguida, nota-se o aumento da aceleração e velocidade por parte do veículo A para chegar à posição pretendida para se juntar de novo à formação, bem como dos veículos que anteriormente abriram o espaço, mas, desta feita, para o fechar. Em seguida, dá-se um novo processo de abertura de espaço, seguido da junção de A e reajuste do pelotão para que se volte a adquirir uma distância fixa de 5 metros entre os veículos. Por fim, vem a separação do líder inicial, acelerando e deixando o grupo.

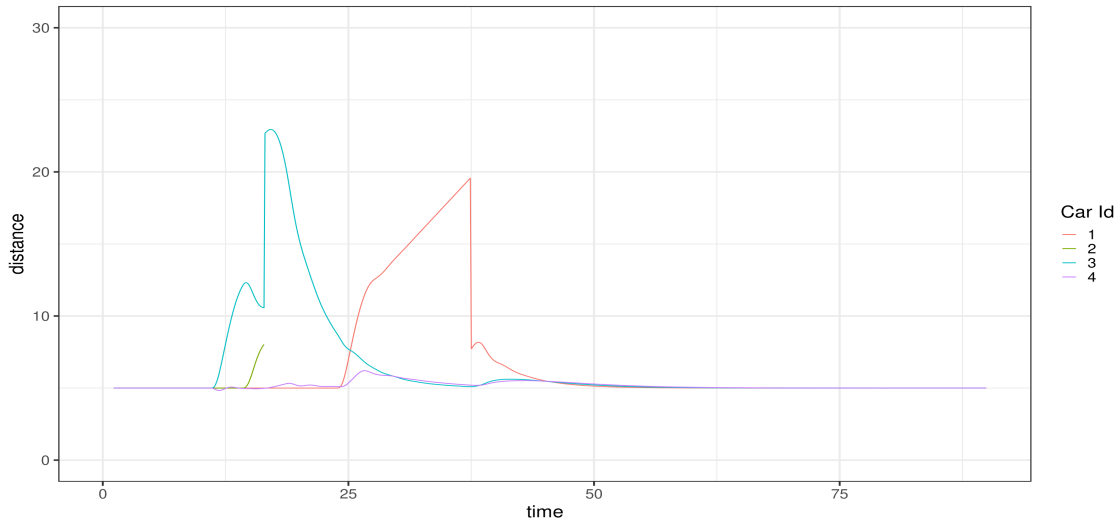
Partindo agora para a execução que utiliza o par de valores maior para o CACC, podemos ver a variação dos mesmos campos na figura 4.5.



(a) Aceleração de cada veículo.



(b) Velocidade de cada veículo.

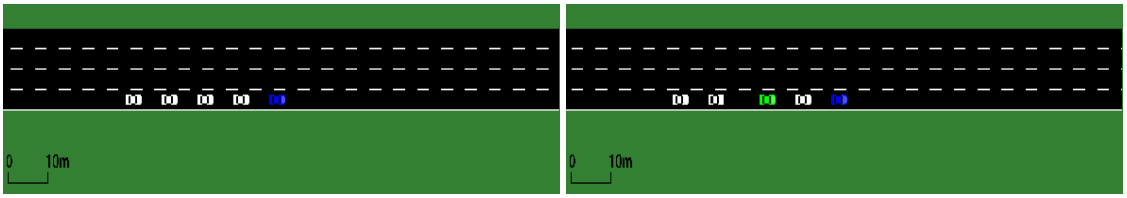


(c) Distância de cada veículo para o da frente.

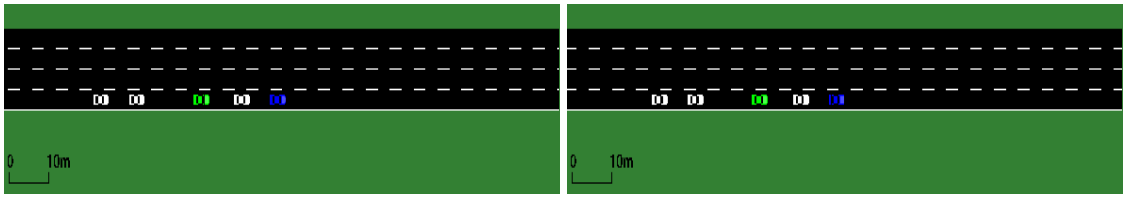
Figura 4.5: Variações nas acelerações, velocidades e distâncias entre os veículos para o segundo cenário.

Neste caso, já se pode ver variações mais significativas nos três campos em estudo. Isto acontece porque com valores mais altos para ξ e w_n a manobra é executada de forma mais agressiva.

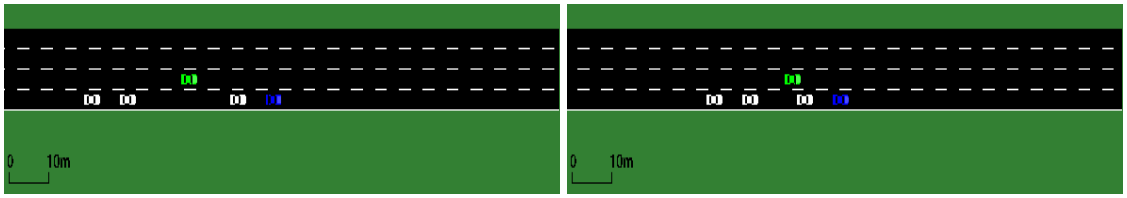
Na figura 4.6 pode-se ver como decorre a execução da manobra para o cenário 2.



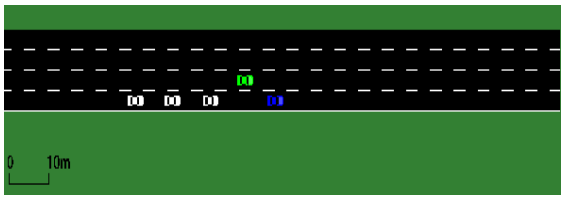
(a) *Frame* 1. (b) *Frame* 2.



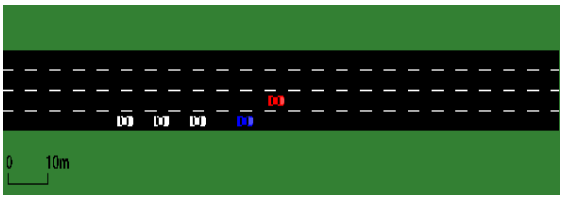
(c) *Frame* 3. (d) *Frame* 4.



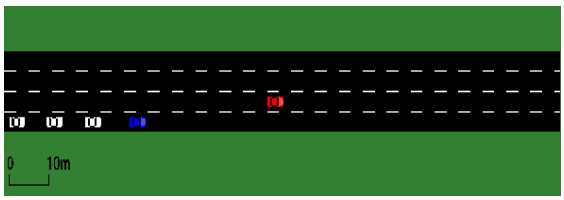
(e) *Frame* 5. (f) *Frame* 6.



(g) *Frame* 7.



(h) *Frame* 8.

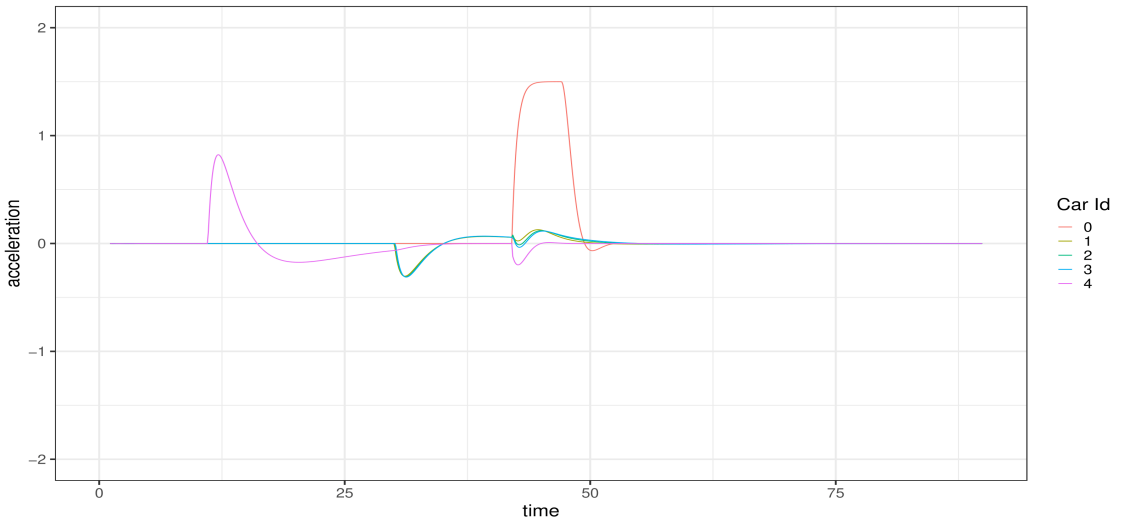


(i) *Frame* 9.

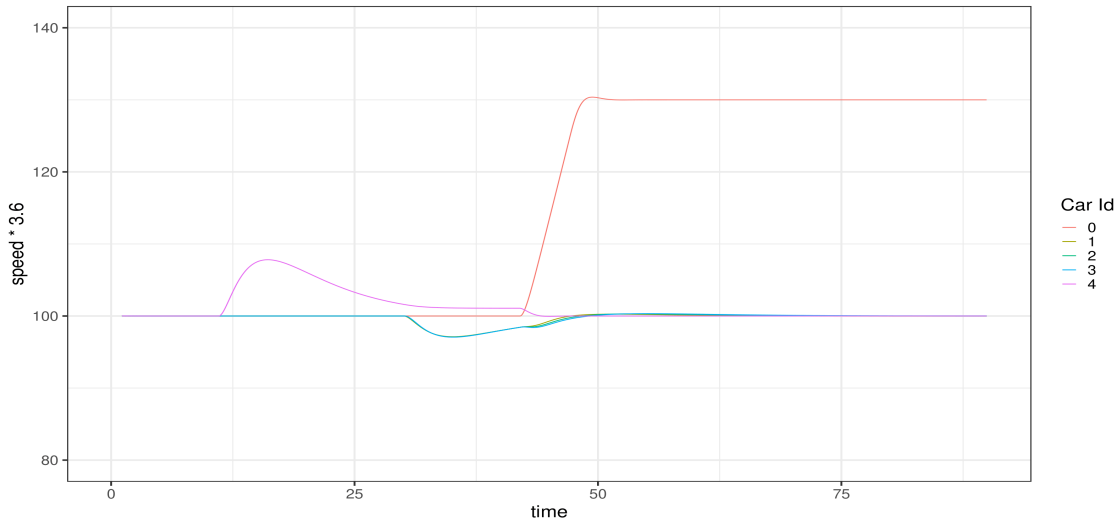
Figura 4.6: *Frames* da manobra no cenário 2.

Assim que o líder inicial avisa os restantes da sua vontade de deixar a formação, o veículo que se disponibiliza para assumir o papel de novo líder muda de cor para verde. Nos *frames* 2, 3, 4 e 5 vê-se o processo de criação de espaço para a mudança de faixa por parte do veículo verde. Em seguida, verifica-se a movimentação deste para a posição onde deverá mudar de novo de faixa após nova abertura de espaço de segurança e, por fim, dá-se a saída do líder inicial, mostrada nos *frames* 8 e 9.

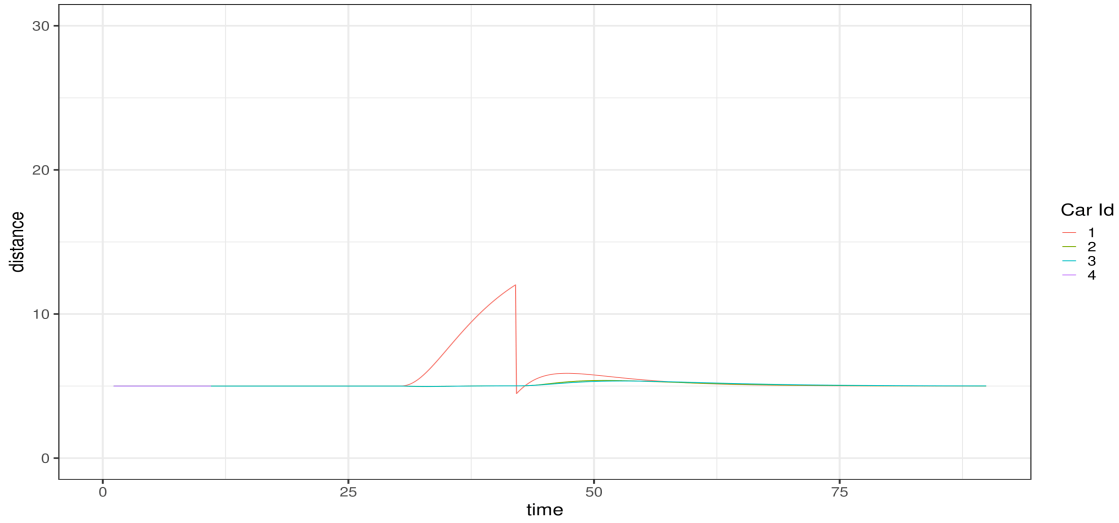
Partindo para o terceiro cenário com valores mais baixos para o CACC, na figura 4.7 estão compilados os gráficos com ele relacionado.



(a) Aceleração de cada veículo.



(b) Velocidade de cada veículo.

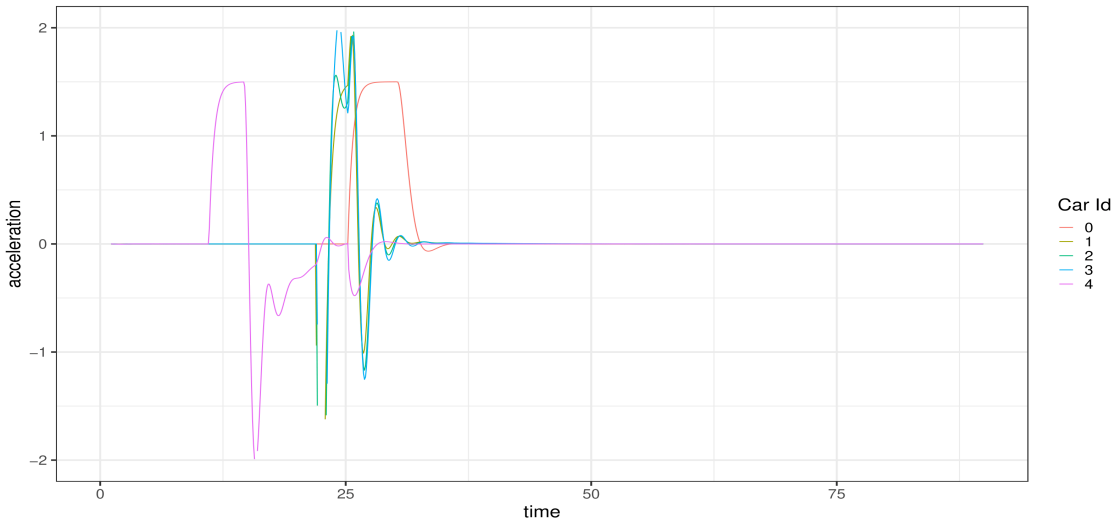


(c) Distância de cada veículo para o da frente.

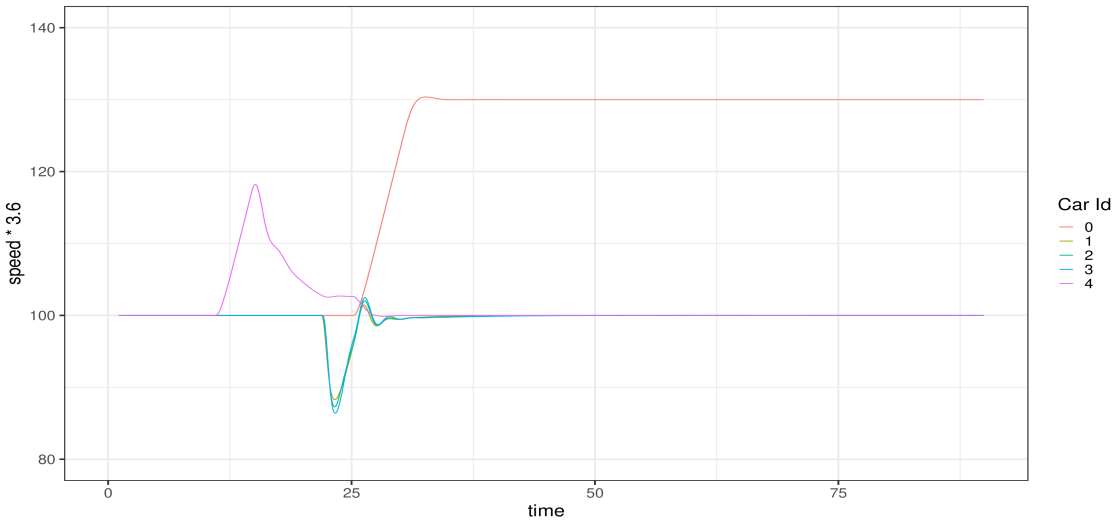
Figura 4.7: Variações nas acelerações, velocidades e distâncias entre os veículos para o terceiro cenário.

Neste cenário, assim que a manobra começa, o veículo A (linha verde nos gráficos) separa-se do pelotão e tem um pico de aceleração e aumento de velocidade para chegar à zona frontal da formação. Assim que chega ao ponto pretendido, dá-se uma manobra de criação de espaço tal como se viu no cenário 2. A partir daqui, os procedimentos são iguais aos do cenário anterior.

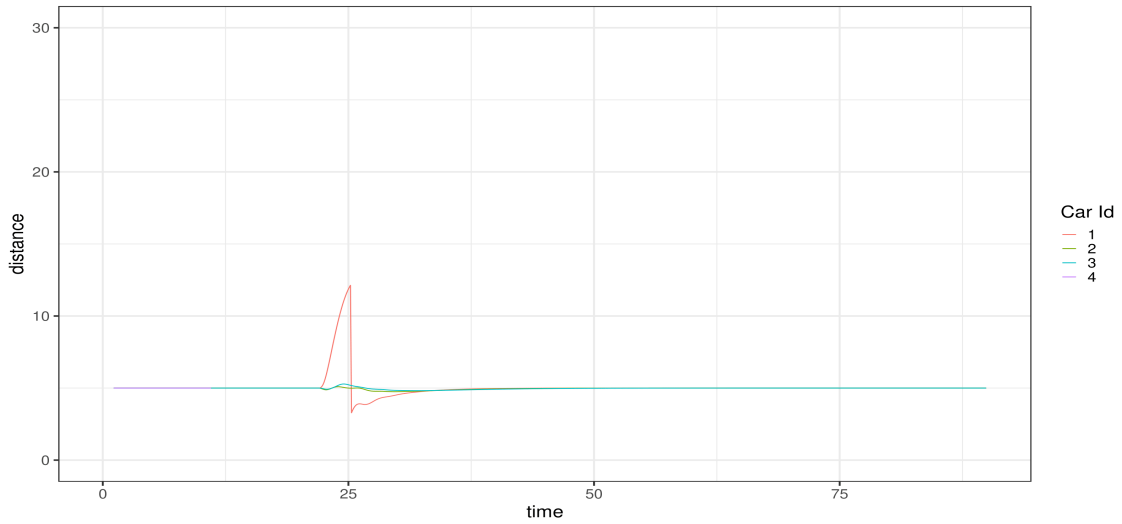
Partindo agora para o mesmo cenário mas com o par de valores para os parâmetros de CACC, a figura 4.8 organiza os gráficos associados ao mesmo.



(a) Aceleração de cada veículo.



(b) Velocidade de cada veículo.

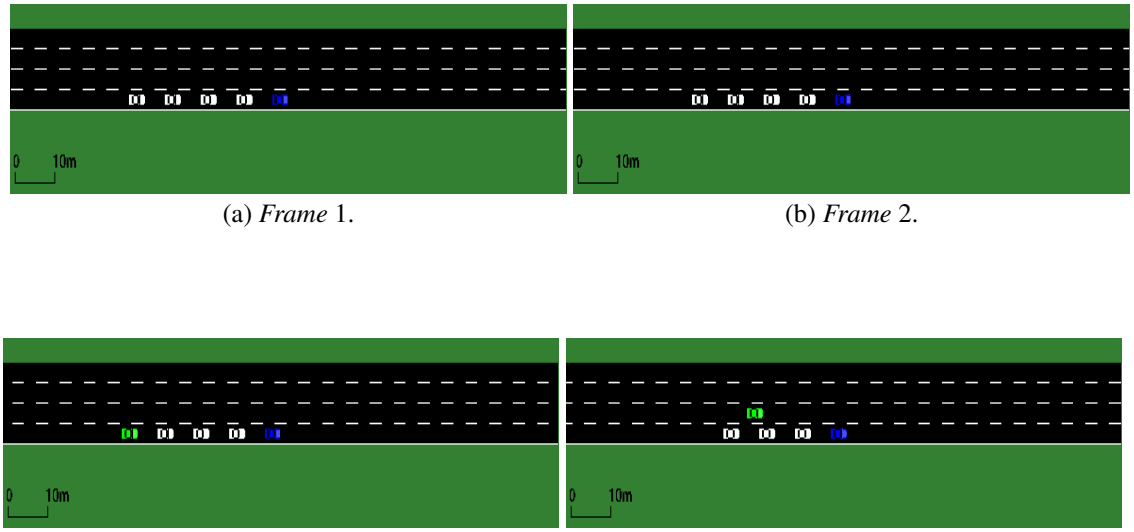


(c) Distância de cada veículo para o da frente.

Figura 4.8: Variações nas acelerações, velocidades e distâncias entre os veículos para o segundo cenário.

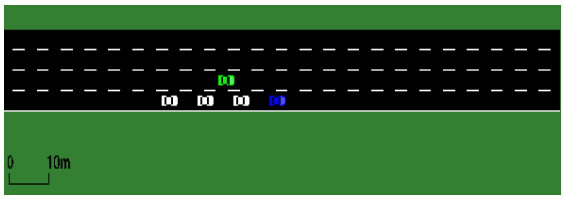
Novamente, nota-se que as execuções têm comportamentos diferentes a nível de conforto e segurança, tendo em conta a agressividade da manobra. Tal como explicado pelos cálculos em [42] e [51], há diferenças entre execuções da simulação quando se introduzem parâmetros diferentes no CACC. Essas diferenças são notórias na forma como a aceleração e velocidade dos veículos oscila nas duas execuções. É possível constatar que a execução que utiliza parâmetros de valor superior demonstra uma manobra muito mais rápida, mas com maior instabilidade. Portanto, como forma de manter a segurança da manobra, bem como o conforto durante a condução, seria ideal manter os valores *default* do simulador, apesar do acréscimo de tempo na duração total da manobra.

A figura seguinte mostra o que acontece no cenário 3.

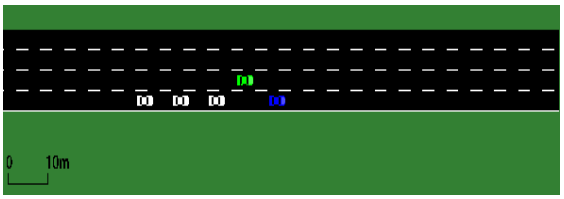


(a) *Frame* 1. (b) *Frame* 2.

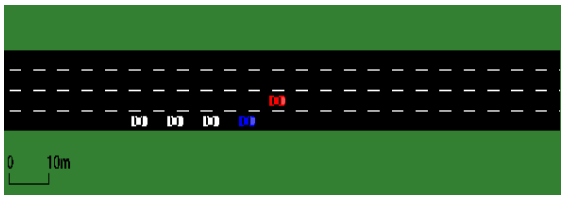
(c) *Frame* 3. (d) *Frame* 4.



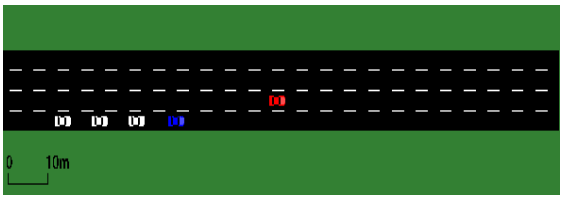
(e) *Frame* 5.



(f) *Frame* 6.



(g) *Frame* 7.



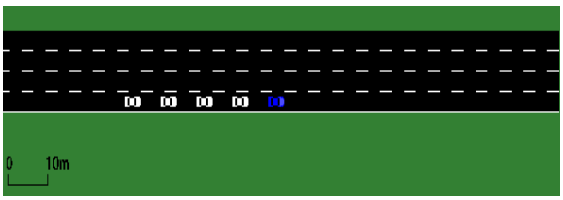
(h) *Frame* 8.

Figura 4.9: *Frames* da manobra no cenário 3.

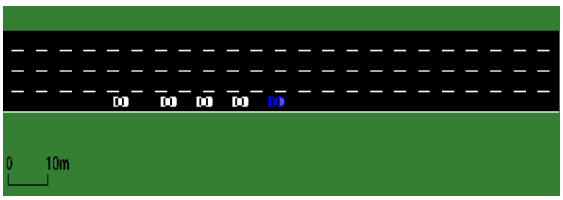
Neste cenário, os procedimentos são muito parecidos com os do cenário 2, sendo que a única diferença recai no processo de abertura no primeiro instante. Como o veículo verde se encontra na última posição, apenas tem de mudar de faixa para se mover para a posição indicada para se juntar de novo ao pelotão. A partir desse momento, a manobra decorre de igual forma do cenário 2.

No que diz respeito ao cenário 4, não havia interesse em estudar estes gráficos sendo que o processo de dissolução do pelotão é refletido numa mudança de faixa e mudança de velocidade para se distanciar do pelotão por parte de cada veículo.

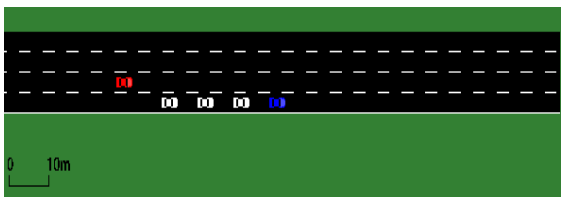
No entanto, para se ter alguma noção de como se procede à dissolução da formação, a figura 4.10 mostra os *frames* referentes a esse cenário.



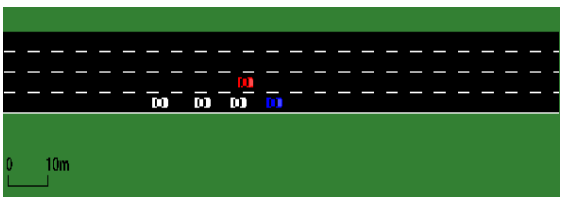
(a) *Frame* 1.



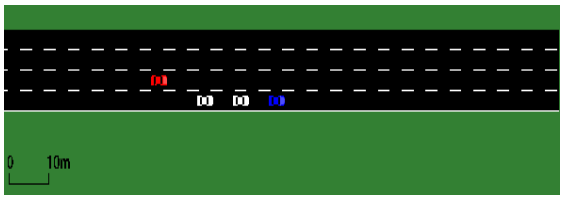
(b) *Frame* 2.



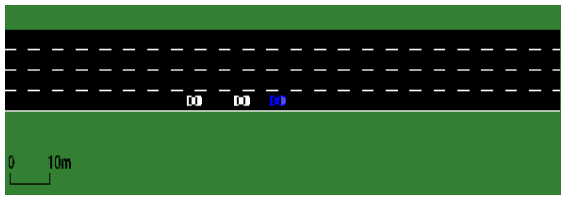
(c) *Frame* 3.



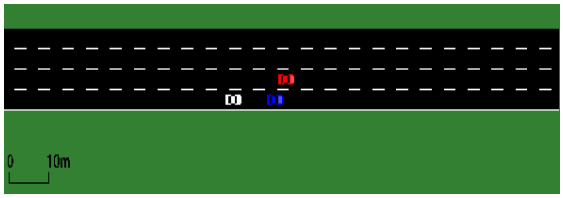
(d) *Frame* 4.



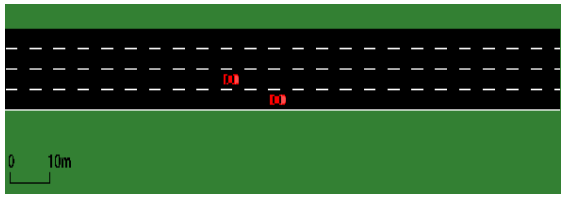
(e) *Frame* 5.



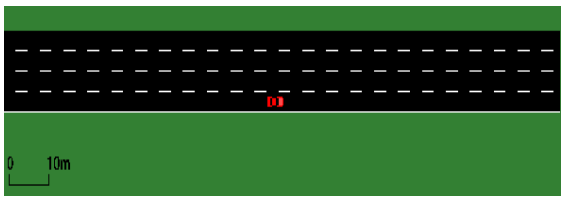
(f) *Frame* 6.



(g) *Frame* 7.



(h) *Frame* 8.



(i) *Frame* 9.

Figura 4.10: *Frames* da manobra no cenário 4.

No caso de não haver ninguém disponível para assumir a liderança do pelotão, inicia-se o processo de dissolução do mesmo. Neste cenário, os veículos que se encontram na última posição da formação vão saindo à vez, afastando-se ligeiramente do veículo da frente, mudando de faixa e separando-se do grupo. A manobra de dissolução está completa quando sobra apenas o líder inicial, que por sua vez também se tornou um veículo externo, ou seja, não pertencente a nenhum outro pelotão.

4.2 Simulação com perdas

Passando para a fase seguinte da simulação, tornou-se pertinente o estudo da manobra num cenário de perdas de dados nos pacotes trocados entre os veículos.

A extensão do Plexe para *platooning* oferece uma funcionalidade que reenvia um pacote sempre que este não chega ao seu destino. O número de tentativas por pacote é de 7, sendo que a manobra é interrompida caso a mensagem não seja entregue. Ou seja, sendo P_i a probabilidade de perda de uma mensagem, a probabilidade de falha da entrega de uma mensagem (P_f) é:

$$P_f = P_i^7 \tag{4.1}$$

A partir daqui, o objetivo era perceber a permiabilidade da manobra em relação a perdas que fossem até aos 50%. Para tal, foram feitos 50 testes para cada percentagem de perda[1] e os resultados são os apresentados na figura 4.11. A tabela 4.2 organiza as principais características dos veículos responsáveis pelas interferências na comunicação intra-pelotão.

Especificações dos veículos interferentes	
Tamanho do pacote	200 *Bytes*
Modelo camadas PHY/MAC	IEEE 802.11p/1609.4 canal único (CCH)
Frequência	5.89GHz
Bitrate	3Mbit/s
Potência de transmissão	100 mW
Velocidade de circulação	100 Km/h
Número de veículos	15

Tabela 4.2: Especificações referentes aos veículos externos das simulações.

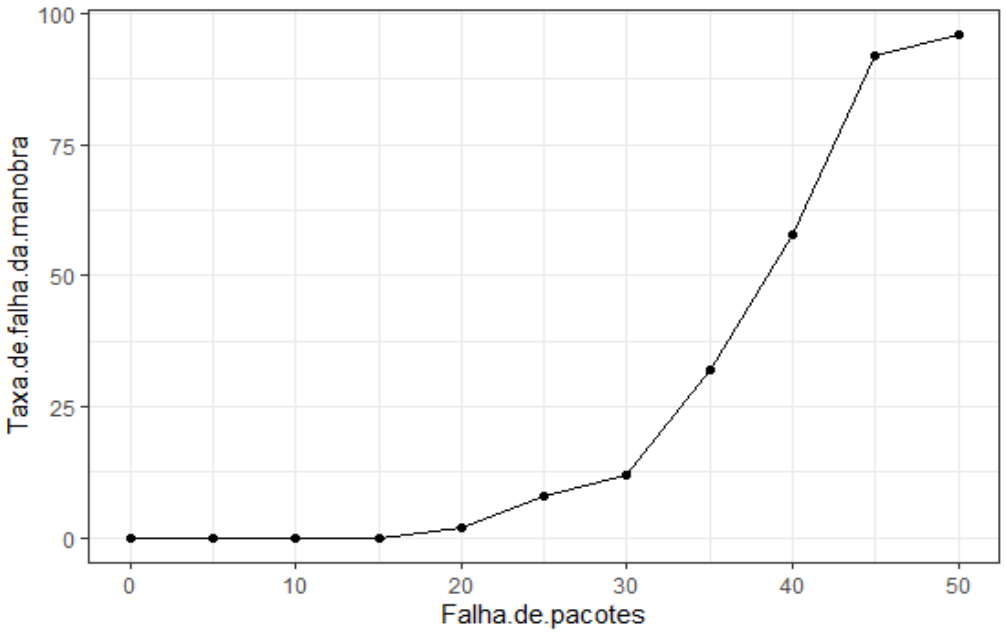


Figura 4.11: Impacto da perda de pacotes no desempenho da manobra.

A manobra apresenta resultados bastante promissores e com uma resistência a perda de pacotes para taxas de perda até 30%, rondando uma taxa de falha de manobra de 12%. A partir desse valor de perda de dados, a probabilidade de falha da manobra começa a tornar-se cada vez mais acentuada, não sendo vantajoso executá-la em tais condições. Por fim, a partir de uma percentagem de falha de mensagens de 45%, é possível prever uma probabilidade de falha da manobra com tendência para 100%.

[1] A partir desta fase de testes, foi apenas testado o cenário 2.

Outro dado interessante e que permite constatar a influência da variação de taxas de perda de mensagens é o número médio de pacotes entregues durante a manobra para a respetiva taxa, demonstrado no gráfico da figura 4.12.

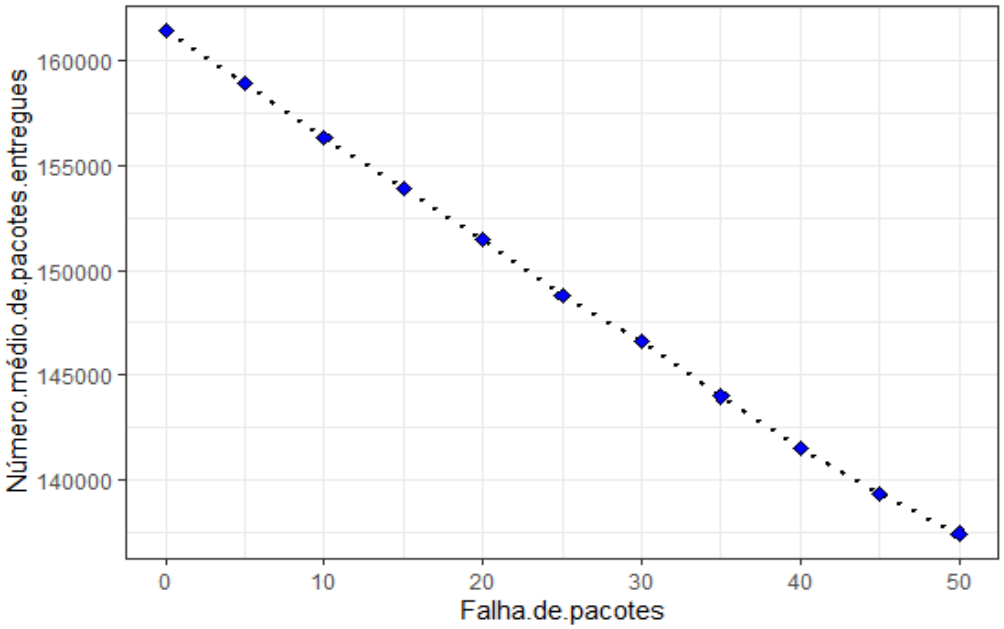


Figura 4.12: Número médio de pacotes entregues durante a execução da manobra.

4.3 Impacto da variação do *bitrate* no desempenho da manobra

Na última fase de simulações, estudou-se o impacto da variação do desempenho da manobra perante a variação da taxa de transmissão de dados. Esta análise foi feita usando o cenário 2, com e sem efeitos de congestionamento para também ser possível analisar até que ponto as comunicações usadas durante a execução da manobra se mantêm imunes a interferências de veículos externos.

Esta fase de simulação tornou-se pertinente graças à análise dos estudos efetuados em [52], [53] e [54] que explicam como o protocolo IEEE 802.11p pode funcionar usando diferentes taxas de tranferências de dados, sendo conclusivo em todos eles que o valor padrão de 6 Mbps pode não ser o mais eficaz tendo em conta o congestionamento dos canais de controlo durante o envio e receção das mensagens. Nos primeiros dois estudos, concluiu-se que valores maiores para taxa de transferência de dados acabam por ser mais vantajosos em casos de congestionamento, algo que se poderá confirmar de seguida. Isto acontece porque o aumento da taxa de transmissão de dados implica uma transmissão dos mesmos num menor espaço de tempo, havendo assim uma probabilidade menor de colisões. No entanto, no terceiro estudo ([54]) já há algum ceticismo em relação a tal informação. Começando por mencionar que o aumento do *bitrate* implica que haja um aumento do limite do SINR (*Signal-to-Noise Ratio*, mas neste caso considera as interferências, ou seja, *Signal-to-Interference+Noise-Ratio*) que permite a receção dos dados. Ou seja, se aumentando o valor de *bitrate*, o valor de SINR aumenta. Isto significa que tem de haver uma

potência de transmissão maior. E aqui surge outra questão que é a do alcance. Aumentando a potência de transmissão, aumenta o alcance não só para as comunicações dos veículos do pelotão, mas também para as comunicações interferentes. Para tornar os testes mais simples, foi definida uma potência de transmissão fixa (usado o valor padrão do simulador) e estudada a variação da taxa de tranferência de dados usando esse mesmo valor. Os valores para *bitrate* testados foram 3, 6, 9, 12, 18, 24, 27 Mbps.

4.3.1 Cenário descongestionado

Na secção 4.2 já se pôde observar a variação da taxa de falha perante perdas de pacotes até 50% sem qualquer tipo de congestionamento ou sobrecarga no canal de controlo (CCH), para um *bitrate* de 6 Mbps. O gráfico apresentado na figura 4.13 ajuda a perceber se, de facto, esse valor para tranferência de dados é o mais adequado para uma maior eficácia para a execução da manobra.

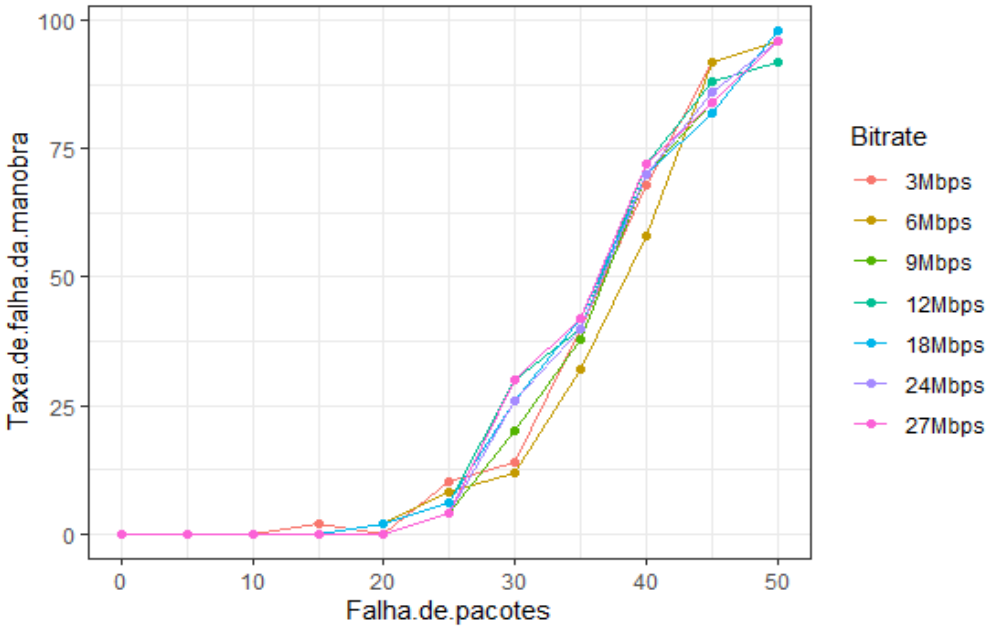


Figura 4.13: Percentagem de falha da manobra perante falha de pacotes para diferentes valores de *bitrate* num cenário sem congestiomento de trânsito.

Pode-se constatar que, no cômputo geral, o valor de 6 Mbps se apresenta como a escolha mais viável, demonstrando um desempenho bastante melhor do que os restantes para taxas de falha de pacotes a partir dos 30%, apesar de não ter a mesma capacidade a partir dos 45%.

No entanto, para perdas de dados até 30%, os valores de 9, 24 e 27 Mbps têm a melhor resistência, sendo que a partir dessa percentagem, o primeiro valor para o *bitrate* demonstra ser mais capaz de ultrapassar as perdas.

Posto isto, apesar de haver diferenças significativas em certos pontos do gráfico, de uma forma geral, todos têm um comportamento similar, pelo que se poderia aceitar o uso do valor norma

para o caso em que não há congestionamento causador de interferências na comunicação. Aliás, nesses mesmos pontos, o valor de 6 Mbps acaba até por ser o que melhor desempenho demonstra, portanto é totalmente plausível que se mantenha o uso deste *bitrate* para este cenário.

4.3.2 Cenário congestionado

O panorama num cenário em que existe congestionamento no canal de controlo provocado principalmente por mensagens externas interferentes já adquire contornos diferentes. Enquanto que no cenário congestionado as linhas referentes aos diferentes *bitrates* seguiam um percurso semelhante entre si, neste caso já há uma diferenciação que não passa despercebida, como se pode ver na figura 4.14.

Para o estudo das interferências no desenrolar da manobra, foram introduzidos no mapa do SUMO 15 veículos para o efeito. Não foi possível inserir mais veículos dado que o poder de processamento do computador usado não o permitia. Contudo, sendo que se trata de uma manobra que requer algum espaço para que se mantenham os requisitos de segurança, não se mostrou de grande pertinência a introdução de mais veículos na estrada do simulador.

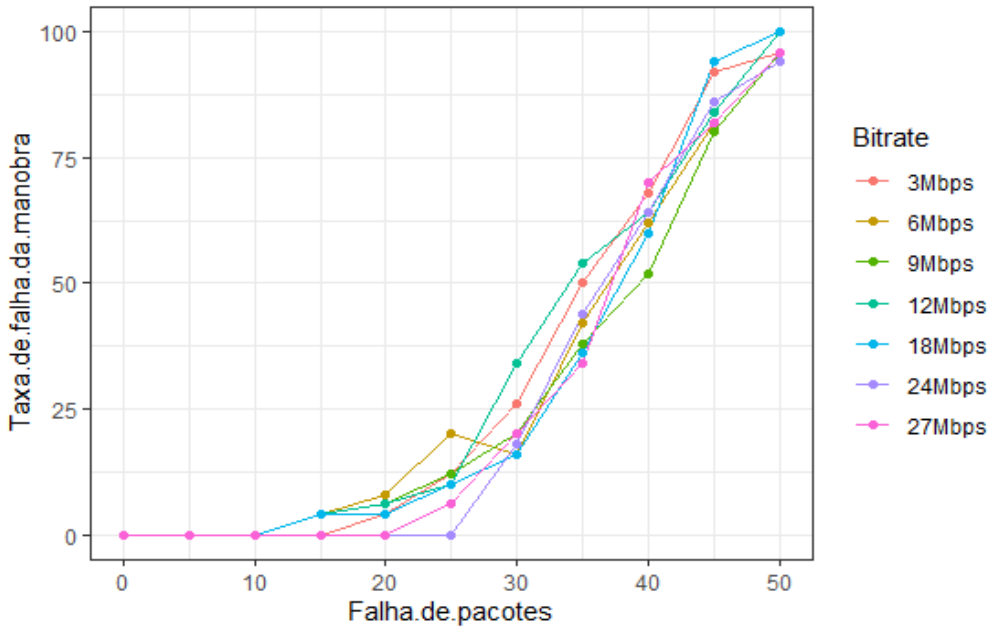


Figura 4.14: Percentagem de falha da manobra perante falha de pacotes para diferentes valores de *bitrate* num cenário com congestiomento de trânsito.

Ao contrário do que se constatou na simulação sem interferências de veículos externos, neste caso, torna-se evidente que o valor norma para a taxa de transferência de dados não é o mais eficaz. Tal como já se tinha mencionado, no estudo efetuado em [52], à medida que o valor de *bitrate* ia aumentado, melhor seria a resposta do sistema à sobrecarga do canal de controlo dos veículos envolvidos na manobra. O gráfico relativo ao cenário congestionado apresenta resultados que vão ao encontro das conclusões desse mesmo estudo.

Até aos 25% de dados perdidos nos pacotes trocados entre os veículos do pelotão, os valores para taxa de transferência de dados que melhor asseguram o bom funcionamento da manobra são os de 18, 24 e 27 Mbps, se bem que a partir dos 30%, o sistema começa a perder permiabilidade à falha de pacotes para uma taxa de tranferência de 24 Mbps.

A seguir aos valores mencionados nos parágrafos anteriores, vem o valor norma de 6 Mbps que não se adequa nem responde tão bem aos problemas inerentes a interferências de diversos veículos externos à formação, apenas conseguindo apresentar um desempenho melhor do que 3 e 12 Mbps e do que 24 Mbps para perdas acima de 30%.

Por último, é importante destacar que o sistema respondeu bem à perda de dados num cenário congestionado usando um valor de 9 Mbps de *bitrate*, pelo que também seria um valor a ter em conta.

Porém, tal como é dito em [52], as limitações em termos de potência de transmissão requirida para a utilização de *bitrates* iguais a 24 ou 27 Mbps não se revelam favoráveis para que estes valores se tornem valores *standard*. Outra razão para haver estas limitações parte da forma como o *hardware* responsável pela transmissão dos dados não apresentar condições que priveligiem a utilização desses valores. Assim sendo, poder-se-ia escolher o uma taxa de transferência de dados de 9 ou 18 Mbps, sendo que em casos adversos de falha, o primeiro valor revele maior capacidade de resposta.

4.4 Comparação com outras abordagens

Outra questão que surge é se esta manobra seria a mais adequada para a reorganização do pelotão. Por exemplo, e partindo do pressuposto que há um veículo disponível para liderar entre os seguidores iniciais, o que se poderia fazer seria a dissolução do pelotão e os veículos que mantinham a intenção de pertencer ao pelotão formavam um novo pelotão com esse mesmo veículo disponível a tomar a posição frontal. No entanto, em termos temporais e de estabilidade da manobra em si, esta abordagem não seria tão eficaz. Através do exemplo que o simulador usado trazia implementado em que se procedia à junção de um veículo na traseira da formação, verificou-se que os tempos para a manobra se realizar com sucesso eram parecidos com o tempo que a manobra implementada neste projeto leva para reorganizar o pelotão. Partindo de um pelotão com 5 veículos, saindo o líder original e mantendo a posição fixa do futuro líder, tem-se 3 veículos a pedir para se juntar de novo ao pelotão, ou seja, 3 vezes o tempo para um veículo se juntar, o que implica que demore 3 vezes mais tempo do que a manobra de reorganização da formação de veículos estudada e desenvolvida.

Como também já foi falado no capítulo 3, secção 3.6, outra abordagem poderia ser de tornar líder temporário o veículo que se posicionasse imediatamente atrás do líder original e libertar este último das suas funções. Depois, o veículo disponível poderia sair do pelotão temporariamente e juntar-se de novo ao mesmo pela frente. No entanto, esta abordagem não é muito viável pelos níveis de estabilidade comparativamente à manobra estudada, tendo em conta que o veículo que se tornaria o novo líder precisaria de informação sobre a posição e velocidade do pelotão, mesmo

estando fora dele. Esta informação é mais precisa e mais fácil de se retirar num cenário em que o futuro líder se junta de novo no pelotão atrás do líder original, em vez de se juntar à frente do líder temporário.

Capítulo 5

Conclusão

5.1 Conclusões principais

Em suma, é possível dizer que esta manobra pode revelar-se uma ferramenta útil para o mundo automóvel, mais especificamente do *platooning* graças à sua capacidade de manter a formação intacta, movendo apenas o veículo disponível para se tornar líder para uma posição externa à mesma. Toda esta forma organizada de atualizar o comandante do pelotão permite manter a área de execução o mais organizada possível de modo a evitar embaraços na estrada. Para além disso, esta manobra garante a segurança de todos os seus participantes mantendo sempre distâncias de segurança adequadas para cada fase da sua execução. Esta manobra cobre todos os cenários possíveis, incluindo o pior deles todos que obriga à dissolução do pelotão. Esta dissolução pode tornar-se problemática pelo simples facto de poder causar congestionamento e situações de perigo ao sair do pelotão, sendo assim preferível manter ao máximo a formação inalterada.

Em termos temporais, notou-se que a formação responde bem à manobra, conseguindo executá-la num espaço de 75 segundos para todos os cenários à exceção do cenário de dissolução e do cenário em que o futuro líder se encontra imediatamente atrás do líder original. Neste caso demora cerca de 15 segundos. Estes tempos revelam-se favoráveis, principalmente em viagens longas e com espaço de manobra, pela relação tempo/utilidade.

No que diz respeito ao lado mais técnico da questão, nomeadamente do controlador CACC, foi possível verificar que aumentando os seus parâmetros ξ e ω_n, a manobra se desenrola mais rapidamente, perdendo, no entanto, estabilidade resultante das mudanças brutas na aceleração dos veículos. Esta perda de estabilidade pode causar desconforto durante a condução, pelo que não se revela uma prioridade alterar os valores padrão do controlador, independentemente de levar mais tempo a concluir o processo de reorganização.

A resposta da manobra perante perdas de pacotes também se revelou promissora visto que a taxa de execução da manobra perante uma perda de pacotes até 20% se manteve praticamente igual a 100% (com 90% de eficácia até 30% de pacotes perdidos). Sendo este um sistema que requer segurança máxima e que está dependente do bom desempenho das tecnologias comunicativas,

assume-se que a perda dos pacotes não irá para além dos 5%, pelo que se pode concluir que a manobra não terá problemas deste foro, em condições normais.

Por fim, numa análise relacionada com a resposta da manobra a interferências externas nas comunicações, concluiu-se que o valor padrão do *bitrate* de 6 Mbps poderá não ser o mais adequado para a execução da manobra, dependendo do estado de congestionamento da via em que o pelotão se insere:

- **Cenário descongestionado -** Para este caso, todos os valores disponíveis para *bitrate* têm um comportamento idêntico, havendo apenas algumas taxas de perda para as quais existem algumas discrepâncias. No entanto, o valor padrão acaba por ser o que tem um desempenho geral mais competente, não havendo necessidade ou utilidade na sua mudança;
- **Cenário congestionado -** Ao contrário do caso anterior, neste já existem diferenças nos desempenhos para valores de *bitrate* diferentes. Concluiu-se que a taxa de tranferência de dados mais adequada perante um cenário de grande congestionamento seria de 9 Mbps ou 18 Mbps, sendo que o primeiro valor apresenta uma resposta mais eficaz em condições mais adversas.

Novamente e antes de passar para possíveis trabalhos futuros relacionados com esta manobra ou comunicação destinada a *platooning*, é de mencionar que esta forma de reorganizar o pelotão reúne características muito interessantes mesmo sob perdas e condições anormais, podendo vir a ser uma grande aquisição no mundo dos automóveis autónomos, mais concretamente no mundo do *platooning*.

Este projeto também se revelou essencial para o desenvolvimento de novos conhecimentos sobre o que acontece no universo dos veículos autónomos e suas comunicações, desde a forma como é feita a comunicação entre eles e como funcionam as operações multicanal. Permitiu também adquirir aptidões no que diz respeito ao trabalho com o Veins, Plexe-Veins, OMNeT++, SUMO e R (linguagem de tratamento de dados), e melhorar a capacidade de programação em C++. Tornou-se ainda um trabalho muito gratificante para desenvolvimento a nível pessoal em aspetos de autonomia, organização e auto-superação.

5.2 Trabalhos futuros

Como forma de complementar esta manobra, mas também como ferramenta para uso em projetos relacionados com IVC e *platooning* seria interessante implementar algumas funcionalidades. Dentro deste leque de possíveis inovações perfilam:

- Uma algoritmo de análise e determinação do grau de capacidade de um determinado condutor/veículo tem para verificar se pode ser escolhido como líder de um pelotão. Desta forma será possível fazer uma triagem inicial e perceber quem está ou não apto para assumir o comando de qualquer tipo de pelotão. A par deste algoritmo, pode ser desenvolvido um outro capaz de fazer a escolha de líder mais adequada tirando partido da análise de competências dos diferentes candidatos;

- Um estudo sobre o melhor conjunto de potência de transmissão e *bitrate* que permita um melhor desempenho nas comunicações da manobra em situações de congestionamento;
- Um mecanismo que permita isolar o pelotão e impedir que mensagens externas que não tenham qualquer tipo de interesse para o mesmo constituam uma forma de interferência. Apesar de ser uma tarefa muito difícil, este mecanismo poderia melhorar bastante os desempenhos do ramo comunicativo em cenários de grande congestionamento que implica que haja um número muito maior de mensagens partilhadas.

Apenas algumas ideias dentro de um universo em que ainda há muito por explorar, mas com um único sentido - o da inovação.

Referências

[1] BMW. The path to autonomous driving. https://www.bmw.com/en/automotive-life/autonomous-driving.html. Acedido em: 26 de Dezembro de 2019.

[2] BMW. The road to autonomous driving. https://www.bmw.com/en/innovation/the-development-of-self-driving-cars.html. Acedido em: 26 de Dezembro de 2019.

[3] Mercedes. Daimler AG and BMW Group launch cooperation for automated driving. https://www.mercedes-benz.com/en/innovation/daimler-ag-and-bmw-group-launch-cooperation-for-automated-driving/. Acedido em: 27 de Dezembro de 2019.

[4] Continental. Autonomous mobility. https://www.continental-automotive.com/en-gl/Passenger-Cars/Autonomous-Mobility. Acedido em: 02 de Janeiro de 2020.

[5] Continental. Autonomous mobility. https://www.continental-automotive.com/en-gl/Passenger-Cars/Vehicle-Networking. Acedido em: 02 de Janeiro de 2020.

[6] PORDATA. Acidentes de viação com vítimas, feridos e mortos - Continente. https://www.pordata.pt/Portugal/Acidentes+de+via%C3%A7%C3%A3o+com+v%C3%ADtimas++feridos+e+mortos+++Continente-326. Acedido em: 15 de Janeiro de 2019.

[7] SAE. SAE International Releases Updated Visual Chart for Its "Levels of Driving Automation" Standard for Self-Driving Vehicles. https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles. Acedido em: 12 de Janeiro de 2020.

[8] Karen Hao. The three challenges keeping cars from being fully autonumous. https://www.technologyreview.com/s/613399/the-three-challenges-keeping-cars-from-being-fully-autonomous/. Acedido em: 28 de Janeiro de 2020.

[9] Daniel Jesus. Estará a população a perder confiança nos carros autónomos? https://pplware.sapo.pt/high-tech/

estarao-os-carros-autonomos-a-perder-a-confianca-da-populacao/. Accessed: 28 de Janeiro de 2020.

[10] Dorothy J. Glancy. Autonomous and Automated and Connected Cars—Oh My! First Generation Autonomous Cars in the Legal Ecosystem. *Minnesota Journal of Law*, páginas 623–625, 2015. Volume 2.

[11] Giulio Vivo, Paola Dalmasso, Erik Nordin, Marco Dozza, Paolo Cravini, Fabio Codeca, Vincenzo Manzoni, Javier Ibanez-Guzman, e A. Mokaddem. V2V Applications in the SAFESPOT European Project: The OEMs Experience. Em *16th World Congress and Exhibition on Intelligent Transport Systems and Services, ITS 2009*, 01 2009.

[12] Ravi Puvvala. How V2X Sensors can Augment ADAS for Connected & Automated Driving. Em *2018 International Wireless Industry Consortium*, página 14, 2018.

[13] Michael Lammert, Adam Duran, Jeremy Diez, Kevin Burton, e Alex Nicholson. Effect of platooning on fuel consumption of class 8 vehicles over a range of speeds, following distances, and mass. *SAE International Journal of Commercial Vehicles*, 7:626–639, 05 2014.

[14] Mani Amoozadeh, Hui Deng, Chen-Nee Chuah, e Dipak Ghosal. Platoon management with cooperative adaptive cruise control enabled by vanet. *Vehicular Communications*, 2, 03 2015.

[15] Tim Dawkins. Autonomous Cars 101: What Sensors Are Used in Autonomous Vehicles? https://levelfivesupplies.com/sensors-used-in-autonomous-vehicles/. Aceido em: 13 de Janeiro de 2019.

[16] UnitedLex. From Google to Tesla, it's a War of LiDAR or RADAR. https://www.unitedlex.com/news/from-google-to-tesla-its-a-war-of-lidar-or-radar. Acedido em: 12 de Janeiro de 2017.

[17] BOSCH. Fourth generation long-range radar sensor (lrr4). Relatório técnico 292000P0ZHC/CCA-201407-En, BOSCH, 2014.

[18] Velodyne Lidar. Alpha prime™. Relatório técnico pode ser acedido por https://velodynelidar.com/products/alpha-prime/, Velodyne Lidar, 2019.

[19] BOSCH. Ultrasonic sensor. Relatório técnico pode ser acedido por https://www.bosch-mobility-solutions.com/media/global/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/construction-zone-assist/ultrasonic-sensor/onepager_ultrasonic_sensor_en_200203.pdf, BOSCH, 2019.

[20] BOSCH. Multi purpose camera. Relatório técnico pode ser acedido por https://www.bosch-mobility-solutions.com/media/global/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/multi-camera-system/multi-purpose-camera/summary_multi-purpose-camera_en.pdf, BOSCH, 2019.

[21] W. Liu, S. Muramatsu, e Y. Okubo. Cooperation of V2I/P2I Communication and Roadside Radar Perception for the Safety of Vulnerable Road Users. Em *2018 16th International Conference on Intelligent Transportation Systems Telecommunications (ITST)*, Oct 2018.

[22] Qualcomm. *Leading the world to 5G: Cellular Vehicle-to-Everything (C-V2X) technologies*, Junho de 2016.

[23] Autotalks. DSRC vs. C-V2X for Safety Applications. `https://www.auto-talks.com/technology/dsrc-vs-c-v2x-2/`. Acedido em: 29 de Janeiro de 2020.

[24] Beylot Andre-Luc. *Vehicular networks: models and algorithms*. ISTE, 2013.

[25] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, e R. Herrtwich. Design of 5.9 ghz dsrc-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, October 2006.

[26] Q. Chen, D. Jiang, e L. Delgrossi. Ieee 1609.4 dsrc multi-channel operations and its implications on vehicle safety communications. Em *2009 IEEE Vehicular Networking Conference (VNC)*, páginas 1–8, 2009.

[27] Claudia Campolo, Antonella Molinaro, e Alexey V. Vinel. Understanding the performance of short-lived control broadcast packets in 802.11p/wave vehicular networks. *2011 IEEE Vehicular Networking Conference (VNC)*, páginas 102–108, 2011.

[28] Xuyu Wang, Shiwen Mao, e Michelle Gong. An Overview of 3GPP Cellular Vehicle-to-Everything Standards. *GetMobile: Mobile Computing and Communications*, 21:19–25, 11 2017.

[29] Autotalks. Accelerating V2X Global Deployment Road for Safety. Relatório técnico, Autotalks, 2019.

[30] Autotalks. Craton2. `https://www.auto-talks.com/wp-content/uploads/2019/03/CRATON2-2019.pdf`. Acedido em: 29 de Dezembro de 2020.

[31] Autotalks. Secton. `https://www.auto-talks.com/wp-content/uploads/2019/03/SECTON-2019-2.pdf`. Acedido em: 29 de Dezembro de 2020.

[32] Qualcomm. *The wireless edge transformation has begun and will realize 5G's full potential*, Agosto de 2018.

[33] Kelechi Eze, Matthew Sadiku, e Sarhan Musa. 5g wireless technology: A primer. *International Journal of Scientific Engineering and Technology*, 7:62–64, 07 2018.

[34] Mohsen Attaran. 5g wireless: A transformative, disruptive technology. *Industrial Management (Norcross, Georgia)*, 61:16–21, 06 2019.

[35] Morgan Kurk. Fast-forward to the future with 5g networks. *CommScope*, 2018.

[36] Qualcomm. Snapdragon x55 5g modem-rf system. `https://www.qualcomm.com/products/snapdragon-x55-5g-modem`. Acedido em: 29 de DEzembro de 2020.

[37] EUCAR. *EUCAR Project Book 2020*. EUCAR, 2020.

[38] 5G MOBIX. 5g mobix. `https://www.5g-mobix.com/`. Acedido em: Outubro de 2020.

[39] C-Roads. C-roads. https://www.c-roads.eu/platform.html. Acedido em: Outubro de 2020.

[40] Cherise Threewitt. Cars That Are Almost Self-Driving. https://cars.usnews.com/cars-trucks/cars-that-are-almost-self-driving. Acedido em: 26 de Dezembro de 2019.

[41] Mani Amoozadeh, Hui Deng, Chen-Nee Chuah, H Michael Zhang, e Dipak Ghosal. Platoon management with cooperative adaptive cruise control enabled by vanet. *Vehicular Communications*, 2(2):110–123, 2015.

[42] Michele Segata, Stefan Joerer, Bastian Bloessl, Christoph Sommer, Falko Dressler, e Renato Lo Cigno. PLEXE: A Platooning Extension for Veins. Em *6th IEEE Vehicular Networking Conference (VNC 2014)*, páginas 53–60, Paderborn, Germany, 12 2014. IEEE.

[43] Christoph Sommer, Reinhard German, e Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing (TMC)*, 10(1):3–15, January 2011.

[44] OMNeT++. Omnet++ 5.5.1 released. https://omnetpp.org/software/2019/05/31/omnet-5-5-released. Acedido em: Maio de 2020.

[45] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, e E. Wiessner. Microscopic traffic simulation using sumo. Em *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, páginas 2575–2582, 2018.

[46] Renato Ferreira. renatoferreira96/platooning. https://github.com/renatoferreira96/platooning, Sep 2020.

[47] Christoph Sommer, David Eckhoff, Alexander Brummer, Dominik Buse, Florian Hagenauer, Stefan Joerer, e Michele Segata. *Veins: The Open Source Vehicular Network Simulation Framework*, páginas 215–252. Springer, 05 2019.

[48] German Aerospace Center (DLR) et al. Traci. https://sumo.dlr.de/docs/TraCI.html#v2x_simulation. Acedido em: Maio de 2020.

[49] Michele Segata. Plexe documentation. http://plexe.car2x.org/documentation/. Acedido em: Maio de 2020.

[50] R Rajamani. *Vehicle Dynamics and Control*. Springer, 01 2006.

[51] Michele Segata. *Safe and Efficient Communication Protocols for Platooning Control*. Phd thesis, University of Innsbruck, 2 2016.

[52] Miguel Sepulcre, J. Gozalvez, e Baldomero Coll-Perales. Why 6 mbps is not (always) the optimum data rate for beaconing in vehicular networks. *IEEE Transactions on Mobile Computing*, PP:1–1, 04 2017.

[53] Xiaofeng Liu e Jaekel Arunita. Congestion control in v2v safety communication: Problem, analysis, approaches. *Electronics*, 8, 05 2019.

[54] Qing Xu, Tony Mak, Jeff Ko, e Raja Sengupta. Vehicle-to-vehicle safety messaging in dsrc. Em *Proceedings of the First ACM International Workshop on Vehicular Ad Hoc Networks, Held in Conjunction with MobiCom 2004, October 1, 2004, Philadelphia, Pennsylvania, USA.*, páginas 19–28, 01 2004.

Anexo A

Implementação de manobras de reorganização de veículos em pelotão usando comunicações inter-veículares

Renato A. S. Ferreira*, Rui E. Araújo†, António F. Lopes‡
Faculdade de Engenharia da Universidade do Porto

***Resumo*—Ao longo dos últimos 30 anos, diversas têm sido as inovações no domínio dos carros autónomos, principalmente no que diz respeito a *platooning*. Como forma de melhorar a capacidade comunicativa dentro dos pelotões, surgiu um novo conceito chamado IVC (*Inter-vehicle communications*). Através deste modo comunicativo foi possível implementar diversas manobras executadas pelo pelotão como a entrada ou saída de um veículo da formação. Caso o veículo que quisesse sair fosse o líder, problemas de estabilidade e segurança seriam levantados. Então, para colmatar alguns desses problemas, foi criada manobra de reorganização do pelotão que localiza um novo líder e coloca-o no comando da formação. Será ainda possível verificar o comportamento da manobra em condições de perdas de pacotes e perante taxas de tranferências de dados diferentes, com especial atenção para a avaliação do melhor valor dessa mesma taxa em comparação ao valor padrão de 6 MBPs.**

***Index Terms*—*Bitrate*, IVC, líder, pelotão.**

I. Introdução

AO longo dos anos, a inovação e investigação feitas no âmbito do desenvolvimento de abordagem comunicativa capaz de auxiliar o processo de autonomização de veículos permitiu a criação do conceito de *platooning* em que há um conjunto ordenado de carros que formam uma fila e partilham informações entre si através de IVC, sendo que o carro à frente do pelotão é o líder. A partir daqui foi possível criar manobras para entrada e saída de carros na formação. No entanto, surgiu um problema: saída do líder e necessidade de reorganização do pelotão. Como solução, em [1] é apresentada uma solução para tal. Contudo, não apresenta uma exploração da reorganização espacial, assumindo que o novo líder é o primeiro seguidor, pelo que foi criada uma manobra que efetivamente procede à reorganização no espaço da formação. Esta manobra foi baseada na tecnologia de DSRC, recorrendo principalmente aos protocolos IEEE 802.11p e IEEE 1609.4 para comunicações veículares e operações multi-canal, respetivamente. Para a implementação, usou-se o Plexe-Veins [2], uma extensão *open-source* para estudo de pelotões para Veins [3]. Tanto a arquitectura como a organização de camadas de comunicação do Veins são detalhadamente explicadas em [4].

II. Implementação do protocolo

Assumindo que existe um pelotão constituído por cinco veículos, sendo o *Leader*, o *Available* e o *Follower* representados por L, A e F, respetivamente, esta é a sugestão para a implementação da manobra de reorganização do pelotão:

1) L envia uma mensagem a todos os membros do pelotão informando-os da sua intenção de deixar o pelotão;
2) Apenas carros do tipo A podem responder de volta a esta mensagem. Caso não haja nenhum carro disponível para ser o novo líder, passa-se ao processo de dissolução da formação;
3) Após receber a informação sobre o seu substituto, L informa o carro do tipo F imediatamente atrás de A para abrir espaço de segurança para que este possa executar a próxima ação sem causar problemas na via. Para tal, F deve tornar-se líder temporário do pelotão formado por ele próprio e pelos carros que se encontram atrás dele. Esta ação é fulcral para que esses mesmos carros não percam uma referência assim que se retirar um veículo do meio do pelotão;
4) Assim que as condições estiverem reunidas, A muda de faixa de rodagem e abandona temporariamente o pelotão;
5) L ordena que o espaço que foi aberto anteriormente seja fechado e que A se coloque na posição desejada para se juntar de novo à formação;
6) Assim que A chega à sua nova posição, notifica L;
7) L ordena a nova criação de espaçamento para que A mude para a faixa em que o pelotão circula, ocupando a segunda posição da formação;
8) Depois de se fechar de novo o espaço criado, A torna-se o novo líder do pelotão;
9) L torna-se num carro externo e abandona o pelotão.

Sempre que um veículo se torna externo ao pelotão, ainda que temporariamente, o seu controlador torna-se do tipo ACC de modo a evitar possíveis situações de colisão com outros carros presentes na via de circulação, por exemplo. Sempre que se junta a um pelotão, o controlador muda para CACC.

III. Resultados

A manobra apresenta uma boa resposta, tendo sido estudada a sua permiabilidade a perdas de pacotes, sendo que o gráfico com os resultados está na figura 1. Na tabela I pode-se ver as especificações usadas nas simulações.

*up201405113@fe.up.pt
†raraujo@fe.up.pt
‡antonio.lopes@fe.up.pt

Tabela I
ESPECIFICAÇÕES DAS SIMULAÇÕES.

Especificações	
Tamanho do pacote	200 *Bytes*
Taxa de perda de pacotes	0%
Modelo camadas PHY/MAC	IEEE 802.11p/1609.4 canal único (CCH)
Frequência	5.89GHz
Bitrate	6Mbit/s
Potência de transmissão	100 mW
Velocidade de circulação	100 Km/h
Número de carros utilizados	5
Número máximo de carros possível	8

Os resultados mostraram-se promissores visto que uma taxa de perda não deve atingir valores superiores a 5% e a manobra funciona na perfeição perante taxas de perdas até esse valor.

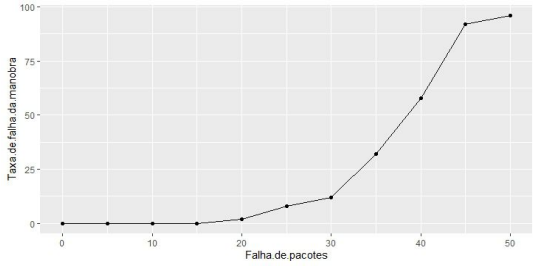


Figura 1. Variação da taxa de falha de manobra consoante a taxa de perda de pacotes.

A manobra apresenta resultados bastante promissores e com uma resistência a perda de pacotes para taxas de perda até 30%, rondando uma taxa de falha de manobra de 12%, tal como pode ser visto na figura 1. A partir desse valor de perda de dados, a probabilidade de falha do protocolo começa a tornar-se cada vez mais acentuada, não sendo vantajoso executar a manobra em tais condições. Por fim, a partir de uma percentagem de falha de mensagens de 45%, é possível prever uma probabilidade de falha da manobra com tendência para 100%.

Em seguida foi feita uma avaliação do comportamento da manobra variando o *bitrate*, visto que há estudos que defendem que há casos em que este valor pode não ser o mais viável a usar, como por exemplo [5], principalmente em casos em que há interferências nas comunicações que causam colisões nos canais de comunicação. No entanto, foi feita uma avaliação também para o caso em que não há interferências e os resultados revelaram que, de facto, o valor padrão de 6 MBPs, no cômputo geral, apresenta resultados que favorecem o melhor desempenho e eficiência na comunicação necessária para o desenrolar da manobra, com uma capacidade de resposta muito melhor do que os restantes valores a partir do 30%. De notar que o valor de 9 MBPs também apresenta resultados interessantes.

As simulações terminaram com a avaliação do *bitrate* ideal para o melhor funcionamento da manobra em situações em que há interferências causadas por carros externos. Era esperado que aumentando a taxa de transferência de dados, a probabilidade de colisões no canal de controlo diminuísse. No entanto, tal como é dito em [6], este aumento requer potências de transmissão maiores, o que pode implicar maior de alcance dos sinais interferentes. Foi definida uma potência de 100mW, analisando os resultados a partir desse valor. Foram usados 15 carros externos para o efeito de interferências. Tornou-se evidente que o valor norma para a taxa de transferência de dados não é o mais eficaz para o caso em que há interferências nas comunicações. Tal como já se está mencionado no estudo efetuado em [5], à medida que o valor de *bitrate* aumenta, melhor será a resposta do sistema à sobrecarga do canal de controlo dos veículos envolvidos na manobra. Foi possível determinar que o valor de bitrate com melhor desempenho geral acabou por ser o de 9 MBPs, tendo em conta limitações e requisitos adjacentes ao uso dos valores de 24 e 27 MBPs [5].

IV. CONCLUSÃO

A manobra revela um desempenho promissor, visto que:

- Em termos temporais, ocupa, no máximo, 75 segundos e mantém a via mais organizada ao evitar a separação de carros que pode causar situações de embaraço;
- Revela ainda bastante resistente a perdas de pacotes mantendo uma taxa de sucesso de praticamente 100% até 20% de pacotes perdidos;
- Apresenta melhor resposta em cenários com interferências quando usa um *bitrate* de 9MBPs ou 18 MBPs, sendo que o primeiro valor apresenta melhor resposta para casos mais adversos.

Estes dados revelam que esta manobra pode tornar-se uma importante ferramenta no mundo do *platooning*.

AGRADECIMENTOS

Agradecimentos especiais aos meus pais, ao meu irmão - os meus pilares - e restantes familiares. Aos meus amigos da faculdade por me acompanharem na vida académica e aos amigos mais próximos pelo acompanhamento na vida. Um agradecimento ao Professor Doutor Michele Segata pela ajuda para entender o simulador usado. Por fim, mas não menos importantes, um agradecimento ao meu orientador, Professor Rui Esteves Araújo, e co-orientador, Engenheiro António Manuel Figueiredo Lopes pelos conhecimentos e partilha de ideias.

REFERÊNCIAS

[1] M. Amoozadeh, H. Deng, C.-N. Chuah, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by vanet," *Vehicular Communications*, vol. 2, 03 2015.

[2] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. Lo Cigno, "PLEXE: A Platooning Extension for Veins," in *6th IEEE Vehicular Networking Conference (VNC 2014)*. Paderborn, Germany: IEEE, 12 2014, pp. 53–60.

[3] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing (TMC)*, vol. 10, no. 1, pp. 3–15, 1 2011.

[4] C. Sommer, D. Eckhoff, A. Brummer, D. Buse, F. Hagenauer, S. Joerer, and M. Segata, *Veins: The Open Source Vehicular Network Simulation Framework*. Springer, 05 2019, pp. 215–252.

[5] M. Sepulcre, J. Gozalvez, and B. Coll-Perales, "Why 6 mbps is not (always) the optimum data rate for beaconing in vehicular networks," *IEEE Transactions on Mobile Computing*, vol. PP, pp. 1–1, 04 2017.

[6] Q. Xu, T. Mak, J. Ko, and R. Sengupta, "Vehicle-to-vehicle safety messaging in dsrc," in *Proceedings of the First ACM International Workshop on Vehicular Ad Hoc Networks, Held in Conjunction with MobiCom 2004, October 1, 2004, Philadelphia, Pennsylvania, USA.*, 01 2004, pp. 19–28.

Anexo B

B.1 LeaveManeuver.cc

```
#include
 ↪ "veins/modules/application/platooning/maneuver/LeaveManeuver.h"
#include
 ↪ "veins/modules/application/platooning/apps/GeneralPlatooningApp.h"

LeaveManeuver::LeaveManeuver(GeneralPlatooningApp* app)
    :Maneuver(app)
{
}

void LeaveManeuver::onManeuverMessage(const ManeuverMessage* mm)
{
    if (const Leave* msg = dynamic_cast<const Leave*>(mm)) {
        handleLeave(msg);
    }
    else if (const LeaveAck* msg = dynamic_cast<const
     ↪ LeaveAck*>(mm)) {
        handleLeaveAck(msg);
    }
    else if (const Close* msg = dynamic_cast<const Close*>(mm)) {
        handleClose(msg);
    }
    else if (const CloseAck* msg = dynamic_cast<const
     ↪ CloseAck*>(mm)) {
        handleCloseAck(msg);
    }
    else if (const Gap* msg = dynamic_cast<const Gap*> (mm)) {
        handleGap(msg);
```

```
25      }
26      else if (const GapAck* msg = dynamic_cast<const GapAck*> (mm)) {
27          handleGapAck(msg);
28      }
29      else if (const Lane* msg = dynamic_cast<const Lane*> (mm)) {
30          handleLane(msg);
31      }
32      else if (const LaneAck* msg = dynamic_cast<const LaneAck*> (mm))
        ↪ {
33          handleLaneAck(msg);
34      }
35      else if (const MoveToPosition* msg = dynamic_cast<const
        ↪ MoveToPosition*>(mm)) {
36          handleMoveToPosition(msg);
37      }
38      else if (const MoveToPositionAck* msg = dynamic_cast<const
        ↪ MoveToPositionAck*>(mm)) {
39          handleMoveToPositionAck(msg);
40      }
41      else if (const JoinFormation* msg = dynamic_cast<const
        ↪ JoinFormation*>(mm)) {
42          handleJoinFormation(msg);
43      }
44      else if (const JoinFormationAck* msg = dynamic_cast<const
        ↪ JoinFormationAck*>(mm)) {
45          handleJoinFormationAck(msg);
46      }
47  }
48
49  Leave* LeaveManeuver::createLeave(int vehicleId, std::string
    ↪ externalId, int platoonId, int destinationId, double
    ↪ platoonSpeed, int platoonLane, const std::vector<int>&
    ↪ newPlatoonFormation)
50  {
51      Leave* msg = new Leave("Leave");
52      app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
53      msg->setPlatoonSpeed(platoonSpeed);
54      msg->setPlatoonLane(platoonLane);
```

```
55
        ↪ msg->setNewPlatoonFormationArraySize(newPlatoonFormation.size());
56      for (unsigned int i = 0; i < newPlatoonFormation.size(); i++) {
57          msg->setNewPlatoonFormation(i, newPlatoonFormation[i]);
58      }
59      return msg;
60  }
61
62  LeaveAck* LeaveManeuver::createLeaveResponse(int vehicleId,
    ↪ std::string externalId, int platoonId, int destinationId)
63  {
64      LeaveAck* msg = new LeaveAck("LeaveAck");
65      app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
66      return msg;
67  }
68
69  Close* LeaveManeuver::createCloseGap(int vehicleId, std::string
    ↪ externalId, int platoonId, int destinationId)
70  {
71      Close* msg = new Close("Close");
72      app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
73      return msg;
74  }
75
76  CloseAck* LeaveManeuver::createCloseGapAck(int vehicleId,
    ↪ std::string externalId, int platoonId, int destinationId)
77  {
78      CloseAck* msg = new CloseAck("CloseAck");
79      app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
80      return msg;
81  }
82
83  Gap* LeaveManeuver::createGap(int vehicleId, std::string externalId,
    ↪ int platoonId, int destinationId, double distance)
84  {
85      Gap* msg = new Gap("Gap");
```

```
86      app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
87      msg->setDistance(distance);
88      return msg;
89  }
90
91  GapAck* LeaveManeuver::createGapAck(int vehicleId, std::string
    ↪ externalId, int platoonId, int destinationId, double
    ↪ platoonSpeed, int platoonLane, const std::vector<int>&
    ↪ newPlatoonFormation)
92  {
93      GapAck* msg = new GapAck("GapAck");
94      app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
95      msg->setPlatoonSpeed(platoonSpeed);
96      msg->setPlatoonLane(platoonLane);
97
        ↪ msg->setNewPlatoonFormationArraySize(newPlatoonFormation.size());
98      for (unsigned int i = 0; i < newPlatoonFormation.size(); i++) {
99          msg->setNewPlatoonFormation(i, newPlatoonFormation[i]);
100     }
101     return msg;
102 }
103
104 Lane* LeaveManeuver::createLaneChange(int vehicleId, std::string
    ↪ externalId, int platoonId, int destinationId, int newLane)
105 {
106     Lane* msg = new Lane("Lane");
107     app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
108     msg->setNewLane(newLane);
109     return msg;
110 }
111
112 LaneAck* LeaveManeuver::createLaneChangeAck(int vehicleId,
    ↪ std::string externalId, int platoonId, int destinationId)
113 {
114     LaneAck* msg = new LaneAck("LaneAck");
115     app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
```

```
116     return msg;
117 }
118
119 MoveToPosition* LeaveManeuver::createMoveToPosition(int vehicleId,
    ↪ std::string externalId, int platoonId, int destinationId, double
    ↪ platoonSpeed, int platoonLane, const std::vector<int>&
    ↪ newPlatoonFormation)
120 {
121     MoveToPosition* msg = new MoveToPosition("MoveToPosition");
122     app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
123     msg->setPlatoonSpeed(platoonSpeed);
124     msg->setPlatoonLane(platoonLane);
125
        ↪ msg->setNewPlatoonFormationArraySize(newPlatoonFormation.size());
126     for (unsigned int i = 0; i < newPlatoonFormation.size(); i++) {
127         msg->setNewPlatoonFormation(i, newPlatoonFormation[i]);
128     }
129     return msg;
130 }
131
132 MoveToPositionAck* LeaveManeuver::createMoveToPositionAck(int
    ↪ vehicleId, std::string externalId, int platoonId, int
    ↪ destinationId, double platoonSpeed, int platoonLane, const
    ↪ std::vector<int>& newPlatoonFormation)
133 {
134     MoveToPositionAck* msg = new
        ↪ MoveToPositionAck("MoveToPositionAck");
135     app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
136     msg->setPlatoonSpeed(platoonSpeed);
137     msg->setPlatoonLane(platoonLane);
138
        ↪ msg->setNewPlatoonFormationArraySize(newPlatoonFormation.size());
139     for (unsigned int i = 0; i < newPlatoonFormation.size(); i++) {
140         msg->setNewPlatoonFormation(i, newPlatoonFormation[i]);
141     }
142     return msg;
143 }
144
```

```
145 JoinFormation* LeaveManeuver::createJoinFormation(int vehicleId,
    ↪ std::string externalId, int platoonId, int destinationId, double
    ↪ platoonSpeed, int platoonLane, const std::vector<int>&
    ↪ newPlatoonFormation)
146 {
147     JoinFormation* msg = new JoinFormation("JoinFormation");
148     app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
149     msg->setPlatoonSpeed(platoonSpeed);
150     msg->setPlatoonLane(platoonLane);
151
        ↪ msg->setNewPlatoonFormationArraySize(newPlatoonFormation.size());
152     for (unsigned int i = 0; i < newPlatoonFormation.size(); i++) {
153         msg->setNewPlatoonFormation(i, newPlatoonFormation[i]);
154     }
155     return msg;
156 }
157
158 JoinFormationAck* LeaveManeuver::createJoinFormationAck(int
    ↪ vehicleId, std::string externalId, int platoonId, int
    ↪ destinationId, double platoonSpeed, int platoonLane, const
    ↪ std::vector<int>& newPlatoonFormation)
159 {
160     JoinFormationAck* msg = new
        ↪ JoinFormationAck("JoinFormationAck");
161     app->fillManeuverMessage(msg, vehicleId, externalId, platoonId,
        ↪ destinationId);
162     msg->setPlatoonSpeed(platoonSpeed);
163     msg->setPlatoonLane(platoonLane);
164
        ↪ msg->setNewPlatoonFormationArraySize(newPlatoonFormation.size());
165     for (unsigned int i = 0; i < newPlatoonFormation.size(); i++) {
166         msg->setNewPlatoonFormation(i, newPlatoonFormation[i]);
167     }
168     return msg;
169 }
```

B.2 LeaveManeuver.h

```
#ifndef LEAVEMANEUVER_H_
#define LEAVEMANEUVER_H_

#include "veins/modules/application/platooning/maneuver/Maneuver.h"
#include "veins/modules/application/platooning/messages/MoveToPositionAck_m.h"
#include "veins/modules/application/platooning/messages/MoveToPosition_m.h"
#include "veins/modules/application/platooning/messages/ManeuverMessage_m.h"
#include "veins/modules/application/platooning/messages/JoinFormationAck_m.h"
#include "veins/modules/application/platooning/messages/JoinFormation_m.h"
#include "veins/modules/application/platooning/messages/Gap_m.h"
#include "veins/modules/application/platooning/messages/GapAck_m.h"
#include "veins/modules/application/platooning/messages/Lane_m.h"
#include "veins/modules/application/platooning/messages/LaneAck_m.h"
#include "veins/modules/application/platooning/messages/Close_m.h"
#include "veins/modules/application/platooning/messages/CloseAck_m.h"
#include "veins/modules/application/platooning/messages/Leave_m.h"
#include "veins/modules/application/platooning/messages/LeaveAck_m.h"

struct LeaveManeuverParameters {
    int platoonId;
    int leaderId;
};

class LeaveManeuver : public Maneuver {

public:
    /**
     * Constructor
     *
     * @param app pointer to the generic application used to fetch parameters and inform it about a concluded maneuver
```

```
31          */
32         LeaveManeuver(GeneralPlatooningApp* app);
33         virtual ~LeaveManeuver(){};
34
35         virtual void onManeuverMessage(const ManeuverMessage* mm)
            ↪  override;
36
37     protected:
38         /**
39          * Creates a Leave Request message
40          *
41          * @param int vehicleId the id of the sending vehicle
42          * @param int platoonId the id of the platoon of the sending
       ↪  vehicle
43          * @param int destinationId the id of the destination
44          * @param double platoonSpeed the speed of the platoon
45          * @param int platoonLane the id of the lane of the platoon
46          * @param std::vector<int> newPlatoonFormation the original
       ↪  platoon
47          */
48         Leave* createLeave(int vehicleId, std::string externalId, int
            ↪  platoonId, int destinationId, double platoonSpeed, int
            ↪  platoonLane, const std::vector<int>& newPlatoonFormation);
49
50         /**
51          * Creates a Leave response message
52          *
53          * @param int vehicleId the id of the sending vehicle
54          * @param int platoonId the id of the platoon of the sending
       ↪  vehicle
55          * @param int destinationId the id of the destination
56          */
57         LeaveAck* createLeaveResponse(int vehicleId, std::string
            ↪  externalId, int platoonId, int destinationId);
58
59         /**
60          * Creates a Close gap message
61          *
62          * @param int vehicleId the id of the sending vehicle
```

```
63          * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
64          * @param int destinationId the id of the destination
65          */
66         Close* createCloseGap(int vehicleId, std::string externalId, int
             ↪  platoonId, int destinationId);
67
68         /**
69          * Creates a Close gap ack message
70          *
71          * @param int vehicleId the id of the sending vehicle
72          * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
73          * @param int destinationId the id of the destination
74          * @param int currentLaneIndex the index of the current lane of
    ↪  the vehicle
75          */
76         CloseAck* createCloseGapAck(int vehicleId, std::string
             ↪  externalId, int platoonId, int destinationId);
77
78         /**
79          * Creates a Gap message
80          *
81          * @param int vehicleId the id of the sending vehicle
82          * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
83          * @param int destinationId the id of the destination
84          * @param double distance the distance of required spacing to
    ↪  front car
85          */
86         Gap* createGap(int vehicleId, std::string externalId, int
             ↪  platoonId, int destinationId, double distance);
87
88         /**
89          * Creates a Gap ack message
90          *
91          * @param int vehicleId the id of the sending vehicle
92          * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
93          * @param int destinationId the id of the destination
```

```
94          * @param double platoonSpeed the speed of the platoon
95          * @param int platoonLane the id of the lane of the platoon
96          * @param std::vector<int> newPlatoonFormation the platoon
    ↪  formation after
97          * the join maneuver
98          */
99         GapAck* createGapAck(int vehicleId, std::string externalId, int
            ↪  platoonId, int destinationId,  double platoonSpeed, int
            ↪  platoonLane, const std::vector<int>& newPlatoonFormation);
100
101        /**
102         * Creates a Lane change message
103         *
104         * @param int vehicleId the id of the sending vehicle
105         * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
106         * @param int destinationId the id of the destination
107         * @param int newLane the index of the required lane for the
    ↪  vehicle
108         */
109        Lane* createLaneChange(int vehicleId, std::string externalId,
            ↪  int platoonId, int destinationId, int newLane);
110
111        /**
112         * Creates a Lane ack message
113         *
114         * @param int vehicleId the id of the sending vehicle
115         * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
116         * @param int destinationId the id of the destination
117         */
118        LaneAck* createLaneChangeAck(int vehicleId, std::string
            ↪  externalId, int platoonId, int destinationId);
119
120        /**
121         * Creates a MoveToPosition message
122         *
123         * @param int vehicleId the id of the sending vehicle
124         * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
```

```
125         * @param int destinationId the id of the destination
126         * @param int platoonSpeed the speed of the platoon
127         * @param int platoonLane the id of the lane of the platoon
128         * @param std::vector<int> newPlatoonFormation the platoon
    ↪  formation after
129         * the join maneuver
130         */
131        MoveToPosition* createMoveToPosition(int vehicleId, std::string
            ↪  externalId, int platoonId, int destinationId, double
            ↪  platoonSpeed, int platoonLane, const std::vector<int>&
            ↪  newPlatoonFormation);
132
133        /**
134         * Creates a MoveToPositionAck message
135         *
136         * @param int vehicleId the id of the sending vehicle
137         * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
138         * @param int destinationId the id of the destination
139         * @param int platoonSpeed the speed of the platoon
140         * @param int platoonLane the id of the lane of the platoon
141         * @param std::vector<int> newPlatoonFormation the platoon
    ↪  formation after
142         * the join maneuver
143         */
144        MoveToPositionAck* createMoveToPositionAck(int vehicleId,
            ↪  std::string externalId, int platoonId, int destinationId,
            ↪  double platoonSpeed, int platoonLane, const
            ↪  std::vector<int>& newPlatoonFormation);
145
146        /**
147         * Creates a JoinFormation message
148         *
149         * @param int vehicleId the id of the sending vehicle
150         * @param int platoonId the id of the platoon of the sending
    ↪  vehicle
151         * @param int destinationId the id of the destination
152         * @param int platoonSpeed the speed of the platoon
153         * @param int platoonLane the id of the lane of the platoon
```

```
154         * @param std::vector<int> newPlatoonFormation the platoon
    ↪   formation after
155         * the join maneuver
156         */
157        JoinFormation* createJoinFormation(int vehicleId, std::string
            ↪  externalId, int platoonId, int destinationId, double
            ↪  platoonSpeed, int platoonLane, const std::vector<int>&
            ↪  newPlatoonFormation);
158
159        /**
160         * Creates a JoinFormationAck message
161         *
162         * @param int vehicleId the id of the sending vehicle
163         * @param int platoonId the id of the platoon of the sending
    ↪   vehicle
164         * @param int destinationId the id of the destination
165         * @param int platoonSpeed the speed of the platoon
166         * @param int platoonLane the id of the lane of the platoon
167         * @param std::vector<int> newPlatoonFormation the platoon
    ↪   formation after
168         * the join maneuver
169         */
170        JoinFormationAck* createJoinFormationAck(int vehicleId,
            ↪  std::string externalId, int platoonId, int destinationId,
            ↪  double platoonSpeed, int platoonLane, const
            ↪  std::vector<int>& newPlatoonFormation);
171
172         /**
173         * Handles a Leave msg in the context of this application
174         *
175         * @param Leave msg to handle
176         */
177        virtual void handleLeave(const Leave* msg) = 0;
178
179         /**
180         * Handles a LeaveAck msg in the context of this application
181         *
182         * @param LeaveAck msg to handle
183         */
184        virtual void handleLeaveAck(const LeaveAck* msg) = 0;
```

```
185
186         /**
187          * Handles a Close gap msg in the context of this application
188          *
189          * @param Close msg to handle
190          */
191         virtual void handleClose(const Close* msg) = 0;
192
193         /**
194          * Handles a Close ack gap msg in the context of this
    ↪  application
195          *
196          * @param CloseAck msg to handle
197          */
198         virtual void handleCloseAck(const CloseAck* msg) = 0;
199
200         /**
201          * Handles a Gap msg in the context of this application
202          *
203          * @param Gap msg to handle
204          */
205         virtual void handleGap(const Gap* msg) = 0;
206
207         /**
208          * Handles a Gap ack msg in the context of this application
209          *
210          * @param GapAck msg to handle
211          */
212         virtual void handleGapAck(const GapAck* msg) = 0;
213
214         /**
215          * Handles a Lane msg in the context of this application
216          *
217          * @param Lane msg to handle
218          */
219         virtual void handleLane(const Lane* msg) = 0;
220
221         /**
222          * Handles a Lane ack msg in the context of this application
223          *
```

```
224         * @param LaneAck msg to handle
225         */
226        virtual void handleLaneAck(const LaneAck* msg) = 0;
227
228        /**
229         * Handles a MoveToPosition in the context of this application
230         *
231         * @param MoveToPosition msg to handle
232         */
233        virtual void handleMoveToPosition(const MoveToPosition* msg) =
           ↪  0;
234
235        /**
236         * Handles a MoveToPositionAck in the context of this
       ↪  application
237         *
238         * @param MoveToPositionAck msg to handle
239         */
240        virtual void handleMoveToPositionAck(const MoveToPositionAck*
           ↪  msg) = 0;
241
242        /**
243         * Handles a JoinFormation in the context of this application
244         *
245         * @param JoinFormation msg to handle
246         */
247        virtual void handleJoinFormation(const JoinFormation* msg) = 0;
248
249        /**
250         * Handles a JoinFormationAck in the context of this application
251         *
252         * @param JoinFormationAck msg to handle
253         */
254        virtual void handleJoinFormationAck(const JoinFormationAck* msg)
           ↪  = 0;
255    };
256
257    #endif
```

B.3 Leader_Leave.cc

```
#include "veins/modules/application/platooning/maneuver/Leader_Leave.h"
#include "veins/modules/application/platooning/apps/GeneralPlatooningApp.h"

Leader_Leave::Leader_Leave(GeneralPlatooningApp* app)
    : LeaveManeuver(app)
    , leaveManeuverState(LeaveManeuverState::IDLE)
{
}

void Leader_Leave::startManeuver(const void* parameters)
{
    LeaveManeuverParameters* pars = (LeaveManeuverParameters*)
     ↪ parameters;
    if(leaveManeuverState == LeaveManeuverState::IDLE) {
        ASSERT(app->getPlatoonRole() == PlatoonRole::LEADER);
        ASSERT(!app->isInManeuver());

        //Leader sends leader message to all
        for (unsigned int i = 1; i <
         ↪ positionHelper->getPlatoonSize(); i++) {
            Leave* l = createLeave(positionHelper->getId(),
             ↪ positionHelper->getExternalId(),
             ↪ positionHelper->getPlatoonId(),
             ↪ positionHelper->getMemberId(i),
             ↪ positionHelper->getPlatoonSpeed(),
             ↪ positionHelper->getPlatoonLane(),
             ↪ positionHelper->getPlatoonFormation());
            app->sendUnicast(l, positionHelper->getMemberId(i));
        }

        leaveManeuverState = LeaveManeuverState::L_WAIT_REPLY;
    }
}

void Leader_Leave::abortManeuver()
{
```

```
29 }
30
31 void Leader_Leave::onPlatoonBeacon(const PlatooningBeacon* pb)
32 {
33     if (leaveManeuverState ==
       ↪ LeaveManeuverState::A_MOVE_IN_POSITION) {
34         //check correct role
35         ASSERT(app->getPlatoonRole() == PlatoonRole::AVAILABLE);
36
37         //if the message comes from the leader
38         if (pb->getVehicleId() == positionHelper->getLeaderId()) {
39
               ↪ traciVehicle->setLeaderVehicleFakeData(pb->getControllerAcceleration
               ↪ pb->getAcceleration(), pb->getSpeed());
40         }
41         //if message comes from front vehicle, in this case is the
           ↪ leader
42         int frontId = positionHelper->getLeaderId();
43         if (pb->getVehicleId() == frontId) {
44             //get front vehicle position
45             Coord frontPosition(pb->getPositionX(),
               ↪ pb->getPositionY(), 0);
46             //get my position
47             Veins::TraCICoord traciPosition =
               ↪ mobility->getManager()->omnet2traci(mobility->getCurrentPosition());
48             Coord position(traciPosition.x, traciPosition.y);
49             //compute distance
50             double distance = position.distance(frontPosition) -
               ↪ pb->getLength();
51
               ↪ traciVehicle->setFrontVehicleFakeData(pb->getControllerAcceleration(
               ↪ pb->getAcceleration(), pb->getSpeed(), distance);
52             MoveToPositionAck* ack =
               ↪ createMoveToPositionAck(positionHelper->getId(),
               ↪ positionHelper->getExternalId(),
               ↪ positionHelper->getPlatoonId(),
               ↪ positionHelper->getLeaderId(),
               ↪ positionHelper->getPlatoonSpeed(),
               ↪ positionHelper->getPlatoonLane(),
               ↪ positionHelper->getPlatoonFormation());
```

```
53              app->sendUnicast(ack, positionHelper->getLeaderId());
54              leaveManeuverState = LeaveManeuverState::A_WAIT_GAP;
55          }
56      }
57  }
58
59  void Leader_Leave::handleLeave(const Leave* msg)
60  {
61      if (msg->getDestinationId() != positionHelper->getId()) return;
62      if (app->getPlatoonRole() != PlatoonRole::AVAILABLE) return;
63
64      /*save data from original platoon*/
65      originalPlatoonData->from(msg);
66
67      LeaveAck* lack = createLeaveResponse(positionHelper->getId(),
        ↪  positionHelper->getExternalId(),
        ↪  positionHelper->getPlatoonId(),
        ↪  positionHelper->getLeaderId());
68      app->sendUnicast(lack, positionHelper->getLeaderId());
69
70      leaveManeuverState = LeaveManeuverState::A_WAIT_GAP;
71  }
72
73  void Leader_Leave::handleLeaveAck(const LeaveAck* msg)
74  {
75      if (msg->getDestinationId() != positionHelper->getId()) return;
76      if (app->getPlatoonRole() != PlatoonRole::LEADER) return;
77
78      // disable lane changing during maneuver
79      traciVehicle->setFixedLane(traciVehicle->getLaneIndex());
80      positionHelper->setPlatoonLane(traciVehicle->getLaneIndex());
81
82      // save some data. who is rejoining?
83      reJoinerData.reset(new ReJoinerData());
84      reJoinerData->from(msg);
85
```

```
86      Gap* cgap = createGap(positionHelper->getId(),
        ↪  positionHelper->getExternalId(),
        ↪  positionHelper->getPlatoonId(),
        ↪  positionHelper->getMemberId(reJoinerData->reJoinerId + 1),
        ↪  12);
87      app->sendUnicast(cgap,
        ↪  positionHelper->getMemberId(reJoinerData->reJoinerId + 1));
88
89      leaveManeuverState = LeaveManeuverState::L_WAIT_GAP;
90  }
91
92  void Leader_Leave::handleGap(const Gap* msg)
93  {
94      if (msg->getDestinationId() != positionHelper->getId()) return;
95      if (app->getPlatoonRole() != PlatoonRole::FOLLOWER) return;
96      if (leaveManeuverState != LeaveManeuverState::IDLE) return;
97
98      //create a temp platoon to avoid crashes when deaccelerating
99      std::vector<int> temp_formation;
100     for (unsigned int i = positionHelper->getId(); i <
        ↪  positionHelper->getPlatoonSize(); i++) {
101         temp_formation.push_back(positionHelper->getMemberId(i));
102     }
103     positionHelper->setPlatoonFormation(temp_formation);
104
105     // activate faked CACC. this way we can approach the front car
106     // using data obtained through GPS
107     traciVehicle->setCACCConstantSpacing(15);
108     // we have no data so far, so for the moment just initialize
109     // with some fake data
110     traciVehicle->setLeaderVehicleFakeData(0, 0,
        ↪  originalPlatoonData->platoonSpeed);
111     traciVehicle->setFrontVehicleFakeData(0, 0,
        ↪  originalPlatoonData->platoonSpeed, 15);
112     traciVehicle->setActiveController(Plexe::FAKED_CACC);
113
```

```
114     GapAck* cgapack = createGapAck(positionHelper->getId(),
        ↪ positionHelper->getExternalId(),
        ↪ positionHelper->getPlatoonId(), msg->getVehicleId(),
        ↪ positionHelper->getPlatoonSpeed(),
        ↪ positionHelper->getPlatoonLane(), temp_formation);
115     app->sendUnicast(cgapack, msg->getVehicleId());
116
117     leaveManeuverState = LeaveManeuverState::F_WAIT_CLOSE;
118 }
119
120 void Leader_Leave::handleGapAck(const GapAck* msg)
121 {
122     if (msg->getDestinationId() != positionHelper->getId()) return;
123     if (leaveManeuverState != LeaveManeuverState::L_WAIT_GAP)
        ↪ return;
124     if (app->getPlatoonRole() != PlatoonRole::LEADER) return;
125
126     /** talvez seja necessário criar novo pelotão**/
127
128     Lane* lane = createLaneChange(positionHelper->getId(),
        ↪ positionHelper->getExternalId(),
        ↪ positionHelper->getPlatoonId(), reJoinerData->reJoinerId,
        ↪ positionHelper->getPlatoonLane() + 1);
129     app->sendUnicast(lane, reJoinerData->reJoinerId);
130
131     leaveManeuverState = LeaveManeuverState::L_WAIT_LANE_CHANGE;
132
133 }
134
135 void Leader_Leave::handleLane(const Lane* msg)
136 {
137     if (msg->getDestinationId() != positionHelper->getId()) return;
138     if (leaveManeuverState != LeaveManeuverState::A_WAIT_GAP)
        ↪ return;
139     if (app->getPlatoonRole() != PlatoonRole::AVAILABLE) return;
140
141     /* Change lane to the required one*/
142     traciVehicle->setFixedLane(msg->getNewLane(), false);
143     reJoinerData->reJoinerLane = msg->getNewLane();
144
```

```
145     LaneAck* laneack = createLaneChangeAck(positionHelper->getId(),
        ↪  positionHelper->getExternalId(),
        ↪  positionHelper->getPlatoonId(),
        ↪  positionHelper->getLeaderId());
146     app->sendUnicast(laneack, positionHelper->getLeaderId());
147
148     leaveManeuverState = LeaveManeuverState::A_WAIT_INFORMATION;
149 }
150
151 void Leader_Leave::handleLaneAck(const LaneAck* msg)
152 {
153     if (msg->getDestinationId() != positionHelper->getId()) return;
154     if (leaveManeuverState !=
        ↪  LeaveManeuverState::L_WAIT_LANE_CHANGE) return;
155     if (app->getPlatoonRole() != PlatoonRole::LEADER) return;
156
157     Close* cls = createCloseGap(positionHelper->getId(),
        ↪  positionHelper->getExternalId(),
        ↪  positionHelper->getPlatoonId(), reJoinerData->reJoinerId +
        ↪  1);
158     app->sendUnicast(cls, reJoinerData->reJoinerId + 1);
159
160     leaveManeuverState = LeaveManeuverState::L_WAIT_GAP_CLOSE;
161 }
162
163 void Leader_Leave::handleClose(const Close* msg)
164 {
165     if (msg->getDestinationId() != positionHelper->getId()) return;
166     if (leaveManeuverState != LeaveManeuverState::F_WAIT_CLOSE)
        ↪  return;
167     if (app->getPlatoonRole() != PlatoonRole::FOLLOWER) return;
168
169     /*close gap*/
170     positionHelper->setFrontId(reJoinerData->reJoinerFrontId);
171     traciVehicle->setFrontVehicleFakeData(0, 0,
        ↪  originalPlatoonData->platoonSpeed, 15);
172
173     traciVehicle->setActiveController(Plexe::CACC);
174     traciVehicle->setCACCConstantSpacing(5);
175
```

```
176     CloseAck* clsAck = createCloseGapAck(positionHelper->getId(),
        ↪ positionHelper->getExternalId(),
        ↪ positionHelper->getPlatoonId(),
        ↪ positionHelper->getLeaderId());
177     app->sendUnicast(clsAck, positionHelper->getLeaderId()); /**
        ↪ mudar este leader id se necessário novo pelotão**/
178
179     leaveManeuverState = LeaveManeuverState::IDLE;
180 }
181
182 void Leader_Leave::handleCloseAck(const CloseAck* msg)
183 {
184     if (msg->getDestinationId() != positionHelper->getId()) return;
185     if (leaveManeuverState != LeaveManeuverState::L_WAIT_GAP_CLOSE)
        ↪ return;
186     if (app->getPlatoonRole() != PlatoonRole::LEADER) return;
187
188     /*insert platoon in correct position*/
189     reJoinerData->newFormation =
        ↪ positionHelper->getPlatoonFormation();
190     reJoinerData->newFormation.erase(reJoinerData->reJoinerId);
191     reJoinerData->newFormation.emplace(1, reJoinerData->reJoinerId);
192
193     MovetoPosition* mtp =
        ↪ createMoveToPosition(positionHelper->getId(),
        ↪ positionHelper->getExternalId(),
        ↪ positionHelper->getPlatoonId(), reJoinerData->reJoinerId,
        ↪ positionHelper->getPlatoonSpeed(),
        ↪ originalPlatoonData->newFormation);
194     app->sendUnicast(mtp, reJoinerData->reJoinerId);
195
196     leaveManeuverState = LeaveManeuverState::L_WAIT_A_IN_POSITION;
197 }
198
199 void Leader_Leave::handleMoveToPosition(const MoveToPosition* msg)
200 {
201     if (msg->getDestinationId() != positionHelper->getId()) return;
202     if (leaveManeuverState !=
        ↪ LeaveManeuverState::A_WAIT_INFORMATION) return;
203     if (app->getPlatoonRole() != PlatoonRole::AVAILABLE) return;
```

```
204
205     /*set new frontId */
206     positionHelper->setFrontId(originalPlatoonData->frontId());
207
208     // activate faked CACC. this way we can approach the front car
209     // using data obtained through GPS
210     traciVehicle->setCACCConstantSpacing(10);
211     // we have no data so far, so for the moment just initialize
212     // with some fake data
213     traciVehicle->setLeaderVehicleFakeData(0, 0,
        ↪  targetPlatoonData->platoonSpeed);
214     traciVehicle->setFrontVehicleFakeData(0, 0,
        ↪  targetPlatoonData->platoonSpeed, 10);
215     // set a CC speed higher than the platoon speed to approach it
216
        ↪  traciVehicle->setCruiseControlDesiredSpeed(targetPlatoonData->platoonSpeed
        ↪  + (30 / 3.6));
217     traciVehicle->setActiveController(Plexe::FAKED_CACC);
218
219     leaveManeuverState = LeaveManeuverState::A_MOVE_IN_POSITION;
220 }
221
222 void Leader_Leave::handleMoveToPositionAck(const MoveToPositionAck*
    ↪  msg)
223 {
224     if (msg->getDestinationId() != positionHelper->getId()) return;
225     if (leaveManeuverState !=
        ↪  LeaveManeuverState::L_WAIT_A_IN_POSITION) return;
226     if (app->getPlatoonRole() != PlatoonRole::LEADER) return;
227
228     leaveManeuverState = LeaveManeuverState::L_WAIT_GAP;
229 }
230
231 void Leader_Leave::handleJoinFormation(const JoinFormation* msg)
232 {
233     app->setPlatoonRole(PlatoonRole::LEADER);
234     leaveManeuverState = LeaveManeuverState::IDLE;
235 }
236
```

```
237 void Leader_Leave::handleJoinFormationAck(const JoinFormationAck*
    ↪ msg)
238 {
239     app->setPlatoonRole(PlatoonRole::NONE);
240     leaveManeuverState = LeaveManeuverState::IDLE;
241 }
```

B.4 Leader_Leave.h

```
1  #ifndef LEADER_LEAVE_H
2  #define LEADER_LEAVE_H
3
4  #include <algorithm>
5
6  #include
    ↪ "veins/modules/application/platooning/maneuver/LeaveManeuver.h"
7  #include
    ↪ "veins/modules/application/platooning/utilities/BasePositionHelper.h"
8
9  #include "veins/modules/mobility/traci/TraCIConstants.h"
10 #include "veins/base/utils/Coord.h"
11
12 using namespace Veins;
13
14 class Leader_Leave : public LeaveManeuver {
15
16 public:
17     /**
18      * Constructor
19      *
20      * @param app pointer to the generic application used to fetch
    ↪ parameters and inform it about a concluded maneuver
21      */
22     Leader_Leave(GeneralPlatooningApp* app);
23     virtual ~Leader_Leave(){};
24
25     /**
26      * This method is invoked by the generic application to start
    ↪ the maneuver
27      *
```

```
28      * @param parameters parameters passed to the maneuver
29      */
30     virtual void startManeuver(const void* parameters) override;
31
32     /**
33      * Handles the abortion of the maneuver when required by the
    ↪  generic application.
34      * This method does currently nothing and it is meant for future
    ↪  used and improved maneuvers.
35      */
36     virtual void abortManeuver() override;
37
38     /**
39      * This method is invoked by the generic application when a
    ↪  beacon message is received
40      * The maneuver must not free the memory of the message, as this
    ↪  might be needed by other maneuvers as well.
41      */
42     virtual void onPlatoonBeacon(const PlatooningBeacon* pb)
     ↪  override;
43
44     /**
45      * Handles a MoveToPosition in the context of this application
46      *
47      * @param MoveToPosition msg to handle
48      */
49     virtual void handleMoveToPosition(const MoveToPosition* msg)
     ↪  override;
50
51     /**
52      * Handles a MoveToPositionAck in the context of this
    ↪  application
53      *
54      * @param MoveToPositionAck msg to handle
55      */
56     virtual void handleMoveToPositionAck(const MoveToPositionAck*
     ↪  msg) override;
57
58     /**
59      * Handles a JoinFormation in the context of this application
```

```
60          *
61          * @param JoinFormation msg to handle
62          */
63         virtual void handleJoinFormation(const JoinFormation* msg)
            ↪  override;
64
65         /**
66          * Handles a JoinFormationAck in the context of this application
67          *
68          * @param JoinFormationAck msg to handle
69          */
70         virtual void handleJoinFormationAck(const JoinFormationAck* msg)
            ↪  override;
71
72         /**
73          * Handles a Gap in the context of this application
74          *
75          * @param Gap msg to handle
76          */
77         virtual void handleGap(const Gap* msg) override;
78
79         /**
80          * Handles a GapAck in the context of this application
81          *
82          * @param GapAck msg to handle
83          */
84         virtual void handleGapAck(const GapAck* msg) override;
85
86         /**
87          * Handles a Lane in the context of this application
88          *
89          * @param Lane msg to handle
90          */
91         virtual void handleLane(const Lane* msg) override;
92
93         /**
94          * Handles a Lane in the context of this application
95          *
96          * @param LaneAck msg to handle
97          */
```

```
98      virtual void handleLaneAck(const LaneAck* msg) override;
99
100     /**
101      * Handles a Leave in the context of this application
102      *
103      * @param Leave msg to handle
104      */
105     virtual void handleLeave(const Leave* msg) override;
106
107     /**
108      * Handles a LeaveAck in the context of this application
109      *
110      * @param LeaveAck msg to handle
111      */
112     virtual void handleLeaveAck(const LeaveAck* msg) override;
113
114     /**
115      * Handles a Close in the context of this application
116      *
117      * @param Close msg to handle
118      */
119     virtual void handleClose(const Close* msg) override;
120
121     /**
122      * Handles a CloseAck in the context of this application
123      *
124      * @param CloseAck msg to handle
125      */
126     virtual void handleCloseAck(const CloseAck* msg) override;
127
128 protected:
129     /** Possible states a vehicle can be in during Leader leaving
        ↪ maneuver */
130     enum class LeaveManeuverState{
131         IDLE, ///< The maneuver did not start
132         //Leader
133         L_WAIT_REPLY, ///< The leader sent the leave msg and waits
            ↪ for reply
134         L_WAIT_GAP, ///< The leader sent a gap msg and waits for the
            ↪ car to be in position
```

```
135         L_WAIT_LANE_CHANGE, ///< The leader waits for A to change
             ↪ lane
136         L_WAIT_GAP_CLOSE, ///< The leader waits for gap close
137         L_WAIT_A_IN_POSITION, ///< The leader waits for A in postion
138         L_WAIT_A_TO_REJOIN, ///< The leader waits for A to rejoin
             ↪ the platoon
139         //FOLLOWER
140         F_WAIT_CLOSE, ///< The follower send GapAck message and
             ↪ waits for Close message
141         //AVAILABLE
142         A_WAIT_GAP, ///< The available vehicle sends leader leave
             ↪ response and waits for the gap
143         A_WAIT_INFORMATION, ///< The available vehicle waits for
             ↪ information about the Platoon
144         A_MOVE_IN_POSITION, ///< The available vehicle moves to
             ↪ position
145         A_WAIT_REJOIN, ///< The available vehicle waits for rejoin
             ↪ permission
146     };
147
148     /** data that a joiner stores about a Platoon it wants to join
         ↪ */
149     struct OriginalPlatoonData {
150         int platoonId; ///< the id of the platoon to join
151         int platoonLeader; ///< the if ot the leader of the platoon
152         int platoonLane; ///< the lane the platoon is driving on
153         double platoonSpeed; ///< the speed of the platoon
154         std::vector<int> newFormation; ///< the new formation of the
             ↪ platoon
155         Coord lastFrontPos; ///< the last kwown position of the
             ↪ front vehicle
156
157         /** c'tor for TargetPlatoonData */
158         OriginalPlatoonData()
159         {
160             platoonId = INVALID_PLATOON_ID;
161             platoonLeader = INVALID_INT_VALUE;
162             platoonLane = INVALID_INT_VALUE;
163             platoonSpeed = INVALID_DOUBLE_VALUE;
164         }
```

```
165
166         /**
167          * Initializes the TargetPlatoonData object with the
    ↪  contents of a
168          * Leave
169          *
170          * @param Leave msg to initialize from
171          * @see Leave
172          */
173         void from(const Leave* msg)
174         {
175             platoonId = msg->getPlatoonId();
176             platoonLeader = msg->getVehicleId();
177             platoonLane = msg->getPlatoonLane();
178             platoonSpeed = msg->getPlatoonSpeed();
179
                ↪  newFormation.resize(msg->getNewPlatoonFormationArraySize());
180             for (unsigned int i = 0; i <
                ↪  msg->getNewPlatoonFormationArraySize(); i++) {
181                 newFormation[i] = msg->getNewPlatoonFormation(i);
182             }
183         }
184
185         /**
186          * Get the id of the front vehicle
187          *
188          * @return int the id of the front vehicle
189          */
190         int frontId() const
191         {
192             return newFormation.at(0);
193         }
194     };
195
196     /** data that a leader stores about rejoining vehicle */
197     struct ReJoinerData {
198         int reJoinerId; ///< the id of the vehicle joining the
            ↪  Platoon
199         int reJoinerLane; ///< the lane chosen for joining the
            ↪  Platoon
```

```
200         int reJoinerFrontId; ///< the id of rejoiner front vehicle
201         std::vector<int> newFormation;
202 
203         /** c'tor for ReJoinerData */
204         ReJoinerData()
205         {
206             reJoinerId = INVALID_INT_VALUE;
207             reJoinerLane = INVALID_INT_VALUE;
208         }
209 
210         /**
211          * Initializes the JoinerData object with the contents of a
212          * LeaveAck message
213          *
214          * @param LeaveAck  to initialize from
215          * @see LeaveAck
216          */
217         void from(const LeaveAck* msg)
218         {
219             reJoinerId = msg->getVehicleId();
220             reJoinerLane = msg->getCurrentLaneIndex();
221             reJoinerFrontId = msg->getFrontId();
222         }
223     };
224 
225     /** data that the LEADER stores about a temporary platoon */
226     /*struct TemporaryPlatoonData {
227         int platoonId; ///< the id of the platoon to join
228         int platoonLeader; ///< the if ot the leader of the platoon
229         int platoonLane; ///< the lane the platoon is driving on
230         double platoonSpeed; ///< the speed of the platoon
231         std::vector<int> newFormation; ///< the new formation of the
    ↪  platoon
232 
233         /** c'tor for temporary platoon */
234         /*TemporaryPlatoonData()
235         {
236             platoonId = INVALID_PLATOON_ID;
237             platoonLeader = INVALID_INT_VALUE;
238             platoonLane = INVALID_INT_VALUE;
```

```
239             platoonSpeed = INVALID_DOUBLE_VALUE;
240         }
241         /**
242          * Initializes the temporary platoon object with the
    ↪ contents of a
243          *
244          *
245          * @param GapAck msg to initialize from
246          * @see GapAck
247          */
248         /*void from(const GapAck* msg)
249         {
250             platoonId = msg->getPlatoonId();
251             platoonLeader = msg->getVehicleId();
252             platoonLane = msg->getPlatoonLane();
253             platoonSpeed = msg->getPlatoonSpeed();
254 
    ↪ newFormation.resize(msg->getNewPlatoonFormationArraySize());
255             for (unsigned int i = 0; i <
    ↪ msg->getNewPlatoonFormationArraySize(); i++) {
256                 newFormation[i] = msg->getNewPlatoonFormation(i);
257             }
258         }
259 
260         /**
261          * Get the id of the front vehicle
262          *
263          * @return int the id of the front vehicle
264          *//*
265         int frontId() const
266         {
267             return newFormation.at(joinIndex - 1);
268         }*/
269     //};*/
270 
271 
272     /** the current state in the join maneuver */
273     LeaveManeuverState leaveManeuverState;
274 
275     /** the data about the reJoiner */
```

```
276     std::unique_ptr<ReJoinerData> reJoinerData;
277
278     /** the data of the platoon to rejoin */
279     std::unique_ptr<OriginalPlatoonData> originalPlatoonData;
280
281     /** the data of the temporary platoon*/
282     //std::unique_ptr<TemporaryPlatoonData> temporaryPlatoonData;
283
284 };
285
286 #endif
```

B.5 LeaveManeuverScenario.cc

```
1  #include "plexe/scenarios/LeaveManeuverScenario.h"
2
3  namespace plexe {
4
5  Define_Module(LeaveManeuverScenario);
6
7  void LeaveManeuverScenario::initialize(int stage)
8  {
9
10     BaseScenario::initialize(stage);
11
12     if(stage == 1) {
13         app =
            ↪  FindModule<GeneralPlatooningApp*>::findSubModule(getParentModule());
14         prepareManeuverCars(0);
15     }
16 }
17
18 void LeaveManeuverScenario::setupFormation()
19 {
20     std::vector<int> formation;
21     for (int i = 0; i < 5; i++) formation.push_back(i);
22     positionHelper->setPlatoonFormation(formation);
23 }
24
25 void LeaveManeuverScenario::prepareManeuverCars(int platoonLane)
```

```
{
    switch (positionHelper->getId()) {

    case 0: {
        //this is the leader
        plexeTraciVehicle->setCruiseControlDesiredSpeed(100.0 /
         ↪  3.6);
        plexeTraciVehicle->setActiveController(ACC);
        plexeTraciVehicle->setFixedLane(platoonLane);
        app->setPlatoonRole(PlatoonRole::LEADER);
        traciVehicle->setColor(TraCIColor::TraCIColor(0, 0, 255,
         ↪  255));

        // after 30 seconds of simulation, start the maneuver
        startManeuver = new cMessage();
        scheduleAt(simTime() + SimTime(10), startManeuver);
        break;
    }

    //change this setup dependent on which scenario you want to
     ↪  study
    //and in which position you want Available to be
    case 1:
    //case 2:
    case 3:
    case 4:{
        // these are the followers which are already in the platoon
        plexeTraciVehicle->setCruiseControlDesiredSpeed(130.0 /
         ↪  3.6);
        plexeTraciVehicle->setActiveController(CACC);
        plexeTraciVehicle->setFixedLane(platoonLane);
        app->setPlatoonRole(PlatoonRole::FOLLOWER);
        traciVehicle->setColor(TraCIColor::TraCIColor(255, 255, 255,
         ↪  255));
        break;
    }

    case 2: {
        //this is the available car for new leader
```

```
60          plexeTraciVehicle->setCruiseControlDesiredSpeed(130.0 /
             ↪  3.6);
61          plexeTraciVehicle->setActiveController(CACC);
62          plexeTraciVehicle->setFixedLane(platoonLane);
63          app->setPlatoonRole(PlatoonRole::AVAILABLE);
64          traciVehicle->setColor(TraCIColor::TraCIColor(255, 255, 255,
             ↪  255));
65          break;
66      }
67      }
68  }
69
70  LeaveManeuverScenario::~LeaveManeuverScenario()
71  {
72      cancelAndDelete(startManeuver);
73      startManeuver = nullptr;
74  }
75
76  void LeaveManeuverScenario::handleSelfMsg(cMessage* msg)
77  {
78      // this takes car of feeding data into CACC and reschedule the
         ↪  self message
79      BaseScenario::handleSelfMsg(msg);
80
81      if (msg == startManeuver) app->startLeaveManeuver(0, 0);
82  }
83
84  } // namespace plexe
```

B.6 LeaveManeuverScenario.h

```
1  //
2  // Copyright (C) 2012-2019 Michele Segata <segata@ccs-labs.org>
3  //
4  // SPDX-License-Identifier: GPL-2.0-or-later
5  //
6  // This program is free software; you can redistribute it and/or
    ↪  modify
7  // it under the terms of the GNU General Public License as published
    ↪  by
```

```
8  // the Free Software Foundation; either version 2 of the License, or
9  // (at your option) any later version.
10 //
11 // This program is distributed in the hope that it will be useful,
12 // but WITHOUT ANY WARRANTY; without even the implied warranty of
13 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14 // GNU General Public License for more details.
15 //
16 // You should have received a copy of the GNU General Public License
17 // along with this program; if not, write to the Free Software
18 // Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
    ↪  02111-1307  USA
19 //
20 // Modified by Renato Ferreira for a Leave Maneuver Scenario (2020)
21
22 #ifndef LEAVEMANEUVERSCENARIO_H_
23 #define LEAVEMANEUVERSCENARIO_H_
24
25 #include "plexe/scenarios/BaseScenario.h"
26 #include "plexe/apps/GeneralPlatooningApp.h"
27 #include "plexe/messages/ManeuverMessage_m.h"
28
29 namespace plexe {
30
31 class LeaveManeuverScenario : public BaseScenario {
32
33 protected:
34     // message used to start maneuver
35     cMessage* startManeuver;
36     // pointer to protocol
37     GeneralPlatooningApp* app;
38
39 public:
40     static const int MANEUVER_TYPE = 12347;
41
42     virtual void initialize(int stage) override;
43
44 protected:
45     void sendUnicast(cPacket* msg, int destination);
46
```

```
47 private:
48 public:
49     LeaveManeuverScenario()
50     {
51         startManeuver = nullptr;
52         app = nullptr;
53     }
54     virtual ~LeaveManeuverScenario();
55
56 protected:
57     virtual void handleSelfMsg(cMessage* msg) override;
58
59     void prepareManeuverCars(int platoonLane);
60     void setupFormation();
61 };
62
63 } //namespace plexe
64
65 #endif
```

B.7 omnetpp.ini

```
1  [General]
2  cmdenv-express-mode = true
3  cmdenv-autoflush = true
4  cmdenv-status-frequency = 10s
5  num-rngs = 6
6
7  ned-path = .
8  network = Highway
9
10 ##########################################################
11 #            Simulation parameters                       #
12 ##########################################################
13 debug-on-errors = true
14 print-undisposed = true
15
16 *.playgroundSizeX = 65000m
17 *.playgroundSizeY = 25000m
18 *.playgroundSizeZ = 50m
```

```
19
20 sim-time-limit = 90 s
21
22 ###################################################################
23 # Annotation parameters                                           #
24 ###################################################################
25 *.annotations.draw = false
26
27 ###################################################################
28 # Obstacle parameters                                             #
29 ###################################################################
30 *.obstacles.debug = false
31
32 ###################################################################
33 #            WorldUtility parameters                              #
34 ###################################################################
35 *.world.useTorus = false
36 *.world.use2D = false
37
38 ###################################################################
39 #            TraCIScenarioManager parameters                      #
40 ###################################################################
41 *.manager.updateInterval = 0.01s
42 *.manager.host = "localhost"
43 *.manager.moduleType = "org.car2x.plexe.PlatoonCar"
44 *.manager.moduleName = "node"
45 *.manager.moduleDisplayString = ""
46 *.manager.autoShutdown = true
47 *.manager.margin = 25
48
49 ###################################################################
50 #            11p specific parameters                              #
51 #                                                                 #
52 #                    NIC-Settings                                 #
53 ###################################################################
54 *.connectionManager.sendDirect = true
55 *.connectionManager.maxInterfDist = 2600m
56 *.connectionManager.drawMaxIntfDist = false
57
58 *.**.nic.mac1609_4.useServiceChannel = false
```

```
59
60 *.**.nic.mac1609_4.txPower = 100mW
61 *.**.nic.mac1609_4.bitrate = 6Mbps
62
63 *.**.nic.mac1609_4.useAcks = true
64 *.**.nic.mac1609_4.ackErrorRate = 0.0
65 *.**.nic.mac1609_4.frameErrorRate = 0.0
66
67 *.**.nic.phy80211p.minPowerLevel = -94dBm
68 *.**.nic.phy80211p.maxTXPower = 100mW
69 *.**.nic.phy80211p.useNoiseFloor = true
70 *.**.nic.phy80211p.noiseFloor = -95dBm
71 *.**.nic.phy80211p.decider = xmldoc("config.xml")
72 *.**.nic.phy80211p.analogueModels = xmldoc("config.xml")
73 *.**.nic.phy80211p.usePropagationDelay = true
74
75 ##########################################################
76 #                       Mobility                         #
77 ##########################################################
78 *.node[*].mobility.x = 0
79 *.node[*].mobility.y = 0
80 *.node[*].mobility.z = 1.895
81
82
83 ##########################################################
84 #                     Seeds and PRNGs                    #
85 ##########################################################
86 seed-set = ${repetition}
87 **.seed = ${repetition}
88
89 *.node[*].unicast.rng-0 = 1
90 *.node[*].prot.rng-0 = 2
91 *.node[*].appl.rng-0 = 3
92 *.node[*].scenario.rng-0 = 4
93 **.traffic.rng-0 = 5
94
95 #launch config. tells Veins which SUMO configuration to run
96 *.manager.configFile = "./sumocfg/freeway.sumo.cfg"
97
98 ##########################################################
```

```
99  #                      Common parameters                       #
100 ##############################################################
101
102 **.numberOfCars = ${nCars = 5}
103 **.numberOfCarsPerPlatoon = ${platoonSize = 5}
104 **.numberOfLanes = ${nLanes = 1}
105 **.controllerXi = ${caccXi = 1, 2}
106 **.controllerOmegaN = ${caccOmegaN = 0.2, 1 ! caccXi}
107
108 ##############################################################
109 #                       Position helper                        #
110 ##############################################################
111
112 *.node[*].helper_type = "PositionHelper"
113
114 ##############################################################
115 #                 Scenario common parameters                   #
116 ##############################################################
117
118 #controller and engine related parameters
119 *.node[*].scenario.caccC1 = 0.5
120 *.node[*].scenario.caccXi = ${caccXi}
121 *.node[*].scenario.caccOmegaN = ${caccOmegaN} Hz
122 *.node[*].scenario.caccSpacing = 5 m
123 *.node[*].scenario.engineTau = 0.5 s
124 *.node[*].scenario.ploegH = ${ploegH = 0.5}s
125 *.node[*].scenario.ploegKp = 0.2
126 *.node[*].scenario.ploegKd = 0.7
127 *.node[*].scenario.useRealisticEngine = false
128
129 #controllers to be tested
130 **.numericController = 0
131 *.node[*].scenario.controller = "CACC"
132
133 #headway for ACCs
134 *.node[*].scenario.accHeadway = 0.1 s
135 *.node[*].scenario.leaderHeadway = ${leaderHeadway = 1.2}s
136
137 #average leader speed
138 *.node[*].scenario.leaderSpeed = ${leaderSpeed = 100}kmph
```

```
139
140 ##########################################################
141 #                       Application                      #
142 ##########################################################
143
144 *.node[*].appl_type = "GeneralPlatooningApp"
145 #enable statistics recording for the application
146 *.node[*].appl.*.scalar-recording = true
147 *.node[*].appl.*.vector-recording = true
148 #use de Leader Leave implementation of LeaveManeuver
149 *.node[*].appl.leaveManeuver = "Leader_Leave"
150
151 ##########################################################
152 #                 Communication protocols                #
153 ##########################################################
154
155 *.node[*].protocol_type = "SimplePlatooningBeaconing"
156 #enable statistics recording for the network layer
157 *.node[*].prot.*.scalar-recording = true
158 *.node[*].prot.*.vector-recording = true
159 #set the beaconing interval to be 0.1s
160 *.node[*].prot.beaconingInterval = ${beaconInterval = 0.1}s
161 #access category for platooning beacons
162 *.node[*].prot.priority = ${priority = 2}
163 #packet size for platooning beacon
164 *.node[*].prot.packetSize = ${packetSize = 200}
165 #set to 0 the packet loss rate for the unicast protocol
166 *.node[*].unicast.packetLossRate = 0
167
168 ##########################################################
169 #                     Traffic manager                    #
170 ##########################################################
171
172 **.traffic_type = "PlatoonsTrafficManager"
173 #insert platooning vehicles at time
174 **.traffic.platoonInsertTime = 1 s
175 #insert platooning vehicles with a speed of
176 **.traffic.platoonInsertSpeed = ${leaderSpeed}kmph
177 #insert nCars platooning vehicles
178 **.traffic.nCars = ${nCars}
```

```
179 #let platoonSize cars per platoon
180 **.traffic.platoonSize = ${platoonSize}
181 #use nLanes lanes
182 **.traffic.nLanes = ${nLanes}
183 #SUMO vtype for platooning vehicles
184 **.traffic.platooningVType = "vtypeauto"
185 #insert vehicles already at steady-state. distance depends on
    ↪ controller
186 **.traffic.platoonInsertDistance = 5 m
187 **.traffic.platoonInsertHeadway = 0 s
188 **.traffic.platoonLeaderHeadway = ${leaderHeadway}s
189
190
191 #disable statistics recording for all other modules
192 **.scalar-recording = false
193 **.vector-recording = false
194
195 [Config LeaveManeuver]
196
197 repeat = 25
198 *.manager.command = "sumo-gui"
199 *.manager.ignoreGuiCommands = false
200
201 #use the join maneuver scenario
202 *.node[*].scenario_type = "LeaveManeuverScenario"
203
204 output-vector-file =
    ↪ ${resultdir}/${configname}_${caccXi}_${caccOmegaN}_${repetition}.vec
205 output-scalar-file =
    ↪ ${resultdir}/${configname}_${caccXi}_${caccOmegaN}_${repetition}.sca
206
207 [Config LeaveManeuverNoGui]
208 extends = LeaveManeuver
209 *.manager.command = "sumo"
210 *.manager.ignoreGuiCommands = true
211 #force the config name in the output file to be the same as for the
    ↪ gui experiment
212 output-vector-file =
    ↪ ${resultdir}/LeaveManeuver_${caccXi}_${caccOmegaN}_${repetition}.vec
```

```
213 output-scalar-file =
    ↪ ${resultdir}/LeaveManeuver_${caccXi}_${caccOmegaN}_${repetition}.sca
```