

Faculdade de Engenharia da Universidade do Porto



**Arquitectura Baseada em Serviços para Redes
Veículo-a-Veículo**

João Filipe Barreiras Gonçalves

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Rosaldo Rossetti

Janeiro de 2009

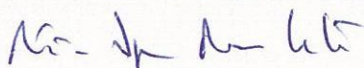
A Dissertação intitulada

“ARQUITECTURA BASEADA EM SERVIÇOS PARA REDES VEÍCULO A VEÍCULO”

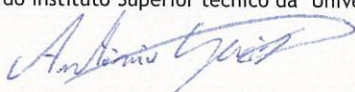
foi aprovada em provas realizadas em 02/Março/2009

o júri

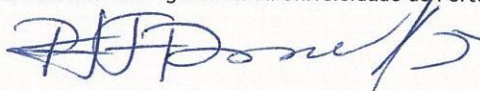
Presidente Professor Doutor Mário Jorge Moreira Leitão
Professor Associado da Faculdade de Engenharia da Universidade do Porto



Professor Doutor António Manuel Raminhos Cordeiro Grilo
Professor Auxiliar do Instituto Superior técnico da Universidade Técnica de Lisboa

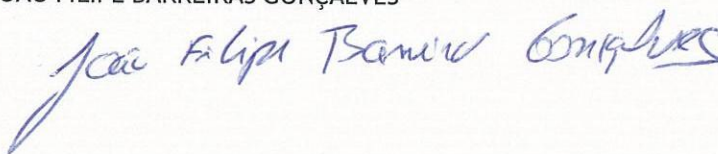


Professor Doutor Rosaldo José Fernandes Rossetti
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto



O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - JOÃO FILIPE BARREIRAS GONÇALVES



Faculdade de Engenharia da Universidade do Porto

Resumo

O trabalho desenvolvido especifica e propõem uma arquitectura baseada em serviços para sistemas inteligentes de transportes (ITS), assente numa rede de comunicações sem fios veículo-a-veículo ou ainda, em inglês, “*vehicle-to-vehicle*” (V2V), em larga escala. O tráfego gerado por estes sistemas normalmente possui requisitos de tempo real, pelo que a arquitectura definida é suficientemente flexível para permitir a aquisição e disseminação de dados em tempo real, tanto na perspectiva do utilizador individual como na perspectiva do sistema em cenários urbanos.

O objectivo foi especificar SOA (*Service-oriented Architecture*) em uma VANET (*Vehicular Ad-Hoc Network*) que, tal como o nome sugere, trata-se de uma rede *Ad-Hoc* móvel constituída pela rede de comunicação gerada entre os veículos. Ou seja, propõem-se um modelo para promover serviços em uma rede *Ad-Hoc* constituída por dispositivos móveis instalados em veículos.

Efectuado um estudo preliminar das tecnologias que viabilizam a aquisição e disseminação de dados em tempo real numa rede veículo a veículo em ambientes urbanos e das implicações da propagação dinâmica de informação de tráfego veicular, foram também identificadas as características e potenciais aplicações de SOA para ITS (*Intelligent Transportation Systems*). A partir desta informação foi definida uma arquitectura SOA para uma rede V2V.

Não sendo viável o recurso à utilização de sistemas reais, devido à sua grande escala e outros condicionalismos inerentes ao estudo de uma rede real de transportes em meios urbanos, foram pesquisadas diferentes ferramentas de simulação de redes de tráfego e de simulação de redes de comunicação que permitissem testar a arquitectura de rede implementada.

Foi conceptualizado e desenvolvido um protótipo funcional que implementa algumas das funcionalidades da arquitectura proposta no âmbito da plataforma MAS-T²er Lab. A plataforma MAS-T²er Lab é uma aplicação de investigação de trânsito e transporte concebida e correntemente em fase de desenvolvimento no NIAD&R/FEUP.

Abstract

The work developed specifies and proposes a service-oriented architecture for intelligent transportation systems (ITS), based on a wireless communication network between vehicles in large scale. The traffic generated by these systems usually poses real-time requirements, so the architecture defined is flexible enough to allow the acquisition and dissemination of data in real time, both from the perspective of the individual user and from the perspective of the system in urban scenarios.

The aim was to specify SOA (Service-oriented Architecture) in a VANET (Vehicular *Ad-Hoc* Network) that, as the name suggests, it consists of a mobile Ad-Hoc network generated by the communication network between vehicles. Thus, a model to promote services in an Ad-Hoc network generated by mobile devices installed in vehicles is proposed.

After a preliminary study of the technologies that enable the acquisition and dissemination of real-time data in a "vehicle-to-vehicle" network applied to urban environments and the implications of the dynamic spread of vehicular traffic information, the characteristics and potential applications of SOA for ITS (Intelligent Transportation Systems) were also identified. With all this information it was defined a SOA architecture for a V2V network.

As dealing directly with real systems was not viable, due to its large scale and all other constraints inherent in the study of a real transportation network applied to urban areas, different tools for simulation of traffic networks and simulation of communication networks were studied to test the network architecture implemented.

A working prototype that implements some of the features of the proposed architecture has been conceptualized and also implemented within the MAS-T²er Lab framework. The MAS-T²er Lab framework is traffic and transportation research platform conceived and currently in the development stage at NIAD&R/FEUP.

Agradecimentos

Gostaria de expressar a minha gratidão a todos aqueles que contribuíram de alguma forma para a realização deste trabalho.

Em primeiro lugar, agradeço ao meu orientador, o Professor Rosaldo Rossetti, pelo apoio, colaboração e orientação prestados. Os seus conselhos e rigor científico constituíram uma preciosa ajuda na realização deste trabalho.

A todo o pessoal do LIACC (Laboratório de Inteligência Artificial e Ciência de Computadores) da FEUP (Faculdade de Engenharia da Universidade do Porto) agradeço a simpatia com que me receberam, o companheirismo e o excelente ambiente de trabalho que me proporcionaram neste laboratório. Um agradecimento especial é dirigido ao Edgar Esteves e ao Paulo Ferreira pelas oportunidades de discussão científica e disponibilidade que sempre demonstraram.

Por último, mas não menos importante, agradeço à minha família e amigos pelo apoio e incentivo que me proporcionaram.

Índice

Resumo	v
Abstract	vii
Agradecimentos	ix
Índice	xi
Lista de figuras	xiii
Lista de tabelas	xvi
Abreviaturas.....	xvii
Capítulo 1	1
Introdução.....	1
1.1 - Contextualização.....	1
1.2 - Objectivos do trabalho.....	2
1.3 - Organização da Tese.....	3
Capítulo 2.....	5
Descrição do Problema	5
Capítulo 3	9
Arquitectura Orientada a Serviços	9
3.1 - Conceitos inerentes.....	9
3.2 - Definição de Arquitectura Orientada a Serviços.....	11
3.3 - Elementos da Arquitectura	13
3.4 - Características de SOA	14
3.5 - Do que não se trata SOA	18
3.6 - Métodos para a implementação de SOA19	
3.7 - SOA aplicada a Redes de Transportes Inteligentes.....	19
Capítulo 4.....	21
<i>Redes de Comunicação Veículo-a-Veículo</i>	<i>21</i>
4.1 - Definição e Contextualização	21
4.2 - Conceitos relacionados.....	22
Capítulo 5	43

Ferramentas de Simulação	43
5.1 - Definição e Contextualização	43
5.2 - Tipos de Sistemas de Simulação	44
5.3 - Simulação de Redes de Comunicação	46
5.4 - Simulação de Tráfego	46
5.5 - Simulação Integrada	49
Capítulo 6	53
Solução Proposta	53
6.1 - Definição da Arquitectura	53
6.2 - Escolha da ferramenta de simulação	66
Capítulo 7	67
Desenvolvimento do protótipo	67
7.1 - Plataforma MAS-T ² er Lab	67
7.2 - Arquitectura da ferramenta de simulação	68
7.3 - Aplicação <i>Simulation Engine Control</i>	70
7.4 - Implementação	73
Capítulo 8	83
Análise de resultados	83
8.1 - Definição do cenário de teste	83
8.2 - Testes básicos	86
8.3 - Resultados das Simulações das diferentes configurações do cenário de teste	87
8.4 - Outras simulações efectuadas	90
Capítulo 9	93
Conclusões	93
9.1 - Principais observações	93
9.2 - Perspectivas de trabalho futuro	94
Referências	97
Anexo A	103
Ficheiros XML da rede usada no cenário de teste	103

Lista de figuras

Figura 2.1 - SOA com servidor central convencional (a) e em uma rede VANET distribuída sem qualquer infra-estrutura de suporte (b).	5
Figura 2.2 - Caso de estudo: SOA aplicada a uma VANET	6
Figura 3.1 - Componentes de SOA [10]	12
Figura 3.2 - Diferentes camadas de uma aplicação orientada a serviços, [11].	12
Figura 3.3 - Paradigma “Localizar-Ligar-Executar [9]	13
Figura 3.4 - Granularidade de serviços: (a) Serviço de Grão Grosso (b) Serviço de Grão Fino [12]	17
Figura 3.5 - Graus de granularidade [9].....	17
Figura 4.1 - Exemplo de uma rede V2V [16].....	22
Figura 4.2 - <i>Intelligent Transportation System (ITS)</i> [17]	23
Figura 4.3 - <i>Intelligent Car</i> [17].....	23
Figura 4.4 - VANET mista incorporando comunicações V2V e V2I [22]	25
Figura 4.5 - Disseminação de informação numa VANET pura V2V [23]	25
Figura 4.6 - Classificação dos protocolos de encaminhamento das redes MANET	29
Figura 4.7 - Funcionamento do algoritmo de inundação [28]	32
Figura 4.8 - Funcionamento com os <i>Multipoint Relays</i> [28].....	32
Figura 4.9 - Funcionamento da mensagem RREQ [28]	33
Figura 4.10 - Funcionamento da mensagem RREP [28].....	33
Figura 4.11 - Formato do datagrama da mensagem RREQ [28]	35
Figura 4.12 - Formato do datagrama da mensagem RREP [28]	36
Figura 4.13 - Formato do datagrama da mensagem RERR [28]	37
Figura 4.14 - Formato do datagrama da mensagem RREP-ACK [28]	37

Figura 4.15 - Comunicação entre componentes do protocolo ZRP [28]	38
Figura 4.16 - Visão dos nós utilizando o protocolo FSR [28].....	39
Figura 5.1 - Componentes se um sistema de simulação [40]	44
Figura 5.2 - Simuladores com aplicação embutida [40]	45
Figura 5.3 - Simulador Integrado de redes V2V.....	45
Figura 5.4 - Screenshots de uma simulação efectuada com SUMO [45]	47
Figura 5.5 - Simulação efectuada com VISSIM [46].....	48
Figura 5.6 - Simulação usando CARISMA [40].....	49
Figura 5.7 - Veículos reais e simulados interagindo no simulador GrooveNet [49].....	50
Figura 5.8 - Simulação usando DIVERT [51]	51
Figura 6.1 - Arquitectura de níveis	54
Figura 6.2 - Caso de estudo: SOA aplicada uma rede V2V	55
Figura 6.3 - Pilha de Protocolos (adaptado de [1])	57
Figura 6.4 - Plug-in: Olsrd [1].....	57
Figura 6.5 - Formato da mensagem SOA (adoptado de [1])	58
Figura 6.6 - Geração da mensagem SOA [1].....	59
Figura 6.7 - Processamento de mensagem SOA [1]	59
Figura 6.8 - Diagrama de desenvolvimento UML para arquitectura SOA no veículo.....	61
Figura 6.9 - Arquitectura SOA para dispositivos móveis [55].....	64
Figura 7.1 - Plataforma MAS-T ² er Lab [57]	68
Figura 7.2 - Arquitectura de alto nível da ferramenta de simulação [59]	69
Figura 7.3 - Arquitectura física da aplicação SEC [59]	70
Figura 7.4 - Arquitectura física da aplicação SEC com comunicação V2V	72
Figura 7.5 - Arquitectura da ferramenta de simulação [59].....	73
Figura 7.6 - Diagrama de estados do ciclo de execução da obtenção da referência aos carros simulados.....	74
Figura 7.7 - Código que implementa a determinação da posição global.....	75
Figura 7.8 - Simulador com comunicação V2V implementada	76
Figura 7.9 - Conexão via Socket	77
Figura 7.10 - Interface gráfica inicial do simulador	79

Figura 7.11 - Interface gráfica do simulador após adaptações implementadas.....	80
Figura 7.12 - Editor de rede gráfico e aplicação de simulação	80
Figura 8.1 - Esquema da rede a usar nas simulações do cenário de teste	84
Figura 8.2 - Simulação da rede do cenário de teste	84
Figura 8.3 - Comparação entre o número de veículos presentes no simulador e o número de veículos que se conectam via V2V num cenário com baixo fluxo de trânsito	87
Figura 8.4 - Comparação entre o número de veículos presentes no simulador e o número de veículos que se conectam via V2V num cenário com fluxo médio de trânsito	88
Figura 8.5 - Comparação entre o número de veículos presentes no simulador e o número de veículos que se conectam via V2V num cenário com alto fluxo de trânsito	88
Figura 8.6 - Comparação da percentagem de rede coberta pela rede V2V com as diferentes configurações do cenário de teste ao longo do tempo de simulação.....	89
Figura 8.7 - Percentagem média de rede coberta pela rede V2V com as diferentes configurações do cenário de teste	89

Lista de tabelas

Tabela 4.1 - Exemplo da tabela de encaminhamento utilizada pelo Protocolo DSDV [28].....	30
Tabela 4.2 - Comparação entre protocolos de encaminhamento de redes MANET (Adaptada de [28]).....	40
Tabela 6.1 - Serviços de Alto nível.....	60
Tabela 7.1 - Protocolo de Comunicação	78
Tabela 8.1 - Parâmetros para <i>Car-following</i> e <i>Lane-changing</i> usados nas simulações.....	85

Abreviaturas

ABS	Anti-lock Braking System
API	Application Programming Language
AODV	<i>Ad-Hoc</i> On-Demand Distance Vector
C2CCC	European Car-to-Car Communication Consortium
C2P2	Car-to-car/peer-to-peer
DIVERT	Development of Inter-Vehicular Reliable Telematics
DSDV	Destination-Sequenced Distance-Vector
DSR	Dynamic Source Routing
DSRC	Dedicated Short Range Communications
ESP	Electronic Stability Program
FEUP	Faculdade de Engenharia do Porto
FSR	Fisheye State Routing
GUI	Graphical User Interface
GPRS	General Packet Radio Service
GPS	Global Positioning System
ITS	Intelligent Transportation System
LIACC	Laboratório de Inteligência Artificial e de Ciência de Computadores
IPC	Internet Process Communication
MA	Moveable Agent
MANET	Mobile Ad-Hoc Network
MAS	Multi-Agent System
MAS-T ² er Lab.	Laboratory for Multi-Agent base traffic and Transportation engineering research
MPR	Multipoint Relays
NIAD&R	Distributed Artificial Intelligence & Robotics Group
NDM	Network Data Manager
NS2	Network Simulator 2
OBU	Onboard Unit
OFDM	Orthogonal frequency-divison multiplexing
OLSR	Optimized Link State Routing Protocol
OSGI	Open Services Gateway Initiative
RERR	Route Error
RREP	Route Reply

RREP-ACK	Route Reply Acknowledgment
RREQ	Route Request
RIP	Routing Information Protocol
RS	Road Segment
SA	Semaphore Agent
SEC	Simulator Engine Controller
SOA	Service-oriented Architecture
STL	StereoLitho
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Universal Modeling Language
VANET	Vehicular Ad-Hoc Network
VITP	Vehicular Information Transfer Protocol
V2I	Vehicle-to-Infrastructure
V2R	Vehicle-to-Road
V2V	Vehicle-to-Vehicle
WAVE	Wireless Access in Vehicular Environments
WSDL	Web Service Description Language
XML	eXtensible Markup Language
ZRP	Zone Routing Protocol

Capítulo 1

Introdução

Este capítulo pretende fornecer ao leitor uma visão global do trabalho desenvolvido e reportado na presente dissertação. Após um breve enquadramento do tema abordado, são descritos os objectivos deste trabalho. Por último é apresentada a estrutura do documento.

1.1 - Contextualização

O número de pessoas a residir nos espaços urbanos tem crescido a olhos vistos. A maior parte das pessoas opta por usar transportes individuais, congestionando as redes de transportes. Estudos e simulações de tráfego têm sido feitos durante décadas sem obter grandes resultados na optimização dessas redes. Contudo, num passado recente, a integração das comunicações entre veículos no contexto dos sistemas de transportes inteligentes (ITS - *Intelligent Transportation Systems*) tem vindo a revelar-se uma mais-valia nesta área. Os veículos estão a ser conectados através de ligações sem fios e equipados com um conjunto de sensores e actuadores que permitem conhecer e fornecer a sua localização, aumentar a segurança dos seus passageiros e proporcionar ferramentas de entretenimento.

Comunicações entre veículos têm-se revelado um dos campos de interesse das telecomunicações que mais rapidamente cresceu. Veículos equipados com dispositivos de comunicação sem fios de curto alcance podem formar uma rede móvel *Ad-hoc*, chamada VANET (*vehicular Ad-hoc Network*). A existência destas redes e a disseminação de informação nelas terão provavelmente um papel importante na segurança e eficiência das redes de tráfego num futuro próximo. De forma a avaliar em que medida isso será verdade, simuladores de tráfego tem sofrido uma gigantesca evolução nos últimos anos com vista a incorporarem as comunicações entre veículos. Varias vias para atingir esse efeito tem sido propostas e analisadas. A grande parte dos estudos caminha na direcção da criação de simuladores que integrem a simulação de tráfego e a simulação de redes de comunicação na mesma ferramenta. Contudo outros estudos optam pela utilização de dois simuladores

independentes, um simulador de tráfego e um simulador de redes de comunicação, interligados por uma aplicação que assegura a troca de informação entre eles. Outros tópicos que têm sido investigados para além do tipo de simulador a usar são: o tipo de arquitectura de simulação que melhor se adequa a estas redes, que tipo de informação pode ser trocada nelas, que tipo de serviços podem ser disponibilizados e se este tipo de tecnologia realmente ajudará na optimização e melhoria da segurança das redes de tráfego.

Esta dissertação é sobre arquitectura orientada a serviços com ênfase nas comunicações veículo-a-veículo. Se os dispositivos instalados nos veículos e no ambiente que os rodeia são vistos como serviços, aplicações podem ser criadas para compor um conjunto de serviços que regule o fluxo de dados entre eles. Contudo a composição de serviços não é tudo o que importa para implementar SOA numa rede V2V. Para realizar SOA nestas redes, as aplicações devem lidar com desafios como a heterogeneidade dos dispositivos, resistência ao dinamismo da rede, mobilidade e descoberta e disponibilidade dos serviços

1.2 - Objectivos do trabalho

O trabalho desenvolvido, e que culminou na escrita desta dissertação, teve como objectivo principal propor e especificar uma arquitectura baseada em serviços para sistemas inteligentes de transportes, assente numa rede de comunicação sem fios V2V em larga escala. Pretendia-se que tal rede fosse suficientemente flexível para permitir a aquisição e disseminação de dados em tempo real, tanto na perspectiva do utilizador individual como na perspectiva do sistema em cenários urbanos. Os objectivos definidos foram assim:

- Estudar as tecnologias que viabilizam a aquisição e disseminação de dados em tempo real numa rede de veículos em ambiente urbanos;
- Entender e avaliar as implicações da propagação dinâmica de informações de tráfego veicular;
- Identificar potenciais aplicações de SOA (*Service-oriented architecture*) para ITS (*Intelligent Transportation Systems*);
- Identificar diferentes ferramentas de simulação de tráfego e de simulação de redes de comunicação;
- Identificar o potencial de desenvolvimento e as diferentes aplicações práticas de uma rede V2V (como por exemplo planeamento de rotas, detecção de colisões, resolução de congestionamentos, etc);
- Propor uma arquitectura SOA para uma rede V2V;
- Definir cenários de teste da aplicação desenvolvida.

As actividades desse projecto foram conduzidas no âmbito da colaboração com o Departamento de Controlo e Gestão de Tráfego da Câmara do Porto, pelo que todas as simulações efectuadas pretendiam ser relativas a uma parte da rede de tráfego da cidade do porto.

Alguns desafios deste trabalho estavam particularmente relacionados com temas como a comunicação sem fios, processamento de dados e informação, sistemas e protocolos auto-organizáveis, optimização colaborativa e computação ubíqua. Os resultados deste projecto serão integrados na plataforma MAS-T²er Lab, concebida e em desenvolvimento no Laboratório

de Inteligência Artificial e Ciência de Computadores (LIACC) da FEUP. O objectivo básico desta plataforma é estudar, desenvolver e testar diferentes tecnologias inteligentes de transportes, considerando os padrões da indústria e diferentes requisitos dos seus diversos intervenientes, integrados em três subsistemas: o mundo real, o domínio virtual, e o indutor de estratégias de controlo e gestão inteligente.

1.3 - Organização da Tese

Após a presente introdução organiza-se o presente relatório nos seguintes capítulos:

- O capítulo 2 apresenta a descrição detalhada do problema em análise;
- O capítulo 3 documenta a revisão do estado da arte das arquitecturas SOA com ênfase na sua aplicação a redes de transportes inteligentes baseadas em comunicações V2V;
- O capítulo 4 apresenta os conceitos relativos à implementação de uma rede V2V orientada a ambientes urbanos;
- O capítulo 5 examina a simulação das redes V2V. São descritas as características que uma ferramenta de simulação deve possuir para simular uma rede V2V e é feito um estudo comparativo entre algumas ferramentas de simulação pesquisadas;
- O capítulo 6 apresenta a solução proposta para uma arquitectura baseada em serviços para redes V2V;
- O capítulo 7 descreve o planeamento e implementação do protótipo desenvolvido para simulação de redes V2V. Começa-se por uma visão geral da implementação e posteriormente são apresentados níveis de detalhe mais específicos, descrevendo-se a implementação dos principais algoritmos;
- O capítulo 8 descreve alguns resultados experimentais obtidos das simulações de uma rede simples, usando o protótipo implementado;
- O capítulo 9 apresenta as conclusões e perspectivas de trabalho futuras para o presente tema.

Capítulo 2

Descrição do Problema

Como já foi definido no capítulo 1, o objectivo deste trabalho é implementar SOA em uma MANET (*Mobile Ad-Hoc Network*) sem qualquer tipo de infra-estrutura em que os nós dessa rede são veículos, ou seja uma VANET. Assim, pretende-se utilizar totalmente os benefícios oferecidos por uma rede VANET e ao mesmo tempo construir um ambiente baseado em serviços que siga os conceitos de SOA.

Convencionalmente, em SOA todos os serviços são publicados num servidor central. Todos os nós precisam de saber a localização do servidor de serviços central e devem estar conectados a este para publicar e descobrir serviços. Na arquitectura que é proposta não existe servidor central, a localização dos serviços é distribuída pela rede VANET. Tal é ilustrado na figura seguinte.

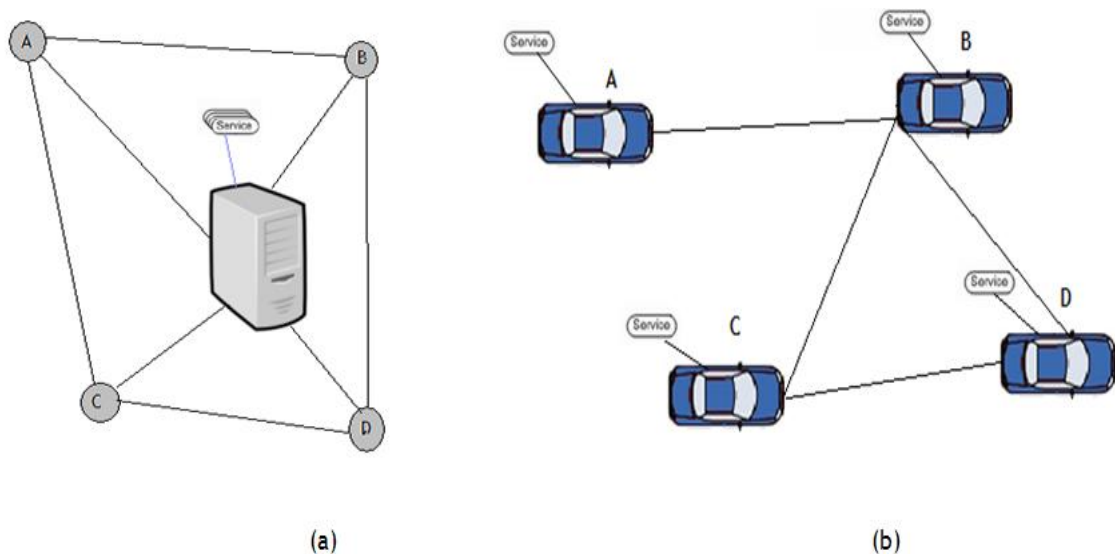


Figura 2.1 - SOA com servidor central convencional (a) e em uma rede VANET distribuída sem qualquer infra-estrutura de suporte (b).

Para constituir o tipo de ambiente pretendido, dois componentes básicos são claramente necessários: o protocolo de *routing* da rede VANET e a realização de SOA. O protocolo de *routing* da rede VANET é responsável pela conectividade da rede, manutenção da sua topologia e pelo encaminhamento dos pacotes de informação. A camada de routing constitui, assim, a espinha dorsal do ambiente, já que é responsável pela topologia da rede e em análise final conecta os serviços aos seus consumidores.

Duas decisões importantes tiveram que ser tomadas: i) escolher o protocolo a utilizar da vasta gama de protocolos existentes e ii) como implementar a funcionalidade SOA.

As principais considerações para implementação de uma arquitectura SOA são a realização da descoberta de serviços, definição da linguagem de descrição de serviços e do formato das mensagens não sendo definidos quaisquer tipos de protocolos específicos para este tipo de arquitectura.

Para ilustrar a ideia de implementar SOA em uma rede VANET foi definido o caso de utilização, adoptado de [1], ilustrado na figura 2.2.

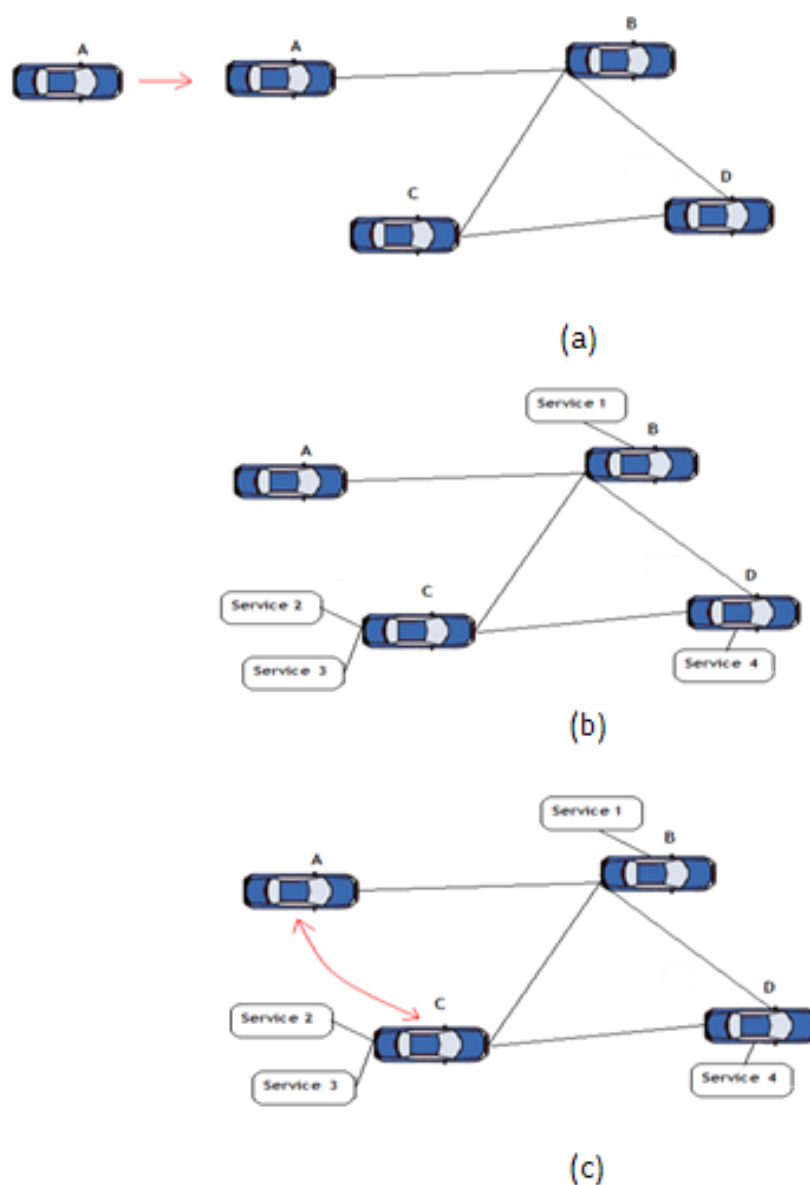


Figura 2.2 - Caso de estudo: SOA aplicada a uma VANET

Na figura 2.2 (a) demonstra a formação da rede em si. Todos os nós (A a D) representam veículos equipados com dispositivos móveis de comunicação V2V. No princípio, o nó A está desconectado não tendo qualquer outro nó na sua área de cobertura. Quando outros nós entram na sua área de cobertura, ele liga-se a estes. Todos os nós conectados formam assim uma VANET. Depois das configurações automáticas necessárias a VANET está pronta a encaminhar mensagens originadas em qualquer nó e destinadas a qualquer outro.

Quando a rede VANET está pronta a funcionalidade SOA tem que ser implementada. Todos os nós provedores de serviços publicam a descrição dos respectivos serviços, assim os restantes nós podem descobrir esses serviços. O processo de descoberta dos serviços deve ser implementado de tal maneira que qualquer serviço na rede VANET deve ser detectado sem qualquer configuração manual. Na figura 2.2 (b) todos os nós que contêm serviços publicaram esses mesmos serviços e a descrição de qualquer um dos serviços é visível por todos os nós da rede.

Depois da rede VANET estar formada e SOA ter sido implementada a rede de serviços está pronta a ser utilizada. Um exemplo da utilização de um serviço é demonstrado na figura 2.2 (c). O nó A conecta-se à interface do serviço 3, que se situa no nó C. Toda a informação necessária à troca de mensagens entre o consumidor de serviços (nó A) e o provedor de serviços (nó C) está contida na descrição do serviço 3, não sendo necessário ao nó A conhecer quaisquer outras propriedades do nó C. O nó A e o nó C não se encontram ligados directamente, assim as mensagens têm que ser encaminhadas pelo nó B. O encaminhamento (*routing*) na rede VANET é independente da funcionalidade SOA.

Capítulo 3

Arquitectura Orientada a Serviços

Neste capítulo é documentada toda a revisão do estado da arte relativo a *Service-oriented Architectures* (SOA).

3.1 - Conceitos inerentes

Antes de definir no que consiste *Service-oriented Architecture* (SOA) ou ainda, em português, Arquitectura Orientada a Serviços, alguns conceitos inerentes devem ser clarificados.

3.1.1 - Definição de Arquitectura de software

Investigadores e profissionais da área de administração de sistemas de informação têm desenvolvido muitas definições para o termo **arquitectura de software** ao longo dos últimos anos. Descrevem-se a seguir algumas definições:

“A arquitectura de software de um programa ou de um sistema computacional é a estrutura ou as estruturas do sistema, que compreendem elementos de software, as propriedades visíveis e externas desses elementos, e as relações entre eles.” [2];

“Arquitectura é o termo que muitas pessoas tentam definir com pouca concordância. Há dois elementos comuns: um deles trata do mais alto nível de

quebra de um sistema em partes; o outro trata de uma decisão difícil de ser mudada.” [3]

“A arquitetura de software é um conjunto de declarações, que descreve os componentes de software e atribui funcionalidades de sistema para cada um deles. Ela descreve a estrutura técnica, limitações e características dos componentes, bem como as interfaces entre eles. A arquitetura é o esqueleto do sistema e, por isso, torna-se o plano de mais alto-nível da construção de cada sistema.” [4]

A definição usada pelo ANSI/IEEE afirma que uma arquitetura de software trata basicamente de como os componentes fundamentais de um sistema se relacionam intrinsecamente e extrinsecamente [5].

Uma importante preocupação na realização de grandes sistemas de computação é a arquitetura de software do sistema.

Quando uma arquitetura de software é criada para um sistema, é importante assegurar que um conjunto de preocupações é tido em conta. Além dos requisitos funcionais que descrevem o que é suposto o sistema fazer, um conjunto de requisitos de qualidade restringe o comportamento do sistema em relação a propriedades como disponibilidade, desempenho, segurança e testabilidade.

Tal como é descrito em [6], a um nível mais elevado, um estilo arquitectónico é um conjunto de regras que rege o desenho da arquitectura. Definir um estilo arquitectónico torna mais fácil manter um sistema coerente porque melhora a integridade conceptual do sistema. Alguns estilos são mais adequados para determinados fins do que outros. Decisões arquitectónicas tomadas de início têm uma grande influência no sistema resultante e no processo de criação deste. Se uma má decisão é tomada numa fase inicial, pode significar a diferença entre o sucesso e o fracasso do sistema. A escolha de um estilo de arquitectura em particular não garante, por si só, o sucesso do sistema desenvolvido e, portanto, detalhes de como esse estilo deve ser aplicado são no mínimo tão importantes como as pistas para que estilo usar.

3.1.2 - Definição de Serviço

Também é importante definir o conceito de serviço. O conceito de serviço é definido de várias formas, principalmente na literatura não académica, sendo algumas delas até certo ponto contraditórias. Já na literatura académica, encontram-se um maior consenso; mas em certos casos, o que na literatura não académica é conhecido como serviço, na literatura académica é conhecido algumas vezes como componentes ou contratos. Contudo algumas definições de serviço já foram desenvolvidas e aceites. A título de exemplo, passa-se a citar:

“ Serviço é uma unidade de trabalho completada por um provedor de serviços, que produz resultados finais desejados para um consumidor. Estes resultados podem modificar o estado do consumidor, do provedor, ou de ambos” [7]

O serviço, no ponto de vista da arquitectura SOA, é uma função de um sistema computacional que é disponibilizada para outro. Um serviço deve funcionar de forma independente do estado de outros serviços e deve possuir interface bem definida.

3.2 - Definição de Arquitectura Orientada a Serviços

O conceito do que é uma Arquitectura orientada a Serviços ainda não é consensual, já que o conceito de serviço, revisto na secção anterior, também não o é. Contudo, existem já algumas definições de SOA que reúnem algum consenso. Descrevem-se a seguir algumas definições:

“ A arquitectura orientada a serviços é uma arquitectura que possui características únicas. É um modelo de componente que inter-relaciona as unidades funcionais de uma aplicação, chamadas serviços, através de interfaces bem definidas e contractos entre os serviços.” [8]

“A Sun definiu SOA rigorosamente na década de 90 para descrever o Jini, um ambiente para encontrar e usar dinamicamente serviços de uma rede. O objectivo da criação do Jini era criar um ambiente dinâmico de rede para dispositivos, serviços e aplicações. Nesse ambiente serviços e dispositivos poderiam ser adicionados e removidos da rede dinamicamente [9]

Pode-se, assim, definir SOA como um estilo de arquitectura de software cujo princípio fundamental preconiza que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços. Frequentemente, estes serviços são organizados através de um "barramento de serviços" (*service bus*, em inglês) que disponibiliza interfaces, ou contratos, acessíveis através de *Web services* ou outra forma de comunicação entre aplicações. Na figura 3.1, é possível visualizar todos os componentes que definem SOA e como se estruturam segundo esta definição.

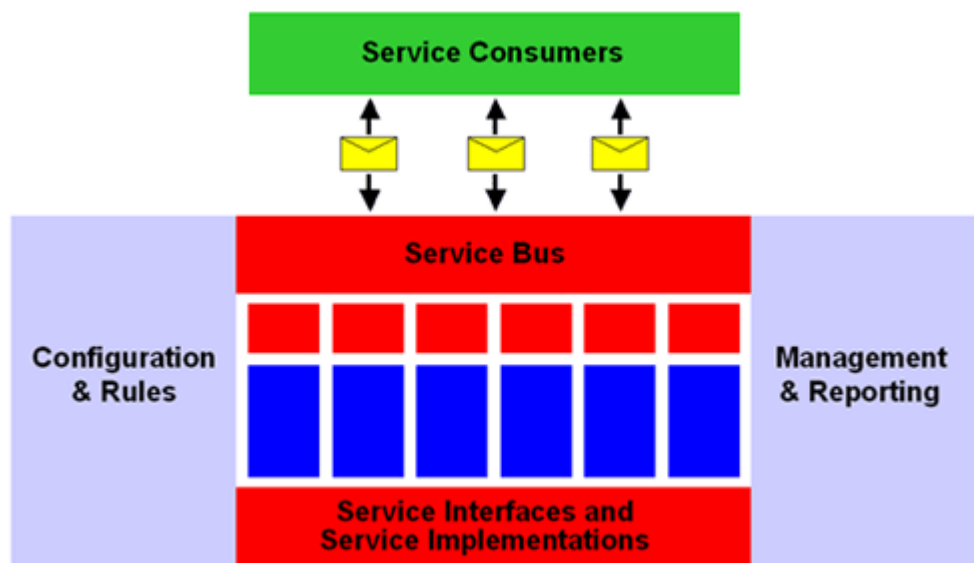


Figura 3.1 - Componentes de SOA [10]

A arquitectura SOA é baseada nos princípios da computação distribuída. Como qualquer outra aplicação distribuída é definida em múltiplas camadas, que incluem as camadas de apresentação, lógica de negócio e persistência. SOA adiciona mais dois elementos na sua estrutura, a camada de serviços e a camada de processos de negócio. Na figura 3.2 é possível ver essa estrutura em camadas.

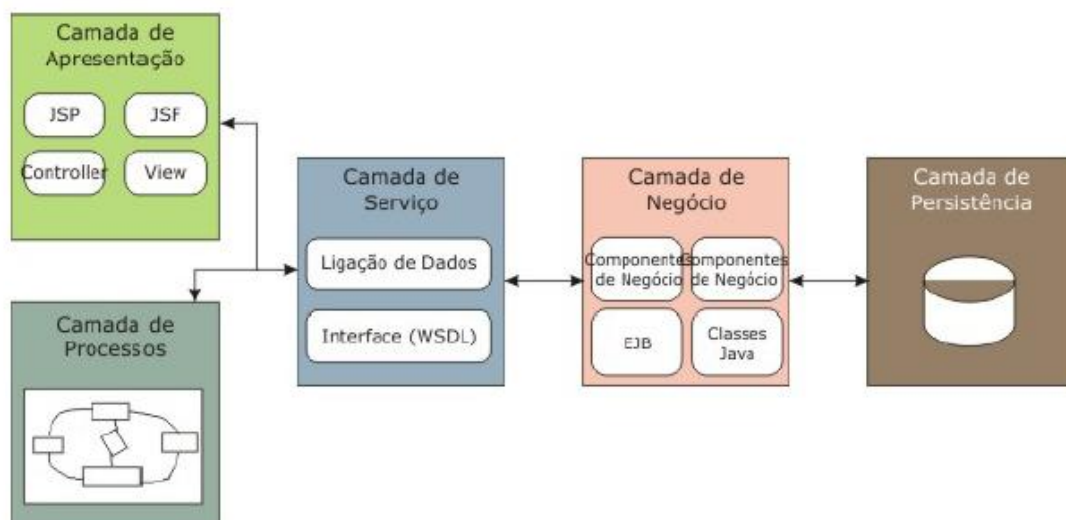


Figura 3.2 - Diferentes camadas de uma aplicação orientada a serviços, [11].

Além da perspectiva estritamente técnica, a Arquitectura orientada a Serviços também se relaciona com determinadas políticas e conjuntos de "boas práticas" que pretendem criar um processo para facilitar a tarefa de encontrar, definir e gerir os serviços disponibilizados.

O principal objectivo de SOA é alinhar o mundo empresarial com o mundo das tecnologias da informação de uma forma que torna os dois mais eficazes. SOA é uma ponte que cria uma simbiose e sinérgico relacionamento entre os dois, que é mais poderoso e valioso do que qualquer coisa que tenha ocorrido no passado.

3.3 - Elementos da Arquitectura

A arquitectura SOA pode ser bem representada a partir do paradigma "*find-bind-execute*", o que significa aproximadamente paradigma de localizar-ligar-executar. Tal conceito é análogo a "*Ciclo de Deming*" aplicado a serviços, que define o ciclo que envolve o planeamento, a execução, a monitorização e a tomada de acção pró-activa para a melhoria da qualidade. O paradigma de localizar-ligar-executar permite que o consumidor de um serviço peça ao registo um serviço que é compatível com o seu critério. Se o registo encontra esse serviço, passa ao consumidor um contracto e o endereço para o serviço.

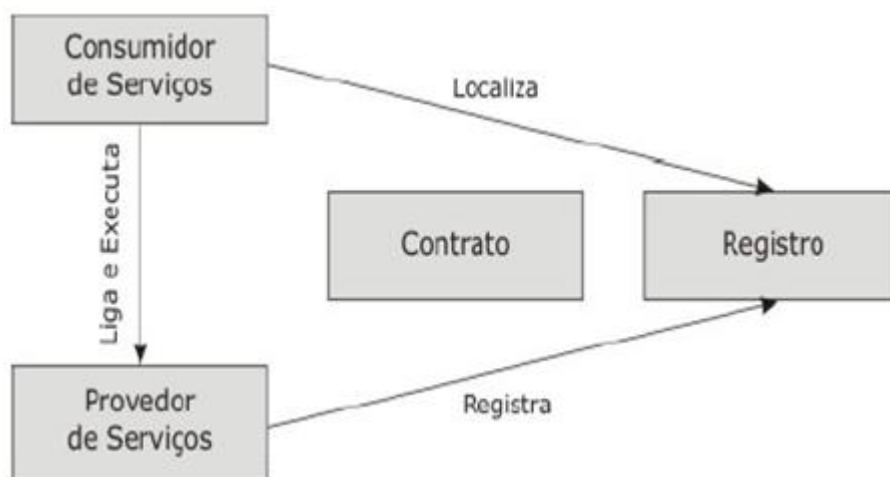


Figura 3.3 - Paradigma "Localizar-Ligar-Executar [9]

A seguir são descritas as entidades que suportam este paradigma de acordo com [9].

a) Consumidor de Serviço

O consumidor de serviço é uma aplicação, um serviço ou um outro tipo de software que requer um serviço. É a entidade que inicia a localização do serviço. O consumidor executa o serviço enviando uma requisição formatada de acordo com o contrato.

b) Provedor de Serviço

O provedor de serviços é a entidade endereçável que aceita e executa requisições de consumidores. Pode ser um *mainframe*, um componente ou outro tipo de software que executa a requisição do serviço. O provedor publica o seu contracto no registo para que seja acedido por consumidores de serviços.

c) Registo de Serviço

O registo de serviço é um directório de rede que possui os serviços disponíveis. É uma entidade que aceita e armazena contratos de provedores de serviços e provê esses contratos para consumidores de serviços.

d) Contrato de Serviço

Um contrato é uma especificação da forma de como um consumidor de serviços deve interagir com o provedor de serviços. Especifica o formato da requisição e da resposta do serviço. Um contrato de serviço pode requerer um conjunto de pré-condições e pós-condições, que especificam o estado em que o serviço deve estar para executar uma certa função. O contrato pode também especificar a qualidade de serviço.

e) Proxy de Serviço

O provedor de serviço fornece um *proxy* de serviços para o consumidor. O consumidor executa a requisição chamando uma função da API do *proxy*. O *proxy* encontra um contrato e referencia o provedor do serviço no registo. Então, formata a mensagem de requisição, e executa-a em benefício do consumidor.

f) "Aluguer" de Serviço

O "aluguer" de serviço, que é concedido pelo registo para o consumidor, especifica a quantidade de tempo durante a qual é válido. Quando o "aluguer" termina, o consumidor deve requisitar outro do registo.

3.4 - Características de SOA

Toda a arquitectura de software reflecte os diferentes princípios e conjunto de *tradeoffs* (troca de uma característica por outra melhor) usados pelo projectista. De acordo com [9] a arquitectura orientada a serviços tem as seguintes características:

- Serviços “descobertos” e ligados dinamicamente;
- Serviços independentes e modulares;
- Serviços enfatizam interoperabilidade;
- Serviços possuem baixo acoplamento;
- Serviços possuem uma interface de rede endereçável;
- Interfaces de serviços são de granularidade grossa;
- Serviços possuem localização transparente;
- Serviços podem ser compostos;
- SOA suporta auto-tratamento.

Em seguida, aprofunda-se cada uma dessas características:

a) “Descoberta” e ligação dinâmica dos serviços

SOA suporta o conceito de descoberta de serviços. Um consumidor descobre em tempo de execução qual o serviço a usar baseado num conjunto de critérios.

Já a ligação dinâmica ocorre da seguinte maneira:

1. O consumidor pede ao registo todos os serviços que executam uma certa actividade.
2. O registo retorna todas as entradas que suportam essa actividade. Essas entradas possuem, todas elas, um custo de transacção.
3. O consumidor escolhe o serviço baseado no menor custo de transacção.
4. O consumidor usando um apontador para a entrada do registo liga-se ao provedor do serviço.
5. O consumidor formata a mensagem de requisição de dados de acordo com a descrição fornecida pelo apontador.
6. O consumidor associa a mensagem a um tipo de transporte esperado pelo provedor.
7. O provedor de serviço executa e envia a mensagem de resposta, que também está descrita na descrição do serviço.

A única dependência entre o consumidor e o provedor é o contracto, que é fornecido pelo registo. Essa dependência é feita em tempo de execução e não em tempo de compilação. Os clientes não necessitam de informações em tempo de compilação.

As interfaces dos serviços são descobertas dinamicamente, da mesma forma que as mensagens.

b) Independência e modularidade

Os serviços em SOA são independentes e modulares. Um dos conceitos mais importantes de SOA é a modularidade. Um serviço suporta um conjunto de interfaces, as quais devem ser coesas, ou seja, devem estar relacionadas umas com as outras no contexto de um módulo. O princípio da modularidade deve ser implementado no projecto dos serviços que suportam uma aplicação, para que novos serviços possam ser facilmente agregados a uma aplicação apenas com poucas dependências bem conhecidas.

c) Interoperabilidade

SOA satisfazem a interoperabilidade, que é a habilidade de sistemas usarem diferentes plataformas e linguagens para comunicarem uns com os outros. O princípio da interoperabilidade é apontado por alguns autores como a característica mais importante de SOA. É fundamental que todos os componentes comuniquem, independentemente da linguagem em que foram construídos, do sistema operativo em que estão a ser executados e da arquitectura de hardware.

Cada serviço fornece uma interface que pode ser invocada por um tipo de conector. Um conector interoperável consiste em um protocolo e em um formato de dados que cada um dos potenciais clientes entenderá.

d) Baixo acoplamento

Acoplamento refere-se ao número de dependências entre os módulos. SOA alcança um baixo acoplamento através do uso de contratos e ligações. Analisando como são executados os diferentes passos necessários à “descoberta” e ligação de serviços apercebemo-nos de que o cliente não depende directamente da implementação do serviço, só depende do contracto que o serviço suporta. Como um serviço pode ser ao mesmo tempo consumidor e provedor de outros serviços, a dependência apenas do contracto só reforça a noção de baixo acoplamento.

e) Interface de rede endereçável

O papel da rede é central no conceito de SOA. Um serviço deve ter uma interface de rede endereçável. Um consumidor pertencente à rede deve ser capaz de invocar serviços através da mesma. A rede permite que serviços sejam reutilizados por qualquer consumidor em qualquer altura. A habilidade de uma aplicação assimilar um conjunto de serviços reutilizáveis em máquinas diferentes só é possível se os serviços suportarem uma interface de rede. A rede também permite que os serviços sejam independentes da sua localização, ou seja, a localização física dos serviços é irrelevante.

f) Interfaces de serviços de granularidade grossa

O conceito de granularidade aplica-se aos serviços de duas formas: primeiro aplica-se ao âmbito do domínio que todos os serviços implementam; segundo, aplica-se ao âmbito do domínio que cada método implementa dentro das interfaces.

O nível de granularidade apropriado para um serviço e seus métodos é relativamente grosso. Um serviço normalmente suporta um único e distinto conceito de negócio ou processo. Contém o software que implementa o conceito de negócio para que possa ser utilizado em vários sistemas distribuídos.

Para exemplificar, imagine um cenário onde são necessários vários parâmetros para a execução de uma certa tarefa. Pode-se criar um serviço que espera todas as informações necessárias de uma só vez e executa a tarefa (granularidade grossa). Por outro lado, pode-se definir um conjunto de serviços com essa mesma finalidade (granularidade fina): um serviço

para criar uma tarefa, outro para definir qual algoritmo será utilizado, um terceiro para receber um parâmetro (que pode ser chamado várias vezes) e um quarto para executar a tarefa. Estes dois tipos de granularidade são ilustrados na figura 3.4.

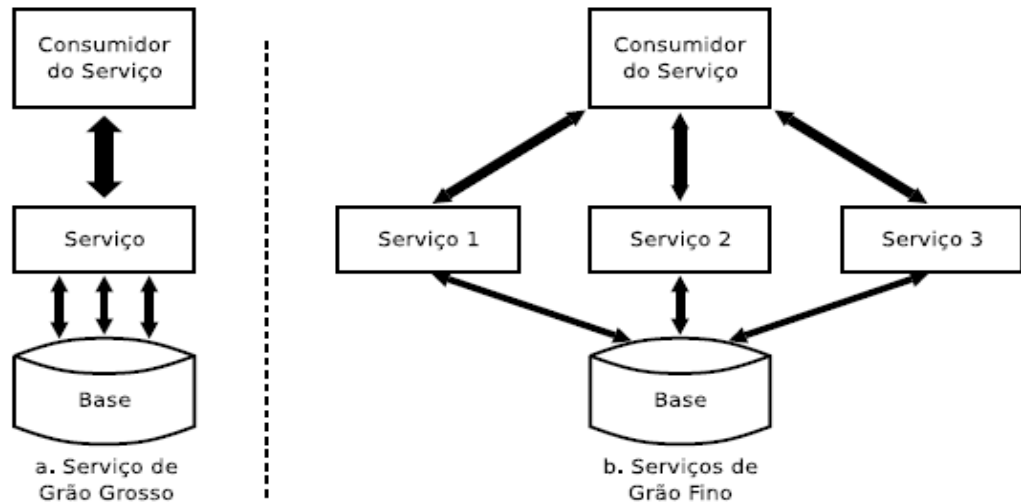


Figura 3.4 - Granularidade de serviços: (a) Serviço de Grão Grosso (b) Serviço de Grão Fino [12]

A opção pelo uso de serviços de grão grosso economiza tráfego na rede, pois há menos trocas de mensagens entre clientes e provedor. Porém, não permite muita flexibilidade nem a fácil reutilização dos serviços, já que oferece um pacote fechado de funcionalidades ao utilizador. Deve-se encontrar um equilíbrio para a granularidade dos serviços, a fim de evitar o excesso de comunicação sem perder a característica de reutilização.

Em SOA, a granularidade encontra-se escalonada de acordo com o apresentado na figura 3.5.

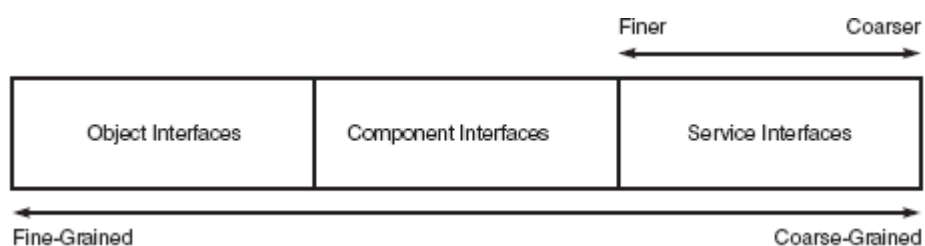


Figura 3.5 - Graus de granularidade [9]

g) Localização transparente

Consumidores de um serviço não sabem onde o serviço está localizado até que o localizam no registo. A localização e ligação a um serviço em tempo de execução permitem que um serviço se mova de um local para o outro sem que o cliente saiba. Esta característica aumenta a disponibilidade e o desempenho do serviço.

h) Habilidade de composição

Esta habilidade está relacionada com a estrutura modular dos serviços. A estrutura modular dos serviços permite que estes sejam assimilados em aplicações que o projectista não tinha noção aquando do desenvolvimento da aplicação. Com a utilização de serviços preexistentes e testados aumenta-se a qualidade de um sistema e o retorno do investimento devido à facilidade de reutilização.

i) Auto tratamento

Devido ao tamanho e complexidade das aplicações modernas, a capacidade de um sistema recuperar de erros e falhas sem necessitar de intervenção humana está-se a tornar uma característica importante.

As arquitecturas que suportam ligação dinâmica e execução de componentes em tempo de execução tem maior capacidade de recuperar de erros do que as restantes arquitecturas. Um sistema baseado em serviços tem ainda uma maior capacidade de recuperação de erros, já que, estando eles ligados dinamicamente em tempo de execução, se um serviço falha o cliente pode ligar-se a outro serviço, desde que este forneça a mesma ou um contracto de interface similar.

3.5 - Do que não se trata SOA

Existem várias ideias erradas em relação à Arquitectura Orientada a Serviços, de acordo com [13]. Algumas delas são:

- Trata-se de um produto que pode ser adquirido, não pode, pois trata-se de uma filosofia de design que informa como a solução deve ser construída;
- O objectivo é construir uma Arquitectura Orientada a Serviços, não é, pois a Arquitectura Orientada a Serviços é o meio para se alcançar um fim;
- A Arquitectura Orientada a Serviços requer uma reforma completa no processo tecnológico e de negócio, não é verdade, pois as soluções da Arquitectura Orientada a Serviços devem ser incrementais e construídas sobre os investimentos já existentes.

A Arquitectura Orientada a Serviços também é muitas vezes equacionada como Serviços Web, e os termos usados são passíveis de mudança. Apesar de ser verdade que a ampla adopção de protocolos e padrões baseados em Serviços Web facilita e difunde o conceito de SOA, são coisas distintas. SOA é uma abordagem para o desenho de sistemas - o projecto arquitectural - que direcciona como os recursos de Tecnologias de Informação devem ser integrados e quais os serviços que devem ser expostos para o uso. Em contraste, os serviços web são uma metodologia de implementação que utiliza padrões e protocolos de linguagem específicos para executar uma solução de Arquitectura Orientada a Serviços.

3.6 - Métodos para a implementação de SOA

Os métodos para a construção de aplicações orientadas a serviços diferem dos métodos tradicionais de desenvolvimento de sistemas, em que projectistas desenvolvem software basicamente empregando alguma variação do ciclo editar-compilar-ligar. No modelo da Arquitectura Orientada a Serviços o software a desenvolver é descrito em função de um serviço oferecido e não mais das necessidades dos utilizadores. Os serviços são configurados para atenderem um conjunto específico de necessidades num determinado momento e serem executados e descartados a seguir - visão de um serviço instantâneo. Esta é uma mudança significativa de visão que pode trazer alguma confusão ao projectista.

3.7 - SOA aplicada a Redes de Transportes Inteligentes

O trabalho a desenvolver tem como um dos objectivos propor uma arquitectura baseada em serviços para uma rede V2V aplicada a ambientes urbanos.

O conceito de Arquitectura Orientada a Serviços já foi aplicado a redes de transportes inteligentes. A título de exemplo, na “2005 IEEE International Conference on Services Computing” foi apresentado o STISAG (*Shanghai Transportation Information Service Application Grid*) [14] um sistema de transportes inteligentes que é usado para lidar com os sérios problemas de congestionamentos na cidade. É baseado em SOA, *Grid* e tecnologias *Web services*. O sistema de desenho e alguns dos principais problemas técnicos são discutidos no documento apresentado nesta conferência. O projecto STISAG não só representa um caso de estudo de SOA, como também fornece uma nova e eficaz solução para ITS. Outros projectos em investigação, quer a nível académico quer a nível de grandes empresas, têm seguido esta hipótese de projectar redes de transportes inteligentes com base em SOA, devido ao facto de as características deste tipo de arquitectura de software se enquadrarem bem nos requisitos estruturais das redes de transportes inteligentes.

Capítulo 4

Redes de Comunicação Veículo-a-Veículo

Neste capítulo é descrito todo o estudo feito sobre redes de comunicação veículo-a-veículo, de onde foram posteriormente retirados os conceitos inerentes a este tema e que serviram de base para a realização do trabalho apresentado.

4.1 - Definição e Contextualização

“*Vehicle-to-vehicle communication networks*”, ou ainda, em português, redes de comunicação veículo-a-veículo, são, tal como o nome indica, redes formadas por vários veículos equipados com dispositivos de comunicação sem fios de curto alcance que podem comunicar entre si.

Comunicações entre veículos têm-se revelado um dos campos de interesse das telecomunicações que mais rapidamente cresceu. Veículos equipados com dispositivos de comunicação sem fios de curto alcance podem formar um tipo especial de rede *Ad-hoc* móvel com características particulares, denominada “*Vehicular Ad-hoc Network*” (VANET).

Uma das maiores características da comunicação V2V é revelar-se como um sistema de ajuda ao motorista. O sistema pode transmitir a localização do seu veículo e monitoriza a posição de centenas de outros igualmente equipados. A título de exemplo, o sistema desenvolvido pela GM Motors transmite a localização do veículo e monitoriza a posição de outros num raio de 300 metros, três vezes mais longe que um radar tradicional, 10 vezes por segundo. É o mesmo sistema de aproximação utilizado na aeronáutica, que por definição deve evitar que uma aeronave possa colidir com outra no ar [15].

O V2V “visualiza”, num raio em redor do veículo, outros veículos, podendo comunicar a sua posição, velocidade, direcção e outras características.

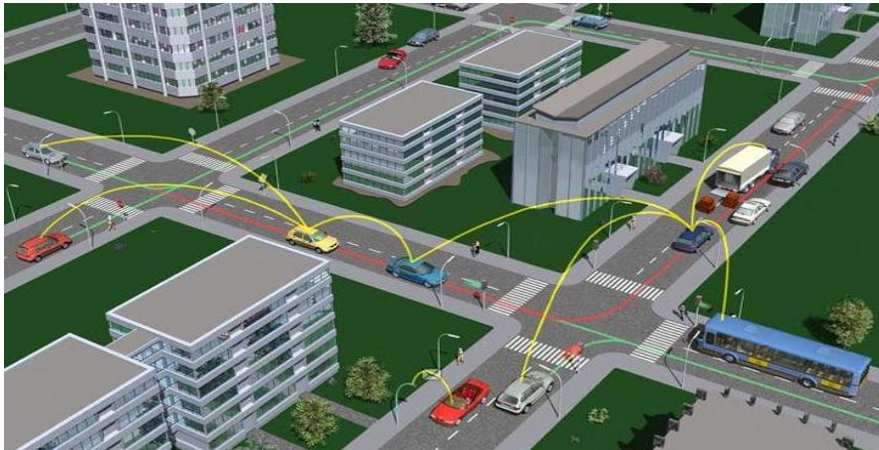


Figura 4.1 - Exemplo de uma rede V2V [16]

A existência de redes V2V, e a disseminação de informação através das mesmas, terão um papel importante na segurança e eficiência das redes de tráfego num futuro próximo. Tais redes inserem-se, assim, no contexto dos ITS (*Intelligent Transportation Systems*) e na iniciativa “*Intelligent Car*”. [17]

4.2 - Conceitos relacionados

Na secção anterior deste relatório foram apresentados alguns conceitos inerentes a definição das redes V2V que convém esclarecer.

4.2.1 - ITS (*Intelligent Transportation System*)

Sistemas inteligentes de transporte são projectos que visam integrar as mais modernas e recentes tecnologias de comunicação e informação existentes em sistemas de gestão de transportes, a fim de optimizar a eficiência funcional, energética e segurança das redes de tráfego nos grandes centros urbanos.

ITS estão-se a tornar cada vez mais complexos, englobando várias tecnologias como a comunicação V2V, V2R (*vehicle-to-road*), V2I (*vehicle-to-infrastructure*), arquitecturas de comunicação, dispositivos de detecção, comunicação via satélite, entre outros.

Hoje em dia o maior enfoque nesta área é mesmo a incorporação da comunicação V2V aos ITS.

A nível europeu, a implementação dos ITS encontra-se relativamente atrasada em relação ao Japão e EUA. Contudo várias conferências, normas europeias e bolsas de investigação têm focado a sua atenção nos ITS, que estão a sofrer uma rápida evolução e são já uma grande área de investigação a nível europeu. Existem vários grupos de trabalho que investigam o campo das VANET devido as suas possíveis aplicações e contributo para os ITS. Na Europa

destaca-se o consórcio C2CCC (European Car-to-Car Communication Consortium). O IEEE em parceria com o C2CCC encontra-se a desenvolver o protocolo IEEE 802.11p. Este protocolo consiste numa adaptação do IEEE 802.11 para cenários que englobem VANET, sendo equivalente ao DSRC utilizado nos Estados Unidos [18]. É também conhecido por WAVE (wireless Access in Vehicular Environments).

Uma possível ilustração de um ITS é apresentada na figura 4.2. Nesta figura é possível observar a complexidade para que tendem os ITS, ilustrando-se os diferentes tipos de comunicação e infra-estruturas envolvidas.

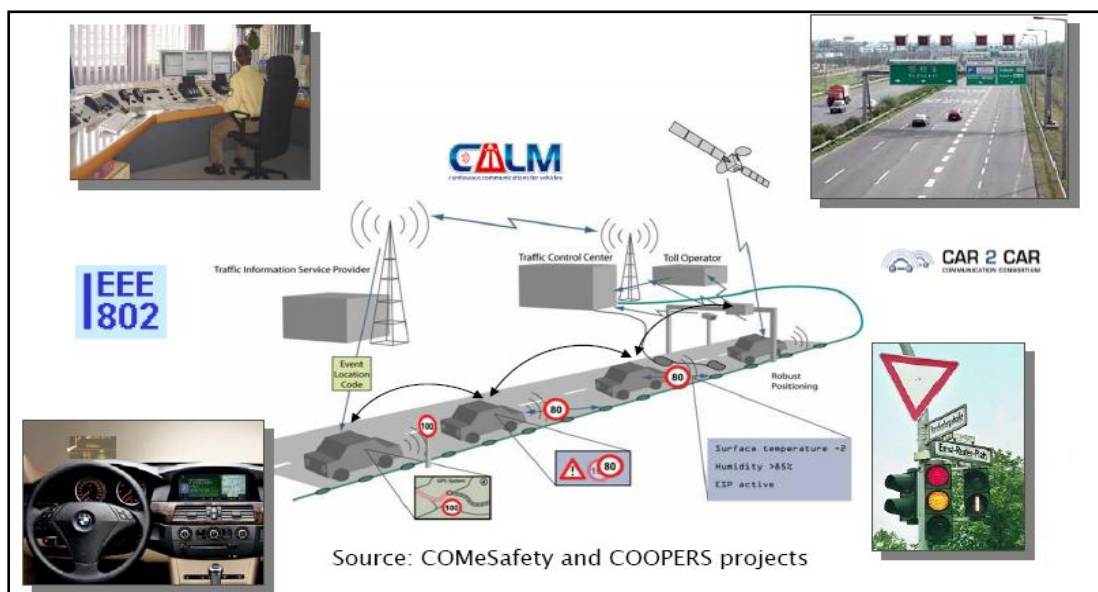


Figura 4.2 - Intelligent Transportation System (ITS) [17]

4.2.2 - Intelligent Car

Intelligent Car, em português, Carro inteligente, é um veículo dotado de inteligência artificial.

Intelligent car é uma das iniciativas da i2010 “European Information Society 2010 for growth and employment”. A i2010 tem como meta promover o desenvolvimento de veículos mais inteligentes, seguros e menos poluentes, com base nas tecnologias de informação e comunicação.

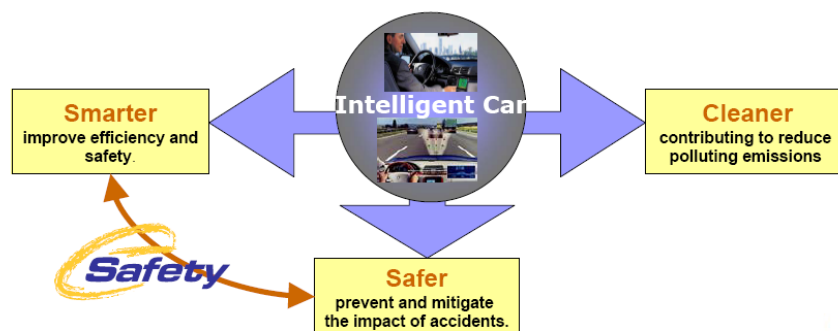


Figura 4.3 - Intelligent Car [17]

4.2.3 - Vehicular Ad-hoc Network (VANET)

Tal como foi referenciado na secção 3.1, as redes V2V podem formar um tipo especial de redes móveis *Ad-hoc* denominadas VANET (*Vehicular Ad-hoc Networks*).

VANET são um tipo de MANET (*Mobile Ad-Hoc Network*) que promovem comunicação entre veículos. Sendo assim, as VANETs herdam algumas características das MANET, mas contêm também características próprias que as diferenciam das restantes redes *Ad-hoc* móveis, tais como (baseado em [19]):

- Alta mobilidade, nós da rede (veículos) estão quase sempre em constante movimento;
- Rede aberta e de topologia altamente dinâmica, devido às velocidades relativas entre os veículos. Consequentemente, há uma alteração constante na vizinhança dos nós, ou seja, o grupo de veículos com que se pode interagir em cada momento. Como resultado, a rede fragmenta-se frequentemente e a sua redundância é comprometida;
- Limitada, devido ao facto de os veículos tenderem a executar movimentos não aleatórios, por estarem condicionados a determinados trajectos, como as vias rodoviárias;
- Potencialidade para atingir grandes escalas;
- Particionada;
- Todos os nós são provedores, encaminhadores e consumidores de dados;
- A transmissão sem fios está sujeita a interferências ainda pouco estudadas, principalmente devido às altas velocidades relativas à obstrução momentânea causada por prédios e outros obstáculos.
- Não existem limitações rígidas quanto à fonte de energia;
- Qualquer serviço disponibilizado através de comunicação entre veículos deve ser de carácter complementar e não-crítico, ou seja, a condução (e/ou fluxo) dos veículos deve ser possível também sem este serviço.

Estas características tornam as VANET suficientemente diferentes das outras redes e afectam significativamente as suas propriedades. Por exemplo, em [20] demonstra-se que o movimento dos veículos tem um efeito significativo na latência de entrega das mensagens.

As VANET podem ser de dois tipos: redes puramente sem fios, veículo-a-veículo, ou redes mistas, promovendo a comunicação entre veículos e entre veículos e equipamentos próximos. As redes sem fios puras, veículo-a-veículo, têm a enorme vantagem de não necessitarem de outras infra-estruturas além dos dispositivos de comunicação sem fios de curto alcance instalados nos veículos, o que diminui consideravelmente o seu custo de implementação e de manutenção. Outro aspecto que contribui para esse baixo custo é o facto das tecnologias sem fios de curto alcance (como a 802.11b/g [21]) não terem qualquer custo associado, além do custo dos próprios dispositivos.

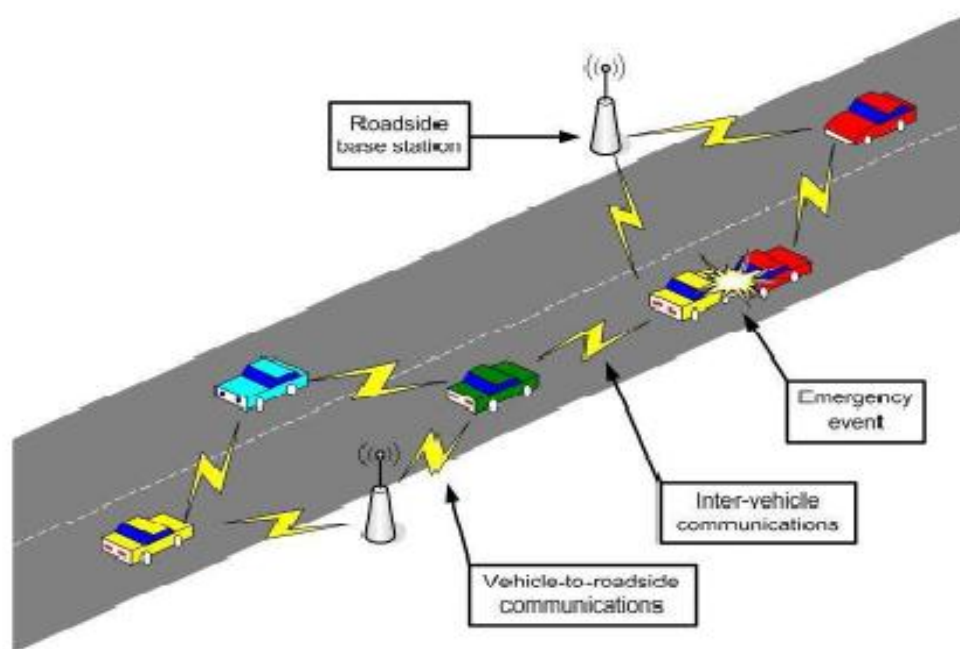


Figura 4.4 - VANET mista incorporando comunicações V2V e V2I [22]

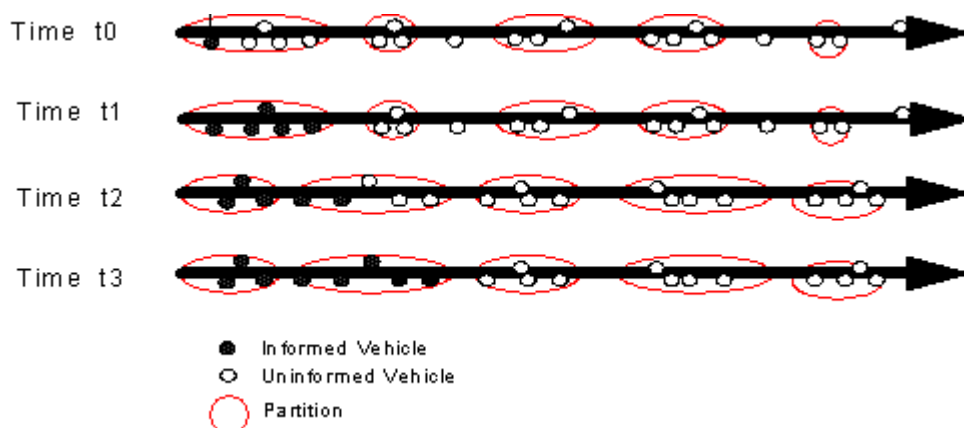


Figura 4.5 - Disseminação de informação numa VANET pura V2V [23]

Os principais objectivos de uma VANET são, nomeadamente, promover segurança e conforto dos passageiros dos veículos, bem como um planeamento eficiente das rotas de tráfego. Várias aplicações a serem implementadas em VANET encontram-se já em desenvolvimento, para atingir esses objectivos bem como outros. Exemplos dessas aplicações incluem o aviso/evitamento de colisões, monitorização de tráfego e vias, aviso de aproximação de veículos prioritários, detecção e aviso de sinais, acesso a internet, assistência de viagem, assistência turística e entretenimento. Exemplos de aplicações seguras em VANET incluem avisos de colisão e outros avisos sobre a segurança do veículo. Aplicações

não-seguras incluem informações sobre o estado do tráfego (congestionamentos), entretenimento móvel entre outras. Ou seja, denominam-se de aplicações seguras aquelas que empregam métodos de manutenção de segurança do motorista/passageiros do veículo. A segurança, portanto, refere-se não à aplicação em si, mas ao ser humano [22].

A maioria das aplicações a serem desenvolvidas em VANETs requer um tipo de disseminação de dados. Tal necessidade revela-se um tópico desafiador devido às características particulares dessas redes. Alguns estudos [24] demonstram que os protocolos de routing clássicos não se adequam bem às redes veiculares devido, nomeadamente, à alta mobilidade dos nós. A necessidade de criar protocolos próprios para as VANET tornou-se assim uma necessidade preponderante. Na secção seguinte, serão apresentados alguns dos protocolos para VANET.

4.2.3.1 - Protocolos criados para VANETs

A descrição dos protocolos apresentada nesta subsecção foi adaptada de [25].

- **DSRC**

DSRC é a sigla usada para *Dedicated Short Range Communications* (Comunicações Dedicadas em Curto Alcance). Este protocolo foi desenvolvido para oferecer comunicações entre veículos e entre veículos e equipamentos próximos. O uso mais conhecido de DSRC é o pagamento electrónico de estacionamento. Contudo, o futuro almeja aplicações mais ousadas, tais como evitar colisões, avisar os motoristas da proximidade de veículos prioritários (por exemplo ambulâncias), inspecção de segurança em veículos, entre outros.

O protocolo possui alta taxa de dados, até 27Mbps, de alcance até 1km, num padrão multicanal baseado no IEEE 802.11/802.11^a. Este protocolo já se encontra implementado nos EUA em um espectro licenciado pelo governo de 75MHz em 5.9GHz, para aplicações em ITS.

Embora este protocolo baseie a sua camada física no IEEE 802.11a, o DSRC diferencia-se deste sobretudo por operar numa banda já dita licenciada, além de permitir aplicações adaptadas às altas velocidades dos veículos (até 190km/h) e 7 canais de 10MHz cada para suporte a aplicações seguras e não seguras.

A banda de 5.9GHz possui sete canais de 10MHz, que incluem um controlo de canal e seis controlos de serviços. O DSRC, que envolve comunicações V2V (Veículo-a-veículo) e V2I (Veículo-a-infra-estrutura), deverá assim suportar aplicações seguras e não-seguras. No entanto, a prioridade é dada às aplicações seguras. Isto deve-se ao facto de se supor que aplicações seguras em VANET devam ser as responsáveis por assegurar vidas, alertando os condutores de eventuais situações.

A camada física do DSRC usa modulação OFDM (*Orthogonal frequency-division multiplexing*) para a multiplexação dos dados.

- VITP

O VITP, *Vehicular Information Transfer Protocol*, é um protocolo de aplicação que especifica a sintaxe e semântica de mensagens de consulta, sensíveis ao local, entre nós das VANET. É uma proposta feita por *Dikaikos et al* no *2nd ACM internacional workshop on VANET*.

Entende-se por “sensível ao local” a possibilidade de fornecer explicitamente a localização do pedido. No contexto de veículos, que têm movimentação limitada às malhas rodoviárias, pode-se assumir que estas são o domínio geográfico.

Um cliente VITP instalado em computadores incorporados em veículos seria responsável pela gestão do funcionamento do protocolo.

Um esquema de codificação de localidades organizaria e representaria simbolicamente segmentos rodoviários. Este esquema possibilitaria as consultas sensíveis ao local, e pelo suporte de outros protocolos geográficos que fazem uso de serviços *on-board*, transformaria as representações simbólicas em coordenadas GPS.

Algumas características a implementar no protocolo visam permitir optimizações de desempenho (*caching* de mensagens, redução de tráfego VITP), assegurar qualidade para resultados VITP e a protecção da privacidade dos condutores.

Diferentemente dos sistemas de monitorização actuais, baseados em disseminação contínua de condições de tráfego via VANET, VITP propõe recuperação de tráfego em *pull-based*, accionado por pedidos de consultas sensíveis ao local feitas por veículos munidos de VITP. O protocolo propõe ainda a propagação de mensagens por *push-based*, mecanismo para disseminar vários alertas que contêm informação sobre emergências e outras ocorrências.

Como a centralização de servidores de dados em redes móveis, em especial veiculares, é inviável, em termos de custo e escalabilidade, o VITP não assume nenhuma infra-estrutura além de veículos e equipamentos de acostamento (denominação normalmente usada para referenciar as infra-estruturas físicas de apoio que se encontram junto às vias de tráfego). Assim, o servidor que fornece respostas a consultas VITP é uma colecção dinâmica de clientes VITP, que ora constituem um veículo que se move dentro da área de localização do alvo da consulta, ora é um veículo capaz de contribuir com o resultado da consulta a partir de informações armazenadas na memória local dos seus dispositivos. Essa colecção de veículos que se estabelece forma uma rede *Ad-hoc*, e depende da VANET constituída na região entre a consulta e seu resultado. À alocação dinâmica de utilizadores de VITP que estão na localização-alvo de uma consulta VITP e influenciam no diagnóstico desta, chama-se *Virtual Ad-Hoc Server*, VAHS. Note que esta colecção de clientes VITP constituirá uma estrutura de melhor-esforço. Ou seja, não fornecerá garantias da recuperação de mensagens perdidas.

- PAVAN

O PAVAN é um protocolo de monitorização de disponibilidade de conteúdos de média, projectado para entretenimento intraveicular, em dispositivos C2P2 (Car-to-Car / Peer-to-Peer), É uma proposta de *Ghandeharizade et al.*, em artigo submetido ao *1st ACM international workshop on VANETs*, 2004.

Dispositivos C2P2 podem formar uma rede *ad-hoc* para troca de áudio e vídeo, para aplicações como vídeo sob pedido (on-demand). Um sistema C2P2 pode actuar sob três papéis, simultaneamente:

- Exibir um título (de conteúdo de média);
- Fazer *streaming* de um clipe armazenado localmente para outro dispositivo C2P2;
- Encaminhar dados entre dispositivos C2P2.

4.2.3.2 - Protocolos herdados das MANETs

As VANET são um subconjunto das MANET. Vários estudos já realizados, a título de exemplo [26], avaliaram a possibilidade de utilizar protocolos de encaminhamento das redes MANET para simulação de redes VANET, apesar das características específicas destas redes. O IETF (Internet Engineering Task Force) [27] está a impulsionar um grupo de trabalho para a sua standardização. Na actualidade, existem mais de 80 protocolos de encaminhamento. A maior parte está em fase de investigação e, dentro do IETF, apenas alguns protocolos são escolhidos como candidatos, nomeadamente:

- AODV
- OLSR
- TBRPF
- DSR

Assim era fulcral para a realização desta tese estudar também com detalhe os protocolos de encaminhamento das redes MANET.

As MANET, devido a sua topologia, possuem já um vasto leque de protocolos propostos.

Existem várias formas de classificar os protocolos de encaminhamento deste tipo de redes. Apresenta-se primeiramente um esquema destas para depois entrar-se em detalhe:

- Segundo o alcance: unicast e multicast
- Segundo o esquema de descobrimento das rotas: pró-activos, reactivos e híbridos
- Segundo o tipo de algoritmo em que se baseiam: Vector de Distância (*Distance Vector*), Estado de Enlace (*Link State*), com ajuda de informação geográfica e baseados em zonas.

Os protocolos unicast são aqueles que transmitem informação de um único emissor a um único receptor. Em contrapartida, os protocolos multicast são aqueles em que a informação se envia a um grupo, previamente criado, de nós. Existem dois casos particulares: Broadcast, em que se envia a informação a todos os nós de uma rede e o anycast, em que o destinatário é único mas não é especificado. Um caso especial de multicast de grande utilidade nas VANET é o geocast, que consiste em enviar informação desde um emissor para um grupo de receptores situados numa certa área geográfica. Neste caso não é preciso definir previamente o grupo de nós receptores, só pela sua posição um nó receberá os pacotes enviados.

Atendendo ao esquema de descobrimento das rotas, as MANET são classificadas em dois tipos diferentes. Os algoritmos pró-activos são aqueles que periodicamente actualizam as rotas da rede, a todo o momento têm conhecimento da topologia da rede, tentando ter as rotas optimizadas para o momento em que seja necessário enviar informação. A troca de pacotes de controlo e actualização das tabelas de encaminhamento é contínua. Todos os nós são conhecidos à partida. Em oposição a estes, existem os protocolos reactivos (também

conhecidos por *on-demand*) que somente descobrem a rota até a um destino, no momento em que necessitam de enviar informação. Nestes o descobrimento de rotas é feito por pedido, não existindo assim rotas permanentes. Os protocolos híbridos combinam os dois esquemas utilizando, por exemplo, pró-actividade na vizinhança do nó mas fazendo a descoberta das rotas até aos nós distantes somente no momento em que são necessárias. O melhor mecanismo de descoberta de rotas a aplicar depende de cada cenário e padrão de mobilidade da rede. Contudo em qualquer caso há que chegar a um compromisso entre a actualização das rotas, o *overhead* e a latência no descobrimento.

Quanto ao tipo de algoritmo em que os protocolos de encaminhamento se baseiam, podem-se encontrar vários grupos. Nos baseados em vector de distâncias (*Distance vector*) todos os nós da rede intercambiam periodicamente os seus vectores de distância, com seus custos associados. Ou seja, cada nó tem a informação dos custos para cada destino. Devido à sua aplicabilidade para encaminhamento de pacotes na Internet, ficou conhecido como *Routing Information Protocol* (RIP) ou *Distributed Bellman-Ford* (DBF). Cada *router* (nó) mantém uma tabela (um vector) contendo o menor caminho conhecido para cada destino. Essas tabelas são actualizadas realizando troca de mensagens com os vizinhos, através da comparação da tabela recebida com a sua tabela. Caso apresente uma melhor rota, o *router* actualiza a sua tabela e armazena também a origem desta informação.

Nos algoritmos baseados no estado de enlace (*link state*), os nós monitorizam o estado de enlace com os seus vizinhos e difundem essa informação. Cada nó conhece o estado de todos os enlaces, determina a topologia completa da rede e determina localmente o caminho mais curto.

Nos algoritmos que utilizam informação geográfica fazem-se cálculos de onde se estima que poderão estar os nós e encaminha-se a informação nessa direcção.

Nos algoritmos iniciados na fonte (*source routing*) uma rota só é criada quando o nó fonte precisa. Após a ser criada a rota, é mantida por um protocolo de manutenção. Não há necessidade de actualizações periódicas.

Na figura 4.6 são descritos os protocolos que serão estudados bem como a sua classificação de acordo com o que foi descrito:

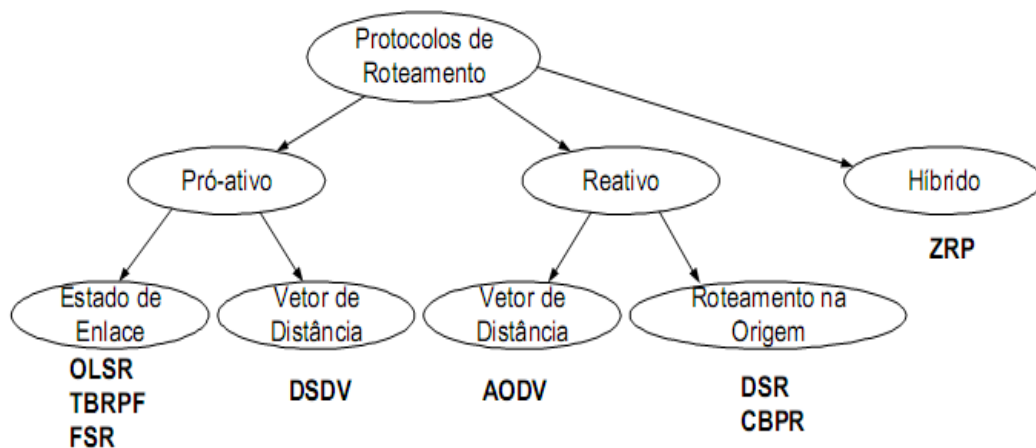


Figura 4.6 - Classificação dos protocolos de encaminhamento das redes MANET

Em seguida analisam-se alguns dos principais protocolos. Textos adaptados de [28].

Destination-Sequenced Distance-Vector (DSDV)

O protocolo DSDV é baseado no algoritmo vector distância, ou seja, cada nó da rede possui uma tabela com as informações que serão enviadas, por *broadcast*, e também uma tabela de encaminhamento com todas as rotas para cada um dos nós da rede e a quantidade de saltos para alcançar cada destino.

A tabela de encaminhamento é preenchida através de um campo chamado *destination sequence number* e o valor desse campo é informado pelo nó de destino durante o processo de descoberta da rota. Na tabela é realizada a manutenção da rota através do envio de mensagens periódicas por cada nó, contendo informação das alterações que ocorreram nas suas tabelas devido às mudanças na topologia da rede. As mensagens para actualização de rotas são de dois tipos:

- Mensagens curtas: contendo apenas as últimas rotas que sofreram alguma modificação;
- Mensagens Completas: contendo toda a informação da tabela de encaminhamento, gerando com isso uma grande quantidade de tráfego. Para evitar uma sobrecarga da rede, essas mensagens devem ser enviadas com frequência relativamente baixa.

Ao realizar uma mudança na rede, como a perda de um determinado *link*, o nó que se apercebe dessa mudança altera a entrada da sua tabela para essa rota, indicando uma quantidade de saltos igual ao maior valor possível para esse campo. Altera também o valor do *destination sequence number*, sendo este o único caso onde a alteração desse campo é feita por um nó que não é o próprio destino. Devido à sua importância, essa alteração é imediatamente propagada na rede pelo nó que primeiro se apercebeu, ou seja, a mensagem de actualização não aguarda o momento em que as rotas modificadas são periodicamente disseminadas.

Tabela 4.1 - Exemplo da tabela de encaminhamento utilizada pelo Protocolo DSDV [28]

Destino	Próximo Salto	Métrica (Salto)	Nº Seqüência	Tempo	Estab. Dados
A	A	0	A-846	1000	Ptr-A
B	B	1	B-470	1200	Ptr-B
C	B	3	C-920	1500	Ptr-C
D	B	4	D-502	1200	Ptr-D

Na tabela 4.1apresenta-se um exemplo de uma tabela de encaminhamento utilizada pelo protocolo DSDV, sendo os seus campos descritos abaixo.

- O campo Destino aponta os possíveis nós para o envio de informação na rede;
- O Próximo Salto indica para qual nó vizinho deve ser enviada a informação que segue para um destino;

- A Métrica indica a quantos saltos este destino se encontra;
- O número de Sequência informa o número criado pelo destino para garantir a ausência de loops;
- O Tempo de Registro serve para decidir o que deve ser descartado ou não;
- A Estabilidade dos Dados aponta para a tabela que guarda a informação que indica o quanto a rota é estável.

Optimized Link State Routing Protocol (OLSR)

Trata-se de uma otimização do protocolo *link state*. O OLSR reduz o tamanho do pacote de controlo assim como o número desses pacotes enviados para a rede. Essa redução no número de pacotes de controlo deve-se à utilização de Multipoint Relays (MPR), que caracterizam o protocolo OLSR.

Um MPR é um nó escolhido entre os vizinhos para enviar os pacotes de controlo, essa escolha é realizada pelos vizinhos que estão apenas a um *hop* (salto) do nó. As figuras 4.7 e 4.8 ilustram o funcionamento de uma rede através do algoritmo de inundação normal e do protocolo OLSR com a utilização dos MPR, representados na figura pelos nós azuis. Essa característica mostra a sua grande aplicabilidade em redes bastante densas. O nó escolhido como MPR passa sempre as informações da rede a todos que o escolheram, ou seja, repassa a visão de todos os componentes do grupo. O nó para ser considerado como MPR precisa saber o estado do *link* de todos os seus vizinhos que se apresentem na distância de 1 e 2 *hops* (saltos) para que possa ser eleito através de políticas de eleição de grupo.

O OLSR é um protocolo pró-ativo que utiliza dois tipos de pacotes de controlo, o *hello* e o *Topology Control (TC)*. O pacote *hello* é usado para a construção da vizinhança dos nós, é enviado para todos os nós situados a um salto de distância, servindo também para a eleição do MPR. O pacote TC é enviado a todos os nós da rede através de broadcast e contém uma lista com o MPR do grupo. É usado um número de sequência neste pacote para evitar a retransmissão infinita do mesmo, e um outro campo verifica a actualização do referido pacote. O pacote *hello* também é enviado por broadcast periodicamente para a verificação do estado dos enlaces.

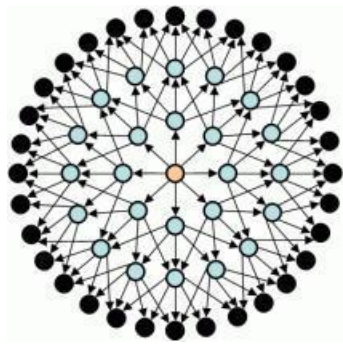


Figura 4.7 - Funcionamento do algoritmo de inundação [28]

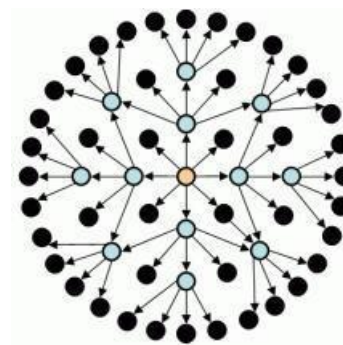


Figura 4.8 - Funcionamento com os *Multipoint Relays* [28]

Dynamic Source Routing (DSR)

É um protocolo de encaminhamento reactivo. Apresenta como principal característica a utilização do encaminhamento por fonte (*source routing*).

O processo de descoberta da rota do nó de origem funciona como mostrado a seguir: o nó de origem envia através de difusão (*broadcasting*) para seus vizinhos um pacote de requisição de rota (RREQ), como mostrado na figura 4.9. Essa mensagem contém o endereço de origem, o destino da comunicação e o registo de rota. Cada nó após receber este pacote verifica na sua *cache* se possui uma rota para o nó de destino. Caso possua a rota, envia ao nó origem um pacote de resposta contendo a sequência de todos os nós até o destino e se não possui a rota, insere o seu endereço no registo da rota e envia-a também por difusão para os seus nós vizinhos.

Quando o primeiro pacote RREQ chega ao destino, é enviado um pacote RREP, conforme mostrado na figura 4.10, informando o caminho da rede (rota) que o pacote deverá percorrer. O nó de origem inicia, assim, o envio dos dados, já sabendo o caminho que os mesmos irão percorrer. Para além disso, os nós intermédios, armazenam em cache as rotas que já foram descobertas, podendo ser utilizadas novamente sem a necessidade de descobrir a rota novamente.

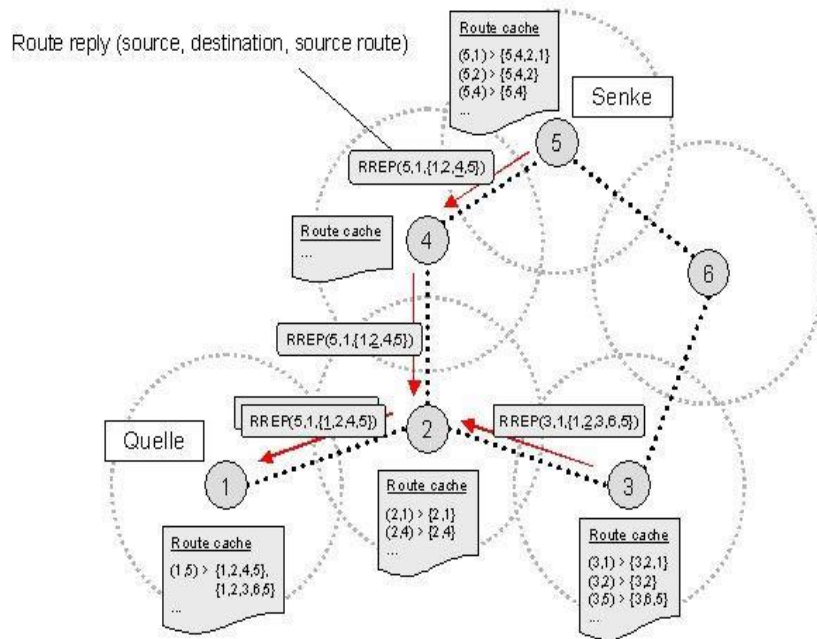


Figura 4.9 - Funcionamento da mensagem RREP [28]

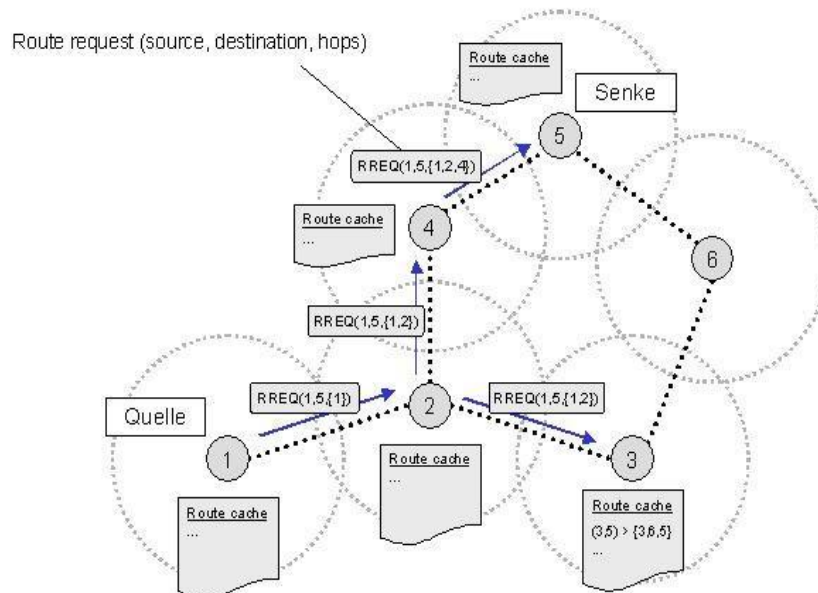


Figura 4.10 - Funcionamento da mensagem RREP [28]

Ad-Hoc On-Demand Distance Vector (AODV)

Trata-se também de um protocolo reactivo, ou seja, a rota para um nó destino só é descoberta quando se deseja enviar um pacote (dados) para este nó. Este protocolo permite encaminhamento dinâmico, onde a rota do pacote pode ser alterada de acordo com a rota em que o dado se está a deslocar caso a rota utilizada esteja indisponível, realizando a descoberta de forma rápida para novos destinos.

O AODV é um protocolo baseado no protocolo *Destination-Sequenced Distance-Vector (DSDV)*. Foi criado basicamente para eliminar os erros do DSDV, como o problema de contagem ao infinito. Este problema deve-se à mudança constante da topologia e à troca em grande número de mensagens de controlo entre os componentes da rede.

Durante a descoberta da rota, o protocolo AODV utiliza como mecanismo de armazenamento uma tabela de encaminhamento tradicional, que armazena apenas uma entrada, ou seja, armazena apenas o próximo salto para o destino. Diferencia-se assim do DSR que armazena múltiplas rotas para um mesmo destino, e também armazena a rota completa da origem ao destino.

O AODV foi projectado para ser utilizado em redes *Ad-hoc* que apresentem desde um pequeno número de nós até milhares deles. O objectivo principal do protocolo é adaptar-se rápida e dinamicamente às variações das condições dos enlaces da rede, descobrindo rotas de forma a proporcionar um QoS desejável, evitando o desperdício de banda, minimizando o uso de memória e processamento nos nós que actuam como *routers*.

Este tipo de protocolo pode ser utilizado em cenários de baixa, média e alta mobilidade. Lida com uma grande variedade de níveis de tráfego de dados, adaptando-se dinamicamente.

O AODV apresenta quatro tipos de mensagens:

1. Route Request (RREQ);
2. Route Reply (RREP)
3. Route Error (RERR)
4. Route Reply Acknowledgment (RREP-ACK)

RREQ funciona de forma similar à mensagem RREQ do DSR. Contudo a conexão entre os nós é bidireccional e simétrica. O nó de origem envia o RREQ aos nós no seu alcance e os nós enviam o RREP ao nó de origem. Com isso a rede aumenta o seu tráfego de pacotes, pois toda requisição terá uma resposta.

Formato das Mensagens:

- Mensagem de Requisição de Rota (*Route Request - RREQ*)

A mensagem Route Request é uma mensagem de requisição de rota enviada através de broadcast. Realiza a disseminação do pedido de rota para os nós da rede para que ocorra a transmissão de um pacote entre um nó de origem e um nó destino. A seguir é mostrado na figura 4.11 o formato do datagrama da mensagem RREQ que possui um total de 192 bits no seu tamanho original.

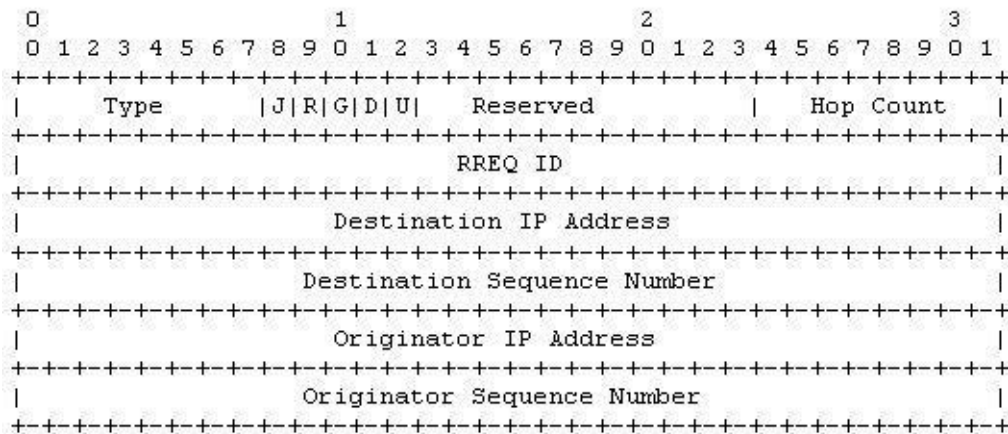


Figura 4.11 - Formato do datagrama da mensagem RREQ [28]

Os campos que constituem a mensagem são os seguintes:

1. *Type*: 1 - mensagem RREQ;
2. J: Reservado para multicast;
3. R: Reservado para multicast;
4. G: Indica se o RREP deve ser unicast ao nó especificado no campo *Destination IP Address*;
5. D: Indica que somente o destino pode responder ao RREQ;
6. U: Número de sequência desconhecido. O número de sequência do destino é desconhecido;
7. *Reserved*: Emitido como 0 (zero), deve ser ignorado na recepção;
8. *Hop Count*: O número de saltos da fonte até ao nó que contém a mensagem de requisição.
9. RREQ ID: Número de sequência que identifica o RREQ, juntamente com o endereço IP (*Internet Protocol*) do nó de origem.
10. *Destination IP Address*: O endereço IP do destino da rota desejada.
11. *Destination Sequence Number*: O último número de sequência recebido de algum nó para a descoberta da rota.
12. *Originator IP Address*: O endereço IP do nó que originou o pedido de rota.
13. *Originator Sequence Number*: Número de sequência actual a ser usado na entrada da rota e que aponta para nó de origem do RREQ.

- Mensagem de Resposta da Rota (*Route Reply - RREP*)

A mensagem *Route Reply* é uma mensagem de controlo. Quando um nó recebe um RREQ e ele é o destino ou possui informações sobre o destino em sua tabela de encaminhamento, ele envia um RREP *unicast* com a rota até o destino. A seguir é mostrado na figura 4.12 o formato do datagrama da mensagem RREP.

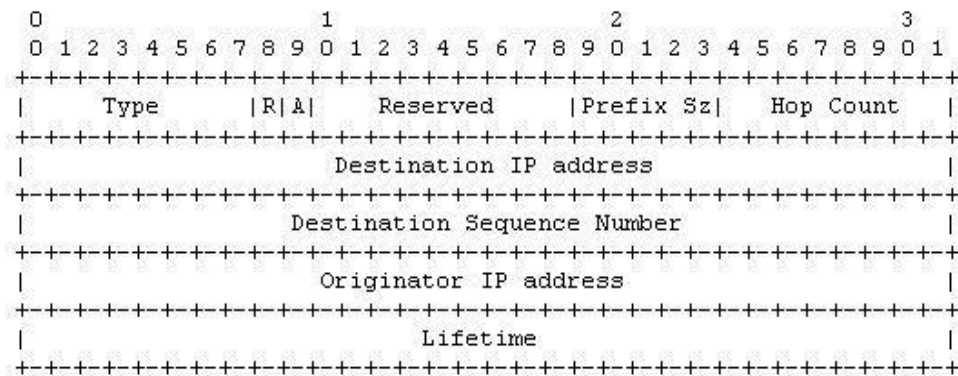


Figura 4.12 - Formato do datagrama da mensagem RREP [28]

Os campos que constituem a mensagem são os seguintes:

1. *Type*: 2 - mensagem RREP;
2. *R*: Usado para *multicast*;
3. *A*: Reconhecimento requerido (*Acknowledgment*);
4. *Reserved*: Emitido como 0 (zero), deve ser ignorado na recepção;
5. *Prefix Size*: Especifica o salto seguinte. Pode ser usado por todos os nós com o mesmo prefixo de encaminhamento;
6. *Hop Count*: Número de saltos do endereço IP de origem até ao endereço IP de destino;
7. *Destination IP Address*: O endereço IP do destino para a rota fornecida;
8. *Destination Sequence Number*: O número de sequência do destino que associou a rota;
9. *Originator IP Address*: O endereço IP do nó que originou o RREQ
10. *Lifetime*: O tempo, em milissegundos, para que os nós que receberam o RREP considerem a rota válida.

- Mensagem de Erro da Rota (*Route Error - RERR*)

A mensagem *Route Error* é uma mensagem do tipo requisição, que informa quando uma rota não está mais disponível devido a uma falha na rota. Um nó gera uma mensagem RERR e pode enviá-la a muitos precursores ou a apenas a um. Contudo, nos dois casos considera-se ser apenas uma mensagem de controlo, ou seja, um nó não pode gerar mais de uma mensagem de controlo por segundo. A seguir é ilustrado na figura 4.13 o formato do datagrama da mensagem RERR.

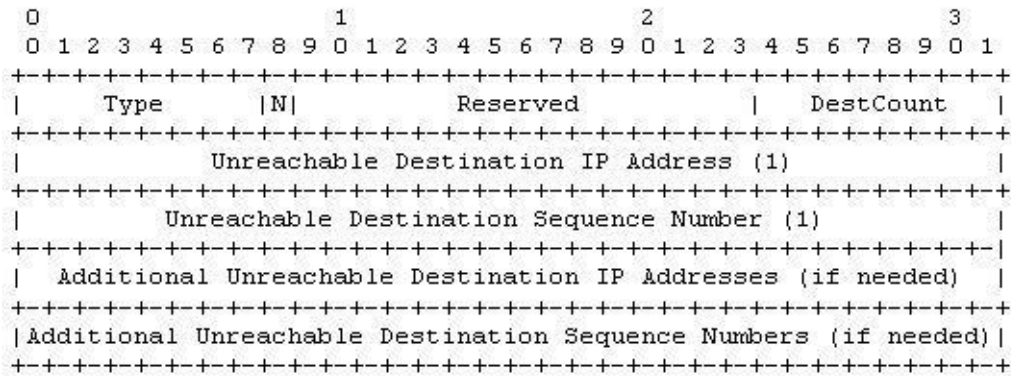


Figura 4.13 - Formato do datagrama da mensagem RERR [28]

Os campos da mensagem são os seguintes:

1. *Type*: 3 - mensagem RERR;
2. *N*: Não apagar a rota. Quando um nó executou um reparo em um *enlace*, os nós acima não devem apagar a rota;
3. *Reserved*: Emitido como 0 (zero), deve ser ignorado na recepção;
4. *DestCount*: Número de destinos incluídos na mensagem. Deve apresentar pelo menos 1;
5. *Unreachable Destination IP Address*: O endereço IP de destino é tornado inalcançável devido à ruptura de um enlace
6. *Unreachable Destination Sequence Number*: O número de sequência na entrada da tabela de encaminhamento para o destino está listado no campo de Endereços IP de destinos inalcançáveis.

- Mensagem de Reconhecimento da Resposta da Rota (*Route Reply Acknowledgment - RREP-ACK*)

A mensagem Route Reply Acknowledgment é uma mensagem de controlo que realiza a confirmação da recepção do RREQ pelo nó de origem. A seguir é mostrado na figura 4.14 o formato do datagrama da mensagem RREP-ACK.

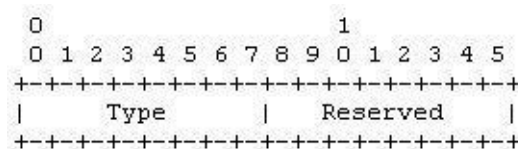


Figura 4.14 - Formato do datagrama da mensagem RREP-ACK [28]

Os campos para a mensagem RREP-ACK são os seguintes:

1. *Type*: 4 (mensagem RREP-ACK);
2. *Reserved*: Emitido como 0 (zero), deve ser ignorado na recepção.

Zone Routing Protocol (ZRP)

O ZRP trata-se de um protocolo híbrido. Procura usar as vantagens dos protocolos pró-activos e reactivos. Para a descoberta da vizinhança local, utiliza a característica pró-activa através do protocolo *Intrazone Routing Protocol (IARP)* e para a comunicação das vizinhanças utiliza técnicas reactivas através do protocolo *Interzone Routing Protocol (IERP)*. A comunicação ocorre com o protocolo *Broadcast Resolution Protocol (BRP)*, no qual é responsável pela formação do pacote de requisição de rota.

O IARP é um protocolo pró-activo baseado no protocolo estado do enlace onde cada nó monitoriza o seu estado de enlace, ou seja, monitoriza todos os seus vizinhos locais. Um nó envia um pacote de controlo a todos os seus vizinhos contendo informações dos seus enlaces, é também carregado no pacote um valor *time to live (TTL)* contendo um número igual a 1, o que significa que este pacote será apenas para os nós vizinhos. Cada nó ao receber o pacote de controlo decrementa o valor do TTL, assim esta variável armazenará o valor igual a zero. Quando um outro nó receber este pacote deve rejeitá-lo, pois o valor é igual a zero. Isto limita o pacote apenas a uma única zona. A seguir, na figura 4.15, é mostrado os principais componentes do protocolo ZRP, e como ocorre a comunicação entre eles e o protocolo IP (internet protocol).

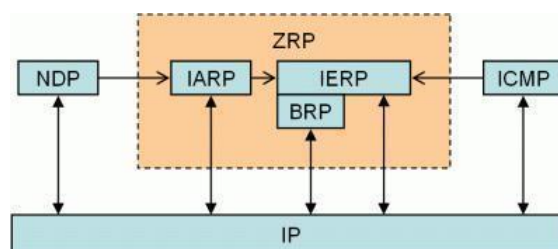


Figura 4.15 - Comunicação entre componentes do protocolo ZRP [28]

Fisheye State Routing (FSR)

O FSR é um melhoramento do protocolo *Global State Routing (GSR)* e ambos são baseados no algoritmo de estado do enlace. Contudo, o GSR utiliza um grande número de mensagens de actualização, o que ocupa bastante largura de banda. No FSR a mensagem de actualização não contém informações sobre todos os nós da rede, mas apenas informações dos vizinhos,

que são identificados pelo número de saltos (*hops*). A informação sobre actualização é trocada com mais frequência entre os nós mais próximos (*hop* = 1). O nó central possui a informação exacta sobre todos os nós do círculo interno, e quando a distância aumenta, a precisão da informação vai diminuindo. A visão do nó central pode ser mostrada na figura 4.16.

Como todos os nós têm informações dos seus vizinhos, os pacotes estarão distribuídos correctamente. Isto significa que o protocolo apresenta um *overhead* bem controlado.

O protocolo FSR é similar ao *Link State (LS)* porque mantém um mapa da topologia da rede em cada nó e, como já mencionado, a informação sobre o link é mais precisa nos nós a sua volta. Com isso o FSR pode imediatamente fornecer a informação da rota quando necessário.

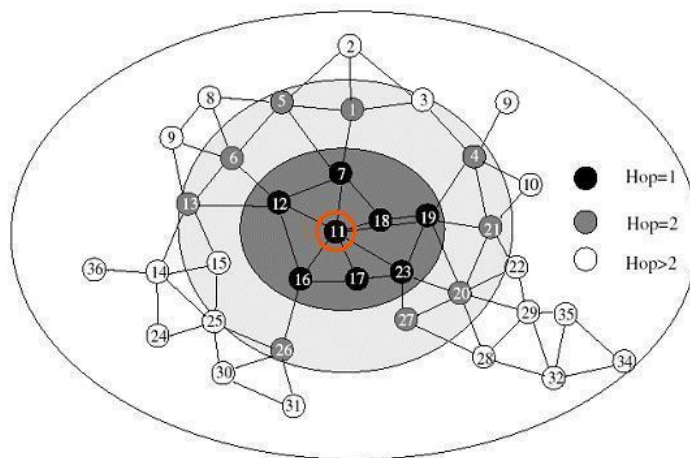


Figura 4.16 - Visão dos nós utilizando o protocolo FSR [28]

O FSR é apropriado para redes com elevado número de nós e ambientes altamente móveis, pois a informação de controlo é restrita ao número de saltos entre os nós, caracterizando-se como um protocolo robusto. Contudo,, o protocolo apresenta uma elevada complexidade de armazenamento em sua tabela de encaminhamento e exige grande quantidade de processamento.

Análise dos protocolos

De acordo com a literatura estudada e com conhecimento adquirido, pode-se concluir que para MANETs os protocolos Pró-activos poderão apresentar melhor desempenho, pois antes de haver a necessidade de envio de informação esses protocolos apresentam armazenados uma estrutura de como a rede se encontra organizada. Entretanto, estes apresentam algumas desvantagens como o problema de contagem ao infinito [29], onde um pacote pode percorrer por um longo período a rede. Já os protocolos reactivos não possuem alto desempenho,

porém controlam alguns problemas como o de contagem ao infinito e o de inundação de pacotes na rede, apesar de o seu desempenho ser inferior, visto que somente iniciam a comunicação quando ocorrer um evento, ou quando há necessidade de envio de informação.

Na tabela 4.2, a seguir, é mostrado um estudo comparativo entre os protocolos estudados anteriormente.

Tabela 4.2 - Comparação entre protocolos de encaminhamento de redes MANET (Adaptada de [28])

Propriedades	Protocolos					
	DSDV	OLSR	DSR	AODV	ZRP	FSR
Classificação	Pró-ativo	Pró-ativo	Reativo	Reativo	Híbrido	Pró-ativo
Protocolo de Origem	DBF	LS	-	DSDV	-	GSR (LS)
Flooding	-	-	-	x	-	x
Flooding Controlado	-	x	x	-	x	-
Livre de loops	x	-	x	x	x	x

4.2.4 - Trabalhos relacionados

A discussão apresentada nesta secção tem como base a discussão apresentada em [30].

Comunicações entre veículos têm-se revelado um dos campos de interesse das telecomunicações que mais rapidamente cresceu. Vários são os projectos de investigação nesta área. Um aspecto comum à maioria das propostas apresentadas é a disseminação de informação. Uma delas é o SODAD (*Segment-oriented Data Abstraction*) [31], que assume que cada veículo está equipado com uma mapa digital, onde as estradas são divididas em segmentos de tamanho conhecido. As informações sobre os segmentos são modeladas como tuplas e difundidas em *broadcasts* locais entre veículos. Dependendo da relevância da tupla esta é armazenada, retransmitida ou descartada.

Outra arquitectura [32] divide os veículos em aglomerados (*clusters*) dentro dos quais o encaminhamento de mensagens é pró-activo. A manutenção do estado de vizinhança (veículos alcançáveis sem encaminhamento) é feita por mensagens periódicas (*beacons*). A informação de tráfego é obtida por meio de requisição de tarefas, que especificam de que veículos e sensores se pretende obter informação.

Hao wu, richard Fujimoto e Randal Guensler procederam a vários estudos sobre *Vehicular Networks* aplicadas a sistemas de transporte urbano.

Sob liderança do Instituto de pesquisa Automobilística do Japão, o projecto *Probe Information System* [33] transmite as leituras dos mais de 120 sensores de cada veículo para uma estação central.

As limitações de uma arquitectura ITS centralizada motivaram trabalhos que propõem independência relativamente às infra-estruturas. Goel et al., [34], propõem a disseminação

das informações de tráfego cooperativamente entre os veículos, eliminando a necessidade das infra-estruturas de apoio. Uma publicação mais recente que também propõe a otimização de rotas de trânsito independentemente da infra-estrutura, foi apresentada em [31] que apresenta o *SOTIS (Self-organizing Traffic Information System)*

Em contraste com o paradigma da computação de rotas individualmente, a ideia de se usar uma *grade (grid) Ad-hoc* tem sido tema de publicações recentes. A computação em tempo real dos dados possibilitaria a direcção cooperativa, em [35] e [36] discutem-se algumas possibilidades nessa área.

Em [24] é apresentada uma discussão interessante sobre alguns dos principais desafios para o desenvolvimento de serviços baseados em comunicações V2V.

Vários trabalhos se desenrolaram na área de QoS , a título de exemplo [37].

Outras grandes fontes de informação são naturalmente as normas, conferências e papers e outras publicações do IEEE, bem como os vários seminários e conferências decorrentes num grande número de países.

Capítulo 5

Ferramentas de Simulação

Neste Capítulo são descritas as características que uma ferramenta de simulação deve possuir para simular uma rede V2V e é feito um estudo comparativo entre algumas ferramentas de simulação estudadas.

5.1 - Definição e Contextualização

A evolução das aplicações e protocolos associados às VANET pode ser estudada através de modelização de redes de tráfego reais, que devem envolver um número elevado de nós de forma a estudar a evolução do desempenho das redes V2V em ambientes urbanos. Contudo, realizar experiências e estudos em tão grande escala revela-se uma tarefa extremamente difícil e dispendiosa. Assim a simulação revela-se uma ferramenta indispensável.

A simulação de redes V2V requer dois componentes diferentes: um simulador de redes de comunicação, capaz de simular as propriedades de uma rede sem fios e um simulador de tráfego veicular, capaz de promover e monitorizar a mobilidade dos nós de uma VANET. Estudos recentes [38] provaram que o modelo de mobilidade veicular é muito importante para obter resultados relevantes e deve estar bem integrado com o modelo de redes de comunicação sem fios. O uso de um modelo de mobilidade incorrecto como o popular “*random waypoint model*” (que pode funcionar muito bem para algumas redes *Ad-hoc* moveis mas sem dúvida não se adequa a representação da mobilidade numa rede sem fios veicular) pode conduzir a resultados erróneos [38] [39].

Necessitando as VANET de dois componentes de simulação diferentes nomeadamente um simulador de redes de comunicação e um simulador de tráfego, os simuladores de tráfego

têm sofrido uma gigantesca evolução nos últimos anos com vista a incorporarem as comunicações entre veículos. Várias vias para atingir esse efeito têm sido propostas e estudadas. A grande parte dos estudos caminha na direcção da criação de simuladores que integrem a simulação de tráfego e a simulação de redes de comunicação numa única ferramenta de simulação. Contudo, outros estudos optam pela utilização de dois simuladores independentes nomeadamente um simulador de tráfego e um simulador de redes de comunicação, interligados por uma aplicação que assegura a troca de informação entre eles.

Nas subsecções seguintes desta parte do relatório serão apresentados os tipos de sistemas de simulação, uma visão sobre a simulação de redes de comunicação e sobre a simulação de tráfego.

5.2 - Tipos de Sistemas de Simulação

O sistema de simulação, como sugerem alguns dos estudos desenvolvidos [40], é constituído por três componentes lógicos (visualizados na figura 5.1). O primeiro representa um simulador de tráfego, que periodicamente monitoriza novas posições para um certo número de veículos, dentro de um cenário geográfico específico. O segundo componente é constituído por um simulador de redes de comunicação sem fios, que é responsável por simular todas as funcionalidades de uma rede sem fios real. O simulador de rede deve ser permanentemente notificado das posições dos veículos que participam na rede de forma a ter disponível o padrão de conectividade corrente. O terceiro e último componente é a aplicação em si, sendo responsável por controlar todo o ambiente de simulação. Representando o mesmo comportamento para todos os veículos, avalia as mensagens recebidas e decide qual a acção que deverá ser tomada pelo simulador de tráfego. Aparte disso, também é capaz de gerar novas mensagens e difundi-las via “*broadcast*” pela rede.

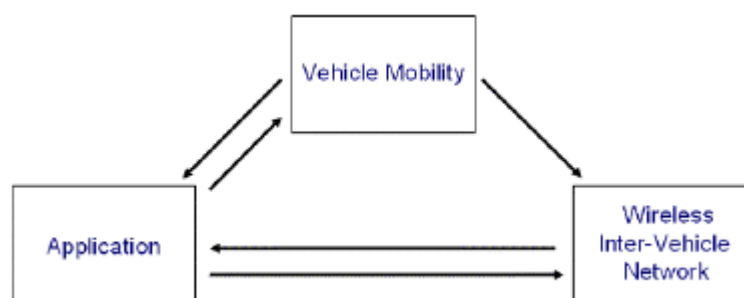


Figura 5.1 - Componentes de um sistema de simulação [40]

Alguns trabalhos integraram a aplicação no simulador de rede, como um módulo adicional. Isto simplifica o sistema e clarifica o trabalho de implementação, já que os

comunicação são apenas necessários entre os dois simuladores. O simulador de rede actua como um cliente requisitando informação ao simulador de tráfego. Por exemplo, o simulador de rede precisa de saber o número exacto de carros que fazem parte da rede e a sua posição geográfica. O simulador de tráfego, por outro lado, actua como um servidor enviando a informação que lhe foi requisitada pelo simulador de rede. Uma representação deste tipo de simulação é apresentada na figura 5.2.

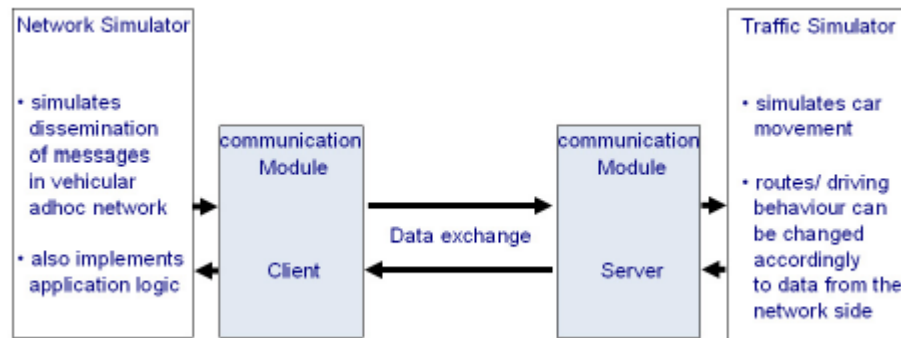


Figura 5.2 - Simuladores com aplicação embutida [40]

A tendência actual é o desenvolvimento e criação de simuladores que integrem a simulação de tráfego e a simulação de redes de comunicação num único simulador. Apesar de projectar e desenvolver um simulador que incorpore os dois tipos de simulação requeridos se revelar uma tarefa mais complicada, o simulador final consiste numa aplicação independente capaz de simular todos os aspectos relacionados com rede V2V, o que facilita a sua comercialização e uso para implementar diferentes cenários de estudo. Um simulador deste tipo é apresentado na figura 5.3.

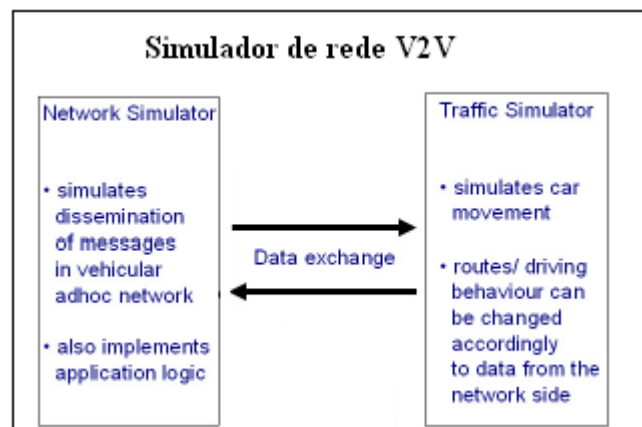


Figura 5.3 - Simulador Integrado de redes V2V

5.3 - Simulação de Redes de Comunicação

Como foi mencionado, uma das partes necessárias à simulação de redes V2V aplicadas a ambientes urbanos é um simulador de redes de comunicação sem fios *Ad-hoc* que interliga veículos. Há vários critérios a ter em conta neste tipo de simulação e na escolha da ferramenta de simulação (simuladores). Um correcto modelo para redes de comunicação móveis sem fios, uma representação dos efeitos específicos como o efeito de sombra causado por edifícios em espaços urbanos ou as reflexões das ondas de rádio nas estradas e paredes dos edifícios devem estar disponíveis e convenientemente ajustados dentro do simulador. Modelos de antena próprios, com poder de transmissão ajustável e limiares de recepção, são necessários para reproduzir as propriedades dos módulos sem fios reais instalados nos veículos. Aparte disso, protocolos de routing adequados devem estar disponibilizados.

Um simulador de rede de comunicação apropriado a uma rede V2V tem também de ter implementado o consensual e aceite protocolo de comunicação standard IEEE 802.11 que especifica os níveis físicos e MAC (*Medium Access Control*) do modelo de arquitectura de redes OSI. O IEEE 802.11 tornou-se a tecnologia de transmissão sem fios prevalente no campo das VANET devido à baixa sensibilidade para altas velocidades, às gamas de transmissão suficientes e aos rápidos tempos de conexão. Para além de um correcto modelo de mobilidade, redes sem fios multi-hop e a implementação do standard de comunicações IEEE 802.11, a possibilidade de incluir trocas de módulos de dados deve estar também implementada. Extensões de rede, como a adição de um novo nó devem também ser possíveis.

Vários simuladores foram desenvolvidos para redes *Ad-hoc* genéricas e que também modelam os canais sem fios com grande detalhe. Um desses simuladores é o **GloMoSim** [41]. Contudo este simulador não é disponível para propósitos académicos e não sofre updates desde 2000. Outro simulador bastante conhecido é o **QualNet** [42]. Um dos simuladores com mais renome, projectado para redes baseadas em IP com e sem fios é o **Network Simulator 2 (NS2)** [43]. Foi escrito em C++ e OTcl, uma linguagem desenvolvida pelo MIT. Este simulador está disponível para uso comercial. A sua arquitectura de software é bem estruturada e contém módulos de software para troca de dados com outros programas, sendo assim ideal para o desenvolvimento de simuladores acoplados (simulador de rede+simulador de tráfego). Além destas características, o NS2 tem incorporado o IEEE 802.11 e é uma plataforma de adaptação fácil.

5.4 - Simulação de Tráfego

Além da simulação da rede de comunicação sem fios, outra componente necessária à simulação de redes V2V é a simulação de tráfego. Para a simulação de uma rede V2V é necessário um simulador de tráfego de alto desempenho que tenha em conta a movimentação dos veículos dentro do cenário geográfico.

Os modelos de simulação de tráfego podem ser classificados em modelos microscópicos, macroscópicos ou mesoscópicos. Nos modelos de simulação microscópica, um veículo individual é estudado e a atenção é focada na sua performance no contexto de todo o sistema da rede de tráfego. As medidas nestes modelos são a velocidade e a localização individual de cada veículo. Este tipo de simulação é adoptado quando se trata de sistemas pequenos ou relativamente simples. Por outro lado, os modelos de simulação macroscópicos agregam a descrição do fluxo de tráfego. As suas medidas de desempenho são a velocidade, o fluxo e a densidade do conjunto de veículos. Este tipo de simulação é usado em grandes redes ou sistemas complexos. Modelos de simulação mesoscópicos agregam aspectos dos sistemas microscópicos e macroscópicos. [44]

Uma das características mais importantes dos simuladores de tráfego é a interactividade. De forma a demonstrar a disseminação de mensagens em VANET e o seu impacto no tráfego, é crucial que seja possível influenciar o comportamento ou mesmo as rotas dos veículos durante a simulação. Um simulador de tráfego deve ser o mais realista possível. Estradas, cruzamentos e regras de trânsito comuns devem estar incorporadas. Sendo assim, um simulador de tráfego microscópico que monitorize a velocidade e posição de cada um dos carros do sistema é requerido.

Um dos simuladores de tráfego mais conhecidos é o **SUMO** [45]. Este simulador é *open source* e foi desenvolvido pela DLR. É um simulador microscópico que corre o espaço continuamente em tempo discreto. Proporciona uma interface gráfica que permite controlar a simulação em execução. Redes de estradas podem ser geradas ou importadas em diversos formatos de mapas, como os do Sistema de Informação Geográfica (Gis) ArcView ou gerados por ferramentas como VISSIM ou VISUM. A grande desvantagem deste simulador, que é implementado em C++, é o facto de as rotas dos veículos terem que ser inseridas antes da simulação. Alterações a este simulador estão a ser efectuadas pelos seus projectistas de forma a incorporar a simulação de redes de comunicação e mesmo a comunicação entre veículos. Contudo tais funcionalidades não se encontram ainda disponíveis.

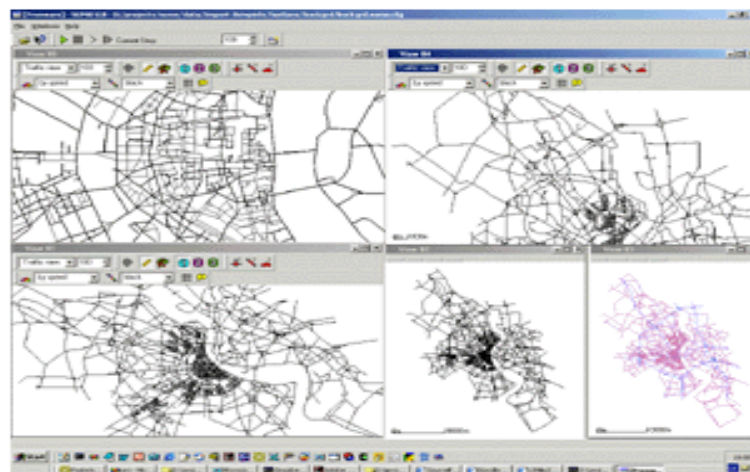


Figura 5.4 - Screenshots de uma simulação efectuada com SUMO [45]

O simulador de tráfego **VISSIM** [46] é uma ferramenta extremamente abrangente, que possui uma brilhante representação gráfica de cenários, incluindo carros, comboios, autocarros e camiões, bem como pessoas e edifícios envolventes. É um simulador microscópico, de tempo discreto, que só corre em sistemas operativos MS Windows. O seu código fonte não é disponível pelo que as suas funcionalidades só podem ser alteradas através do ajuste de certos parâmetros. Contudo, contém uma interface programável e pode ser acoplado a outras aplicações. A licença deste simulador é de custo elevado, em contrapartida é bem suportado e demonstra elevada performance, principalmente no que diz respeito ao ambiente gráfico. Algumas das desvantagens deste simulador revelam-se principalmente ao nível da interactividade e compatibilidade. Mapas de estradas só podem ser usados quando traduzidos com a ajuda de uma ferramenta chamada VISUM. Além disso, a versão standard do VISSIM só possibilita que os veículos se movam ao longo de rotas previamente decididas antes da simulação. Só com muito esforço de programação, as rotas de um veículo em particular poderiam ser alteradas durante o decorrer da simulação.



Figura 5.5 - Simulação efectuada com VISSIM [46]

O simulador de tráfego **CARISMA**, desenvolvido pela BMW, foi implementado em C++ como um simulador para cenários relativamente pequenos. O movimento dos veículos pode ser visualizado depois da simulação terminar. O simulador trabalha com base em tempos discretos e actualiza a posição dos veículos a todos os segundos. Redes de estradas podem ser lidas dos chamados ESRI-Shapefiles. Em contraste com o VISSIM e o SUMO, o CARISMA computa as rotas dos veículos durante a simulação. Redes de tráfego consistem essencialmente em cruzamentos. Este simulador também disponibiliza um modelo de obstrução de ondas rádio que é excepcional para a simulação de tráfego em ambientes urbanos. Assume que edifícios estão localizados ao longo de todas as ruas e periodicamente computa informação de conectividade para todos os veículos.[40]



Figura 5.6 - Simulação usando CARISMA [40]

Outros exemplos de simuladores de tráfego microscópico são o **CORSIM** [47] e **PARAMICS** [48], existindo ainda muitos mais disponíveis, sendo a maior parte deles pouco orientados à adaptação de simulações V2V.

5.5 - Simulação Integrada

Num passado recente foram desenvolvidos alguns simuladores que incorporam a simulação de tráfego e de redes de comunicação num único simulador, como já foi referido mais vezes neste relatório. Parece mesmo que esta é a actual tendência devido às vantagens inerentes ao uso de uma só aplicação para simular todas as funcionalidades das redes V2V. Sendo assim, vários projectos encontram-se em desenvolvimento implementando diferentes arquitecturas para a realização destes simuladores.

O **GrooveNet** [49] é um simulador híbrido, destinado à simulação de redes V2V. Este simulador possibilita comunicação entre veículos simulados, veículos reais e entre veículos reais e simulados. O GrooveNet possibilita a simulação de protocolos de redes veiculares em cenários que integram milhares de veículos virtuais numa topologia de redes de estradas e a simulação em ambiente real de veículos equipados com GPS (*Global Positioning System*) e interfaces de redes sem fios. Em adição, o GrooveNet suporta ainda a comunicação entre veículos simulados e veículos reais. Se estes estiverem na vizinhança uns dos outros podem trocar pacotes de informação. É assim ideal para proporcionar um rápido desenvolvimento, correcção e teste de protocolos de redes veiculares e para a prototipação de cenários de teste para redes de comunicação *multi-hop* em ambiente real. Este simulador foi ainda

desenvolvido para explorar e usar o standard DSRC (*Dedicated Short Range Communication*) e suportar simulação de canais múltiplos e interfaces de rede baseadas em DSRC.

As características deste simulador são:

- Simulador modular baseado em eventos;
- Suporta múltiplos veículos e modelos de mobilidade sobre uma variedade de *links* de rede e modelo físico de camadas;
- Possui uma interface gráfica que permite auto-gerar simulações com milhares de veículos. Todos os veículos obedecem aos limites de velocidade das vias;
- Suporta três tipos de nós de simulação, nomeadamente veículos capazes de disseminar dados sobre um ou mais canais DSRC, infra-estruturas físicas e “gateways” móveis capazes de comunicação veículo-a-veículo e veículo-a-infra-estrutura;
- Suporta vários tipos de mensagens como mensagens GPS que são difundidas periodicamente para informar as posições de veículos vizinhos e mensagens de alerta para proximidade de veículos de emergência;
- Suporta várias interfaces de rede para comunicações entre veículos reais e entre veículos e infra-estruturas como, por exemplo, uma interface DSRC de 5.9 GH, IEEE 802.11^a/b/g, 1xRTT e interfaces celulares EVDO;
- É capaz de suportar comunicação híbrida, isto é, entre veículos reais e veículos simulados;
- Possui conectividade com os computadores de bordo dos veículos, sendo capaz de se ligar a estes e lê códigos de diagnóstico OBD-II.

Na figura 5.7 é apresentado um screenshot do simulador GrooveNet implementado para ambientes Linux.

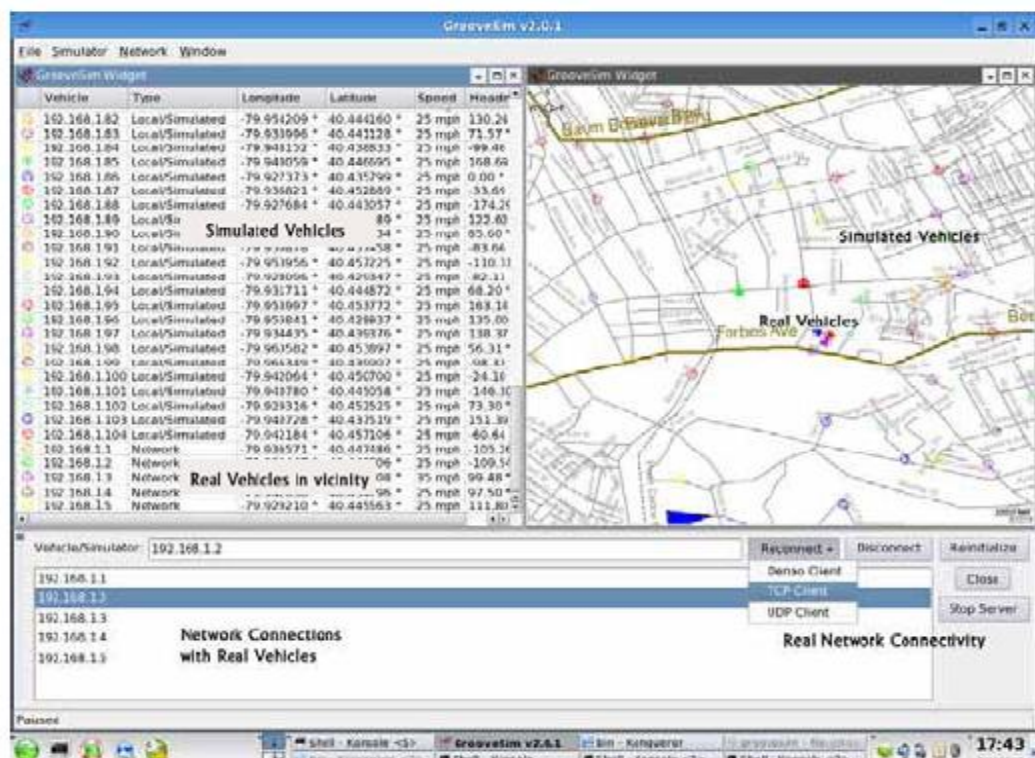


Figura 5.7 - Veículos reais e simulados interagindo no simulador GrooveNet [49]

O projecto **DIVERT** (Development of Inter-VEhicular Reliable Telematics) [50] [51] desenvolvido na Faculdade de Ciências da Universidade do Porto combina um sofisticado simulador de tráfego, funcionando como um servidor de posicionamento global baseado em GPS que fornece informações para simulação microscópica de veículos, com um modelo de comunicação sem fios virtual, permitindo o desenvolvimento de simulações para redes V2V.

Actualmente o DIVERT está implementado sobre uma representação vectorial da rede de estradas do Porto. Contudo, não é dependente deste mapa e pode ser actualizado com mapas de outras cidades ou regiões, desde que sigam as especificações do *Open GeoSpatial Consortium*. São modelados dois tipos de veículos no DIVERT: veículos que circulam e comunicam, chamados de sensores, e veículos que só circulam. Dentre destes dois tipos de veículos o DIVERT divide-os ainda em veículos normais e de tamanho grande, associando padrões de mobilidade indicados para cada um deles. Estes padrões são ainda influenciados por inicialização aleatória de atributos como a aceleração, travagem, agressividade e tolerância ao risco.

O DIVERT usa os seguintes mapas vectoriais da rede de estradas:

- Um mapa simples de dois eixos representando por linhas a topologia das redes de estradas.
- Um mapa mais detalhado, descrevendo em promenor a rede de estrada, incluindo informação sobre segmentos de estrada, visibilidade nos cruzamentos, localização de semáforos, limites de velocidade e parques de estacionamento.

Um conjunto de imagens de satélite melhora a visualização das simulações de tráfego nesta aplicação, que actualmente é baseada em imagens 2D. Entretanto está a ser implementada actualmente uma versão em 3D que permitirá, além de acelerações mais realistas, também um melhoramento nas detecções sem fios que assumirão reflexão e o efeito de sombra de edifícios 3D. Olhando para o campo das rotas, o DIVERT usa actualmente um modelo híbrido, que combina rotas predefinidas com rotas geradas aleatoriamente.

A simulação de tráfego é parametrizada pelo número de veículos e pela referência de quais deles são sensores. A simulação é iniciada posicionando cada um dos veículos num ponto aleatório da sua rota. Veículos que chegam ao seu destino são removidos ou estacionados por um certo intervalo de tempo. Novos veículos surgem também na simulação, de pontos de entrada do mapa ou de parque de estacionamento da rede.

Na figura 5.8 é ainda apresentada um screenshot de uma simulação.



Figura 5.8 - Simulação usando DIVERT [51]

Capítulo 6

Solução Proposta

Neste capítulo é documentada a solução proposta para o objectivo de projectar uma arquitectura que implemente SOA em uma rede V2V. Apresenta-se a arquitectura proposta e ilustra-se a escolha da ferramenta de simulação a usar.

6.1 - Definição da Arquitectura

O principal objectivo deste trabalho, como foi definido no capítulo 1, é especificar uma arquitectura que implemente SOA em uma rede MANET sem qualquer tipo de infra-estrutura, em que os nós desta rede são veículos; ou seja, uma VANET.

Actualmente, os serviços que o condutor e os passageiros podem usar nos veículos requerem uma grande quantidade de hardware. Cada nova funcionalidade implementa-se com um novo dispositivo que tem que ser instalado no habitáculo. Esta estratégia revela-se dispendiosa e não se apresenta compatível com o aumento significativo de serviços que se tem observado. Fornecer cada novo serviço como software executável em uma unidade de bordo (*Onboard Unit* - *OBU*) baseada num computador embutido apresenta inúmeras vantagens.

A capacidade de comunicar com outros veículos é também um importante factor a ter em conta. Como já foi retratado, a comunicação inter-veicular neste trabalho baseia-se numa rede VANET.

De acordo com o apresentado, definiu-se uma arquitectura extensível para serviços baseada num computador embutido com capacidade de comunicação a nível de serviço.

Como a arquitectura proposta se destina a fornecer serviços a utilizadores através de uma rede de comunicação V2V, a abordagem a seguir foi definir uma arquitectura de camadas. Para começar, dividiu-se assim a arquitectura em dois níveis principais: nível de serviços de rede e nível de serviços de utilizador. O primeiro nível é responsável pela “criação” da topologia de rede e pela descoberta e troca de serviços entre veículos. O segundo nível, pela definição da arquitectura necessária à distribuição dos serviços, denominados de serviços de alto nível, aos utentes.

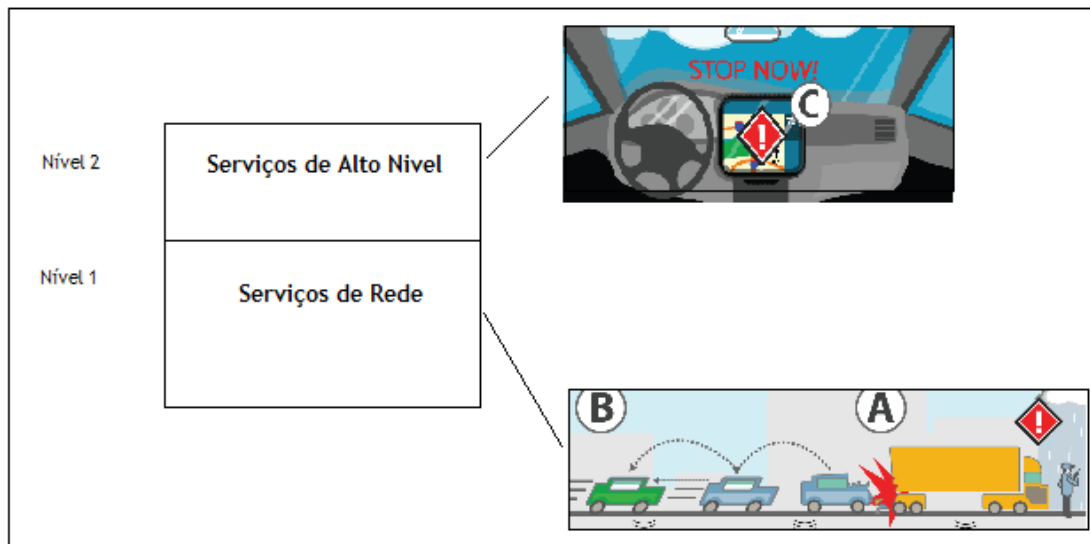


Figura 6.1 - Arquitectura de níveis

6.1.1 - Definição do nível de serviços de rede

Para a definição do primeiro nível da arquitectura que, como já foi retratado, será responsável pela “criação” da topologia de rede e pela troca de serviços entre veículos, partiu-se do caso de utilização definido no capítulo 2 e adoptou-se a solução proposta em [1] para redes VANET. O caso de utilização é novamente demonstrado na figura 6.2 e resumido nos primeiros parágrafos do texto seguinte, relatando-se a seguir a solução proposta para este nível da arquitectura.

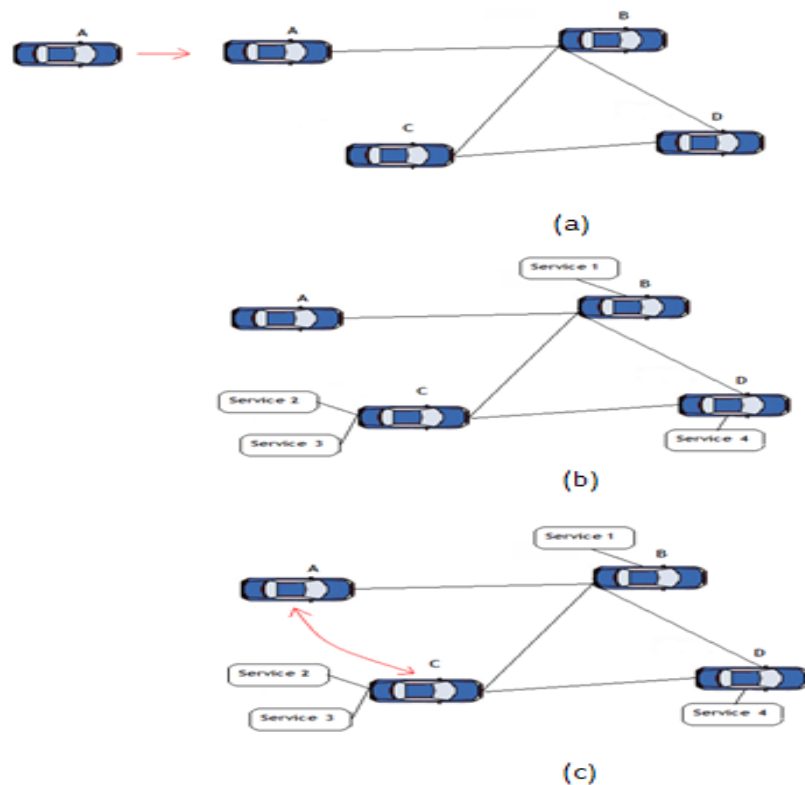


Figura 6.2 - Caso de estudo: SOA aplicada uma rede V2V

Na figura 6.2 (a) demonstra-se a formação da rede em si. Todos os nós (A a D) representam veículos equipados com dispositivos móveis de comunicação V2V. No princípio, o nó A está desconectado, não tendo qualquer outro nó na sua área de cobertura. Quando outros nós entram na sua área de cobertura ele liga-se a estes. Todos os nós conectados formam assim uma VANET. Depois das configurações automáticas necessárias, a VANET está pronta a encaminhar mensagens originadas em qualquer nó e destinadas a qualquer outro.

Quando a rede VANET está pronta, a funcionalidade SOA tem que ser implementada. Todos os nós provedores de serviços publicam a descrição dos respectivos serviços, assim os restantes nós podem descobrir esses serviços. O processo de descoberta dos serviços deve ser implementado de tal maneira que qualquer serviço na rede VANET deve ser detectado sem qualquer configuração manual. Na figura 6.2 (b) todos os nós que contêm serviços publicaram esses mesmos serviços e a descrição de qualquer um dos serviços é visível por todos os nós da rede.

Depois da rede VANET ser formada e da implementação de SOA a rede de serviços está pronta a ser utilizada. Um exemplo da utilização de um serviço é demonstrado na figura (c). O nó A conecta-se à interface de serviço do serviço 3 presente no nó C. Toda a informação necessária à troca de mensagens entre o consumidor de serviços (nó A) e o provedor de serviços (nó C) está contida na descrição de serviço do serviço 3, não sendo necessário ao nó A conhecer quaisquer outras propriedades do nó C. O nó A e o nó C não se encontram ligados directamente, assim as mensagens têm que ser encaminhadas pelo nó B. O encaminhamento (*routing*) na rede VANET é independente da funcionalidade SOA.

Estabelecido o caso de utilização, ficou definido que um dos requisitos mais importantes de um sistema SOA assente numa rede MANET é a realização do mecanismo de descoberta dos serviços. Como não existe servidor central essa descoberta tem que ser feita de forma descentralizada. Duas abordagens são então possíveis para implementar este mecanismo: integrá-lo no mecanismo de *routing* da rede MANET ou implementá-lo separadamente no topo da camada de rede. A integração com o protocolo de routing possui a vantagem de as tabelas de routing e as técnicas de encaminhamento do protocolo serem utilizadas na descoberta de serviços. Optando por uma implementação separada, os benefícios do algoritmo de routing e da informação da topologia de rede não poderão ser assim utilizadas. Logo optou-se pela opção de integração.

O passo seguinte foi escolher o protocolo de *routing* a usar. Tal como já foi descrito, no capítulo 4, as MANET devido a sua topologia possuem já um vasto leque de protocolos propostos, tendo sido alguns dos principais protocolos deste tipo de rede analisados nesse capítulo. Foi também referido que as VANET, hoje em dia, não possuem uma gama tão elevada de protocolos. Através dessa análise e tendo em conta as suas características e as da rede em questão escolheu-se, um para a implementação do nível da arquitectura em estudo. A opção foi usar um protocolo das redes MANET e alterá-lo para as características das VANET e da implementação SOA.

Ficou decidido que a implementação SOA seria integrada no mecanismo de routing da rede MANET. Apesar de alguns estudos afirmarem que os protocolos reactivos têm melhores prestações nas VANET optou-se por um protocolo pró-activo. Esta opção deve-se ao facto de se querer integrar a funcionalidade SOA no protocolo, isto facilita a propagação das descrições dos serviços, já que toda a topologia da rede é conhecida. Outro aspecto a ter em conta nesta escolha foi que se assume que a maior parte dos serviços simulam algum tipo de interacção do utilizador com o serviço. Isto suporta a ideia de distribuir aos utilizadores todos os serviços disponíveis na rede, sem estes terem necessidade de requerer um tipo específico de serviço.

O protocolo escolhido foi assim o OLSR. Dos protocolos pró-activos é o que se encontra mais desenvolvido e toma vantagem no uso de *multipoints relaying* (MPR), como foi visto, para otimizar o *flooding* das mensagens na rede. Outro aspecto vantajoso é de o protocolo OLSR possuir um componente *open-souce* para a sua implementação, o *OLSR daemon* (*oslrld*) [52].

Escolhido o protocolo, é tempo de pensar na implementação SOA.

Routing nas redes MANET é baseado na cooperação dos nós participantes. Todos eles têm de cumprir com o mesmo protocolo de forma a colectivamente implementar o esquema de *routing*. Esta cooperação deve ser estendida também à rede de serviços. Assim, todos os nós da VANET devem encaminhar as mensagens de descoberta de serviços, quer eles façam ou não parte da rede de serviços. Na prática, isto significa que todos os nós encaminham as mensagens, mesmo que não reconheçam o conteúdo destas. Desta maneira, todos os nós que contêm serviços podem utilizar toda a rede e o protocolo de *routing*.

Um nó que queira publicar serviços deve gerar e difundir mensagens SOA periodicamente de acordo com um determinado intervalo de transmissão. Uma mensagem SOA deve ter a descrição de um serviço.

Quando um nó recebe uma mensagem SOA deve retransmiti-la e, se suporta SOA, processá-la. Este processamento inclui fazer o *update* do repositório remoto de serviços e outras tarefas opcionais, como exibir a informação ao utilizador ou outro uso para a

informação recebida.

Todos os nós que suportam SOA devem manter o repositório remoto de serviços actualizado. A informação acumulada nos repositórios provém das mensagens SOA recebidas e processadas. Para todos os serviços remotos conhecidos uma entrada contendo três valores deve ser guardada. As entradas são assim constituídas pelos seguintes componentes: endereço de origem, descrição do serviço e um valor temporal. O endereço de origem é o endereço do nó remoto de *hosting*. A descrição do serviço é a descrição total do serviço correspondente. O valor temporal especifica o tempo em que a entrada expira e deve ser removida do repositório. Sempre que um nó processa uma mensagem SOA, deve actualizar o seu repositório.

A figura 6.3 ilustra assim os componentes mais relevantes para a solução definida: o protocolo de *routing*, a camada SOA e os serviços.

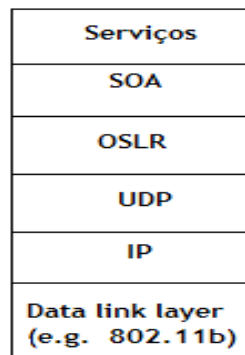


Figura 6.3 - Pilha de Protocolos (adaptado de [1])

Para a implementação pode ser usado um componente de software open-source, o OLSR Daemon (olsrd).

O Olsrd é entregue como um executável único. Em adição proporciona um *Plug-in Framework*, ilustrado na figura 6.4.

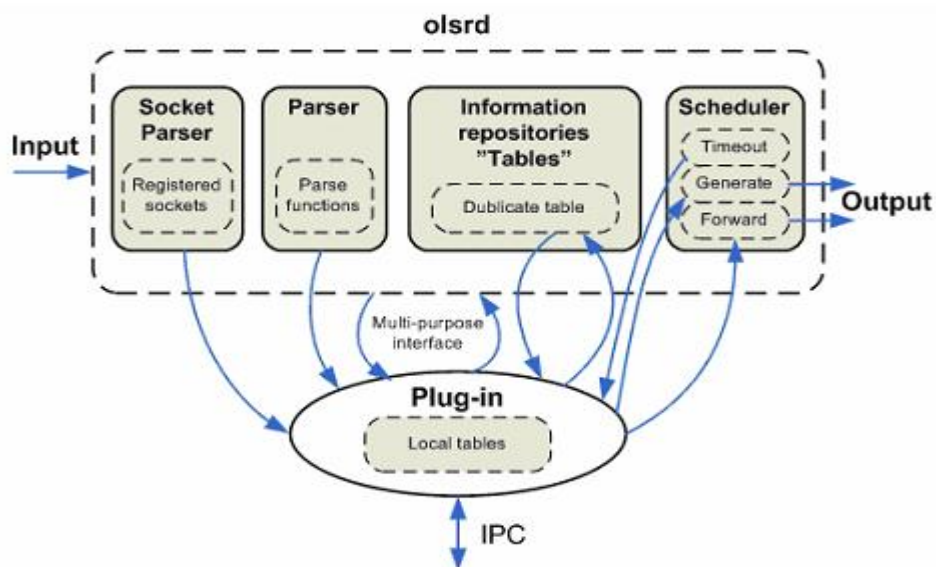


Figura 6.4 - Plug-in: Olsrd [1]

Os *Plug-ins* são bibliotecas dinâmicas ligadas que, como o nome indica, são carregadas dinamicamente em tempo de execução. O Framework do Plug-in fornece a possibilidade de executar módulos de software, que podem implementar funcionalidades associadas ao protocolo de *routing* OLSR.

*Pulg-*ins** podem aceder à funcionalidade *oslrdr* para gerar e processar pacotes, e podem utilizar o algoritmo de encaminhamento e a propriedade dos MPRs. Podem também operar com *sockets* e aceder a informação dos repositórios, isto é as tabelas de *routing*.

Seguindo esta ideia a camada SOA é feita como um Plug-in oslrd, que fornece a descoberta de serviços SOA.

Um *Plug-in* oslrd fornece uma funcionalidade oslrd não estando de maneira nenhuma ligado a SOA. Oslrd apenas transfere mensagens de acordo com o protocolo OSLR, independentemente do conteúdo destas. O Plug-in SOA apenas fornece um *framework* para a rede de serviços não sendo responsável por oferecer qualquer serviço ou aplicação para o utilizador. As principais tarefas do *Plug-in* SOA são criar mensagens SOA, processar as mensagens SOA recebidas e manter a representação dos serviços remotos.

A figura seguinte ilustra o formato da mensagem SOA que é introduzida como um novo tipo de mensagem oslrd.

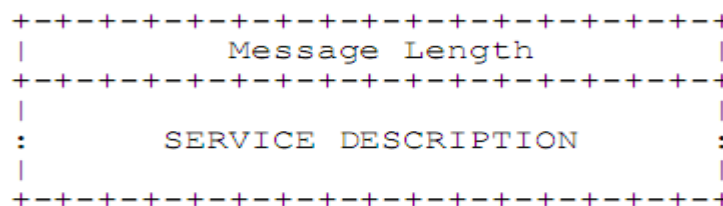


Figura 6.5 - Formato da mensagem SOA (adoptado de [1])

O *message Length* indica o comprimento da mensagem. Em seguida vem a mensagem em si, que é uma descrição SOA do serviço. A mensagem SOA é transportada no pacote OSLR e transmitida pela rede.

A informação transmitida pela rede é a descrição do serviço, contida nas mensagens SOA. Cada mensagem SOA contém precisamente uma descrição de um serviço, que corresponde a um único serviço. Não existem requisitos próprios para a estrutura e conteúdo da descrição de serviços que dependerá do serviço e da forma de implementação. Por exemplo, *Web Services Description Language (WSDL)* pode ser usada, assim como a linguagem XML.

Para gerar mensagens SOA, o Plug-in SOA tem que aceder às descrições locais dos serviços, incluir essa descrição numa mensagem SOA, formatar a mensagem como uma mensagem OSLR e difundir-na na rede. A descrição dos serviços é fornecida pelos provedores de serviços e não pelo Plug-in SOA, logo é necessário um IPC (*inter-process communication*). No Plug-in SOA o IPC é implementado por *Shared files*, que estão alocados numa localização específica no sistema de ficheiros.

A figura 6.6, a seguir, descreve a geração da mensagem SOA, que acontece

periodicamente num intervalo de transmissão definido. É utilizada a funcionalidade *scheduler* do *olsrd*. Quando o *Plug-in* SOA é invocado, este lê uma descrição de serviço do sistema de ficheiros e constrói a mensagem SOA. Esta é convertida para uma mensagem OLSR que é difundida através da invocação do *olsrd* API.

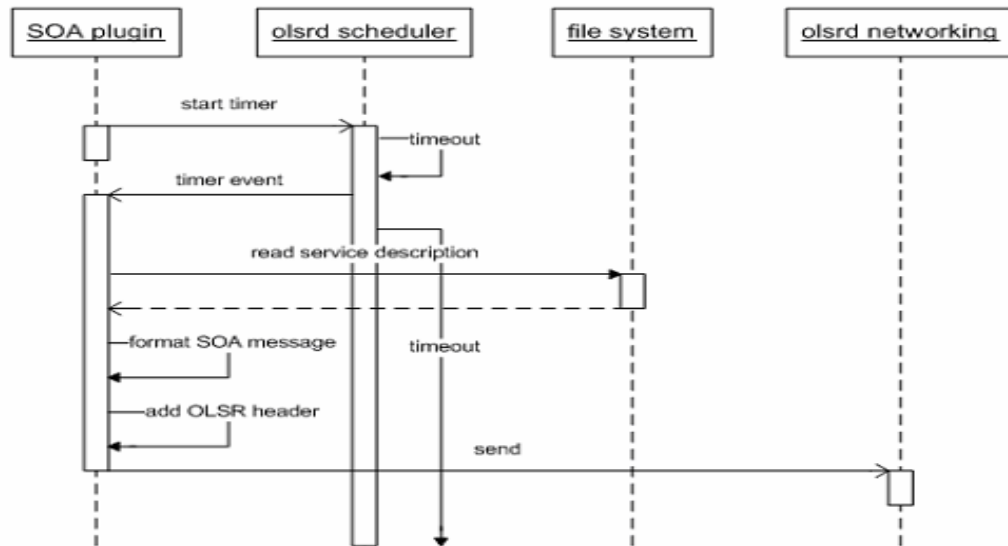


Figura 6.6 - Geração da mensagem SOA [1]

O Plug-in SOA é também responsável pelo processamento das mensagens SOA recebidas, como é ilustrado na figura 6.7.

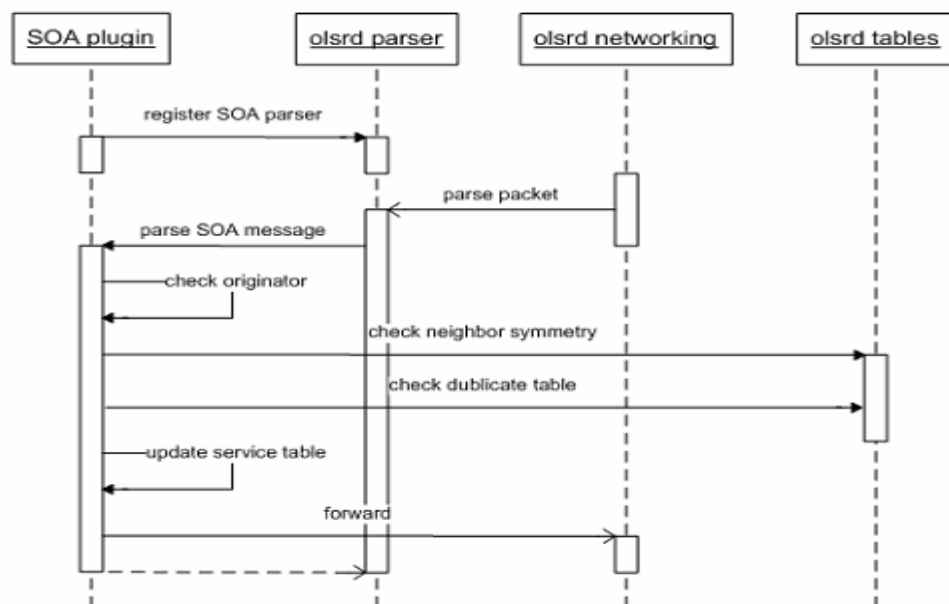


Figura 6.7 - Processamento de mensagem SOA [1]

Os repositórios remotos de serviços são implementados através de uma tabela de serviços, onde cada serviço é representado por uma estrutura de dados contendo o endereço, a descrição do serviço e o intervalo de tempo durante o qual o serviço é válido. Sempre que uma mensagem SOA é processada, a tabela de serviços é actualizada.

6.1.1 - Definição do nível de serviços de Alto nível

O segundo nível consiste na arquitectura necessária à distribuição dos serviços de alto nível aos utentes. Serviços de alto nível são aqueles que são disponibilizados aos utilizadores. Ou seja, os serviços de alto nível são aplicações finais com interface gráfica para o utilizador. Por assim dizer, são todas as aplicações a que os utilizadores têm acesso no veículo.

Assim para a definição deste nível começou-se por identificar os vários serviços a disponibilizar, os utilizadores e provedores de cada serviço e os requisitos necessários a implementação de cada serviço.

Essa informação encontra-se organizada na tabela 6.1 a seguir.

Tabela 6.1 - Serviços de Alto nível

Serviços	Tipo de Cliente	Cliente	Provedor de serviços	Requisitos de implementação
Detecção de colisões	Simples	Condutor	Veiculo	Dispositivos de detecção Comunicação V2V
Detecção de congestionamentos	Simples	Condutor	Veiculo	Dispositivos de detecção Comunicação V2V
Detecção de peões	Simples	Condutor	Veiculo	Dispositivos de detecção Comunicação V2V
Luz de travagem de emergência	Simples	Condutor	Veiculo	Dispositivos de detecção Dispositivo de aviso Comunicação V2V
Detecção de objectos caídos na via	Simples	Condutor	Veiculo	Dispositivos de detecção Comunicação V2V
Chamada de emergência	Simples	Condutor	Central	Dispositivo de segurança Comunicação V2V Comunicação com central
Detecção de aproximação de veículo prioritário	Simples	Condutor	Veiculo	Dispositivos de detecção Comunicação V2V
Alterações climáticas	Simples	Condutor	Veiculo	Dispositivos de comunicação Comunicação V2V

Acesso a internet	Composto	Condutor Passageiros	Central	Dispositivo de acesso a internet Routers
Aplicações de entretenimento	Composto	Condutor Passageiros	Central	Servidor de aplicações Routers
Sistema de telecomunicação	Simples	Condutor	Veículo	Dispositivos de comunicação Comunicação v2v
Alarme de veículo desviado/em perigo	Simples	Central	Central	Dispositivo de segurança Comunicação v2V Comunicação com central
Informação turística	Composto	Condutor Passageiros	Central	Servidor de aplicações Comunicação V2V Routers

Como já foi dito, supõem-se que os veículos estejam equipados com uma OBU.

A figura seguinte apresenta os principais componentes do nível da arquitectura em estudo. A arquitectura é representada através de um diagrama de desenvolvimento UML, que demonstra a distribuição dos componentes dentro do veículo e componentes pertencentes ao ambiente que rodeia o veículo.

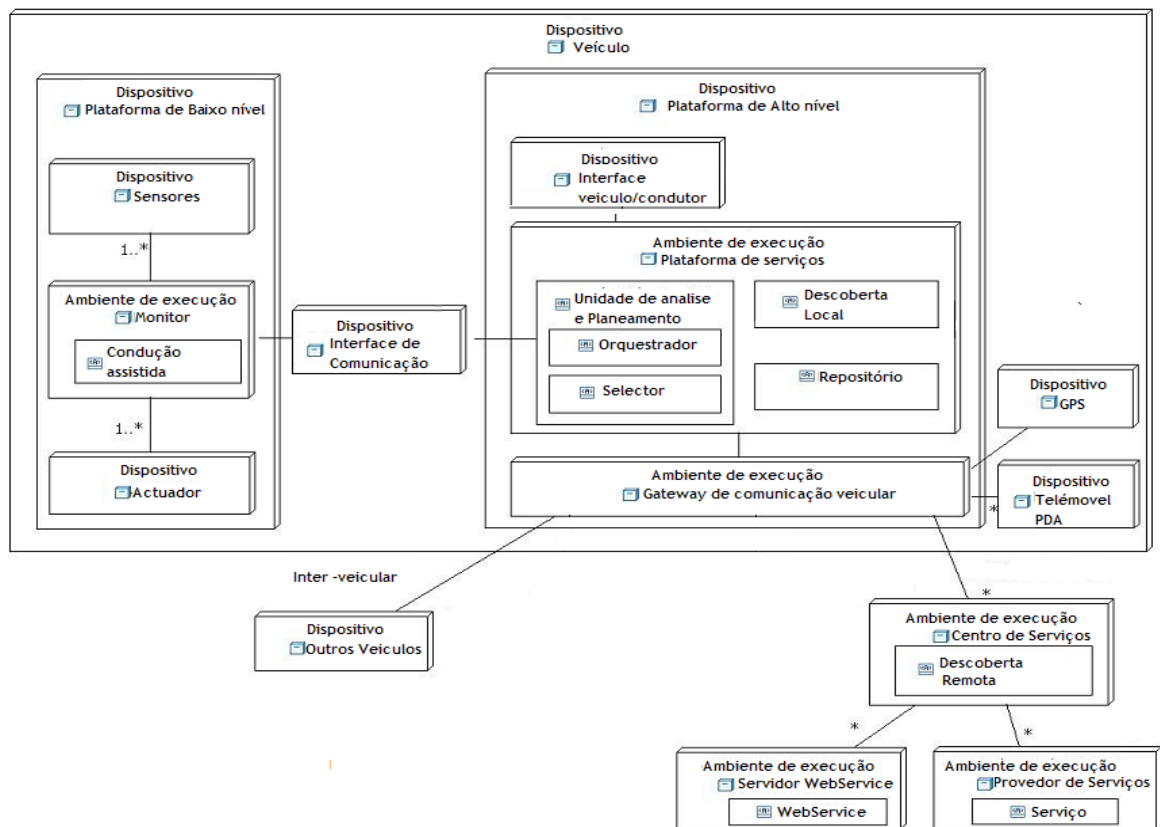


Figura 6.8 - Diagrama de desenvolvimento UML para arquitectura SOA no veículo

Segue-se uma descrição dos diferentes elementos da arquitectura proposta, com base em [53].

Veículo - Veículo equipado com sensores e que possui capacidade de determinar a sua localização geográfica e de interagir com o ambiente que o rodeia.

Plataforma de baixo nível - Contém o hardware e os componentes de software de baixo nível do veículo. Serviços críticos de assistência à condução, como o ABS e o ESP e o software de actuação dos sensores do veículo relacionados com eles são tratados neste baixo nível da arquitectura, para assegurar um tempo de resposta mínimo.

Sensor - Todos os sensores necessários para absorver o estado do veículo.

Monitor - Monitoriza e gere todos os sensores do veículo e alerta no caso de algum dos sensores indicar algum evento. Situações críticas são reportadas para o actuador, enquanto as demais são submetidas à unidade de análise e planeamento.

Condução assistida - Contém todas as funcionalidades de assistência à condução de baixo nível desencadeadas pelos actuadores e geridas no monitor.

Actuador - Desencadeia automaticamente os sistemas de assistência à condução.

Interface de Comunicação - Permite a comunicação entre a plataforma de baixo nível e a plataforma de alto nível do veículo.

Plataforma de Alto Nível - Plataforma de computação para realizar as funcionalidades computacionais de alto nível do veículo.

Interface veículo/condutor - Interface de comunicação entre o veículo e o condutor/passageiros. O condutor e passageiros recebem informação dos serviços activos e podem realizar comandos para desencadear, parar ou adaptar esses serviços.

Plataforma de serviços - Elemento da arquitectura, presente no veículo, que contém um repositório de serviços e software para gerenciar funções específicas aos serviços como a descoberta de serviços e a sua orquestração.

Unidade de análise e planeamento - Analisa eventos reportados pelo monitor, pela interface veículo/condutor ou pelo *gateway* de comunicação veicular e planeia a acção correspondente. O repositório interno de serviços é questionado por um determinado serviço, para uma acção específica. Deve possuir mecanismos de actualização (do repositório de serviços), descoberta e registo de serviços.

Orquestrador - Elemento da arquitectura que é responsável por alcançar um objectivo através da composição de serviços.

Selector - Analisa e selecciona serviços baseado em critérios estabelecidos.

Descoberta local - Serviço local para a descoberta de serviços apropriados.

Repositório - Repositório interno de serviços onde os serviços contidos no veículo ou software preciso para consumir serviços externos são armazenados localmente.

Gateway de comunicação veicular - Envio abstracto de mensagens para componentes externos (por exemplo, outros veículos ou um servidor). A melhor técnica de comunicação é escolhida (por exemplo, GSM, GPRS ou VANET)

Telefone móvel/ PDA - Dispositivo usado para construir conexões celulares de grande alcance. Estes dispositivos podem ser incorporados no veículo ou providos pelos utilizadores destes.

GPS - Receptor da informação do sistema de posicionamento global (*Global Positioning System*)

Centro de serviços - Plataforma de execução remota que oferece descoberta de serviços e aplicações que suportam pedidos de serviços.

Descoberta remota - Serviço remoto para a descoberta de serviços apropriados.

Provedor de serviços - Plataforma de execução remota que oferece serviços.

Servidor Web Service - Plataforma de execução remota contendo serviços disponíveis na internet.

Serviço - Recurso abstracto, que representa a capacidade de executar tarefas de uma funcionalidade coerente do ponto de vista das entidades provedoras e das entidades requisitoras.

Web Service - Serviço disponível na internet.

Os elementos da arquitectura que constituem o diagrama UML são nós e artefactos conectados por associações UML e dependências entre eles. Artefactos são usados para especificar os componentes físicos de informação que são usados ou produzidos por um processo de desenvolvimento de software ou por implementações e operação de um sistema. Nós são recursos computacionais sobre os quais os artefactos devem ser implementados para execução. São usados dois tipos especiais de nós: dispositivos e ambiente de execução. **Dispositivos** são recursos computacionais físicos, com capacidade de processamento sobre os quais os artefactos devem ser implementados para execução. Podem ser bastante complexos e até constituídos por um grupo de outros dispositivos. O **Ambiente de execução** é um nó

que oferece um ambiente de execução para um tipo específico de componentes que são implementados neste na forma de artefactos executáveis.

Os serviços, na arquitectura proposta para este nível, estão classificados de acordo com o seu nível de abstracção. Os serviços de baixo nível são, principalmente, software de acesso aos dispositivos físicos do veículo. Por sua vez, os serviços de alto nível têm a seu cabo tarefas de adopção e transformação de informação. Esta estrutura hierárquica de serviços tem um duplo propósito. Para além da programação modular, resolve problemas de acesso aos dispositivos quando há problemas de sincronização.

Uma maneira de implementar a arquitectura proposta é supondo que todos os veículos possuem uma OBU. Todos os sensores incluídos no veículo estão conectados a essa OBU via enlaces RS232, USB ou Bluetooth. A OBU será, na sua concepção básica, um computador com sistema operativo que não está prefixado, graças ao facto de o software ser baseado em Java. Sobre a máquina virtual Java situam-se várias APIs de programação que facilitam a implementação dos serviços. OSGI (*Open Service Gateway initiative*) [54] pode ser usado para realizar a plataforma de serviços do veículo. Assim todas as interfaces que levam a cabo a comunicação entre as diferentes camadas de serviços serão interfaces Java.

A comunicação inter-veicular será implementada da maneira descrita na secção 6.1.1.

A funcionalidade SOA, que permite o acesso a serviços externos ao veículo, como Web services, quer pelo veículo quer por outros dispositivos móveis que se encontram neste, como PDAs ou telemóveis, pode ser implementada da forma descrita em [55] que apresenta uma arquitectura SOA para dispositivos móveis baseadas em redes celulares de longo alcance, que tem como principais características:

- Integração dinâmica de novos serviços;
- Descoberta dinâmica de novos serviços;
- Uso de software open source para desenvolver a solução.

Basicamente a arquitectura proposta nesse trabalho divide a arquitectura em quatro componentes principais: provedores de serviço, gerenciador de serviços, os clientes e o registo de UDDI. Acrescenta-se assim um componente (gerenciador de serviços) a arquitectura SOA típica.

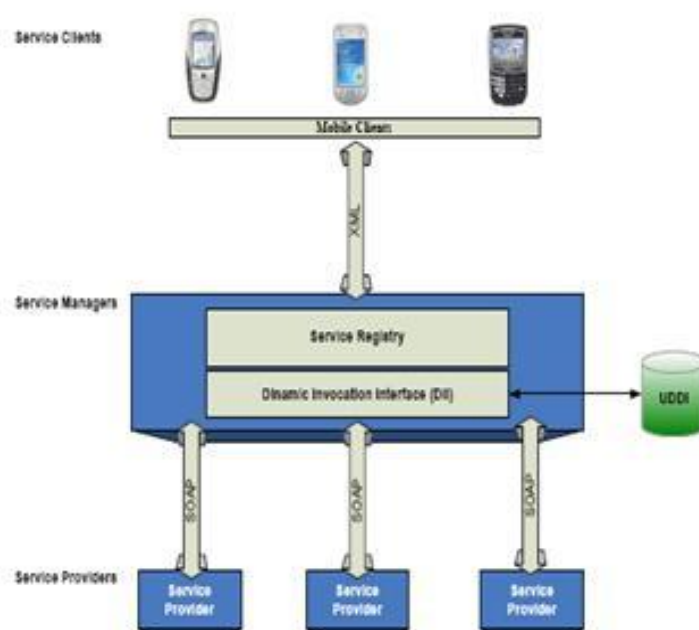


Figura 6.9 - Arquitectura SOA para dispositivos móveis [55]

Os principais componentes da arquitectura proposta nesse trabalho são descritos a seguir,:

- **Provedores de Serviço:** são os principais componentes que implementam e oferecem os serviços disponíveis. Estes serviços são descritos utilizando a linguagem WSDL [56].
- **Clientes:** representam os utilizadores de dispositivos móveis que estão interessados nos serviços disponíveis no ambiente em que ele se encontra. Estes serviços devem ser fornecidos aos clientes de uma forma transparente para estes.
- **Gerenciador de Serviço:** é a camada intermediária entre os dispositivos móveis e o provedor de serviços. É o responsável pelo fluxo das informações entre ambos os componentes. Um gerenciador de serviços é um *web service* que utiliza uma invocação dinâmica de serviços como mecanismo de comunicação entre os diferentes provedores de serviços.
- **Registo UDDI:** estes registos são utilizados para a localização de novos serviços. A descoberta de serviços é computada em tempo de execução pelo gerenciador de serviços, quando o utilizador envia uma requisição de um novo serviço.

O *Framework* proposto pode suportar um ou vários gerenciadores de serviços.

As aplicações dos clientes que residem nos dispositivos móveis apenas interagem com os gerenciadores de serviços.

A linguagem XML pode ser o formato usado para a troca de dados entre as aplicações residentes nos dispositivos móveis e os gerenciadores de serviços.

A interacção entre dispositivos móveis e os provedores de serviços usando gerenciadores de serviços, consiste num processo com cinco passos, descritos a seguir, baseado em [55].

- **Começo:** Quando se inicia um gerenciador de serviços, este processa um registo de serviço. Este registo é uma estrutura que permite aos provedores de serviço armazenar a lista de endereços URL dos serviços disponíveis. Os gerenciadores de serviços mantêm uma estrutura XML como registo. Uma interface de invocação dinâmica é utilizada pelos gerenciadores de serviços a fim de obter a descrição dos serviços em tempo de execução. O uso da interface de invocação dinâmica evita a geração de uma classe *stub* para cada serviço disponível, já que o gerenciador de serviço invoca dinamicamente a descrição do *Web service* e só posteriormente o invoca a ele. Assim, a lista de serviços a oferecer aos utilizadores dos dispositivos móveis pode mudar dinamicamente de acordo com os novos serviços acrescentados a qualquer momento e em tempo de execução.

- **Descoberta de Serviços:** Para descobrir serviços na rede SOA, os clientes móveis enviam um pedido de informação dos serviços disponibilizados pelos provedores de serviços na entidade gerenciador de serviços e/ou serviços no registo UDDI. Clientes móveis requisitam todos os serviços do gerenciador de serviços ou algum tipo de serviço específico. Para a descoberta de um tipo particular de serviço poderá ser utilizado algum meio de autenticação..

- **Descrição da entrega de serviços:** A descrição dos serviços (parâmetros, operações

realizadas, etc) e definida numa mensagem XML que é enviada pelo gerenciador de serviços para a aplicação de cliente residente nos dispositivos moveis. A aplicação de cliente processa a informação recebida e mostra-a ao utilizador móvel.

- **Invocação de serviços:** O gerenciador recebe um pedido codificado como mensagem XML com a informação necessária (Nome do *Web Service*, operação seleccionada, valores de parâmetros introduzidos, etc) de um dispositivo móvel quando um utilizador está interessado em algum serviço que tenha sido previamente entregue ao dispositivo móvel no passo descrição de entrega de serviços. Então, o gerenciador de serviços faz a invocação do *Web Service* seleccionado, usando a interface de invocação dinâmica, ao provedor de serviços.

- **Transmissão resultante:** O gerenciador de serviços envia a informação codificada como mensagem XML para o utilizador móvel, quando recebe a resposta do provedor de serviços correspondente. A informação é então mostrada no ecrã do dispositivo móvel.

6.2 - Escolha da ferramenta de simulação

Como foi visto no capítulo 5 da presente dissertação, há várias maneiras de simular uma rede V2V, quer acoplando um simulador de tráfego a um simulador de redes de comunicação, quer utilizando um simulador que integre estes dois num só. No início do trabalho, ponderou-se usar um único simulador que integrasse todas as funcionalidades requeridas, sendo o escolhido entre os dois apresentados na secção 5.5 (GrooveNet e DIVERT) o DIVERT. Tal escolha residiu nos factos de este simulador cumprir as exigências do trabalho que se pretendia desenvolver e da aplicação simular, neste momento, a rede de tráfego do Porto, que é o cenário que se previa estudar neste trabalho. Pretendia-se assim, adaptar este simulador as características da arquitectura SOA. Para além disso havia, a vantagem da proximidade com a instituição que desenvolveu este projecto. Após contactos com a referida instituição ficou-se a saber que a aplicação não seria disponibilizada em versão *open source* nem para uso académico.

Partiu-se assim para a hipótese do Groovenet, adoptando-se a mesma estratégia que se tinha pensado para o DIVERT. O simulador foi instalado, estudado e foram contactados os seus programadores. Foram feitas várias simulações e começou-se a adopção deste para a arquitectura projectada. Contudo durante o decorrer deste trabalho foi concluída a implementação do primeiro protótipo funcional da plataforma MAS-T²er Lab em desenvolvimento no laboratório em que se desenvolveu esta dissertação. Decidiu-se, assim, implementar de raiz a comunicação veículo-a-veículo no simulador de trânsito desenvolvido de forma a deixar um contributo para este simulador e para a preparação de desenvolvimentos futuros neste.

Capítulo 7

Desenvolvimento do protótipo

Nesta secção é apresentada a concepção e desenvolvido de um protótipo funcional que implementa algumas das funcionalidades da arquitectura proposta no âmbito da plataforma MAS-T²er Lab.

7.1 - Plataforma MAS-T²er Lab

A plataforma MAS-T²er Lab (*Laboratory for MAS-based Traffic and Transportation Engineering Research*) [57] [58] é um sistema multi-agente integrado que aplica uma avaliação metodológica dos conceitos actuais das soluções inteligentes de transportes através do conceito de agentes. O domínio da aplicação MAS-T²er Lab é conceptualizado em termos de agentes e três subsistemas básicos são identificados, nomeadamente: o *real world*, o *virtual domain* e o *control strategies and management policies inductor*. O subsistema *real world* designa o sistema de transporte real em áreas urbanas, onde componentes físicos, como veículos, sistemas de controlo de tráfego e soluções inteligentes de transportes coabitam e interagem. Estes componentes são replicados para o *virtual domain* onde são modelados na forma de agentes. Os agentes de software no *virtual domain* devem emular o comportamento individual dos componentes físicos no *real world*. Finalmente, o *control strategies and management policies inductor* é um subsistema formado por agentes inteligentes, tanto humanos como artificiais, que observam a população sintética do *virtual domain* e podem directamente interagir com ela e aplicar regras de controle que alteram o comportamento de alguns elementos por forma melhorar a performance geral. A interacção entre estes três subsistemas é dinâmica e iterativa, permitindo intervenções em tempo real no mundo real (*real world*). A figura seguinte ilustra os três subsistemas básicos da plataforma MAS-T²er Lab e a forma como eles se interligam.

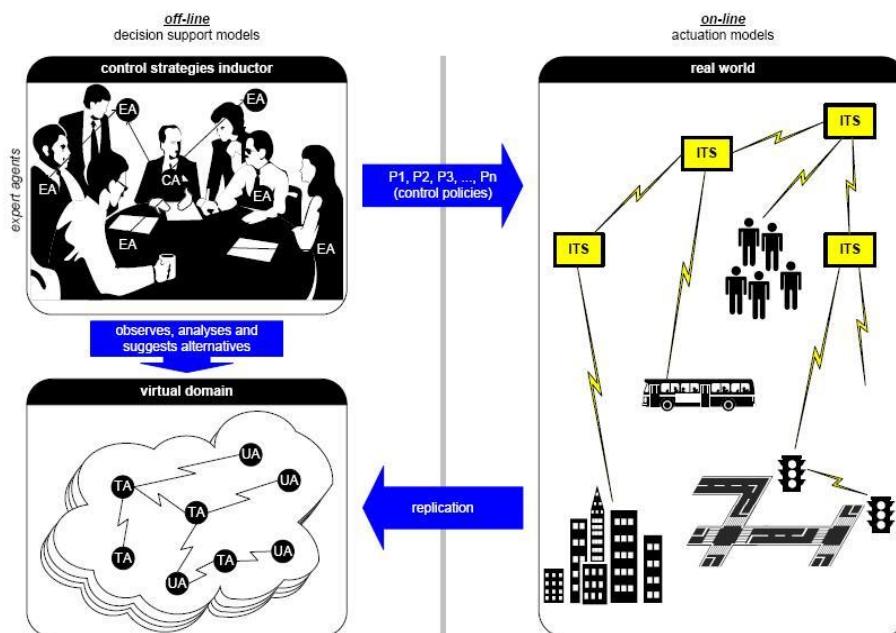


Figura 7.1 - Plataforma MAS-T²er Lab [57]

O primeiro passo que foi realizado para realizar uma plataforma tão cooperativa como o MAS-T²er Lab foi modelar e implementar o subsistema *Virtual Domain*.

O domínio virtual é um dos principais subsistemas da plataforma MAS-T²er Lab. Basicamente pode ser visto como a ferramenta de simulação que suporta representação detalhada de cenários baseada em agentes que representam os elementos da maneira mais realista possível. O *virtual domain* foi num passado recente projectado e um protótipo funcional já se encontra em funcionamento no NIAD&R.

Na secção seguinte serão descritos os componentes fundamentais que foram necessários à implementação da ferramenta de simulação. Detalhes mais precisos de como essa ferramenta foi implementada podem ser encontrados em [59].

7.2 - Arquitectura da ferramenta de simulação

A ferramenta de simulação foi descrita na secção anterior como sendo o subsistema *virtual domain* da plataforma MAS-T²er Lab. A sua especificação foi já alvo de estudo [59]. A solução proposta para a sua arquitectura de alto nível e todos os seus principais componentes são descritos e ilustrados na figura 7.2.

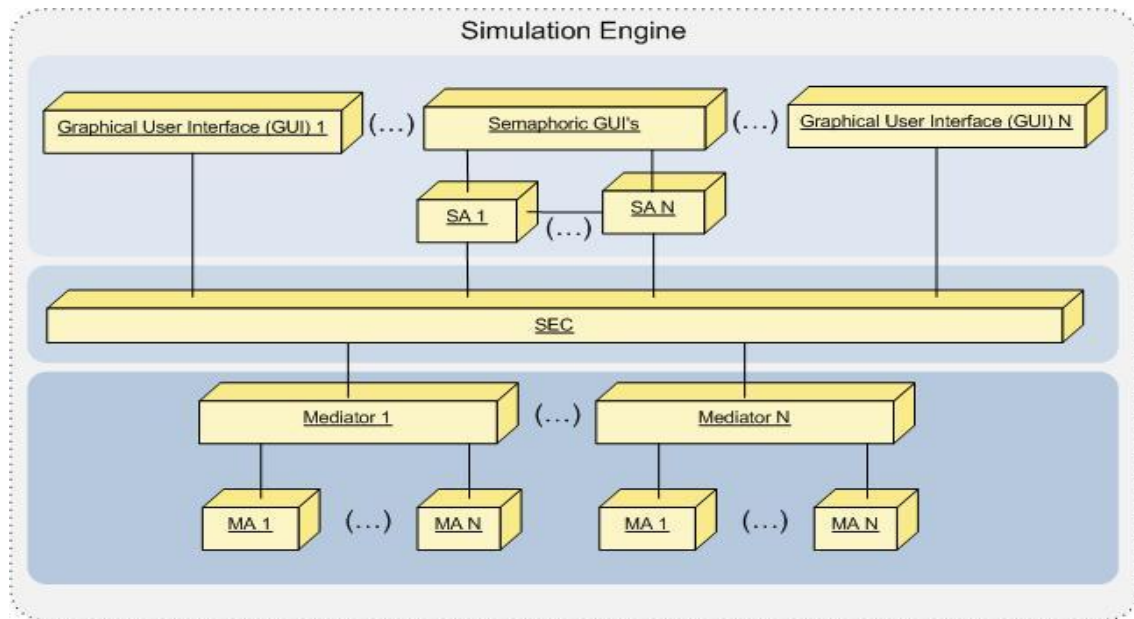


Figura 7.2 - Arquitectura de alto nível da ferramenta de simulação [59]

Como pode ser visto na figura 7.2, a ferramenta de simulação é altamente distribuída e é composta por várias aplicações independentes

A principal aplicação da ferramenta de simulação é o **Simulator Engine Controller (SEC)**. Basicamente o SEC é responsável por receber e permitir acções provenientes de todas as aplicações e executá-las. É também responsável por enviar o estado actualizado da simulação sempre que é necessário.

Os **Moveable Agent (MA)** são agentes que controlam as entidades móveis da rede. Basicamente recebem informação em intervalos de tempo regulares acerca do ambiente que os rodeia e decidem acções que gostariam de executar sobre as entidades que controlam. Cada MA controla apenas uma entidade da rede e as acções que gostaria de executar na respectiva entidade têm que ser enviadas para a aplicação *mediator* que será responsável por executar essas acções.

A aplicação **Mediator** é responsável por coordenar um determinado número de MA. Esta coordenação inclui fazer a translação da informação relativa ao estado do mundo enviada pelo SEC em intervalos de tempo regulares para o ambiente que rodeia cada agente que coordena. Também recebe acções dos agentes, agrega-as, calcula e actualiza as entidades afectadas por essas acções e envia a informação de actualização de novo para o SEC. As aplicações *mediator* também são responsáveis por lançar novos agentes nos pontos de

entrada da área da rede que controlam, bem como eliminar os agentes que alcançam qualquer um dos nós de saída nessa área de controlo.

A aplicação **Graphical User Interface** é responsável por receber o estado de simulação da rede em intervalos de tempo regulares e passá-los para uma interface perceptível pelo utilizador.

As aplicações **Semaphore Agent (SA)** são agentes responsáveis por controlar intersecções com sinalização luminosa (semáforos) com o objectivo de melhorarem o fluxo de trânsito.

7.3 - Aplicação *Simulation Engine Controll*

Como foi mencionado na secção anterior, a principal aplicação da ferramenta de simulação é a unidade SEC. A solução prevista para a sua arquitectura física é ilustrada no diagrama de desenvolvimento UML apresentado na figura 7.3, a seguir, e os seus componentes descritos, de acordo com [59].

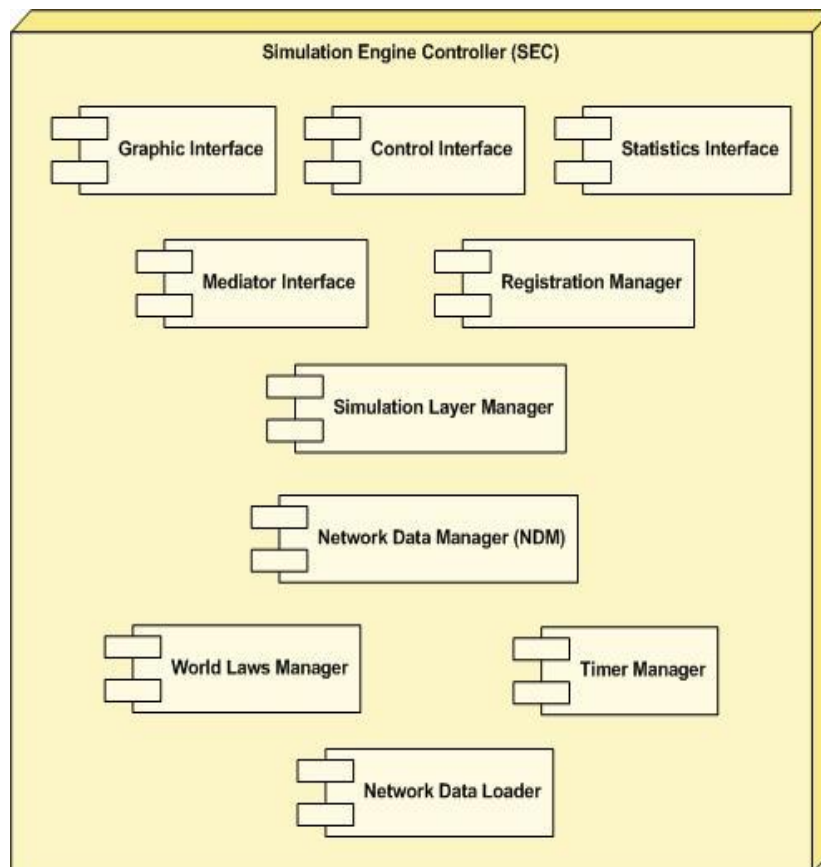


Figura 7.3 - Arquitectura física da aplicação SEC [59]

Graphic interface - Este componente é a interface gráfica de comunicação do SEC que permite que ele mande, em intervalos de tempo regulares, informações do estado da rede para as aplicações de interface gráfica dos utilizadores.

Control interface - Este componente é a interface de comunicação do SEC que permite receber acções de controlo dos utilizadores, bem como acções dos SA.

Satistics Interface - Este componente é a interface de comunicação do SEC que permite enviar, sempre que são pedidas, informações estatísticas, da simulação tais como fluxo de tráfego numa certa área, estrada específica ou numa intersecção específica, informação de controlo semafórico, etc.

Mediator interface - Este componente fornece ao SEC uma interface que permite interacções de comunicação com a aplicação mediator.

Registration Manager - Este componente permite o registo dos agentes inteligentes que podem ser operados por especialistas (agentes que pertencem ao subsistema *Control strategies and Management Policies Inductor*) reais ou artificiais.

Simulation Layer Manager - Este componente permite a filtragem da simulação. Isto quer dizer que é possível filtrar a informação da rede baseada em áreas de especialização específicas. Este mecanismo de filtragem é imperativo pois permite que diferentes especialistas e partições comuniquem enquanto se estuda diferentes perspectivas o mesmo cenário de simulação.

Timer Manager - Permite ao SEC a habilidade de gerir e coordenar o tempo. Se é necessário fazer qualquer ajuste à unidade de tempo de simulação (*simulation time step*), o *time manager* ajusta-o incrementando ou decrementando o *simulation time step*.

Network Data Manager - É o componente fundamental do SEC e contém toda a informação de actualização da rede durante uma simulação. Proporciona também recursos de edição cooperativa da rede incluindo definição de cenário.

Word laws Manager - É o componente que proporciona recursos que permitem editar, salvar e actualizar as regras do mundo, tais como prioridade à direita, definição de sinais de trânsito e regras de transparência ou opacidade dos objectos.

Network Data Loader - Este componente proporciona ao SEC a habilidade de manter a persistência dos dados. Tal permite salvar simulações, reactivar simulações previamente pausadas, carregar redes que se encontram em bases de dados georreferenciadas e em ficheiros (XML files e shape files) e inserir regras do mundo.

Estudados os componentes da plataforma MAS-T²er Lab, tendo particular ênfase na arquitectura da ferramenta de simulação SEC, partiu-se para o projecto e implementação de novas funcionalidades nesta que permitissem a simulação de uma rede V2V.

Assim, a solução projectada passará por adicionar um novo componente à arquitectura física do SEC, a V2V interface como se pode ver na figura 7.4 a seguir.

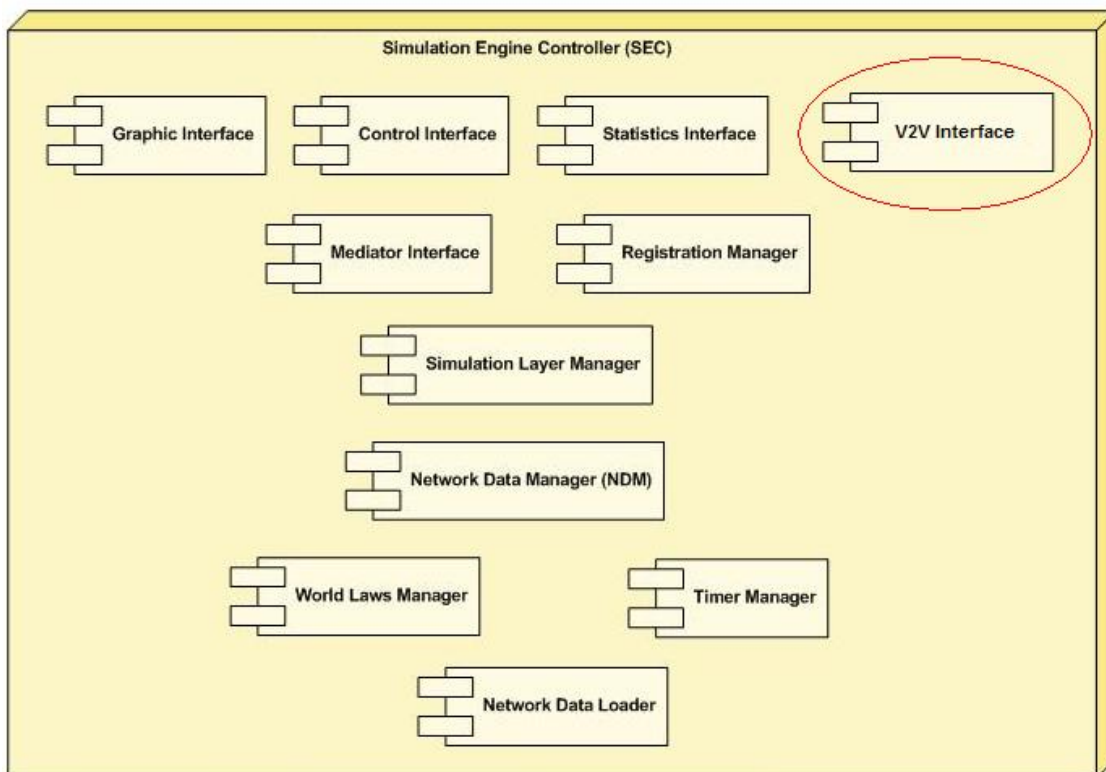


Figura 7.4 - Arquitectura física da aplicação SEC com comunicação V2V

V2V Interface - Este componente é a interface gráfica de comunicação do SEC que permite enviar, sempre que são pedidas, informações da simulação da comunicação veículo-a-veículo, tais como a posição global dos veículos, o seu raio de comunicação e os outros veículos que se encontram dentro dele bem como as mensagens enviadas e recebidas pelos diferentes veículos. Manda em intervalos de tempo regulares informações do estado da rede V2V para as aplicações de interface gráfica dos utilizadores. Permite também receber acções de controlo dos utilizadores, tais como enviar mensagens.

7.4 - Implementação

Como já foi referido, um protótipo funcional de ferramenta de simulação foi implementado num passado recente.

A ferramenta de simulação foi desenvolvida em C++ usando uma versão open source da plataforma de aplicação QT. Esta plataforma oferece uma API intuitiva e rica em bibliotecas C++ e ferramentas integradas para desenvolvimento de GUI. Mais detalhes desta aplicação podem ser encontrados em [60].

A arquitectura do protótipo desenvolvido e que serviu de base ao trabalho realizado é apresentada na figura 7.5, a seguir.

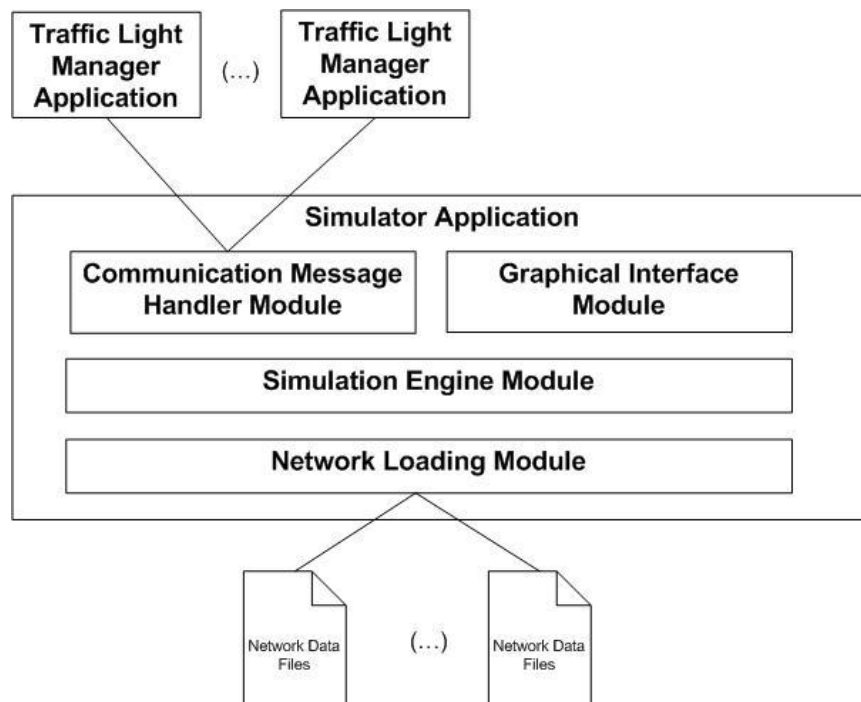


Figura 7.5 - Arquitectura da ferramenta de simulação [59]

A arquitectura do protótipo desenvolvido é semelhante à solução descrita nas secções anteriores. A descrição dos componentes desta arquitectura e detalhes de funcionamento do protótipo referido podem ser encontrados em [59].

Como foi retratado na secção anterior, as alterações a efectuar ao protótipo disponível para permitir a implementação da comunicação passam pela criação da *V2V Interface*.

Tendo em atenção a arquitectura do protótipo disponível e o estudo efectuado relativo ao seu funcionamento, decidiu-se criar uma classe *V2VInterface* que gera um vector com referência a todos os carros simulados. Nos nós de entrada da rede, sempre que um carro é criado, a classe *V2VInterface* atribui um ID ao carro e adiciona-o ao vector. Nos nós de saída da rede, sempre que um carro é apagado, é apagado também na classe *V2VInterface*, ou seja, é eliminado do vector. O acesso é directo nesta fase porque os ID são referentes à posição do vector. Nesta fase os ID são também actualizados. Este ciclo de execuções é descrito nos diagramas de estados apresentados na figura 7.6, a seguir.

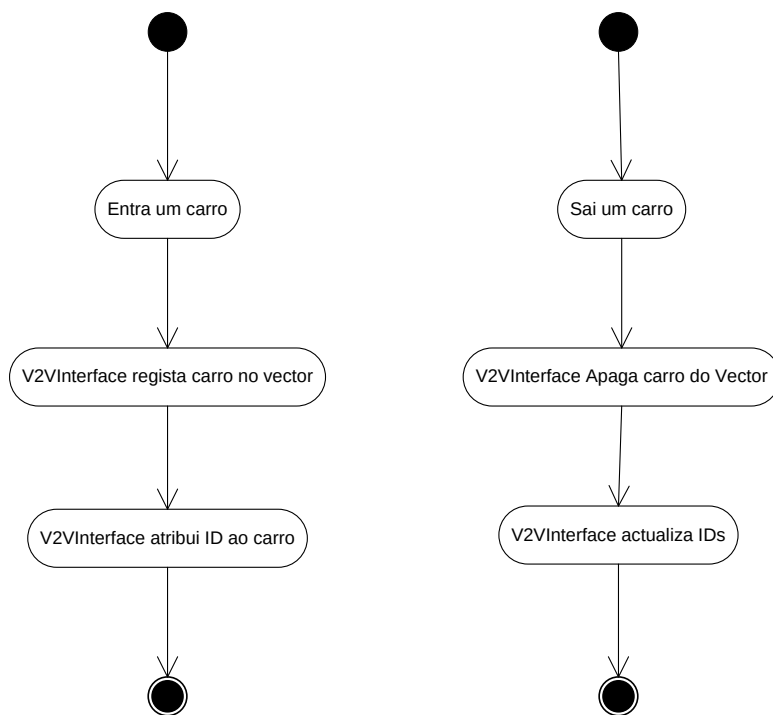


Figura 7.6 - Diagrama de estados do ciclo de execução da obtenção da referência aos carros simulados

Durante a implementação das execuções descritas foram observados alguns problemas que resultavam, sobretudo, da ocorrência de acessos múltiplos e simultâneos ao vector e também quando se tentava aceder a uma posição vazia deste, pois o carro já tinha sido eliminado na aplicação e a classe *V2VInterface* ainda não tinha actualizado o vector. Estes problemas foram solucionados através do uso de *mutexes*. *Mutex* é uma abreviatura para *mutual exclusion object*. *Mutexes* são métodos usados para garantir que duas *threads* não tentem aceder à memória ao mesmo tempo. Permitem assim que threads partilhem recursos mas não simultaneamente.

O passo seguinte foi implementar a detecção da posição global dos carros. Um problema do protótipo disponível era que a translação das *lanes* era sempre feita em x. A solução encontrada para este problema foi efectuar uma rotação à lane. Ou seja, ver o ângulo de rotação da *lane*, actualizar as posições e fazer a translação correcta. Isto é conseguido através de fórmulas matemáticas como é demonstrado no excerto de código apresentado a seguir, na figura 7.7.

```

GlobalPosition Car::getGlobalPosition()
{
    this->lock.lock();

    short direction = 1;
    if (currentLane->positiveDirection == false){
        direction = -1;
    }

    double posX = cos (this->currentRoad->GetInclinationAngleRad()) * position + (direction *
this->currentRoad->GetInitialCoordinates().posx);
    double posY = (this->currentRoad->GetInitialCoordinates().posy) + sin
(this->currentRoad->GetInclinationAngleRad()) * position;

    GlobalPosition gP;
    gP.gPosX = posX;
    gP.gPosY = posY;
    this->lock.unlock();
    return gP;
}

void Car::translate(double dt)
{
    this->lock.lock();
    position=position+dt*velocity;
    positionX=positionX+cos(this->currentRoad->GetInclinationAngleRad())*dt*velocity;
    positionY=positionY+sin(this->currentRoad->GetInclinationAngleRad())*dt*velocity;
    this->lock.unlock();
}

```

Figura 7.7 - Código que implementa a determinação da posição global

A seguir definiu-se um raio de comunicação para os veículos e implementou-se o envio de mensagens. O carro que é seleccionado para enviar mensagem, envia-a a todos os carros que se encontram dentro do seu raio de comunicação. Este envio é recursivo, assim todos os carros que recebem mensagens também a enviam aos carros que se encontram dentro do seu raio de comunicação e assim sucessivamente, gerando-se uma rede VANET.

Para determinar se um carro se encontra dentro do raio de comunicação de um veículo que envia a mensagem, verifica-se para todos os carros do vector se as suas posições globais verificam a seguinte condição matemática, apresentada a seguir.

$$\begin{aligned}
 GPosX &\leq GPosXe + raio \ \&\& \ GPoY \leq GPosYe + raio \ \&\& \\
 GPosX &\geq GPosXe - raio \ \&\& \ GPosY \geq GPosYe - raio
 \end{aligned}$$

Onde GposX é a posição global em X de um carro, GPosY é a posição global em Y de um carro, GPosXe é a posição global em X do carro que envia a mensagem, GPosYe é a posição global em Y do carro que envia a mensagem e raio é o valor do raio de comunicação do carro que envia a mensagem.

Para permitir ao utilizador interagir com a aplicação, foi criada uma janela (*QWidget*), denominada *V2V interface*, no simulador que apresenta uma listagem de todos os carros, apresentando o seu ID, a sua posição global tanto em X como em Y e a mensagem que contém. Foram implementadas funcionalidades nesta janela, tais como:

- Fazendo um clique sobre um dos veículos na tabela o veículo é seleccionado e o seu raio de comunicação é desenhado no simulador.
- Fazendo duplo clique sobre um dos veículos, a aplicação recolhe o ID do carro e é apresentada uma janela de envio de mensagem onde o utilizador pode especificar a mensagem a enviar. Carregando no botão “enviar” desta janela a mensagem é enviada para todos os veículos dentro do raio de comunicação do veículo que enviou a mensagem. A aplicação calcula os veículos através da posição global como foi descrito anteriormente. Todos os veículos que enviaram ou receberam a mensagem apresentam-na na janela *V2V Interface* no campo de mensagem e é desenhado o seu raio de comunicação em seu redor no simulador. O envio é recursivo, como foi descrito anteriormente.

De forma a distinguir as diferentes mensagens enviadas, para além de serem apresentadas no campo de mensagens dos respectivos veículos da janela *V2V Interface*, na janela de simulação os carros que se encontram a reencaminhar mensagens diferentes apresentam raios de comunicação de cores diferentes. As diferentes cores são obtidas pela aplicação através de seis combinações das cores primárias.

É apresentada na figura seguinte um *screenshot* que apresenta a interface gráfica do simulador e da janela *V2V Interface* durante o decorrer de uma simulação, onde são enviadas mensagens entre veículos.

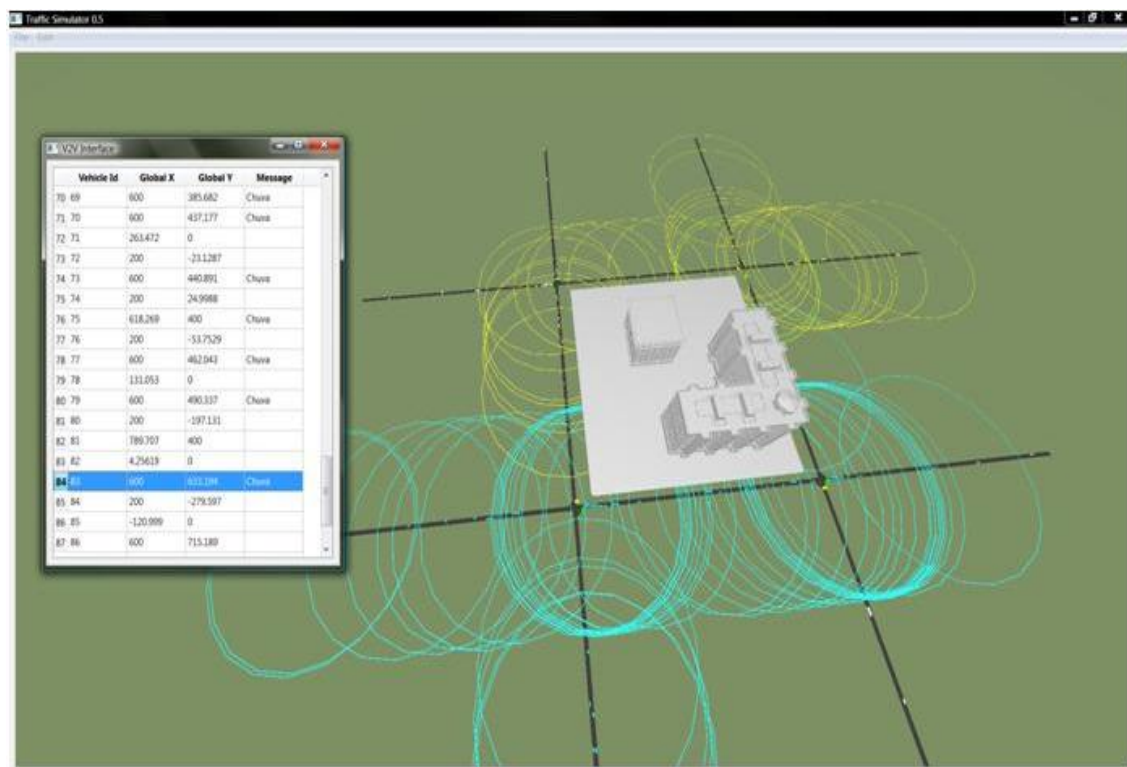


Figura 7.8 - Simulador com comunicação V2V implementada

Foi também implementada a comunicação externa do simulador através de Socket.

Um *socket* [61] representa uma conexão entre duas máquinas onde tal conexão funciona como um canal que permite a transmissão de dados de uma máquina para a outra de forma bidireccional, possibilitando o envio e recebimento simultâneo de dados. Isto significa que cada *socket* possui um canal de entrada e outro de saída, sendo que o que é enviado pelo canal de saída de uma máquina é recebido pelo canal de entrada da outra máquina e vice-versa.

Os sockets operam de dois modos principais, de acordo com [62]:

1. Modo orientado à conexão.

Os sockets orientados à conexão ou stream sockets operam como um telefone; eles têm de estabelecer uma conexão e suspender a ligação. Tudo o que flui entre esses dois eventos chega na mesma ordem em que foi transmitido. Além disso a entrega das mensagens é garantida. Este modo também é chamado de troca confiável de dados e baseia-se no uso do TCP (*Transmission Control Protocol*).

2. Modo sem conexão.

Os sockets sem conexão ou datagram sockets operam como um correio onde a entrega não é garantida. Além disso os diferentes itens recebidos da correspondência podem chegar em uma ordem diferente daquela em que foram enviados. Este modo também é chamado de troca não confiável de dados e baseia-se no uso do UDP (*User Datagram Protocol*).

Para que dois computadores comuniquem, cada um deles deverá utilizar um *socket*, num processo muito similar às acções de leitura e escrita de ficheiros. A única diferença é que tal ficheiro é realmente uma máquina remota que pode "decidir" o que fazer com os dados enviados ou solicitados. Geralmente, um dos computadores é chamado de servidor, e é responsável por abrir um *socket* e "ouvir" eventuais pedidos de conexão. O outro computador, denominado cliente, usualmente "chama" o *socket* servidor para estabelecer a conexão. Para isto é necessário apenas o endereço IP do servidor e o número de porto associado à aplicação com a qual se deseja "conversar". Esta forma de conexão é demonstrada na figura 7.9.

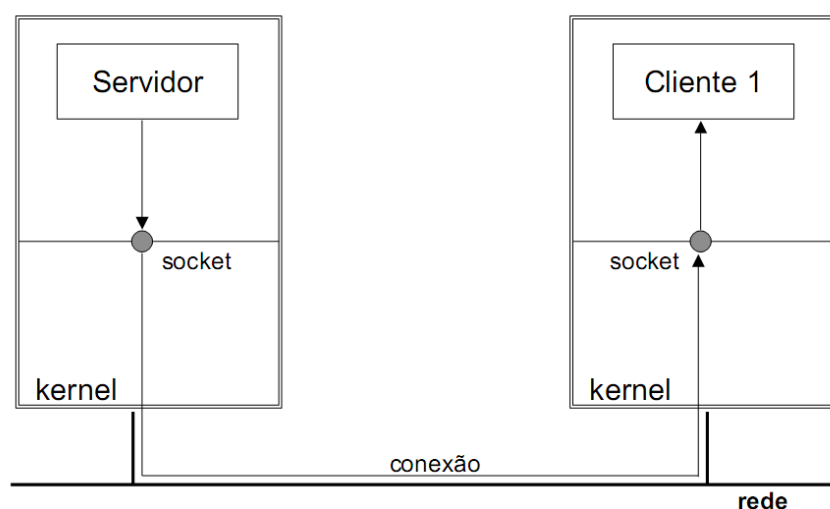


Figura 7.9 - Conexão via Socket

Tendo em conta as características apresentadas sobre a comunicação via *sockets*, foi implementado um *server socket*, do tipo *stream socket* via TCP. A porta associada a este foi a porta 8000.

Foi definido e implementado um protocolo de comunicação que é demonstrado na tabela 7.1 a seguir:

Tabela 7.1 - Protocolo de Comunicação

Cliente	Servidor
Envia comando “help”	Devolve a seguinte lista de comandos: COMMANDS: GETALLPOSITIONS SENDMESSAGE
Envia comando “GETALLPOSITIONS”	Devolve a posição global de todos os veículos no formato: [ID]posx_posy [IDn]posxn_posyn
Envia comando “SENDMESSAGE”	Devolve: USAGE: SENDMESSAGE [ID] [“message”] Que é o formato em que o cliente deve mandar a mensagem
Envia comando de envio de mensagem: SENDMESSAGE_ID_”Mensagem” Exemplo: SENDMESSAGE_35_chuva	Se o carro identificado pelo ID enviado já foi eliminado devolve: <i>ID out of range</i> Se o carro ainda se encontra no vector de carros, executa o mesmo ciclo de implementação que é executado fazendo duplo clique sobre um carro na janela V2V interface, enviando o carro seleccionado a mensagem que foi passada pelo servidor.

O passo seguinte necessário à implementação da funcionalidade SOA da arquitectura apresentada no capítulo 6 necessitava que os veículos/condutores estivessem implementados como agentes. Contudo, no protótipo disponível, os veículos não foram implementados como agentes, o que impossibilitou esta acção. Revelou-se temporalmente impossível alterar a ferramenta de simulação implementando os veículos como agentes, para além de tal acção ser também impossibilitada pelo facto de outros projectos estarem a utilizar o protótipo. Um agente é basicamente uma unidade autónoma que possui a habilidade de perceber o meio onde está inserida (através de sensores) e agir sobre este mesmo meio (através de actuadores). Agentes são capazes de se comunicar com outros agentes e tomar decisões “racionalis” ou seja, tomar decisões que resultam da inferência sobre aquilo que eles conhecem. A modelação dos veículos/condutores em agentes é necessária à implementação

da funcionalidade SOA porque, como foi definido na arquitectura proposta, os serviços são, em última análise, disponibilizados aos condutores e não ao objecto veículo. Portanto, é necessário haver uma excelente representação da unidade decisória (condutor) neste tipo de sistemas, assim como das unidades artificiais que o podem representar. Modelando os veículos como agentes permitiria que eles funcionassem como provedores e consumidores de serviços.

Contudo, o conceito da arquitectura proposta é provado. A rede base de comunicação V2V encontra-se implementada e a funcionar. O conceito de troca de informação na rede VANET é provado pela troca de mensagens e a implementação da comunicação exterior pode ser vista como a modelação da comunicação dos veículos com entidades exteriores (como entidades provedoras de serviços, PDAs, telemóveis, etc).

A interface gráfica de todo o simulador foi também melhorada, como trabalho extra. Essas melhorias passam por carregamento de modelos mais realistas dos veículos e do ambiente que os rodeia, fazendo-se o carregamento de alguns edifícios e imagens mais realistas do céu do cenário. A luminosidade do simulador foi também melhorada.

A melhoria dos modelos da interface gráfica foi conseguida após permitir, através de implementação, que a aplicação carregasse modelos STL (stereolitho).

STL funciona sobre o conceito de triangularização. Uma malha de polígono em um arquivo de STL é incluída de um conjunto de faces triangulares, ou seja, todos os polígonos são vistos como triângulos. Mais detalhes modelos STL podem ser encontrados em [63].

Implementou-se, assim, um *parser* que permite ler e manipular ficheiros STL.

Nas figuras seguintes demonstram-se as melhorias efectuadas na interface gráfica.

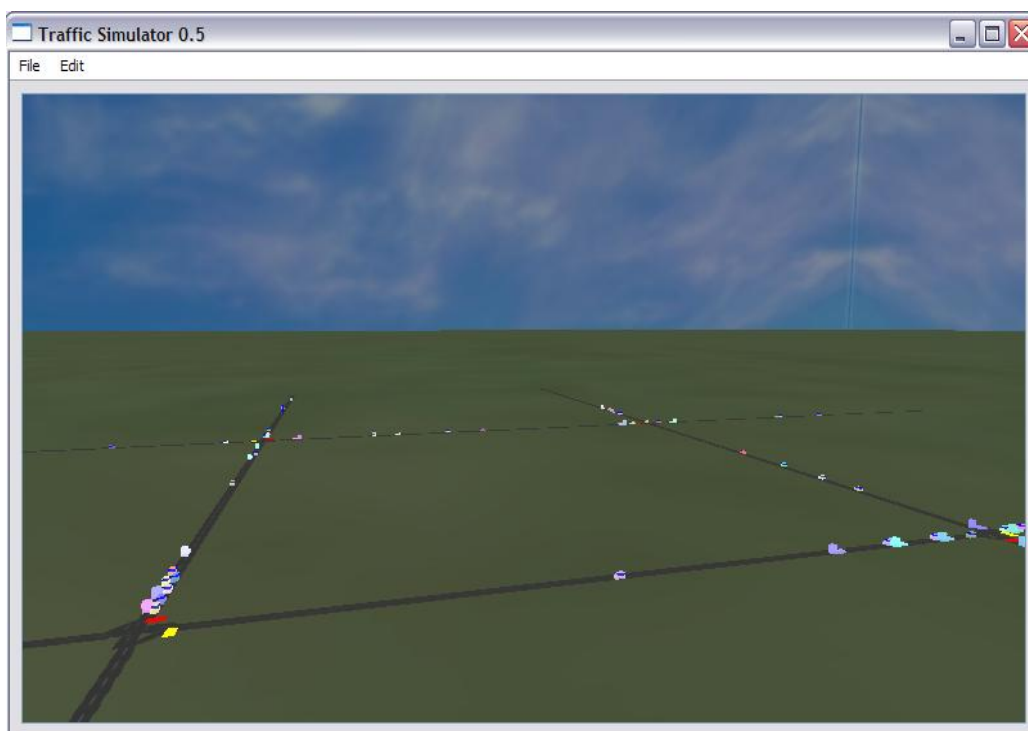


Figura 7.10 - Interface gráfica inicial do simulador

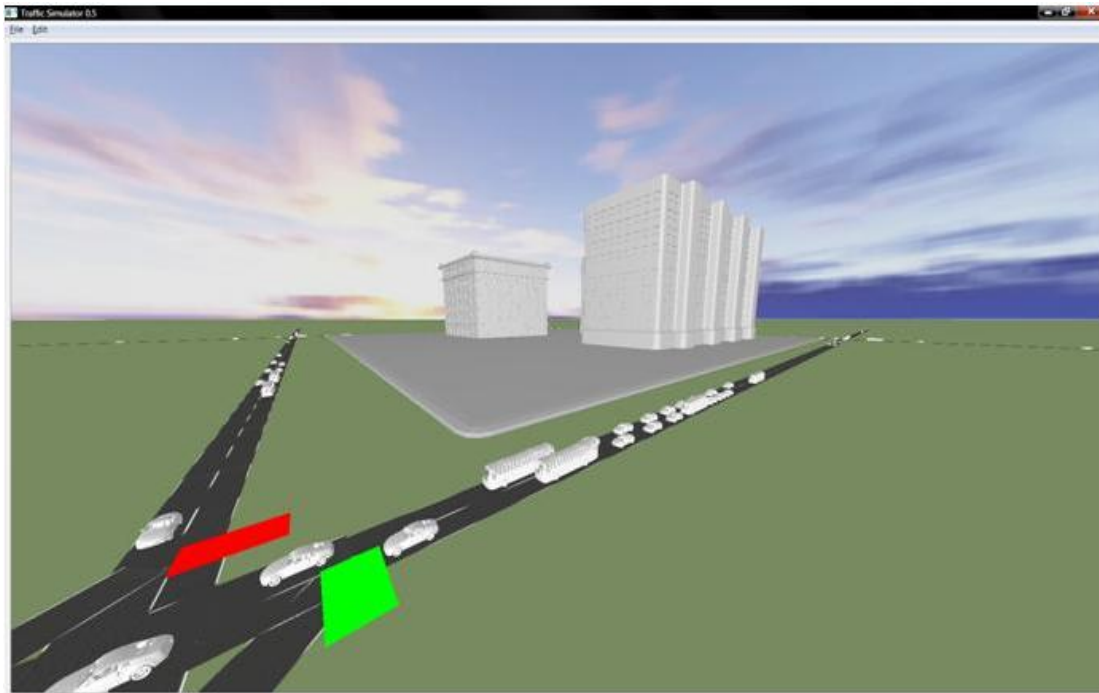


Figura 7.11 - Interface gráfica do simulador após adaptações implementadas

Um editor gráfico de redes está também a ser desenvolvido como parte do projecto MAS-T²er Lab. Este projecto foi acompanhado de perto no decorrer deste trabalho e o editor foi usado para produzir redes e cenários de teste. Actualmente este editor permite a construção de redes e exportação destas para o formato XML, pronto a ser carregado pelo simulador. A figura 7.12 ilustra a interface gráfica do editor de redes com a representação esquemática de uma rede a ser editada. A mesma rede é carregada no simulador.

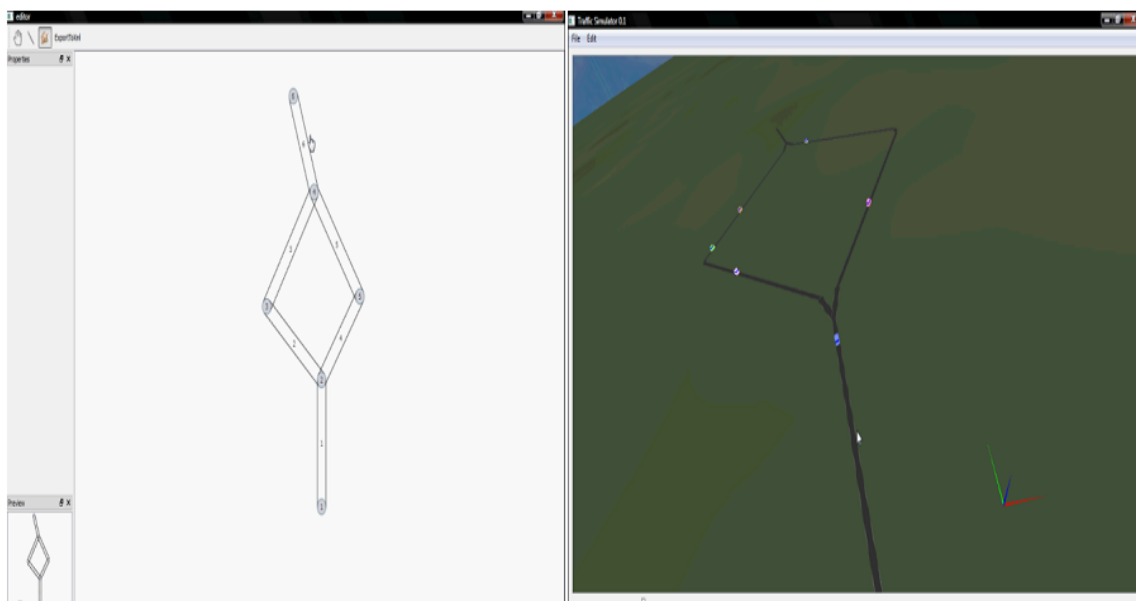


Figura 7.12 - Editor de rede gráfico e aplicação de simulação

Aquando da escrita desta dissertação estavam a ser implementadas funcionalidades para o carregamento de ficheiros shapefile pelo editor. Depois que esta funcionalidade estiver concluída, será possível simular áreas de maior extensão, nomeadamente partes da rede do Porto.

Capítulo 8

Análise de resultados

Este capítulo descreve os resultados obtidos na simulação de uma rede simples, aplicando algumas alterações a esta de forma a avaliar o output promovido pela ferramenta de simulação e da rede V2V. Os resultados obtidos são posteriormente analisados.

8.1 - Definição do cenário de teste

O cenário de teste definido é baseado numa rede muito simples. Essa rede é codificada no formato XML definido para o protótipo implementado. O código XML da rede usada pode ser consultado no Apêndice A. Um exemplo esquemático da rede é apresentado na figura 8.1. Neste esquema são visíveis a topografia da rede, quais os nós de entrada e saída da rede e a localização dos sinais luminosos.

Na figura 8.2 apresenta-se um screenshot de uma simulação. É possível ver a rede esquematizada carregada no simulador, a janela de interface V2V e a os veículos apresentando o seu raio de comunicação, com cores diferentes para troca de mensagens diferentes.

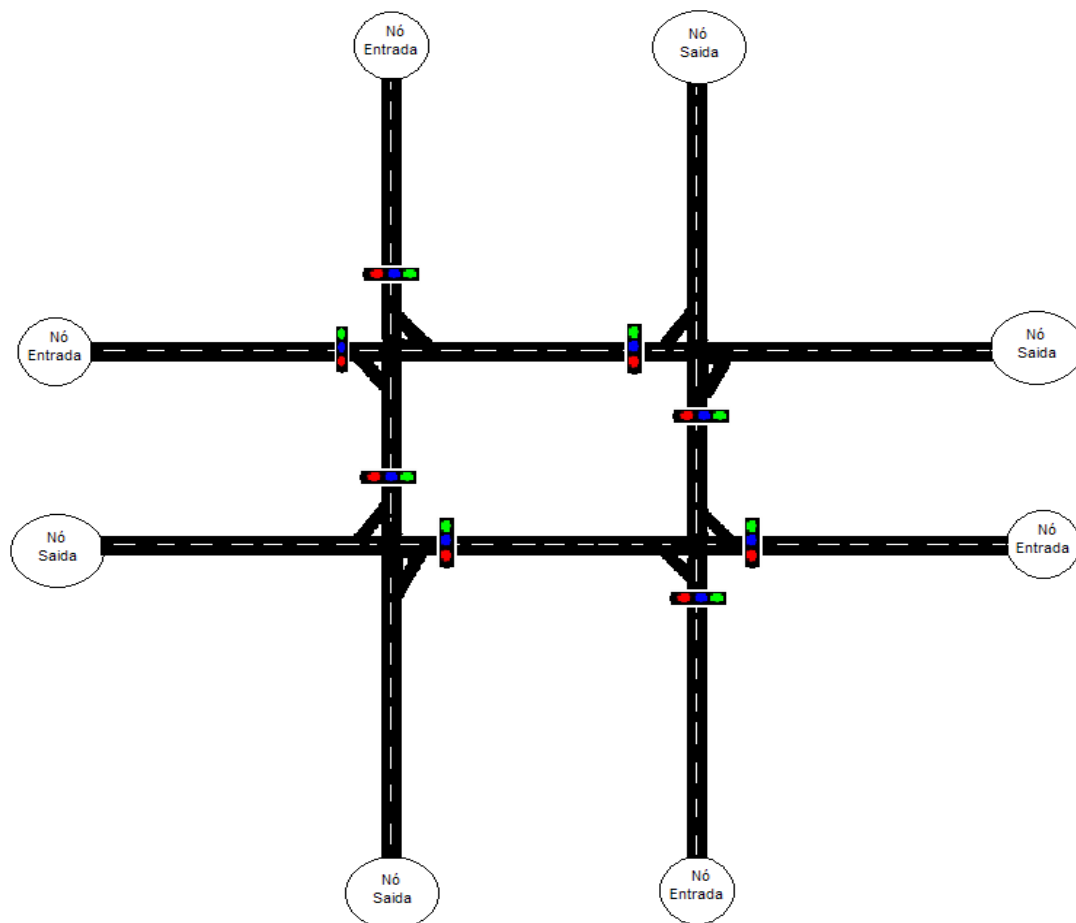


Figura 8.1 - Esquema da rede a usar nas simulações do cenário de teste

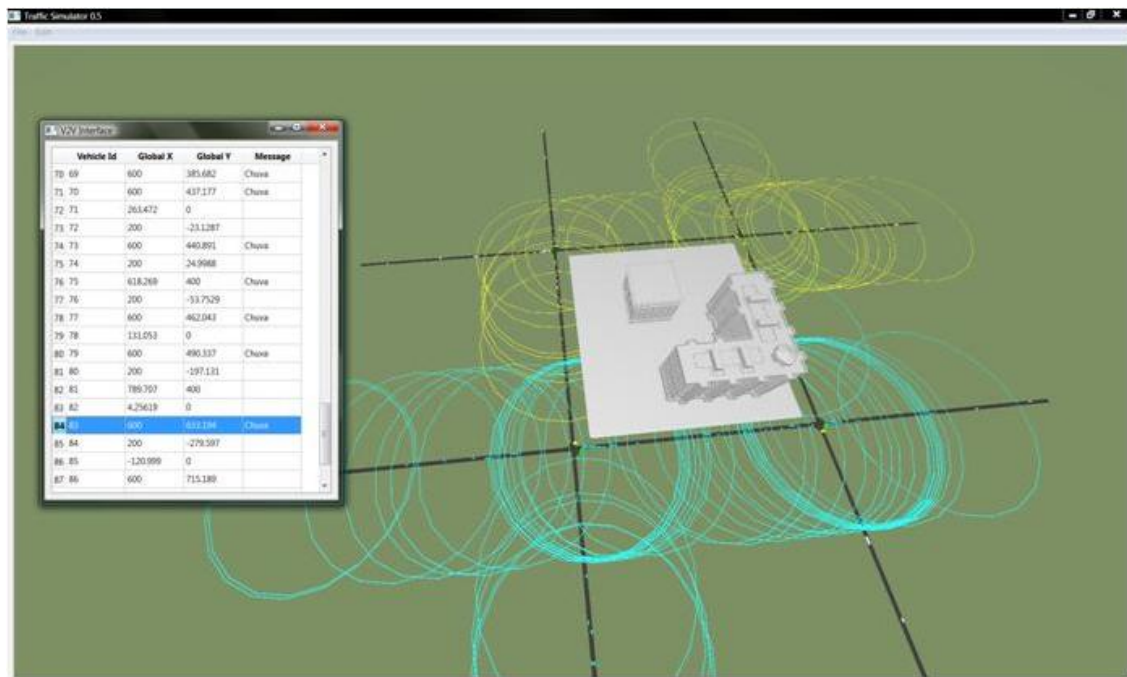


Figura 8.2 - Simulação da rede do cenário de teste

Foram definidas e testadas três configurações de trânsito com o objectivo de testar o comportamento da rede de comunicação V2V implementada no simulador.

1.ª Configuração: Cenário com baixo fluxo de trânsito

Neste cenário de teste as intersecções fonte gerarão fluxos de veículos entre os 100 e 250 veículos por hora.

2.ª Configuração: Cenário com fluxo médio de trânsito

Neste cenário de teste as intersecções fonte gerarão fluxos de veículos entre os 250 e 850 veículos por hora.

3.ª Configuração: Cenário com alto fluxo de trânsito

Neste cenário de teste as intersecções fonte gerarão fluxos de veículos entre os 850 e 1300 veículos por hora.

As diferentes configurações do cenário de teste foram inspirados no perfil diário de uma rede de tráfego urbana que normalmente é caracterizada por três cenários distintos. Baixo fluxo de trânsito (normalmente entre as 22h00 e as 06h00), fluxo médio de trânsito (normalmente entre as 12h00 e as 14h30) e alto fluxo de trânsito, as denominadas horas de ponta (normalmente entre as 07h00 e as 9h30 e entre as 19h00 e as 21h30).

Foram efectuadas várias simulações para todas as configurações propostas. Os resultados são apresentados e discutidos na secção seguinte com uma média dos valores obtidos nas diferentes várias simulações efectuadas.

Não foi definido como objectivos no plano de testes da simulação a avaliação dos modelos de *car-following* e *lane-changing* usados pelos condutores para decidir as suas acções. Os parâmetros usados para os modelos nas simulações são os sugeridos pelos autores [64] [65].

Os parâmetros usados em todas as configurações de teste descritas neste capítulo foram sempre os mesmos e encontram-se descritos na tabela 8.1, a seguir.

Tabela 8.1 - Parâmetros para *Car-following* e *Lane-changing* usados nas simulações

Parâmetro	Condutores/carros	Condutores/autocarros
Politeness Factor	Entre 1% e 50%	Entre 0% e 50%
Maximum Safe Deceleration	Entre 3 m/s ² e 6 m/s ²	Entre 3 m/s ² e 6 m/s ²
Threshold	0.2	0.2
Bias to the right Lane	0.2 m/s ²	0.2 m/s ²
Desired velocity	Aleatória entre 35 km/h e 100 km/h	Aleatória entre 30 km/h e 80 km/h
Time headway	Aleatória 1s e 1.5s	Aleatória entre 1.3s e 2s
Minimum gap	2.0m	2.0m
Acceleration	0.5 m/s ²	0.3 m/s ²
Deceleration	2.0 m/s ²	1.0 m/s ²

Como se depreende da observação da tabela, dois tipos diferentes de condutores e claro de veículos com parâmetros diferentes são usados nas simulações. A percentagem dos dois tipos de Condutores/veículos é a mesma para todas as configurações de teste, ou seja, 20% de carros e 80% de autocarros.

A atribuição de viagens (*trip assignment*) ajustada para todas as intersecções fonte da rede definida para o cenário de teste é também a mesma em todas as configurações de teste.

O controlo semaforico também não é tido em conta nas simulações descritas.

Todas as configurações de teste correram com o mesmo valor de *time step*, 100ms. Os tempos de simulação são acelerados três vezes em relação ao tempo real o que significa que cada 100ms no tempo real correspondem a 300ms em tempo simulado.

O período de tempo real durante o qual decorreram as simulações foi de 10min que correspondem a 30 min no tempo de simulação. Periodicamente, de dois em dois minutos, um carro que se encontra numa das intersecções envia uma mensagem. Foi definido que os carros enviariam mensagens nas intersecções para maximizar o número de carros no alcance pois assim o raio de comunicação abrange vários segmentos de vias e carros movendo-se em várias direcções. A intersecção onde se envia a mensagem varia no sentido do ponteiro dos relógios.

Para além das configurações de teste definidas anteriormente, foram também definidas testes preliminares básicos para a rede de comunicação V2V. Esses testes consistiram em testar se a rede é correctamente formada, verificando se os veículos enviam mensagens para todos os veículos no seu raio de comunicação, se a mensagem enviada e recebida é a mesma e se esta é encaminhada de forma recursiva. Estes testes são efectuados correndo uma simulação e observando o output gráfico do simulador e da janela V2V interface implementada.

Foram também efectuadas algumas simulações alterando o parâmetro velocidade dos veículos e o seu raio de comunicação para ver a influência destes na propagação da mensagem. Efectuaram-se simulações em que os veículos não enviam mensagens nas intersecções mas sim noutros pontos da rede e simulações com envio de várias mensagens por vários carros.

Por último, foi também testada a comunicação externa implementada. Este teste consistiu em implementar um socket cliente noutro computador e tentar a partir deste obter a lista de todos os carros e enviar mensagens a partir de um carro específico.

8.2 - Testes básicos

Estes testes, como foi descrito na secção anterior, são feitos correndo uma simulação, enviando uma mensagem e observando o output gráfico do simulador e da janela V2V Interface. Procedendo a esta sequência de acções várias vezes foi observado que a rede de comunicação V2V funciona correctamente, já que sempre que se envia uma mensagem a partir de um veículo todos os veículos no seu raio de comunicação recebem a mensagem. Isto foi testado não permitindo recursividade no envio das mensagens e verificando por observação que todos os carros no raio de comunicação do veículo que envia a mensagem a recebem, verificando no simulador que todos os carros que se encontravam dentro do raio

ficaram com raio de comunicação activo (desenhado) e que na janela V2V Interface apresenta-se a mensagem correspondente à mensagem enviada. Para facilitar a observação no simulador, decidiu-se, temporariamente, que apenas o carro que envia a mensagem desenha o seu raio de comunicação e os carros que a recebem, em vez de desenharem o seu raio mudam de cor.

Seguidamente activou-se a funcionalidade de recursividade no envio das mensagens e verificou-se mais uma vez por observação que esta funcionalidade funcionava correctamente. Estas observações são possíveis aplicando um fluxo de tráfego baixo no simulador que permite seguir a mensagem de carro para carro através de observação directa do simulador.

8.3 - Resultados das Simulações das diferentes configurações do cenário de teste

8.3.1 - Cenário com baixo fluxo de trânsito

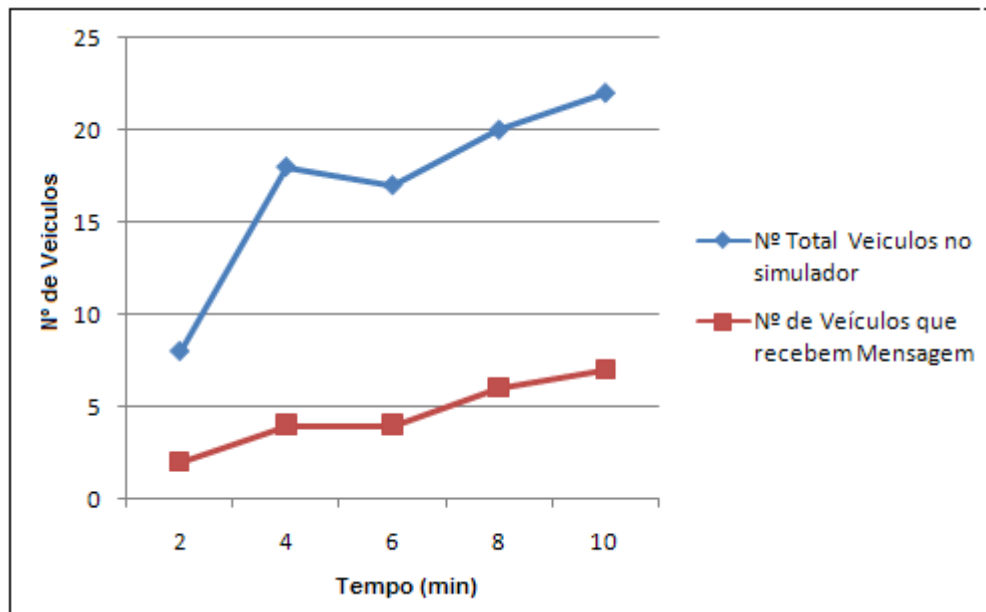


Figura 8.3 - Comparação entre o número de veículos presentes no simulador e o número de veículos que se conectam via V2V num cenário com baixo fluxo de trânsito

Como era de prever, apenas um número pequeno de veículos se consegue conectar e receber a mensagem. Este cenário reporta para as horas de baixo fluxo de uma rede urbana, podendo também ser visto como a simulação de uma área rural que por norma contém um fluxo reduzido de trânsito.

8.3.2 - Cenário com fluxo médio de trânsito

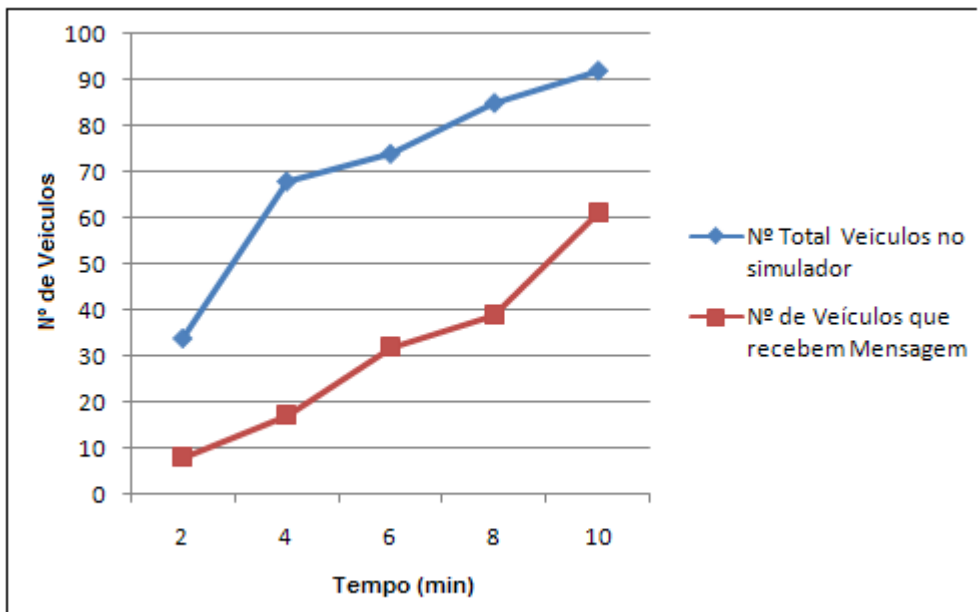


Figura 8.4 - Comparação entre o número de veículos presentes no simulador e o número de veículos que se conectam via V2V num cenário com fluxo médio de trânsito

Nos cenários com fluxo médio de trânsito o número de veículos que se consegue conectar e receber a mensagem aumenta. Cerca de metade da rede de tráfego é coberta pela comunicação V2V. A disseminação de uma mensagem de controlo de tráfego já se verifica viável neste tipo de cenários pois um número de veículos razoável recebe essa mensagem podendo optar por rotas alternativas.

8.3.3 - Cenário com alto fluxo de trânsito

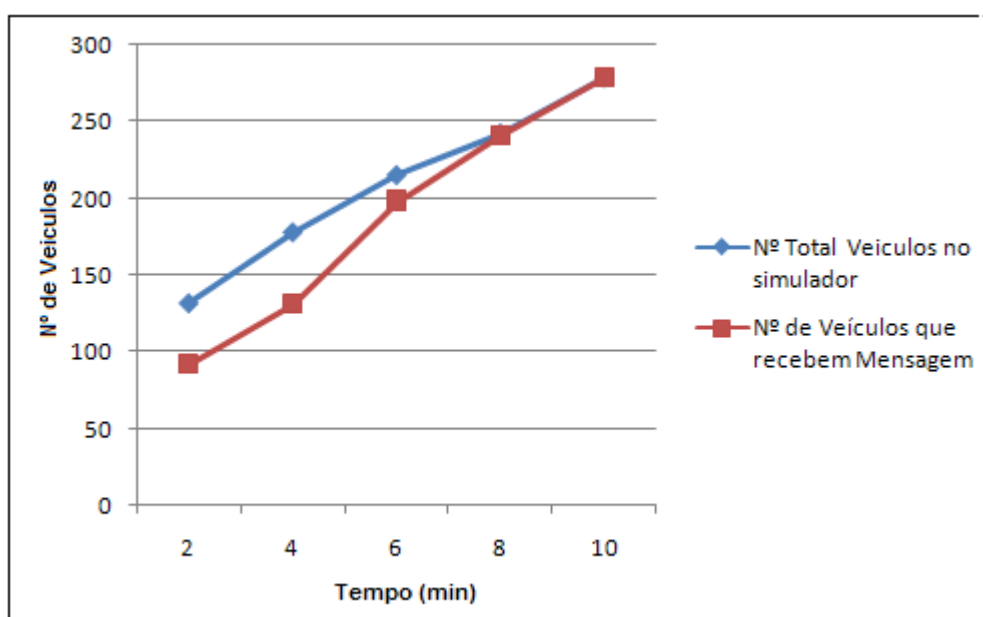


Figura 8.5 - Comparação entre o número de veículos presentes no simulador e o número de veículos que se conectam via V2V num cenário com alto fluxo de trânsito

Nos cenários com alto fluxo de trânsito verificou-se que a maioria dos veículos presentes no simulador consegue conectar-se e receber a mensagem. Na maioria dos casos a totalidade da rede de tráfego é coberta pela rede de comunicação V2V. Verificou-se também que a percentagem de rede coberta aumenta ao longo do tempo devido à formação de congestionamentos nas intersecções com controlo semafórico, o que leva a um aumento do número de veículos presente no cenário em simultâneo e consequentemente aumentando também o número de nós disponíveis para a comunicação V2V.

8.3.3 - Comparação entre os cenários de teste

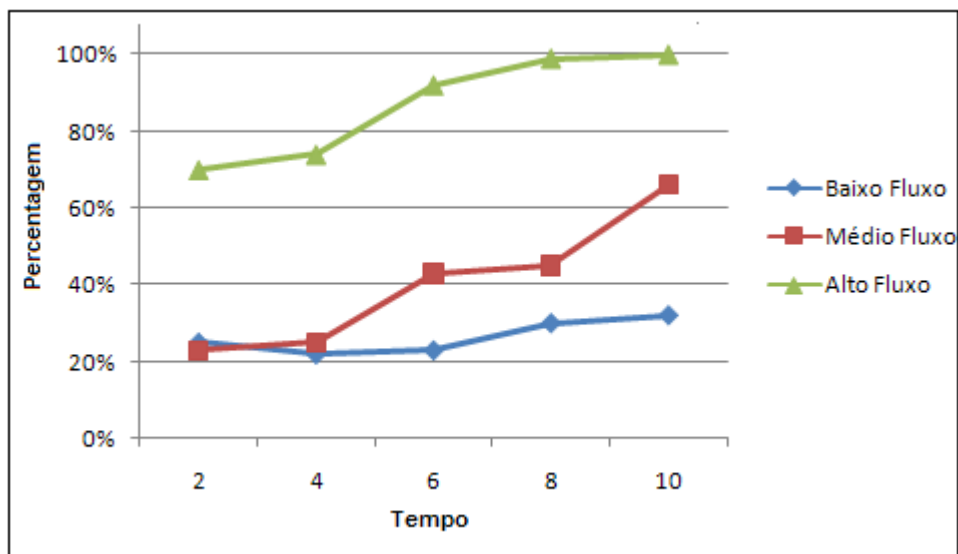


Figura 8.6 - Comparação da percentagem de rede coberta pela rede V2V com as diferentes configurações do cenário de teste ao longo do tempo de simulação

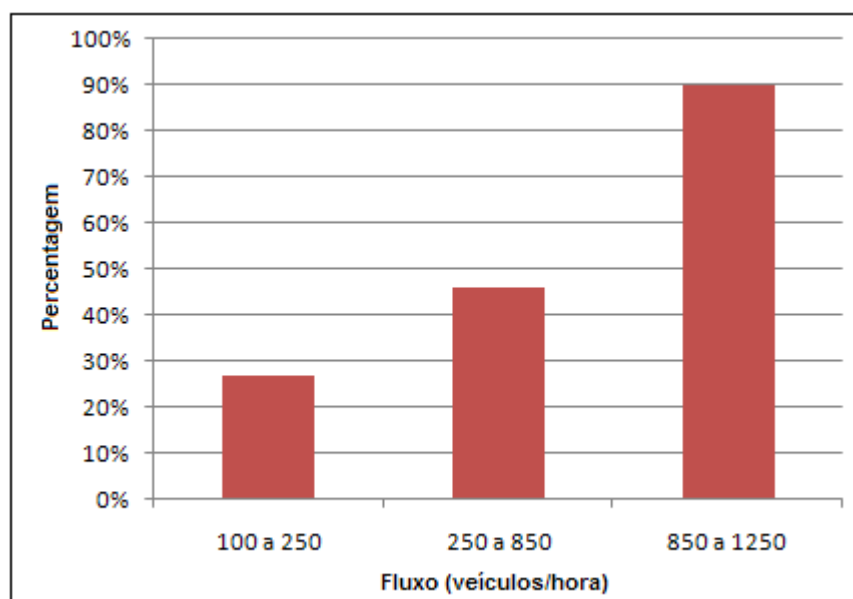


Figura 8.7 - Percentagem média de rede coberta pela rede V2V com as diferentes configurações do cenário de teste

Tal como seria de esperar, os resultados obtidos da simulação das diferentes configurações do cenário de teste revelam que, em redes com baixo fluxo a disseminação da mensagem via comunicação V2V é muito pobre, resultando numa pequena cobertura da rede (27%), muitas vezes, com este tipo de fluxo, a mensagem nem chega a alcançar nenhum veículo. Nas redes com fluxo médio de tráfego cerca de metade da rede é coberta. A rede só foi coberta na sua totalidade nas redes com alto fluxo de tráfego.

Em todas as simulações há um aumento da área coberta pela rede ao longo do decorrer da simulação, tal deve-se ao facto de a rede ser uma rede pequena e com controlo semafórico, o que faz com que os veículos fiquem algum tempo parados nos sinais vermelhos, aumentando o número de veículos presente no cenário em simultâneo e consequentemente aumentando também o número de nós disponíveis para a comunicação V2V.

As diferentes configurações do cenário de teste foram inspiradas no perfil diário de uma rede de tráfego urbana, a configuração de baixo fluxo pode também reportar a áreas rurais onde a densidade de tráfego é normalmente baixa. Tendo isto em conta e o resultado das simulações, fica demonstrado que as comunicações V2V são uma óptima ferramenta de disseminação de informação em ambientes urbanos, nomeadamente para a troca de informação relativa a congestionamentos e controle de tráfego, pois como foi provado em áreas congestionadas a informação alcança quase toda a rede.

8.4 - Outras simulações efectuadas

Como foi definido, para além das diferentes configurações do cenário de teste, foram efectuadas outras simulações.

8.4.1 - Simulações com alteração do parâmetro de velocidade.

Foram realizadas simulações com alteração da velocidade dos veículos. Com estas simulações pretendia-se modelar a diferença da formação da rede V2V em vias rápidas e meios urbanos. Verificou-se que à medida que se aumenta a velocidade diminui-se o número de veículos que a recebem. Pois devido à grande mobilidade, os veículos encontram-se durante pouco tempo no raio de comunicação dos veículos emissores.

.

8.4.2 - Simulações com alteração do raio de comunicação.

Os sistemas de comunicação de curto alcance possuem uma gama distinta de raios de comunicação. Em muitas aplicações reais ele é alterado através da adição de antenas, programação, etc. Tendo em vista esse aspecto realizaram-se simulações com raios de

alcance diferente. Como seria de prever, dessas simulações retira-se que a percentagem de cobertura da rede é directamente proporcional ao tamanho do raio de comunicação dos veículos.

8.4.3 - Simulações com envio de mensagem em pontos diferentes da rede

No cenário de teste todas as mensagens eram enviadas nas intersecções de forma a maximizar o processo de criação da rede V2V, pois é nesses pontos que se encontram as maiores concentrações de veículos na rede. Realizaram-se também simulações com envio em pontos diferentes da rede. Logicamente essas simulações permitiram uma menor cobertura de rede e em muitos dos casos a rede não se forma pois não encontra nenhum carro no seu alcance. Contudo, estas simulações foram realizadas porque na solução proposta, espera-se que na implementação real da ferramenta sempre que um tipo específico de sensor do veículo, nomeadamente os de detecção das condições da estrada, é activado há uma tentativa automática de envio da mensagem de aviso, mesmo que nenhum veículo se encontre no seu raio de comunicação.

8.4.4 - Simulações com envio simultâneo de várias mensagens

Simulou-se também o envio simultâneo de várias mensagens em pontos diferentes da rede. Observou-se que o protótipo desenvolvido comporta-se como esperado, ou seja, possibilita esse tipo de simulação, sendo as várias mensagens difundidas como seria de esperar e os veículos guardam sempre a última mensagem que receberam. Numa implementação real, os veículos possuiriam um repositório para guardar as mensagens, que na vida real seriam serviços, para acesso posterior aquando do pedido do utilizador.

8.4.5 - Simulação da comunicação externa do servidor

Para realizar este tipo de simulação foi implementado um socket cliente num computador diferente ao que alojava o simulador. Foram realizados então testes de conexão. A conexão ao servidor foi bem sucedida e testaram-se todos os comandos do protocolo de comunicação. Verificando-se o envio correcto da posição global de todos os veículos para o cliente quando ele requisita tal informação e que o cliente consegue enviar uma mensagem por ele definida e através de um carro por ele escolhido, verificando-se que o processo normal de envio de mensagens é disparado no simulador.

Capítulo 9

Conclusões

9.1 - Principais observações

Comunicações entre veículos têm-se revelado um dos campos de interesse das telecomunicações que mais rapidamente cresceu. Veículos equipados com dispositivos de comunicação sem fios de curto alcance podem formar uma rede móvel Ad-hoc, chamada VANET (*vehicular Ad-hoc Network*). A existência destas redes e a disseminação de informação nelas terão provavelmente um papel importante na segurança e eficiência das redes de tráfego num futuro próximo.

Esta dissertação recaiu sobre Arquitectura Orientada a Serviços com ênfase nas comunicações veículo-a-veículo. Se os dispositivos instalados nos veículos e no ambiente que os rodeia são vistos como serviços, aplicações podem ser criadas para compor um conjunto de serviços que regule o fluxo de dados entre eles. Contudo a composição de serviços não é tudo o que importa para implementar SOA numa rede V2V. Para implementar SOA nestas redes, as aplicações devem lidar com desafios como a heterogeneidade dos dispositivos, resistência ao dinamismo da rede, mobilidade e descoberta e disponibilidade dos serviços

O objectivo principal desta dissertação era propor e especificar uma arquitectura baseada em serviços para sistemas inteligentes de transportes, assente numa rede de comunicação sem fios V2V em larga escala. Foi proposta e especificada uma arquitectura de camadas, baseada no nível de abstracção dos serviços. A arquitectura em dois níveis principais: nível de serviços de rede e nível de serviços de utilizador/serviços de alto nível. O primeiro nível é responsável pela “criação” da topologia de rede e pela descoberta e troca de serviços entre veículos. O segundo nível consiste na arquitectura necessária à distribuição dos serviços, denominados de serviços de alto nível, aos utilizadores. Para o primeiro nível foi proposta uma arquitectura que integra a funcionalidade de descoberta de serviços de SOA no protocolo de encaminhamento da rede VANET. O resultado é uma especificação dinâmica e adaptativa que continua de acordo com a natureza abstracta e universal requeridas por SOA. A realização SOA neste nível é totalmente descentralizada, não existindo registo ou

repositório central de serviços. Para o segundo nível, apresentou-se uma arquitectura modular e extensível para a criação de serviços a bordo dos veículos. Especificou-se a arquitectura veicular necessária a implementação de serviços no veículo e a arquitectura para obtenção de serviços de unidades externas, para além dos outros veículos.

Outro grande objectivo proposto era identificar o potencial de desenvolvimento e as diferentes aplicações práticas de uma rede V2V.

Foi conceptualizado e desenvolvido um protótipo funcional que implementa a rede de comunicação V2V na plataforma MAS-T²er Lab. O protótipo permite a comunicação entre veículos e a comunicação com entidades exteriores. O conceito de troca de informação entre veículos é provado pela troca de mensagens e a implementação da comunicação exterior pode ser vista como a modelação da Comunicação dos veículos com entidades exteriores.

As várias simulações efectuadas demonstram que a funcionalidade da rede V2V é altamente influenciada pela densidade de tráfego e pelo *layout* das estradas. Em auto-estradas, os veículos em aproximação só se encontram dentro do raio de comunicação por alguns segundos. Contudo, em situações de congestionamento, veículos que viagem na mesma direcção podem estar dentro do raio de comunicação durante horas. Assim, comunicação inter-veicular através de redes VANET revela-se como uma óptima ferramenta de disseminação de informação em ambientes urbanos, nomeadamente para a troca de informação relativa a congestionamentos e controlo de tráfego. Além do mais, a alternativa que são as comunicações celulares de longo alcance possuem um conjunto de desvantagens como baixa largura de banda na comunicação nó a nó, áreas sem cobertura (por exemplo túneis) e normalmente esta tecnologia requer um tipo de pagamento.

De um modo geral, os objectivos do trabalho proposto foram cumpridos e a primeira etapa para a implementação da arquitectura proposta implementada num protótipo funcional. Pode também concluir-se que as funcionalidades implementadas e o trabalho extra realizado, como a melhoria da interface gráfica do simulador, constituem um contributo para a realização da plataforma MAS-T²er Lab.

9.2 - Perspectivas de trabalho futuro

Encontrando-se implementada a comunicação V2V e a comunicação externa da ferramenta de simulação da plataforma MAS-T²er Lab devem planear-se os trabalhos futuros.

A próxima etapa deste projecto deve ser a modulação das Veículos/condutores em agentes de forma a finalizar a implementação da arquitectura SOA planeada para a rede V2V. Além de que a implementação dos veículos/condutores sob a forma de agentes é uma das características da solução arquitectónica proposta para a plataforma MAS-T²er Lab. Para converter os veículos/ condutores em agentes a aplicação *Mediator* deve ser implementada.

Um trabalho paralelo a ser realizado é permitir que o editor de rede reconheça ficheiros *shapefile* e os carregue directamente para simulação, o que permitirá realizar simulações de redes mais complexas. Pretende-se também que o editor carregue também ficheiros de bases de dados georreferenciadas.

Outros trabalhos que se deverão realizar, tendo em vista o melhoramento geral do desempenho da ferramenta de simulação plataforma MAS-T²er Lab são: a separação da interfaces GUI da ferramenta de simulação, o que permitiria a visualização e análise da mesma simulação por vários utilizadores, e a validação e calibração do modelo de simulação de dados.

Se continuarem a ser realizados trabalhos como o que foi alvo esta dissertação e outros já realizados sobre a plataforma MAS-T²er Lab prevê-se que futuramente se conclua a implementação da solução apresentada para o *virtual domain* e a longo termo a implementação dos outros dois subsistemas da plataforma, tornando o MAS-T²er Lab numa poderosa ferramenta de análise e controlo de sistemas inteligentes de transportes reais.

Referências

- [1] Halonen T., Ojaja T., “Cross-Layer Design for Providing Service Oriented Architecture in a Mobile Ad hoc Network.”, 2006.
- [2] Bass L., Clemens P., Kazman R., “Software Architecture in Practice.”, Addison Wesley, 2003.
- [3] Fowler M., “Patterns of Enterprise Application Architecture”, 2002.
- [4] Krafzig D., Banke K., Slama D., “Enterprise SOA: Service-Oriented Architecture Best Practices”, 2004.
- [5] ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems. Disponível em <http://www.ieee.org/>. Acesso em 18/Setembro/2008.
- [6] Dewayne E., Alexander L., “Foundations for the study of software architecture” SIGSOFT, Softw. Eng. Notes, 1992.
- [7] CONSORTIUM. Disponível em <http://www.w3c.org>. Acesso em 18/ Setembro/2008.
- [8] GROUP, I. S, “New to SOA and Web Services.”, 2004. Disponível em <http://www-128.ibm.com/developerworks/webservices/newto/>. Acesso em 18/Setembro/2008.
- [9] McGovern J. et al., “Java Web Services Architecture”, 1a. ed, Elsevier Science and Technology Books, 2003.
- [10] http://p.blog.csdn.net/images/p_blog_csdn_net/winniepu/EntryImages/20090104/r_jw-0613-soa2.gif. Acesso em 13 Outubro de 2008.

- [11] Panda D., “An Introduction to Service-Oriented Architecture from a Java Developer Perspective”, 2005.
- [12] Grossi B., “Estudo do Modelo de Computação Orientada a Serviços e sua Aplicação a um Sistema de Mineração de Dados”, 2005
- [13] “Conheça a Arquitetura Orientada a Serviços”. Disponível em: <http://www.micrisoft.com/brasil/servidores/biztalk/soa>. Acesso em 21/Setembro/2008.
- [14] Tao X., Jiang C., Yaojun H., “Applying SOA to Intelligent Transportation System.”, 2005
- [15] Vieira J. L., “V2V, a nova sigla do futuro próximo.”, Disponível em: http://www.webmotors.com.br/wmpublicador/wmpublicador/Colunistas_Conteudo.vxlpub?hmid=37932. Acesso 29/Setembro/2008.
- [16] http://blog.wired.com/cars/images/2007/05/17/v2v_3.jpg. Acesso em 29/Setembro/2008.
- [17] Jaaskelainen J., “About ITS Architectures in Europe”, workshop on ITS Architectures, 12 -13 Novembro 2007.
- [18] Eichler, S., Schroth, C., Eberspächer, J., “Car-to-Car Communication”, Institute of Communication Networks, Technische Universität München and Institute of Media and Communication Management, SAP Research CEC, University of St. Gallen, München, 2006.
- [19] Mello H., Endler M., “Identificação de Região de Congestionamento através de Comunicação Inter-veicular”, 2006.
- [20] Chen Z. D., Kung H., Vlah D., “Ad hoc relay wireless networks over moving vehicles on highways”, 2001
- [21] IEEE. Disponível em: www.ieee.org
- [22] Raya M., Hubaux J., “The security of Vehicular Ad Hoc Networks”, 2005
- [24] Blum J., Eskandarian A., Hoffman L., “Challenges of inter-vehicle ad hoc network”, *IEEE Transactions on Intelligent Transportation Systems*, 2004.
- [25] Pivotto C. V. C., “VANET - Vehicular Ad-hoc Networks”, capítulo 2. Disponível em http://www.gta.ufrj.br/grad/06_2/pivotto/cap02.html/.

- [26] Naumov V., Baumann R., Gross, T., “An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces.”, em MobiHoc, 2006.
- [27] IETF - Internet Engineering Task Force. Disponível em <http://www.ietf.org/>.
- [28] Farias M. M. “Estudo Comparativo em Algoritmos de Roteamento para Redes Wireless Ad Hoc”, 2006
- [29] Tanenbaun, Andrew S., “Redes de Computadores”, Elsevier, 2003.
- [30] Mello H., Endler M., “Identificação de Região de Congestionamento através de Comunicação Inter-veicular”, 2006.
- [31] Wischhof L., Ebner A., Rohling H., “Information dissemination in self-organizing inter vehicle networks”, *IEEE Transactions on Intelligent Transportation Systems*, 2005.
- [32] Thomas, M., Peytchev, E., Al-Dadbass, D., “Auto-sensing and distribution of traffic information in vehicular ad hoc networks.”, *International Journal of Simulation*, 2005.
- [33] Maekawa, “ITS (Intelligent Transportation System) solutions”, *NEC Journal of Advanced Technology*, n. 3, July 2004.
- [34] Goel et al., Poster abstract: “Sensors on wheels - towards a zeroinfrastructure solution for intelligent transportation systems”. In: *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, New York, 2003.
- [35] Gaynor et Al, “Integrating wireless sensor networks with the grid, *IEEE Internet Computing*, 2004.
- [36] Nekovee M., “Sensor networks on the road: the promises and challenges of vehicular ad hoc networks and grids”, In: *Workshop on Ubiquitous Computing and e-Research*, 2005.
- [37] Chakrabarti S. e Mishra A., “*Quality of service in mobile ad hoc networks*”; 2003.
- [38] Choffnes D.R., Bustamante F.E., “An Integrated Mobility and Traffic Model for Vehicular Wireless Networks”, 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET), 2005.

- [39] Saha A., Johnson D., “Modelling Mobility for Vehicular Ad-hoc Networks”, ACM International Workshop on Vehicular Ad Hoc Networks (VANET) , 2004.
- [40] Schroth C., Dötzer F., Kosch T., Ostermaier B.; Strassberger M., “Simulating the traffic effects of vehicle-to-vehicle messaging systems”, 2005.
- [41] Glomosim. Disponível em <http://pcl.cs.ucla.edu/projects/glomosim/>. Acesso em 12/Outubro/2008
- [42] QualNet. Disponível em <http://www.scalable-networks.com/>. Acesso em 12/Outubro/2008
- [43] NS2 - Network Simulator 2. Disponível em: <http://www.isi.edu/nsnam/ns/>. Acesso em 12/Outubro/2008
- [44] Boxill S. A., Yu L., “An Evaluation of Traffic Simulation Models for Supporting ITS Development.”, 2000.
- [45] SUMO. Disponível em <http://sumo.sourceforge.net>. Acesso em 12/Outubro/2008.
- [46] VISSIM. Disponível em http://www.ptv.de/cgi-bin/traffic/traf_vissim.pl. Acesso em 12/Outubro/2008
- [47] CORSIM User’s Manual, ITS Research Division, FHWA, 2000.
- [48] PARAMICS. Disponível em <http://www.paramics-online.com/>
- [49] Mangharam R., Weller D., Rajkumar R., Mudalige P., Bai F., “GrooveNet: A Hybrid Simulator for Vehicle-to-Vehicle Networks”, 2006.
- [50] Conceição H., Damas L., Ferreira M., Barros J.; “*Large-Scale Simulation of V2V Environment*“, 2007
- [51] DIVERT. Disponível em <http://myddas.dcc.fc.up.pt/divert>.
- [52] OLSR daemon. Disponível em <http://www.olsr.org/>.
- [53] Koch N., “Automotive Case Study: UML specification of On Road Assistance Scenario”, 2007.
- [54] OSGi Alliance. Disponível em <http://www.osgi.ordd/>
- [55] Sánchez-Nielsen E., Martín-Ruiz S., Rodríguez-Pedrianes J., “An open and Dynamical Service Oriented Architecture for Supporting Mobile Services.”, 2006.

- [56] Linguagem WSDL. Disponível em <http://www.w3.org/TR/wsdl/>.

- [57] Rossetti R., Oliveira E., Bazzan A., "Towards a specification of a framework for sustainable transportation analysis.," no Workshop on Artificial Intelligence Applied to Sustainable Transportation Systems da 13th Portuguese Conference on Artificial Intelligence, Guimarães, Portugal, 2007.

- [58] Ferreira P., Esteves E., Rossetti R., Oliveira E., "A Cooperative Simulation Framework for Traffic and Transportation Engineering," in CDVE2008: The 5th International Conference on Cooperative, Mallorca, Spain, 2008.

- [59] Ferreira P., Specification and Implementation of an Artificial Transport System", 2008.

- [60] Trolltech, Qt Cross-Platform Application Framework - Trolltech. Disponível em <http://trolltech.com/products/qt/>.

- [61] Alves M. M., "Sockets Linux", 2008

- [62] "O que são sockets?". Disponível em http://br.geocities.com/java_io/jfaq0075.html

- [63] "The STL Format - Standard Data Format for Fabbers". Disponível em <http://www.ennex.com/~fabbers/StL.asp/>.

- [64] Martin T., "The Lane-Change Model MOBIL". Disponível em <http://www.vwi.tudresden.de>. Acesso em 12/Novembro/2008

- [65] Treiber M., Helbing D., "Realistische Mikrosimulation von Straßenverkehr miteinem einfachen Modell," no 16. Symposium "Simulationstechnik ASIM 2002", Rostock,2002, pp. 514-520.

Anexo A

Ficheiros XML da rede usada no cenário de teste

ScenarioNetwork.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Network>
  <DefaultTimeStep>100</DefaultTimeStep>
  <DefaultMultiplier>3</DefaultMultiplier>
  <RightPriority>1</RightPriority>
  <RoadFile>Road.xml</RoadFile>
  <NodeFile>Node.xml</NodeFile>
  <TripAssignmentFile>TripAssignment.xml</TripAssignmentFile>
  <TrafficLightFile>TrafficLight.xml</TrafficLightFile>
</Network>
```

Road.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Roads>
  <Road id="1" node1="0" node2="1">
    <RoadSegment id="0">
      <AdjacentNodes>
        <Connected2Node>0</Connected2Node>
        <Connected2Node>1</Connected2Node>
      </AdjacentNodes>
      <InitialPosition>-200 0</InitialPosition>
      <FinalPosition>200 0</FinalPosition>
      <LaneStruct>
        <LaneWidth>3</LaneWidth>
        <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
        <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
      </LaneStruct>
    </RoadSegment>
  </Road>
</Roads>
```

```

<DeadEndLanes></DeadEndLanes>
    <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
    </InterSegmentLaneConnections>
</LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
    </RoadSegment>
    <NecessaryLaneToReachNode>
        <NodeLaneToLaneConnection>
            <Connection from="1" to="1" node="4" />
            <Connection from="1" to="1" node="3" />
            <Connection from="0" to="0" node="3" />
        </NodeLaneToLaneConnection>
    </NecessaryLaneToReachNode>
</Road>
<Road id="2" node1="2" node2="1">
    <RoadSegment id="0">
        <AdjacentNodes>
            <Connected2Node>2</Connected2Node>
            <Connected2Node>1</Connected2Node>
        </AdjacentNodes>
        <InitialPosition>200 -400</InitialPosition>
        <FinalPosition>200 0</FinalPosition>
        <LaneStruct>
            <LaneWidth>3</LaneWidth>
            <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
            <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
            <DeadEndLanes></DeadEndLanes>
            <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
    </InterSegmentLaneConnections>
</LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
    </RoadSegment>
    <NecessaryLaneToReachNode>
        <NodeLaneToLaneConnection>
            <Connection from="1" to="1" node="4" />
            <Connection from="0" to="0" node="4" />
            <Connection from="0" to="0" node="3" />
        </NodeLaneToLaneConnection>
    </NecessaryLaneToReachNode>
</Road>
<Road id="3" node1="1" node2="3">
    <RoadSegment id="0">
        <AdjacentNodes>
            <Connected2Node>1</Connected2Node>
            <Connected2Node>3</Connected2Node>
        </AdjacentNodes>
        <InitialPosition>200 0</InitialPosition>

```

```

<FinalPosition>600 0</FinalPosition>
  <LaneStruct>
    <LaneWidth>3</LaneWidth>
    <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
    <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
    <DeadEndLanes></DeadEndLanes>
    <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
  </InterSegmentLaneConnections>
  </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
  </RoadSegment>
  <NecessaryLaneToReachNode>
    <NodeLaneToLaneConnection>
      <Connection from="1" to="1" node="10" />
      <Connection from="0" to="0" node="10" />
      <Connection from="0" to="0" node="11" />
    </NodeLaneToLaneConnection>
  </NecessaryLaneToReachNode>
</Road>
<Road id="4" node1="1" node2="4">
  <RoadSegment id="0">
    <AdjacentNodes>
      <Connected2Node>1</Connected2Node>
      <Connected2Node>4</Connected2Node>
    </AdjacentNodes>
    <InitialPosition>200 0</InitialPosition>
    <FinalPosition>200 400</FinalPosition>
    <LaneStruct>
      <LaneWidth>3</LaneWidth>
      <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
      <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
      <DeadEndLanes></DeadEndLanes>
      <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
  </InterSegmentLaneConnections>
  </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
  </RoadSegment>
  <NecessaryLaneToReachNode>
    <NodeLaneToLaneConnection>
      <Connection from="1" to="1" node="5" />
      <Connection from="1" to="1" node="6" />
      <Connection from="0" to="0" node="6" />
    </NodeLaneToLaneConnection>
  </NecessaryLaneToReachNode>
</Road>
<Road id="5" node1="4" node2="5">

```

```

<FinalPosition>600 0</FinalPosition>
    <LaneStruct>
        <LaneWidth>3</LaneWidth>
        <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
        <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
        <DeadEndLanes></DeadEndLanes>
        <InterSegmentLaneConnections>

    <ForwardLaneConnections></ForwardLaneConnections>

    <BackwardLaneConnections></BackwardLaneConnections>
        </InterSegmentLaneConnections>
    </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
    </RoadSegment>
    <NecessaryLaneToReachNode>
        <NodeLaneToLaneConnection>
            <Connection from="1" to="1" node="10" />
            <Connection from="0" to="0" node="10" />
            <Connection from="0" to="0" node="11" />
        </NodeLaneToLaneConnection>
    </NecessaryLaneToReachNode>
</Road>
<Road id="4" node1="1" node2="4">
    <RoadSegment id="0">
        <AdjacentNodes>
            <Connected2Node>1</Connected2Node>
            <Connected2Node>4</Connected2Node>
        </AdjacentNodes>
        <InitialPosition>200 0</InitialPosition>
        <FinalPosition>200 400</FinalPosition>
        <LaneStruct>
            <LaneWidth>3</LaneWidth>
            <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
            <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
            <DeadEndLanes></DeadEndLanes>
            <InterSegmentLaneConnections>

        <ForwardLaneConnections></ForwardLaneConnections>

        <BackwardLaneConnections></BackwardLaneConnections>
            </InterSegmentLaneConnections>
        </LaneStruct>

    <Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

    <Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
        </RoadSegment>
        <NecessaryLaneToReachNode>
            <NodeLaneToLaneConnection>
                <Connection from="1" to="1" node="5" />
                <Connection from="1" to="1" node="6" />
                <Connection from="0" to="0" node="6" />
            </NodeLaneToLaneConnection>
        </NecessaryLaneToReachNode>
    </Road>
<Road id="5" node1="4" node2="5">

```

```

<FinalPosition>600 0</FinalPosition>
  <LaneStruct>
    <LaneWidth>3</LaneWidth>
    <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
    <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
    <DeadEndLanes></DeadEndLanes>
    <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
  </InterSegmentLaneConnections>
</LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
  </RoadSegment>
  <NecessaryLaneToReachNode>
    <NodeLaneToLaneConnection>
      <Connection from="1" to="1" node="10" />
      <Connection from="0" to="0" node="10" />
      <Connection from="0" to="0" node="11" />
    </NodeLaneToLaneConnection>
  </NecessaryLaneToReachNode>
</Road>
<Road id="4" node1="1" node2="4">
  <RoadSegment id="0">
    <AdjacentNodes>
      <Connected2Node>1</Connected2Node>
      <Connected2Node>4</Connected2Node>
    </AdjacentNodes>
    <InitialPosition>200 0</InitialPosition>
    <FinalPosition>200 400</FinalPosition>
    <LaneStruct>
      <LaneWidth>3</LaneWidth>
      <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
      <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
      <DeadEndLanes></DeadEndLanes>
      <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
  </InterSegmentLaneConnections>
</LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
  </RoadSegment>
  <NecessaryLaneToReachNode>
    <NodeLaneToLaneConnection>
      <Connection from="1" to="1" node="5" />
      <Connection from="1" to="1" node="6" />
      <Connection from="0" to="0" node="6" />
    </NodeLaneToLaneConnection>
  </NecessaryLaneToReachNode>
</Road>
<Road id="5" node1="4" node2="5">

```

```

    <RoadSegment id="0">
      <AdjacentNodes>
        <Connected2Node>5</Connected2Node>
        <Connected2Node>4</Connected2Node>
      </AdjacentNodes>
      <InitialPosition>200 400</InitialPosition>
      <FinalPosition>-200 400</FinalPosition>
      <LaneStruct>
        <LaneWidth>3</LaneWidth>
        <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
        <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
        <DeadEndLanes></DeadEndLanes>
        <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
      </InterSegmentLaneConnections>
    </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
  </RoadSegment>
  <NecessaryLaneToReachNode>

<NodeLaneToLaneConnection></NodeLaneToLaneConnection>
  </NecessaryLaneToReachNode>
</Road>
<Road id="6" node1="4" node2="6">
  <RoadSegment id="0">
    <AdjacentNodes>
      <Connected2Node>6</Connected2Node>
      <Connected2Node>4</Connected2Node>
    </AdjacentNodes>
    <InitialPosition>200 400</InitialPosition>
    <FinalPosition>200 800</FinalPosition>
    <LaneStruct>
      <LaneWidth>3</LaneWidth>
      <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
      <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
      <DeadEndLanes></DeadEndLanes>
      <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
    </InterSegmentLaneConnections>
  </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
  </RoadSegment>
  <NecessaryLaneToReachNode>

<NodeLaneToLaneConnection></NodeLaneToLaneConnection>
  </NecessaryLaneToReachNode>
</Road>
<Road id="7" node1="7" node2="4">
  <RoadSegment id="0">

```

```

    <AdjacentNodes>
      <Connected2Node>7</Connected2Node>
      <Connected2Node>4</Connected2Node>
    </AdjacentNodes>
    <InitialPosition>600 400</InitialPosition>
    <FinalPosition>200 400</FinalPosition>
    <LaneStruct>
      <LaneWidth>3</LaneWidth>
      <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
      <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
      <DeadEndLanes></DeadEndLanes>
      <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
      </InterSegmentLaneConnections>
    </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
    </RoadSegment>
    <NecessaryLaneToReachNode>
      <NodeLaneToLaneConnection>
        <Connection from="1" to="1" node="5" />
        <Connection from="0" to="0" node="5" />
        <Connection from="0" to="0" node="6" />
      </NodeLaneToLaneConnection>
    </NecessaryLaneToReachNode>
  </Road>
  <Road id="8" node1="8" node2="7">
    <RoadSegment id="0">
      <AdjacentNodes>
        <Connected2Node>8</Connected2Node>
        <Connected2Node>7</Connected2Node>
      </AdjacentNodes>
      <InitialPosition>600 800</InitialPosition>
      <FinalPosition>600 400</FinalPosition>
      <LaneStruct>
        <LaneWidth>3</LaneWidth>
        <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
        <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
        <DeadEndLanes></DeadEndLanes>
        <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
      </InterSegmentLaneConnections>
    </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
    </RoadSegment>
    <NecessaryLaneToReachNode>
      <NodeLaneToLaneConnection>
        <Connection from="1" to="1" node="3" />
        <Connection from="0" to="0" node="3" />

```

```

        <Connection from="0" to="0" node="4" />
    </NodeLaneToLaneConnection>
</NecessaryLaneToReachNode>
</Road>
<Road id="9" node1="9" node2="7">
    <RoadSegment id="0">
        <AdjacentNodes>
            <Connected2Node>9</Connected2Node>
            <Connected2Node>7</Connected2Node>
        </AdjacentNodes>
        <InitialPosition>1000 400</InitialPosition>
        <FinalPosition>600 400</FinalPosition>
        <LaneStruct>
            <LaneWidth>3</LaneWidth>
            <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
            <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
            <DeadEndLanes></DeadEndLanes>
            <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
            </InterSegmentLaneConnections>
        </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
    </RoadSegment>
    <NecessaryLaneToReachNode>
        <NodeLaneToLaneConnection>
            <Connection from="1" to="1" node="4" />
            <Connection from="1" to="1" node="3" />
            <Connection from="0" to="0" node="4" />
        </NodeLaneToLaneConnection>
    </NecessaryLaneToReachNode>
</Road>
<Road id="10" node1="3" node2="10">
    <RoadSegment id="0">
        <AdjacentNodes>
            <Connected2Node>10</Connected2Node>
            <Connected2Node>3</Connected2Node>
        </AdjacentNodes>
        <InitialPosition>600 0</InitialPosition>
        <FinalPosition>1000 0</FinalPosition>
        <LaneStruct>
            <LaneWidth>3</LaneWidth>
            <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
            <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
            <DeadEndLanes></DeadEndLanes>
            <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
            </InterSegmentLaneConnections>
        </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>

```

```

        </RoadSegment>
        <NecessaryLaneToReachNode>

<NodeLaneToLaneConnection></NodeLaneToLaneConnection>
    </NecessaryLaneToReachNode>
</Road>
<Road id="11" node1="3" node2="11">
    <RoadSegment id="0">
        <AdjacentNodes>
            <Connected2Node>11</Connected2Node>
            <Connected2Node>3</Connected2Node>
        </AdjacentNodes>
        <InitialPosition>600 0</InitialPosition>
        <FinalPosition>600 -400</FinalPosition>
        <LaneStruct>
            <LaneWidth>3</LaneWidth>
            <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
            <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
            <DeadEndLanes></DeadEndLanes>
            <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
    </InterSegmentLaneConnections>
    </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>
    </RoadSegment>
    <NecessaryLaneToReachNode>

<NodeLaneToLaneConnection></NodeLaneToLaneConnection>
    </NecessaryLaneToReachNode>
</Road>
<Road id="12" node1="7" node2="3">
    <RoadSegment id="0">
        <AdjacentNodes>
            <Connected2Node>7</Connected2Node>
            <Connected2Node>3</Connected2Node>
        </AdjacentNodes>
        <InitialPosition>600 400</InitialPosition>
        <FinalPosition>600 0</FinalPosition>
        <LaneStruct>
            <LaneWidth>3</LaneWidth>
            <Node1ToNode2NumLanes>2</Node1ToNode2NumLanes>
            <Node2ToNode1NumLanes>0</Node2ToNode1NumLanes>
            <DeadEndLanes></DeadEndLanes>
            <InterSegmentLaneConnections>

<ForwardLaneConnections></ForwardLaneConnections>

<BackwardLaneConnections></BackwardLaneConnections>
    </InterSegmentLaneConnections>
    </LaneStruct>

<Node1ToNode2BackwardSegment></Node1ToNode2BackwardSegment>

<Node1ToNode2ForwardSegment></Node1ToNode2ForwardSegment>

```

```

        </RoadSegment>
        <NecessaryLaneToReachNode>
            <NodeLaneToLaneConnection>
                <Connection from="1" to="1" node="11" />
                <Connection from="1" to="1" node="10" />
                <Connection from="0" to="0" node="11" />
            </NodeLaneToLaneConnection>
        </NecessaryLaneToReachNode>
    </Road>
</Roads>

```

Node.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<Nodes>
    <Node id="0" posx="-200" posy="0" size="0" isExit="0"
isStart="1">
        <RoadInCounterClockwise>
            <Road>1</Road>
        </RoadInCounterClockwise>
        <EntryRoadConnections>
        </EntryRoadConnections>
        <ExitRoadConnections>
            <Exit toNode="1" road="1" />
        </ExitRoadConnections>
        <Startup>
            <Flow>1250</Flow>
            <CarVSTruck>80</CarVSTruck>
            <StartupTripVec>0</StartupTripVec>
        </Startup>
    </Node>
    <Node id="1" posx="200" posy="0" size="10" isExit="0"
isStart="0">
        <RoadInCounterClockwise>
            <Road>1</Road>
            <Road>2</Road>
            <Road>3</Road>
            <Road>4</Road>
        </RoadInCounterClockwise>
        <EntryRoadConnections>
            <Entry fromNode="0" road="1" />
            <Entry fromNode="2" road="2" />
        </EntryRoadConnections>
        <ExitRoadConnections>
            <Exit toNode="3" road="3" />
            <Exit toNode="4" road="4" />
        </ExitRoadConnections>
        <Startup></Startup>
    </Node>
    <Node id="2" posx="200" posy="-400" size="0" isExit="0"
isStart="1">
        <RoadInCounterClockwise>
            <Road>2</Road>
        </RoadInCounterClockwise>
        <EntryRoadConnections></EntryRoadConnections>
        <ExitRoadConnections>

```

```

        <Exit toNode="1" road="2" />
    </ExitRoadConnections>
    <Startup>
        <Flow>900</Flow>
        <CarVSTruck>80</CarVSTruck>
        <StartupTripVec>1</StartupTripVec>
    </Startup>
</Node>
<Node id="3" posx="600" posy="0" size="10" isExit="0"
isStart="0">
    <RoadInCounterClockwise>
        <Road>3</Road>
        <Road>11</Road>
        <Road>10</Road>
        <Road>12</Road>
    </RoadInCounterClockwise>
    <EntryRoadConnections>
        <Entry fromNode="1" road="3" />
        <Entry fromNode="7" road="12" />
    </EntryRoadConnections>
    <ExitRoadConnections>
        <Exit toNode="10" road="10" />
        <Exit toNode="11" road="11" />
    </ExitRoadConnections>
    <Startup></Startup>
</Node>
<Node id="4" posx="200" posy="400" size="10" isExit="0"
isStart="0">
    <RoadInCounterClockwise>
        <Road>4</Road>
        <Road>7</Road>
        <Road>6</Road>
        <Road>5</Road>
    </RoadInCounterClockwise>
    <EntryRoadConnections>
        <Entry fromNode="1" road="4" />
        <Entry fromNode="7" road="7" />
    </EntryRoadConnections>
    <ExitRoadConnections>
        <Exit toNode="5" road="5" />
        <Exit toNode="6" road="6" />
    </ExitRoadConnections>
    <Startup></Startup>
</Node>
<Node id="5" posx="-200" posy="400" size="0" isExit="1"
isStart="0">
    <RoadInCounterClockwise>
        <Road>5</Road>
    </RoadInCounterClockwise>
    <EntryRoadConnections>
        <Entry fromNode="4" road="5" />
    </EntryRoadConnections>
    <ExitRoadConnections></ExitRoadConnections>
    <Startup></Startup>
</Node>
<Node id="6" posx="200" posy="800" size="0" isExit="1"
isStart="0">
    <RoadInCounterClockwise>
        <Road>6</Road>
    </RoadInCounterClockwise>

```

```

    <EntryRoadConnections>
      <Entry fromNode="4" road="6" />
    </EntryRoadConnections>
    <ExitRoadConnections></ExitRoadConnections>
    <Startup></Startup>
  </Node>
  <Node id="7" posx="600" posy="400" size="10" isExit="0"
    isStart="0">
    <RoadInCounterClockwise>
      <Road>7</Road>
      <Road>12</Road>
      <Road>9</Road>
      <Road>8</Road>
    </RoadInCounterClockwise>
    <EntryRoadConnections>
      <Entry fromNode="8" road="8" />
      <Entry fromNode="9" road="9" />
    </EntryRoadConnections>
    <ExitRoadConnections>
      <Exit toNode="4" road="7" />
      <Exit toNode="3" road="12" />
    </ExitRoadConnections>
    <Startup></Startup>
  </Node>
  <Node id="8" posx="600" posy="800" size="0" isExit="0"
    isStart="1">
    <RoadInCounterClockwise>
      <Road>8</Road>
    </RoadInCounterClockwise>
    <EntryRoadConnections></EntryRoadConnections>
    <ExitRoadConnections>
      <Exit toNode="7" road="8" />
    </ExitRoadConnections>
    <Startup>
      <Flow>850</Flow>
      <CarVSTruck>80</CarVSTruck>
      <StartupTripVec>2</StartupTripVec>
    </Startup>
  </Node>
  <Node id="9" posx="1000" posy="400" size="0" isExit="0"
    isStart="1">
    <RoadInCounterClockwise>
      <Road>9</Road>
    </RoadInCounterClockwise>
    <EntryRoadConnections></EntryRoadConnections>
    <ExitRoadConnections>
      <Exit toNode="7" road="9" />
    </ExitRoadConnections>
    <Startup>
      <Flow>1000</Flow>
      <CarVSTruck>80</CarVSTruck>
      <StartupTripVec>3</StartupTripVec>
    </Startup>
  </Node>
  <Node id="10" posx="1000" posy="0" size="0" isExit="1"
    isStart="0">
    <RoadInCounterClockwise>
      <Road>10</Road>
    </RoadInCounterClockwise>
    <EntryRoadConnections>
      <Entry fromNode="3" road="10" />
    </EntryRoadConnections>

```

```

        </EntryRoadConnections>
        <ExitRoadConnections></ExitRoadConnections>
        <Startup></Startup>
    </Node>
    <Node id="11" posx="600" posy="-400" size="0" isExit="1"
        isStart="0">
        <RoadInCounterClockwise>
            <Road>11</Road>
        </RoadInCounterClockwise>
        <EntryRoadConnections>
            <Entry fromNode="3" road="11" />
        </EntryRoadConnections>
        <ExitRoadConnections></ExitRoadConnections>
        <Startup></Startup>
    </Node>

</Nodes>

```

Trip Assignment.Xml

```

<?xml version="1.0" encoding="UTF-8"?>
<TripAssignment>
    <Start node="0" tripId="0">
        <Path percentage="35">
            <Node>1</Node>
            <Node>3</Node>
            <Node>10</Node>
        </Path>
        <Path percentage="15">
            <Node>1</Node>
            <Node>3</Node>
            <Node>11</Node>
        </Path>
        <Path percentage="25">
            <Node>1</Node>
            <Node>4</Node>
            <Node>5</Node>
        </Path>
        <Path percentage="25">
            <Node>1</Node>
            <Node>4</Node>
            <Node>6</Node>
        </Path>
    </Start>
    <Start node="2" tripId="1">
        <Path percentage="25">
            <Node>1</Node>
            <Node>3</Node>
            <Node>10</Node>
        </Path>
        <Path percentage="15">
            <Node>1</Node>
            <Node>3</Node>
            <Node>11</Node>
        </Path>
        <Path percentage="25">
            <Node>1</Node>
            <Node>4</Node>

```

```

        <Node>5</Node>
    </Path>
    <Path percentage="35">
        <Node>1</Node>
        <Node>4</Node>
        <Node>6</Node>
    </Path>
</Start>
<Start node="8" tripId="2">
    <Path percentage="25">
        <Node>7</Node>
        <Node>4</Node>
        <Node>6</Node>
    </Path>
    <Path percentage="15">
        <Node>7</Node>
        <Node>4</Node>
        <Node>5</Node>
    </Path>
    <Path percentage="35">
        <Node>7</Node>
        <Node>3</Node>
        <Node>11</Node>
    </Path>
    <Path percentage="25">
        <Node>7</Node>
        <Node>3</Node>
        <Node>10</Node>
    </Path>
</Start>
<Start node="9" tripId="3">
    <Path percentage="25">
        <Node>7</Node>
        <Node>4</Node>
        <Node>6</Node>
    </Path>
    <Path percentage="35">
        <Node>7</Node>
        <Node>4</Node>
        <Node>5</Node>
    </Path>
    <Path percentage="25">
        <Node>7</Node>
        <Node>3</Node>
        <Node>11</Node>
    </Path>
    <Path percentage="15">
        <Node>7</Node>
        <Node>3</Node>
        <Node>10</Node>
    </Path>
</Start>
</TripAssignmen

```

TrafficLight.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<TrafficLights>
  <LightIntersection node="1">
    <IntersectionLights>
      <TrafficLight id="1">
        <FromNode>0</FromNode>
        <ToNode>4</ToNode>
        <Position>190 5</Position>
        <Angle>0</Angle>
        <LightType></LightType>
        <GreenTime>30000</GreenTime>
        <YellowTime>3000</YellowTime>
        <RedTime>30000</RedTime>
      </TrafficLight>
      <TrafficLight id="2">
        <FromNode>0</FromNode>
        <ToNode>3</ToNode>
        <Position>190 1</Position>
        <Angle>0</Angle>
        <LightType>1</LightType>
        <GreenTime>30000</GreenTime>
        <YellowTime>3000</YellowTime>
        <RedTime>30000</RedTime>
      </TrafficLight>
      <TrafficLight id="3">
        <FromNode>2</FromNode>
        <ToNode>4</ToNode>
        <Position>204 -10</Position>
        <Angle>0</Angle>
        <LightType>1</LightType>
        <GreenTime>30000</GreenTime>
        <YellowTime>3000</YellowTime>
        <RedTime>30000</RedTime>
      </TrafficLight>
      <TrafficLight id="4">
        <FromNode>2</FromNode>
        <ToNode>3</ToNode>
        <Position>201 -10</Position>
        <Angle>0</Angle>
        <LightType>1</LightType>
        <GreenTime>30000</GreenTime>
        <YellowTime>3000</YellowTime>
        <RedTime>30000</RedTime>
      </TrafficLight>
    </IntersectionLights>
    <IntersectionLightPlan>
      <TrafficLight id="1">
        <InitialLightState>GREEN</InitialLightState>
        <InitialTimeOfState>30000</InitialTimeOfState>
        <SetInternalTime>0</SetInternalTime>
      </TrafficLight>
      <TrafficLight id="2">
        <InitialLightState>GREEN</InitialLightState>
        <InitialTimeOfState>30000</InitialTimeOfState>
        <SetInternalTime>0</SetInternalTime>
      </TrafficLight>
    </IntersectionLightPlan>
  </LightIntersection>
</TrafficLights>

```

```

</TrafficLight>
<TrafficLight id="3">
  <InitialLightState>RED</InitialLightState>
  <InitialTimeOfState>30000</InitialTimeOfState>
  <SetInternalTime>0</SetInternalTime>
</TrafficLight>
<TrafficLight id="4">
  <InitialLightState>RED</InitialLightState>
  <InitialTimeOfState>30000</InitialTimeOfState>
  <SetInternalTime>0</SetInternalTime>
</TrafficLight>
</IntersectionLightPlan>
</LightIntersection>
<LightIntersection node="3">
  <IntersectionLights>
    <TrafficLight id="1">
      <FromNode>1</FromNode>
      <ToNode>11</ToNode>
      <Position>590 1</Position>
      <Angle>0</Angle>
      <LightType></LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="2">
      <FromNode>1</FromNode>
      <ToNode>10</ToNode>
      <Position>590 4</Position>
      <Angle>0</Angle>
      <LightType>1</LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="3">
      <FromNode>7</FromNode>
      <ToNode>11</ToNode>
      <Position>601 10</Position>
      <Angle>0</Angle>
      <LightType>1</LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="4">
      <FromNode>7</FromNode>
      <ToNode>10</ToNode>
      <Position>604 10</Position>
      <Angle>0</Angle>
      <LightType>1</LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
  </IntersectionLights>
</IntersectionLightPlan>
  <TrafficLight id="1">
    <InitialLightState>GREEN</InitialLightState>
    <InitialTimeOfState>30000</InitialTimeOfState>
    <SetInternalTime>0</SetInternalTime>
  </TrafficLight>

```

```

</TrafficLight>
<TrafficLight id="2">
  <InitialLightState>GREEN</InitialLightState>
  <InitialTimeOfState>30000</InitialTimeOfState>
  <SetInternalTime>0</SetInternalTime>
</TrafficLight>
<TrafficLight id="3">
  <InitialLightState>RED</InitialLightState>
  <InitialTimeOfState>30000</InitialTimeOfState>
  <SetInternalTime>0</SetInternalTime>
</TrafficLight>
<TrafficLight id="4">
  <InitialLightState>RED</InitialLightState>
  <InitialTimeOfState>30000</InitialTimeOfState>
  <SetInternalTime>0</SetInternalTime>
</TrafficLight>
</IntersectionLightPlan>
</LightIntersection>
<LightIntersection node="4">
  <IntersectionLights>
    <TrafficLight id="1">
      <FromNode>1</FromNode>
      <ToNode>5</ToNode>
      <Position>201 390</Position>
      <Angle>0</Angle>
      <LightType></LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="2">
      <FromNode>1</FromNode>
      <ToNode>6</ToNode>
      <Position>204 390</Position>
      <Angle>0</Angle>
      <LightType>1</LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="3">
      <FromNode>7</FromNode>
      <ToNode>5</ToNode>
      <Position>210 401</Position>
      <Angle>0</Angle>
      <LightType>1</LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="4">
      <FromNode>7</FromNode>
      <ToNode>6</ToNode>
      <Position>210 404</Position>
      <Angle>0</Angle>
      <LightType>1</LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
  </IntersectionLights>
</LightIntersection>

```

```

</IntersectionLights>
<IntersectionLightPlan>
  <TrafficLight id="1">
    <InitialLightState>GREEN</InitialLightState>
    <InitialTimeOfState>30000</InitialTimeOfState>
    <SetInternalTime>0</SetInternalTime>
  </TrafficLight>
  <TrafficLight id="2">
    <InitialLightState>GREEN</InitialLightState>
    <InitialTimeOfState>30000</InitialTimeOfState>
    <SetInternalTime>0</SetInternalTime>
  </TrafficLight>
  <TrafficLight id="3">
    <InitialLightState>RED</InitialLightState>
    <InitialTimeOfState>30000</InitialTimeOfState>
    <SetInternalTime>0</SetInternalTime>
  </TrafficLight>
  <TrafficLight id="4">
    <InitialLightState>RED</InitialLightState>
    <InitialTimeOfState>30000</InitialTimeOfState>
    <SetInternalTime>0</SetInternalTime>
  </TrafficLight>
</IntersectionLightPlan>
</LightIntersection>
<LightIntersection node="7">
  <IntersectionLights>
    <TrafficLight id="1">
      <FromNode>9</FromNode>
      <ToNode>3</ToNode>
      <Position>610 401</Position>
      <Angle>0</Angle>
      <LightType></LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="2">
      <FromNode>9</FromNode>
      <ToNode>4</ToNode>
      <Position>610 404</Position>
      <Angle>0</Angle>
      <LightType>1</LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="3">
      <FromNode>8</FromNode>
      <ToNode>4</ToNode>
      <Position>601 410</Position>
      <Angle>0</Angle>
      <LightType>1</LightType>
      <GreenTime>30000</GreenTime>
      <YellowTime>3000</YellowTime>
      <RedTime>30000</RedTime>
    </TrafficLight>
    <TrafficLight id="4">
      <FromNode>8</FromNode>
      <ToNode>3</ToNode>
      <Position>604 410</Position>
      <Angle>0</Angle>

```

```

        <LightType>1</LightType>
        <GreenTime>30000</GreenTime>
        <YellowTime>3000</YellowTime>
        <RedTime>30000</RedTime>
    </TrafficLight>
</IntersectionLights>
<IntersectionLightPlan>
    <TrafficLight id="1">
        <InitialLightState>GREEN</InitialLightState>
        <InitialTimeOfState>30000</InitialTimeOfState>
        <SetInternalTime>0</SetInternalTime>
    </TrafficLight>
    <TrafficLight id="2">
        <InitialLightState>GREEN</InitialLightState>
        <InitialTimeOfState>30000</InitialTimeOfState>
        <SetInternalTime>0</SetInternalTime>
    </TrafficLight>
    <TrafficLight id="3">
        <InitialLightState>RED</InitialLightState>
        <InitialTimeOfState>30000</InitialTimeOfState>
        <SetInternalTime>0</SetInternalTime>
    </TrafficLight>
    <TrafficLight id="4">
        <InitialLightState>RED</InitialLightState>
        <InitialTimeOfState>30000</InitialTimeOfState>
        <SetInternalTime>0</SetInternalTime>
    </TrafficLight>
</IntersectionLightPlan>
</LightIntersection>
</TrafficLights>

```

