CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY

ASSIGNMENT OF MASTER'S THESIS

Title:	Car-to-Infrastructure Communication in the Context of Intelligent Traffic Intersections
Student:	Bc. Jan Beran
Supervisor:	Ing. Martin Nemčík, Ph.D.
Study Programme:	Informatics
Study Branch:	Computer Security
Department:	Department of Computer Systems
Validity:	Until the end of winter semester 2017/18

Instructions

- Get familiar with Car-to-infrastructure (C2I) communication networks, focus on Green Light Optimized Speed Advisory systems (GLOSA).
- Describe security of C2I networks and its coverage by existing industry standards.
- Get familiar with the GreenLight mobile application implementing GLOSA and with the MirrorLink technology used within the application.
- Analyze possible security threats for the GreenLight application and propose suitable security measures.
- Propose an algorithm for calculating recommended (optimal) speed for a driver.
- Implement the proposed algorithm and integrate it in the GreenLight application.
- Verify performance of the implemented algorithm in real traffic situations and compare it with the previously implemented simple algorithm.
- Analyze and propose methods for determining intended driving direction through an intersection.

References

Will be provided by the supervisor.

L.S.

prof. Ing. Róbert Lórencz, CSc.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 29, 2016

Czech Technical University in Prague
Faculty of Information Technology
Department of Computer Systems

Master's thesis

Car-to-Infrastructure Communication in the Context of Intelligent Traffic Intersections

Bc. Jan Beran

Supervisor: Ing. Martin Nemčík, Ph.D.

1st July 2016

Acknowledgements

In the first place, I would like to thank my supervisor *Ing. Martin Nemčík, Ph.D.* for the opportunity to work on this exciting topic and his professional guidance during the entire project. I would also like to thank *Mr. Pavel Dyntera* from the *Intens Corporation* for his technical support and help with the application development.

Finally, I wish to thank my brother *Adam* for a detailed proofreading of this thesis and my parents for their invaluable support and encouragement throughout my study.

Declaration

Citation of this thesis

Beran, Jan. *Car-to-Infrastructure Communication in the Context of Intelligent Traffic Intersections.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

Abstrakt

Car-to-infrastructure (C2I) je ad-hoc komunikační schéma používané v inteligentních dopravních systémech. Jednou z hlavních aplikací C2I sítí je Green Light Optimal Speed Advisory (GLOSA). Jejím cílem je podporovat plynulost provozu a umožnit řidičům průjezd křižovatkou bez nutnosti zastavení či prudkých změn rychlosti. Hlavním cílem této práce je navrhnout a implementovat vylepšený algoritmus pro výpočet doporučené rychlosti příjezdu ke křižovatce. Algoritmus je následně integrován do mobilní aplikace GreenLight, která implementuje systém GLOSA na straně řidiče. Dalším tématem je analýza bezpečnostních hrozeb – obecných pro GLOSA systémy a konkrétních pro aplikaci GreenLight. Práce také navrhuje několik metod pro určení směru, kterým zamýšlí řidič projet křižovatkou. Toto umožní automaticky vybrat správný semafor pro výpočet doporučené rychlosti.

Klíčová slova Inteligentní Dopravní Systémy, Car-to-infrastructure, Křižovatka, Semafor, GLOSA, Bezpečnostní hrozby, Doporučená rychlost

Abstract

Car-to-infrastructure (C2I) is an ad-hoc communication scheme used within Inteligent Transportation Systems. Green Light Optimal Speed Advisory (GLOSA) is one of the main C2I applications. It increases an overall traffic flow continuity by helping drivers to avoid unnecessary stops at traffic intersections. Main goal of this thesis is to design and implement an enhanced algorithm for calculating a recommended approaching speed to an intersection. It also describes its integration in GreenLight, a mobile application implementing driver-side part of GLOSA. Furthermore, the thesis identifies common security threats to GLOSA systems and to the GreenLight application. It also proposes several methods for determining direction in which the driver intends to go through an intersection. This will allow for an automatic selection of the proper light signal for which a recommended speed should be calculated.

Keywords Inteligent Transportation Systems, Car-to-infrastructure, Intersection, Traffic light, GLOSA, Security threats, Recommended speed

Contents

List of Figures

Introduction

Road transportation systems have undergone a massive technological improvement during the last two decades. Nowadays, *Intelligent Transportation System* (ITS) is a widely recognized term denoting modern communication technology applied to various transportation systems. This thesis deals exclusively with road transportation and automotive systems, but ITS is also defined for railway, aeronautical and maritime transportation [1].

Existence of *Intelligent Transportation Systems* is an important condition for deployment of autonomous (self-driven) vehicles. It is probably not realistic to expect a complete adoption of self-driven cars in the near future, but at least a partial adoption might be achieved soon. *Society of Automotive Engineers* (SAE) has defined the following six levels of automated vehicle control [2], [3]:

- **Level 0 – No Automation:** The driver controls the vehicle at all times, no automated assistance systems are used. Only notifications and warnings to the driver may be issued automatically (e.g. optimal speed advisory).
- **Level 1 – Driver Assisted:** Assistance systems are used to support the driver and partially control the vehicle (e.g. adaptive cruise control, lane departure warning). However, the *Level 2* systems provide only a limited support and the driver still needs to control the vehicle at all times.
- **Level 2 – Partial Automation:** Steering, accelerating and braking are controlled automatically. All the remaining aspects of the vehicle control are performed by the driver, e.g. lights, turn indicators. The driver also needs to be ready to take over control if the automated systems fail to operate properly, e.g. in bad weather.
- **Level 3 – Conditional Automation:** Within limited environments such as highways, the automated systems can completely control the

car. The driver needs to be ready to take over control in a short time upon a takeover request from the automated systems. This means the driver is allowed to perform secondary tasks such as calling or internet browsing during the autonomous operation.

- **Level 4 – High Automation:** Same as *Level 3* but the driver does not need to provide any fall-back control performance. Within the limited environments, the automated control must handle all the possible conditions by itself.
- **Level 5 – Full Automation:** The car is fully autonomous, the driver is not expected to control the vehicle at any time. Furthermore, no driver needs to be physically present in the vehicle during the journey. The only expected human interaction is determining the route (or only its destination) and activating the autonomous control.

Currently, many *Level 1* systems are being adopted by major car manufacturers. This means the *Advanced Driver Assistance Systems* (ADAS) are under rapid development and there is an incredibly increasing demand for technology support from roadside infrastructure. Furthermore, organizations such as IEEE, ETSI and SAE are working on a standardization of various ITS-related elements and supporting technology. In certain limited domains, systems of higher levels have also been introduced already. Example of such technology is *Remote Valet Parking*^{TM}, a driver-less parking system developed by BMW [4]. It can be considered as a *Level 4* system but only in a very limited application domain.

ITS may include all types of communication in vehicles, between vehicles (car-to-car, C2C), between vehicles and static road infrastructure elements (car-to-infrastructure, C2I) and between infrastructure elements and their backend components (e.g. datacenters). When this communication is used to improve traffic effectiveness, safety or environmental impact, the system is denoted as *Cooperative ITS* (C-ITS) [5].

There are many different C-ITS use cases and applications. In this work, I focus on *Green Light Optimized Speed Advisory* (GLOSA) which is a C-ITS application designed to increase traffic flow efficiency at road intersections. It involves communication between a car and an intelligent traffic intersection (car-to-infrastructure network). In this scenario, an intersection controller broadcasts information about the phase schedule of each traffic light signal to nearby vehicles. Approaching vehicles can calculate the optimal approaching speed and avoid any unnecessary stops [6]. GLOSA is described in detail in section 1.3.

An essential aspect of any *Intelligent Transportation System* is security. There have been several successful attacks already, for example remotely disabled brakes of a moving Jeep Cherokee [7]. Security will get particularly crucial once the autonomous driving systems are adopted (phase 3 as defined above). Therefore, it is important to pay adequate attention to the security aspect when designing or deploying any ITS-related technologies.

Goals of the Thesis

It might be useful to outline the most important goals and desired outcomes of this thesis. The list is not complete, some partial goals are neglected here:

- Introduce car-to-infrastructure networks and their applications in C-ITS (especially GLOSA). Expected outcome: Reader should get a general understanding of car-to-infrastructure networks and the role they play in C-ITS systems.

- Identify and describe common security threats in car-to-infrastructure networks. On top of this analysis, provide suitable security measures for *GreenLight*, a mobile application representing a car-side implementation of a GLOSA system. Expected outcome: Reader should get an overall overview of security threats in C2I-based systems. Moreover, this should provide a simple guideline for a security design of future *GreenLight* versions.

- Design, implement and evaluate an enhanced algorithm for calculating the advised speed in the *GreenLight* application. Expected outcome: The newly implemented algorithm should improve performance and result quality of the original (simple) algorithm.

- Analyze possible methods for determining an intended driving direction through an intersection. This is necessary for selecting a proper traffic light signal in the *GreenLight* application when the car is approaching an intersection. Expected outcome: An algorithm for determining a proper traffic light is expected to be implemented by the end of 2016. The implementation will be based on this theoretical analysis.

Thesis Structure

The first chapter briefly introduces car-to-infrastructure communication networks and their usage in C-ITS systems. Special focus is given to the recommended speed advisory systems (GLOSA). Furthermore, the chapter outlines related technology background and looks into the current industry standards.

Chapter 2 moves focus to an existing mobile application implementing GLOSA in a vehicle – *GreenLight*. The application is described in detail together with the *MirrorLink*^{TM} technology it uses.

The third chapter presents common security threats to C2I-based GLOSA systems and also for the *GreenLight* application. Based on the identified threats, several possible security measures for the application are discussed. These should be considered for the future development of the application.

There are often more than one different traffic lights at an intersection approach (e.g. separate signal for left / right / straight directions). Current version of the *GreenLight* application displays only the straight direction signal which is clearly not optimal and might not be acceptable for a production version of the application. Therefore, a theoretical analysis of possible methods for determining a proper traffic light signal is provided in chapter 4.

Another important feature of a GLOSA system is the algorithm for calculating a recommended speed displayed to the driver. Current version of the *GreenLight* application contains just a very simple algorithm and its performance is insufficient. Therefore, an enhanced calculation of the advised speed is presented in chapter 5.

Chapter 6 deals with the implementation and testing of the enhanced speed calculation proposed in the previous chapter and its integration in the *GreenLight* application. *GreenLight* is an *Android* application written in *C#* using *.NET* libraries and the *Xamarin* platform. Therefore, the same programming language and tools are used for the new speed algorithm. After that, updated *GreenLight* with the newly implemented speed algorithm is tested in real traffic. In addition, a theoretical comparison with the original algorithm is also presented in this chapter.

Chapter 1

Car-to-Infrastructure Networks (C2I)

1.1 Overview of C2I networks

An *Intelligent Transportation System* involves a wireless communication between vehicles and roadside infrastructure elements. It may vary in the communication scheme and actual technology used. The basic classification is the following:

- **Centralized approach:** All the communication is routed through central nodes using an existing network infrastructure. For the automotive usage, the most common technologies are mobile networks such as 3G/LTE.

- **Ad-Hoc approach:** Usually denoted as *Vehicular Ad Hoc Network* (VANET). Decentralized approach where the communication does not rely on an existing network infrastructure. Instead, each node participates in the routing by forwarding data for other nodes [8]. The most common technology is IEEE 802.11p which is an amendment to the well-known *Wi-Fi* protocol (more in section 1.2.1).

Car-to-infrastructure is an ad-hoc communication scheme involving an *On-Board ITS Unit* (OBU) in a vehicle, a static *Roadside Infrastructure ITS Unit* (RSU) and a wireless communication network between these two. Similarly, car-to-car is a scheme involving communication between two or more on-board units. Both schemes may be also referenced as car-to-x (C2X). Possibly, there might be also a *Central ITS Unit* communicating with the RSU units providing them with backend services such as central database access. Since this communication is not ad-hoc, it is, strictly speaking, not part of C2X networking. All these communication schemes are illustrated in figure 1.1.

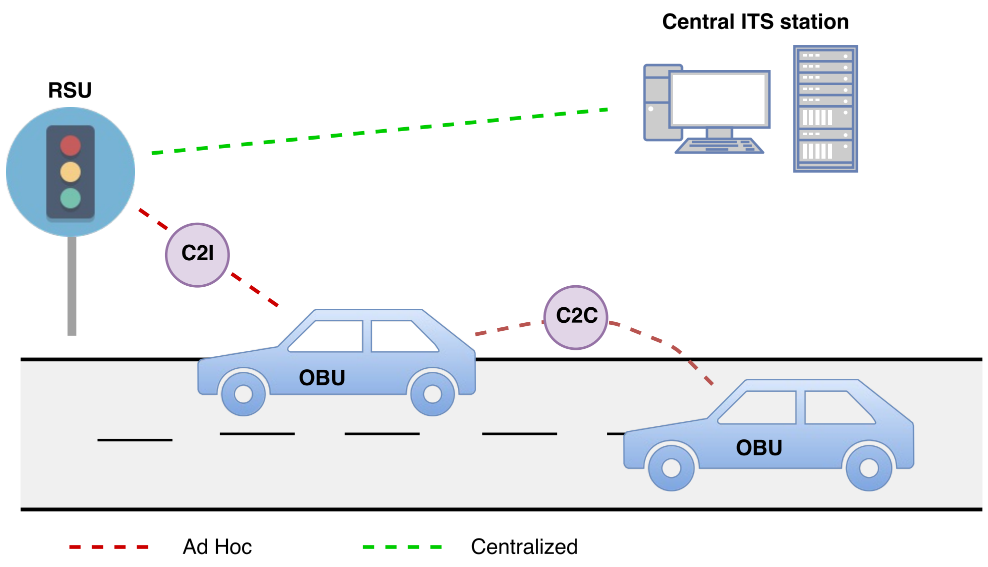


Figure 1.1: Communication Schemes of C2X Networking

There are many possible use cases of C2X communication within C-ITS systems. The *Drive C2X Consortium* has defined and standardized eighteen potentially usable C2X applications (use cases) listed in table 1.1. They are divided into three categories according to their main purpose: *Traffic Safety*, *Traffic Efficiency* and *Infotainment and Business* [9]. Some of the most common uses cases are the following:

- **Traffic jam ahead warning** (C2C, C2I): Vehicles approaching a traffic jam receive warning messages from roadside infrastructure (C2I) or from the cars that are already stuck in the jam (C2C). Probability of a rear-end collision is decreased.

- **Road works warning** (C2I): A roadwork acts as a roadside communication unit and broadcasts warning messages to the approaching vehicles. Similarly to the previous use case, drivers are informed in advance and possible collisions can be avoided.

- **Approaching emergency vehicle** (C2C): An emergency (privileged) vehicle informs nearby vehicles about its approach. The drivers receiving this message may react in advance which can increase both traffic efficiency and safety.

- **In-vehicle signage** (C2I): Traffic signs act as roadside units and inform the nearby vehicles about themselves. This decreases the chance that a driver does not notice an important traffic sign. A different approach is

C2X Use Case	**Purpose**
Traffic Safety	
Traffic jam ahead warning	Avoiding collisions with jammed vehicles
Road works warning	Awareness of potentially dangerous road work conditions
Car breakdown warning	Avoiding collisions with disabled vehicles
Approaching emergency vehicle	Awareness of approaching privileged vehicles
Weather warning	Awareness of critical weather conditions ahead
Emergency electronic brake lights	Avoiding collisions with an unexpectedly braking vehicle ahead
Slow vehicle warning	Avoiding collisions with slower vehicles ahead
Post crash warning	Awareness of traffic accidents ahead
Obstacle warning	Avoiding collisions with obstacles blocking the road
Motorcycle warning	Awareness of approaching motorcycles
In-vehicle signage	Awareness of traffic signs along the route
Traffic Efficiency	
GLOSA	Improving traffic flow at road intersections
Traffic information	Improving traffic flow by route recommendations
Infotainment and Business	
Insurance and Financial Services	Providing information to insurance companies after accidents
Dealer Management	Offering products to drivers (advertising)
Point of interest notification	Awareness of places such as sights and parkings along the route
Fleet management	Fleet management collaboration across organizations
Transparent leasing	Enforcing agreed leasing parameters (e.g. distance driven)

Table 1.1: C2X Use Cases [9]

that a car recognizes traffic signs using a built-in camera. This is part of the *Advanced Driver Assistance Systems* (ADAS).

- **GLOSA** (C2I): An intersection controller broadcasts information about the phase schedule of each traffic light signal to nearby vehicles. Approaching vehicles can calculate an optimal driving speed and avoid any unnecessary stops. This helps to maintain the traffic flow continuity and generally increases the traffic efficiency.

Figure 1.2 illustrates expected phases of C-ITS technology adoption. Since most of the applications defined above provide only notifications and warnings to the driver, their successful adoption can be expected during the first two phases. The third and fourth phase deal primarily with adoption of automated assistant systems and will become important in the future, especially for deployment of partially or fully autonomous cars [10].

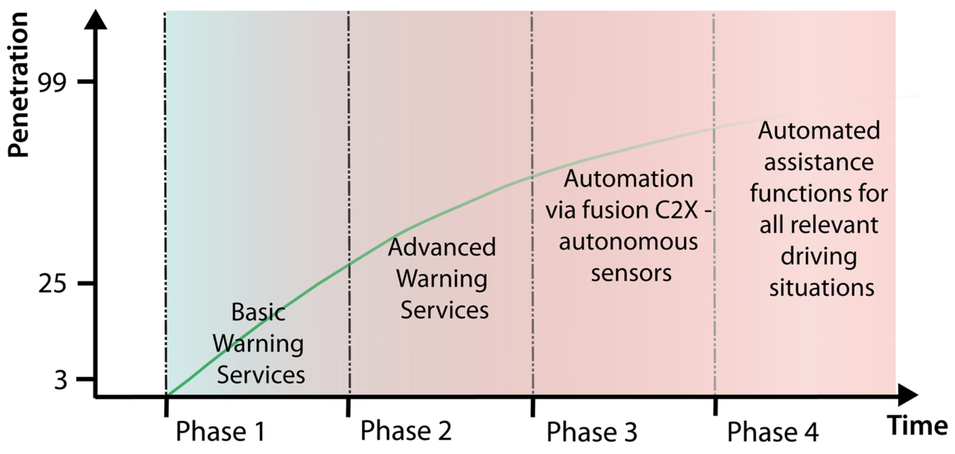


Figure 1.2: Phase Concept of C-ITS Adoption [10]

1.2 Technical Background of C2I Communication Systems

There are different technologies used for car-to-x communication networks around the world. This work focuses on European ITS technologies and protocols as defined by the *European Telecommunications Standards Institute* (ETSI). Although many standards are still under development and change rapidly, ETSI has already defined a complete protocol suite to be used for ITS/C2I systems [11]. Furthermore, the *European Committee for Standardization* (CEN) has published first *European Standards* based on the ETSI technical standards.

Besides ETSI, organizations such as IEEE, SAE (*Society of Automotive Engineers*), ARIB (*Association for Radio Industry and Business*) and ISO are the major players in developing technical ITS standards around the world.

As shown in figure 1.3, the ETSI protocol stack consists of several layers: Access, Networking & Transport, Facilities, Applications, Management and Security. The rest of this section describes each layer in detail.

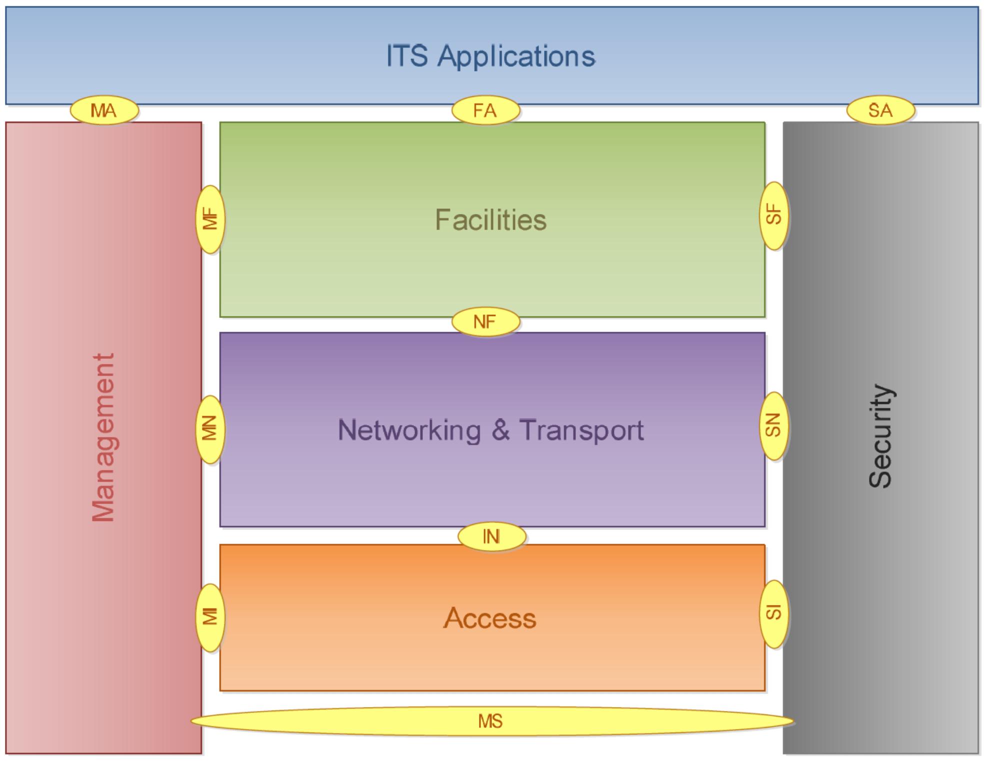


Figure 1.3: ETSI ITS Network Architecture [11]

(Abbreviations MS, MN, MF etc. are names of interfaces between the corresponding layers)

1.2.1 Access Layer

The *Access Layer* corresponds to the physical (L1) and data link (L2) layers of the well-known ISO/OSI networking model [11]. Thus, it deals with the hardware connection to the physical medium, link control and transmission of the data frames between the communicating devices.

It is possible to use various technologies within the access layer according to the application needs and available resources. For the car-to-x networks ETSI defines a protocol called ITS-G5 which is just a very slightly modified IEEE 802.11p. The only distinction is a different frequency range since the 802.11p is designed to be used primarily in North America, while ITS-G5 targets Europe. Both protocols operate in the licensed ITS band of 5.9 GHz, but ITS-G5 uses the 5.855–5.925 GHz range and IEEE 802.11p the 5.850–5.925 GHz range. In comparison, a different frequency range of 5.770–5.850 GHz is used in Japan [12][13]. These frequency bands are shown in figure 1.4 together with a corresponding frequency band reserved by the *Radiocommunication Sector* of the *International Telecommunication Union* (ITU-R) for industrial, scientific and medical application (*ISM band*).

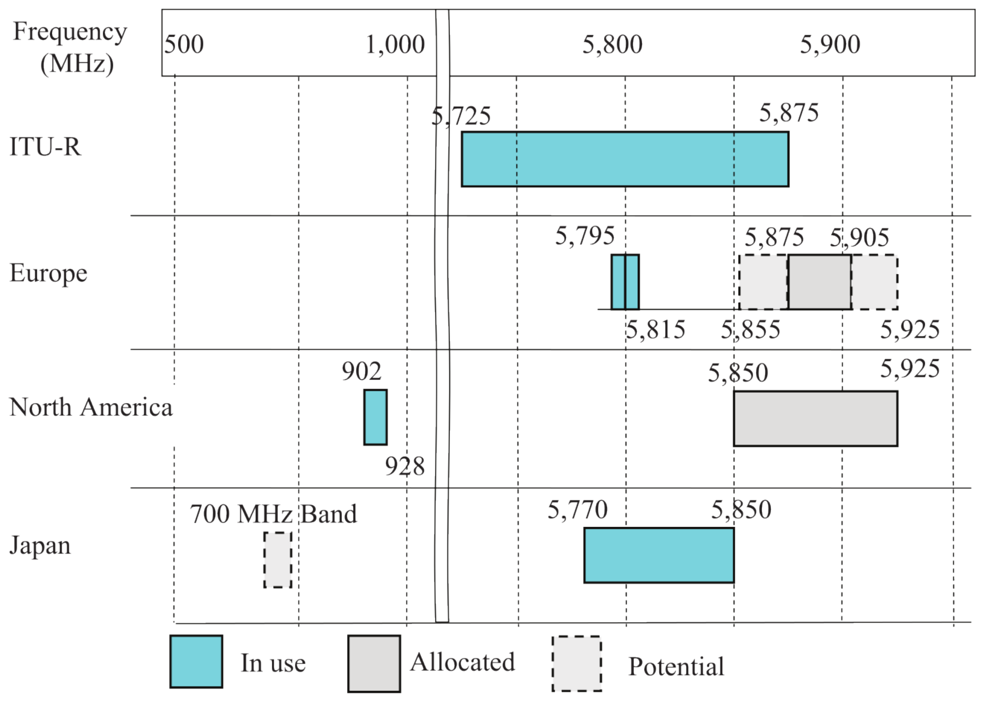


Figure 1.4: C2X Frequency Bands [13]

IEEE 802.11p is an amendment to the 802.11 standard for WLAN, better known as *Wi-Fi*. The main difference is that 802.11p can operate outside a *Basic Service Set* (BSS), while the traditional *Wi-Fi* cannot [14]. It means data can be transmitted at any time with no need to wait for an association and authentication to a wireless access point. In the car-to-infrastructure networks, this eliminates an unacceptably long delay before an actual data transfer may take place. However, since there is no authentication, there is also no identity protection and no data confidentiality. Thus, all the security services must be provided by upper network layers.

Theoretical communication range of the 802.11p is 1,000 meters line of sight as specified by the standard. However, several experimental evaluations have proved that this requirement cannot be satisfied in a real traffic. For example, a reliable communication distance of only 300 meters was achieved in a city environment (Hamburg, Germany) as presented in [15]. Another evaluation determined the maximum distance as 750 meters (with a 90% packet delivery rate) but in a highway scenario instead of city [16].

1.2.2 Network & Transport Layer

Analogously to the ISO/OSI model, ETSI *Network & Transport Layer* corresponds to the ISO/OSI network (L3) and transport (L4) layers. They deal mainly with addressing of communication nodes, message routing and reliable data delivery. ETSI defines two protocols to be used within this layer: *GeoNetworking* and *Basic Transport Protocol* (BTP). Alternatively, TCP/IP protocols such as IP, TCP and UDP may be used for specific applications [11].

GeoNetworking is a network-layer protocol that provides two basic functions – geographical addressing and geographical forwarding. Instead of traditional addressing scheme where an address is based on a node identity (e.g. MAC and IP addresses), GeoNetworking address is based on a geographical location of the communicating node. It means each packet carries an exact geographical location of both sender and receiver. When a node receives a packet, it may forward it further depending on the destination address (multi-hop transfer). Thus, all nodes may act as routers/forwarders, but there is no need to setup routing tables or any similar infrastructure in advance. This also allows for a data transfer between distant nodes that cannot communicate directly [17].

Five different routing and forwarding schemes for GeoNetworking have been defined. These are illustrated in figure 1.5 and described below:

- **GeoUnicast:** A packet is destined for a single receiver (geographical point). The route may involve multiple hops and each node along the way forwards the packet until it reaches its destination.

- **GeoMulticast:** A packet is destined for a specific group of nodes. This may consist of multiple GeoUnicast transmissions.

- **GeoBroadcast:** A packet is destined for all nodes in a specific geographical area. Similarly to the GeoUnicast routing, the packet is being forwarded until it reaches the destination area. The receiving nodes within the destination area rebroadcast the packet so it can reach all the nodes in that specific area.

- **GeoAnycast:** A packet is destined for any node in a specific geographical area. Routing is the same as GeoBroadcast, but there are no rebroadcasts within the destination area.

- **Topologically-scoped broadcast:** A packet is destined for all nodes in an n-hop neighborhood (i.e. nodes that can be reached in up to n hops). Routing is similar to a traditional broadcast, but a number of allowed rebroadcasts is limited by the value of n. This also corresponds to broadcasting packets with a time-to-live value equal to n.

Basic Transport Protocol is a transport-layer protocol similar to UDP. It provides a connection-less data transport between end nodes. This means that any intermediate hops (e.g. caused by the GeoNetworking forwarding) are completely transparent to BTP. It is a lightweight protocol with unreliable

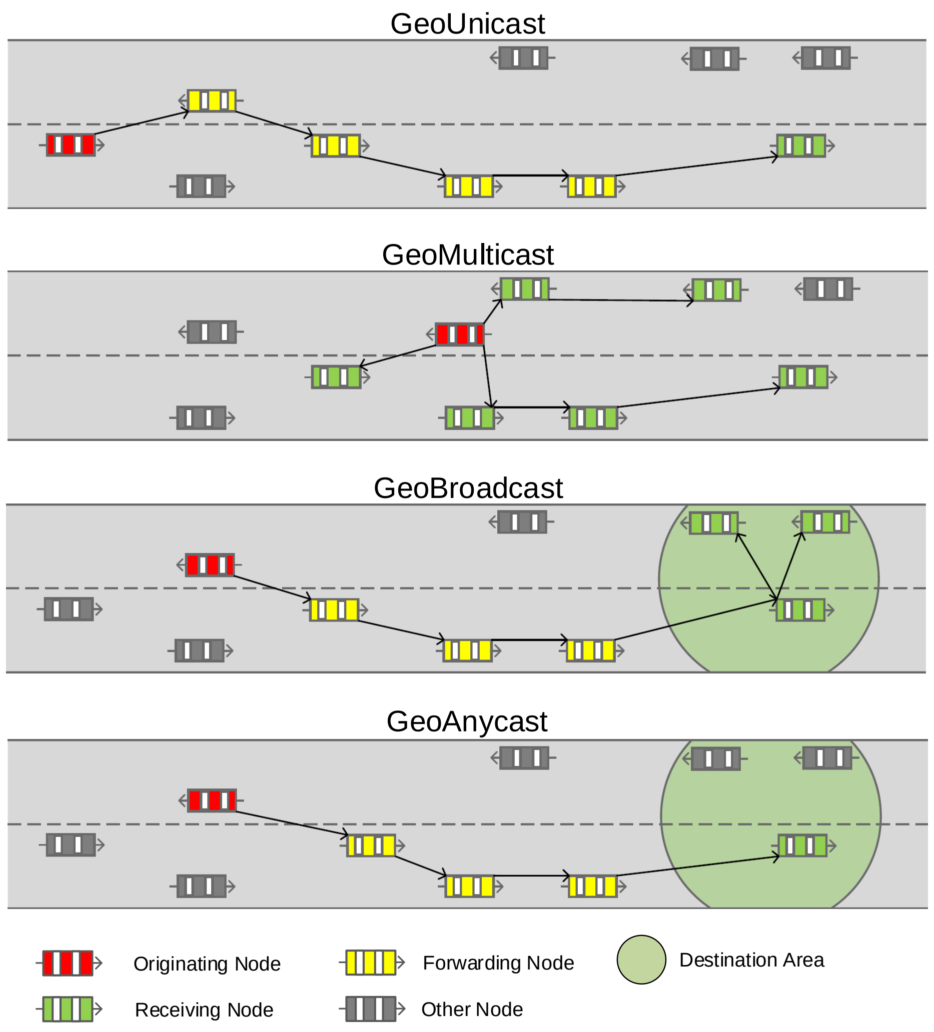


Figure 1.5: GeoNetworking Routing Examples [17]

packet delivery. Packets might be received out-of-order or even dropped during the transfer. The main function of BTP is (de)multiplexing messages of different applications. For this purpose, it uses source and destination port numbers in the same way as UDP and TCP do [18].

1.2.3 Facility Layer

ETSI *Facility Layer* covers the relation (L5), presentation (L6) and partially also application (L7) layers of the ISO/OSI model [11]. Its main purpose is to provide standardized messages (called facilities here) that can be used by applications. It also defines specific facilities to support communication, session and information (data) management [19].

Facilities can be split into several categories. A few examples of existing facilities are listed for each category:

- **Common facilities:** These facilities provide core services common to all applications and should be supported by all ITS stations (communication nodes).
  - Application support facilities: CAM management (see below), time management, security access management etc.
  - Communication support facilities: communication management, addressing mode etc.
  - Information support facilities: position management, data presentation etc.
- **Domain-specific facilities:** These facilities provide specific services relevant to a certain group of applications or only to a single application. Implementation at an ITS station is optional.
  - Application support facilities: DEN management (see below), billing and payment, GIS support etc.
  - Communication support facilities: session support
  - Information support facilities: LDM (see below), map data base, user repository etc.

There are three particularly important facilities that should be mentioned here: *Co-operative Awareness Message* (CAM), *Decentralized Environmental Notification* (DEN) and *Local Dynamic Map* (LDM).

Co-operative Awareness Message is an application-independent facility designed to be periodically (1-10 Hz) exchanged between all road users (ITS stations) including vehicles, roadside infrastructure or possibly even bicycles, pedestrians and similar. This might be achieved by using e.g. GeoBroadcast messages. Basic information such as station position, type, moving direction and speed are transmitted. Such co-operative awareness might be very useful for applications dealing with traffic safety and efficiency [20].

Unlike CAM, *Decentralized Environmental Notification* is a domain-specific event-driven facility, completely controlled by the application. It defines a standard message format for general event notifications. Information such as event position, type (e.g accident, traffic jam, roadworks), duration and status (e.g. new, update, cancellation) are transmitted [21].

Local Dynamic Map is the most important information-support facility. It provides a data storage and information management for the *Application Layer*. For example, data received by CAM and DEN messages are maintained by the LDM. Additional services such as time (and/or location) based data retention rules and data request prioritization are also supported by the LDM [22].

1.2.4 Application Layer

ETSI has defined a *Basic Set of Applications* (BSA) covering most of the considered use cases of an ITS system. This specification includes requirements of functions, performance, message contents etc. It also provides basic implementation guidelines, but exact technical details are left unspecified. All the applications are supposed to use the ETSI protocol stack (ITS-G5, GeoNetworking, BTP, Facilities). However, there might be many other possible applications using different technologies, but these are not specified by ETSI [19].

There are 32 ITS applications (use cases) in the BSA divided into four different categories as outlined below:

- **Active road safety:** Applications mostly intended to warn drivers of unexpected and potentially dangerous conditions along the way. Examples of these applications are: *Emergency Vehicle Warning*, *Intersection Collision Warning*, *Wrong Way Driving Warning*, *Roadwork Warning.*

- **Co-operative traffic efficiency:** Applications intended to help drivers (through notifications or assist systems) to achieve a higher traffic efficiency. Examples of these applications are: *Green Light Optimal Speed Advisory* (GLOSA), *Speed Limits Notification*, *Route Guidance and Navigation*, *Detour Notification.*

- **Co-operative local services:** Location based services, mostly intended for supplementary non-critical purposes. These deal neither with traffic safety nor efficiency. Examples of such applications are: *Point of Interest Notification*, *Local Electronic Commerce*, *Media Downloading.*

- **Global internet services:** Unlike the local services, Global internet services strongly depend on a backend infrastructure such as internet connection. Examples of such applications: *Insurance and Financial Services*, *Fleet Management*, *Vehicle Software/Data Provisioning and Update.*

1.2.5 Management Entity

ETSI *Management Entity* provides a cross-layer management functions with interfaces to all the other layers within the ETSI protocol stack. Its main purpose is to handle common management for the core services and to allow certain cross-layer communication channels that are needed for correct functionality of the ETSI protocol stack [11].

A core component of the *Management Entity* is a *Management Information Base* (MIB) that stores all the parameters (settings) of the ITS communication objects, i.e. of the addressable instances of functionality of ITS stations. The MIB is designed in accordance with RFC 3410 and thus it can be accessed using the *Simple Network Management Protocol* (SNMP) version 2 [23].

Another important feature is a *Decentralized Congestion Control* (DCC). It attempts to optimize the transmit parameters of the outgoing packets based on the current congestion level of the communication channel. It also determines priorities between different types of messages to achieve required quality of service (QoS) [24].

1.2.6 Security Entity

ETSI *Security Entity* consists of two separate parts as depicted in figure 1.6. A cross-layer *Security Management Plane* and layer-by-layer core security services [25].

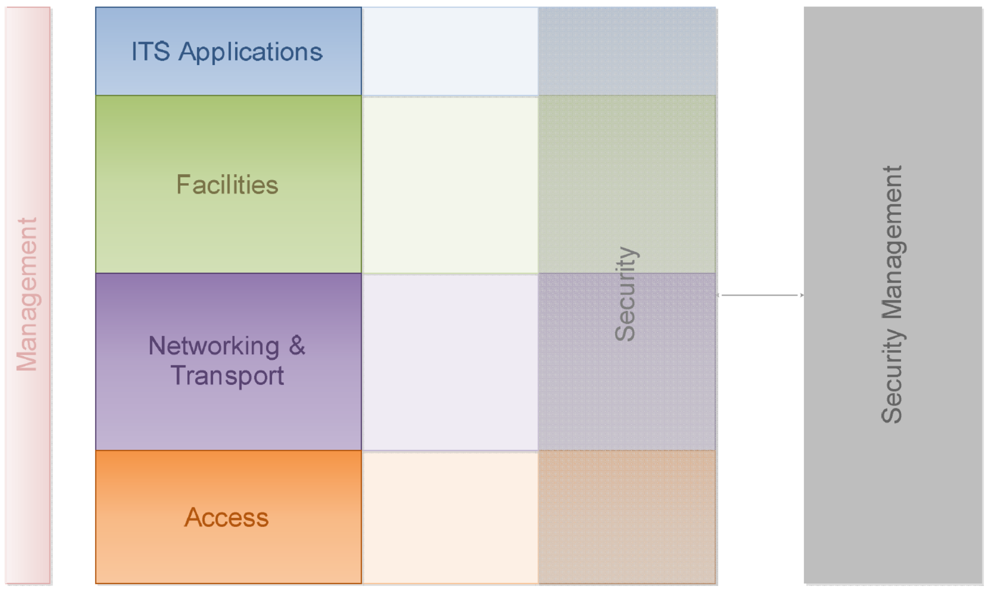


Figure 1.6: ETSI Security Entity [25]

Security Management Plane is a cross-layer entity similar to the *Management Entity* which is described in section 1.2.5. It provides security services common to all the layers within the ETSI protocol stack. Examples of such services are *Firewall and Intrusion Management*, *Identity Management*, *Authentication and Authorization Management*, *Enrolment Management*.

Most of the security services are, however, provided on a layer-by-layer basis so that each service is part of either Access, Network & Transport, Facility or Application Layer. Examples of these services are the following:

- **Access Layer:** *Sender and Receiver Identification*
- **Network & Transport Layer:** *Message Integrity Verification*, *Message Digital Signature*
- **Facility Layer:** *Message Payload Encryption and Decryption*, *Message Payload Timestamp*
- **Application Layer:** Not defined by ETSI.

Definitions of all the security services specified by ETSI can be found in [26], but it is beyond the scope of this work to describe them here. Nevertheless, ETSI specifies neither exact technical methods nor implementation details of any particular service and thus actual implementations may vary.

1.3 Green Light Optimized Speed Advisory Systems (GLOSA)

Green Light Optimized Speed Advisory Systems (GLOSA) is one of the ITS applications defined in the ETSI *Basic Set of Applications.* It is categorized as a speed management application within the co-operative traffic efficiency application class.

Main purpose of GLOSA is to increase traffic efficiency by helping drivers to prevent any unnecessary stops at traffic lights. It involves a car-to-infrastructure communication between a car and an ITS-equipped traffic intersection. The intersection controller broadcasts information about the phase schedule of each traffic light signal to nearby vehicles. The broadcasted messages should contain at least the intersection topology (map), current phase of each light signal (green/yellow/red) and remaining time to the end of that phase. Using this information, the approaching vehicles can calculate an optimal approaching speed and avoid any unnecessary stops [6].

Benefits of a functional GLOSA system are primarily an improved traffic flow continuity, higher travel comfort, lower fuel consumption (up to 20% difference [27]) and lower environmental impact, especially reduced noise and CO2 emissions. This is particularly beneficial in urban areas with a dense road traffic.

1.3.1 ETSI specification of GLOSA

ETSI specified and standardized functional, operational and message content requirements for GLOSA systems [19]. These requirements are specified on a high level of abstraction and do not include any technology or implementation related requirements.

Operational requirements:

1. Availability of a GLOSA application should be announced by an authorized roadside ITS station (intersection controller).

2. The roadside unit should provide information about the signal phase schedule in a *Signal Phase and Timing* (SPaT) message and information about the intersection topology in an *Intersection Topology* (MAP) message. GeoBroadcast should be used for transmitting these messages.

3. SPaT and MAP messages should be broadcasted periodically at a given frequency, exact transmission frequency is not specified by ETSI. The SPaT message should be updated with each transmission. This process is fully controlled by the roadside unit.

4. The on-board unit (vehicle) should keep receiving the broadcasted messages and possibly update the speed recommendation.

The operational requirements presented above cover the basic operation of a GLOSA system. ETSI also defined eleven functional requirements, but these are only formalized principles already contained in the operational requirements. Nevertheless, a few of these requirements are worth mentioning:

- An authorized roadside ITS station shall transmit the relevance area for the traffic light phase and timing information.

- The relevance area shall allow matching to a specific road section or specific lane, if the provided traffic light phase and timing is lane specific.

- The receiving vehicle ITS station shall check the relevance of the information.

- The vehicle ITS station should keep the speed advice information at least when vehicle is still located in the intersection area.

1.3.2 SPaT and MAP messages

The ETSI operational requirements for GLOSA specify usage of the *Signal Phase and Timing* (SPaT) and *Intersection Topology* (MAP) messages. These messages have been defined and standardized by the *Society of Automotive Engineers* as SAE J2735 [28]. However, this standard is intended to be used primarily in North America. A corresponding European standard (ISO/CEN TS 19091) is still under development and not available to public as of March 2016 [29]. This European standard will be, however, based on SAE J2735 with only minor adjustments for European usage. Furthermore, an Europe variant of the SAE J2735 message set definition is available [30]. According to SAE,

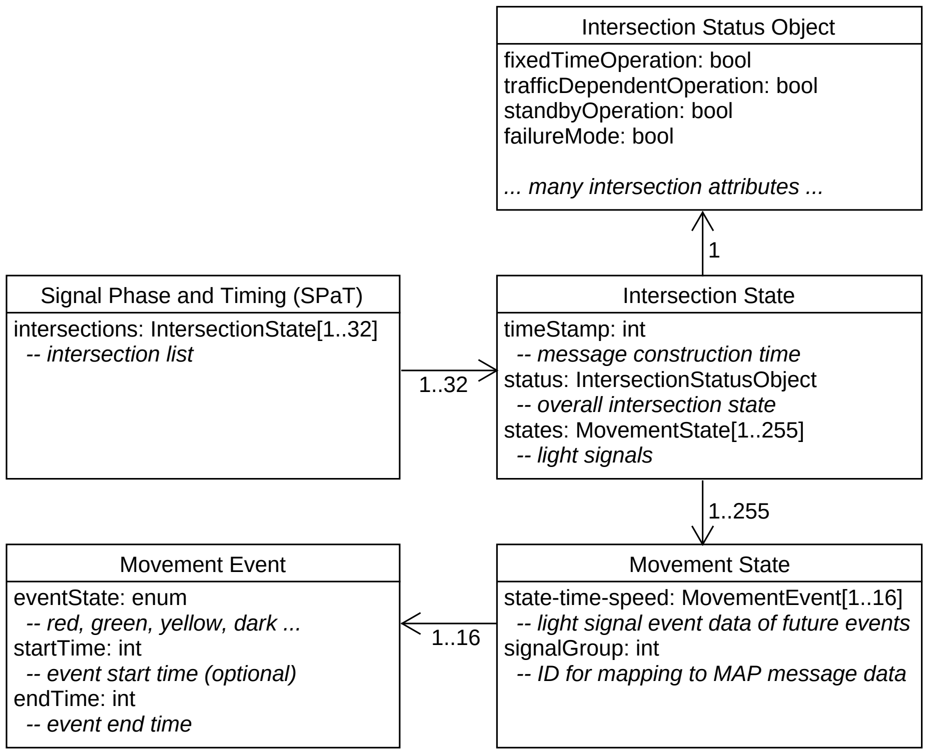

Intersection Status Object
fixedTimeOperation: bool
trafficDependentOperation: bool
standbyOperation: bool
failureMode: bool
... many intersection attributes ...
1
Signal Phase and Timing (SPaT)
intersections: IntersectionState[1..32]
-- intersection list
1..32
Intersection State
timeStamp: int
-- message construction time
status: IntersectionStatusObject
-- overall intersection state
states: MovementState[1..255]
-- light signals
1..255
Movement Event
eventState: enum
-- red, green, yellow, dark ...
startTime: int
-- event start time (optional)
endTime: int
-- event end time
1..16
Movement State
state-time-speed: MovementEvent[1..16]
-- light signal event data of future events
signalGroup: int
-- ID for mapping to MAP message data

Figure 1.7: SPaT Message Structure (simplified) [30]

it is using both SAE J2725 and ISO 19091 draft. The following description is based on the 1.0 version of this document.

The *Signal Phase and Timing* message contains information about phase schedule of each traffic light in one or more (up to 32) intersections. A simplified structure of the message is shown in figure 1.7 in form of a UML class diagram. The *Intersection State* entity represents a single intersection and the *Intersection Status Object* carries its attributes. Examples of the attributes are intersection operation status (ok, failure, standby ...), current signal scheduling mode (fixed, depending on traffic ...) and similar.

An intersection consists of up to 255 *Movement States*. A *Movement State* is a set of lanes belonging to a single traffic light signal. The key information is contained in the *Movement Event* entity. This represents a particular signal phase (e.g. green) and its timing, especially end time of the phase. Other time values such as start time are optional. Up to 16 *Movement Events* might be associated with a single *Movement State*. This allows to include information about the current and also the future signal phases in a single message.

Figure 1.8 demonstrates a SPaT message data mapped to a sample intersection. Each red or green arrow belongs to a particular *Movement State* and

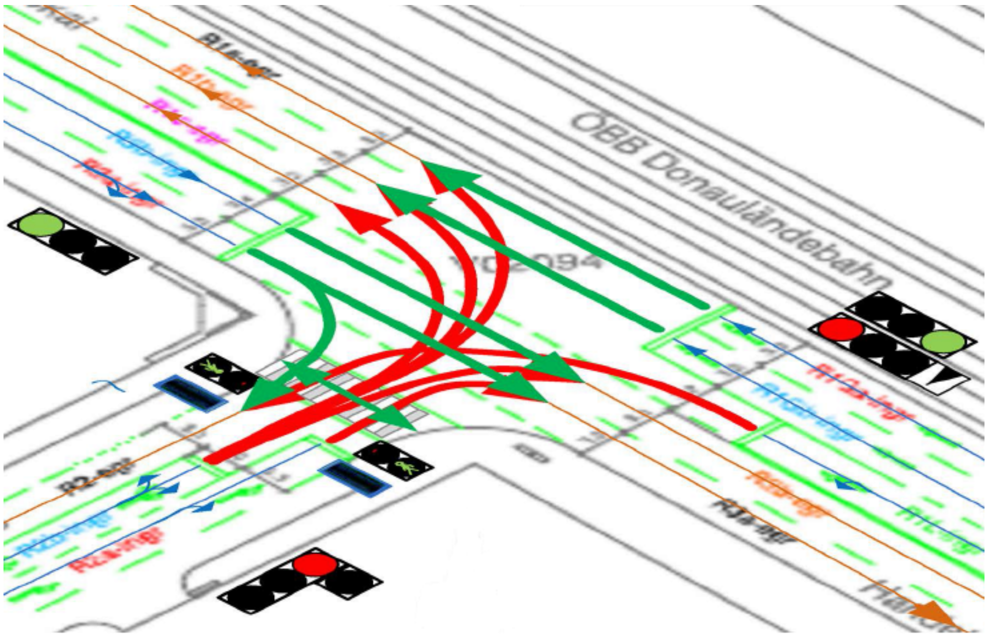


Figure 1.8: Intersection Example (SPaT) [31]

its color represents the current signal phase, i.e. an *Movement Event* entity. Any future phases, which can be included in the message as well, are not shown here.

The *Intersection Topology* (Map Data, MAP) message contains geographical data of one or more (up to 32) intersections. A simplified structure of the message is shown in figure 1.9. Complexity and all the possible content of this message is beyond the scope of this work and thus only the main parts are presented here.

The *Intersection Geometry* entity represents a single intersection consisting of up to 255 road lanes (*Generic Lane*). Each lane has various attributes (*Lane Attributes*) such as its type (car, bus, bicycle) and allowed move direction (one-way, two-way). It is described by a list of geographical points encoded as additive offsets to an *Intersection Reference Point* (any fixed point with exact geographical coordinates). A lane can also connect to other lanes and such connection is represented by the *Connection* entity. This means a vehicle can arrive at the intersection in certain lane and leave it in any of its connecting lanes.

Figure 1.10 demonstrates a MAP message data mapped to a sample intersection. Each colored thick arrow corresponds to a *Generic Lane*. The black circles are its geographical points (*Nodes*) with a single *Intersection Reference Point* among them. The thin black arrows are *Connections* between lanes.

Upon receiving SPaT and MAP messages, it is necessary to map the light

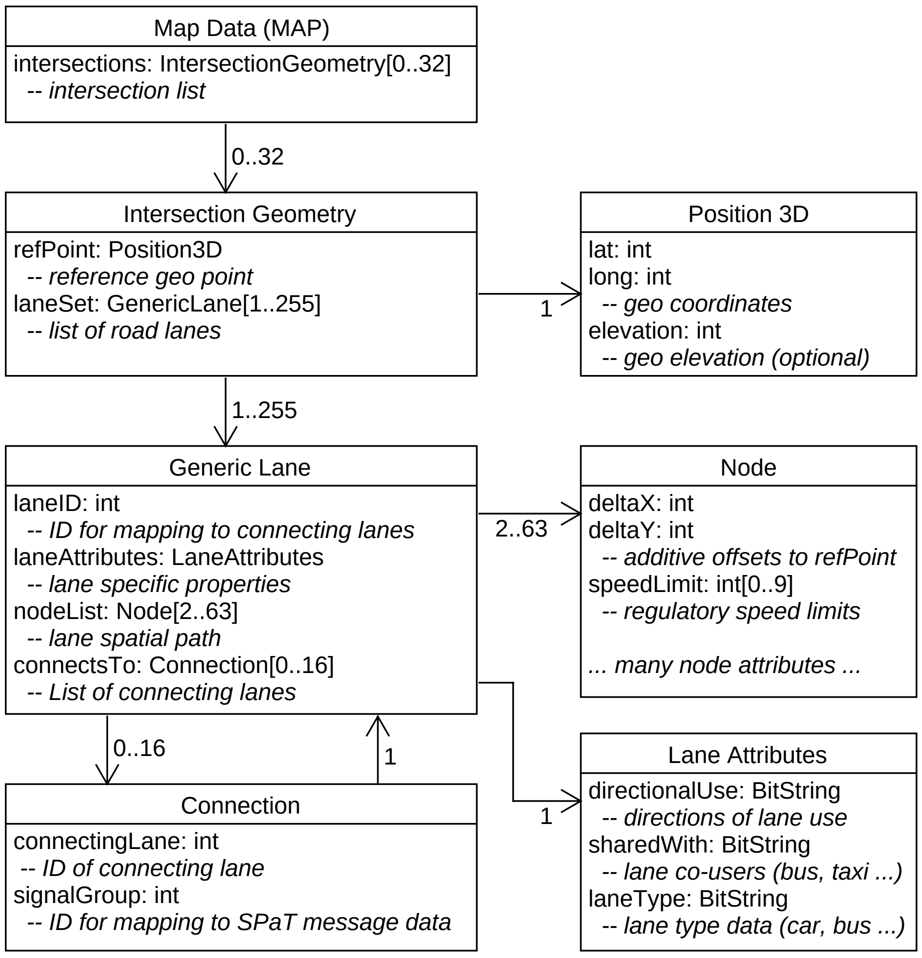

Map Data (MAP)
intersections: IntersectionGeometry[0..32]
-- intersection list
0..32
Intersection Geometry
refPoint: Position3D
-- reference geo point
laneSet: GenericLane[1..255]
-- list of road lanes
1
Position 3D
lat: int
long: int
-- geo coordinates
elevation: int
-- geo elevation (optional)
1..255
Generic Lane
laneID: int
-- ID for mapping to connecting lanes
laneAttributes: LaneAttributes
-- lane specific properties
nodeList: Node[2..63]
-- lane spatial path
connectsTo: Connection[0..16]
-- List of connecting lanes
2..63
Node
deltaX: int
deltaY: int
-- additive offsets to refPoint
speedLimit: int[0..9]
-- regulatory speed limits
... many node attributes ...
0..16
1
Connection
connectingLane: int
-- ID of connecting lane
signalGroup: int
-- ID for mapping to SPaT message data
1
Lane Attributes
directionalUse: BitString
-- directions of lane use
sharedWith: BitString
-- lane co-users (bus, taxi ...)
laneType: BitString
-- lane type data (car, bus ...)

Figure 1.9: MAP Message Structure (simplified) [30]

signals to the corresponding geographical data. This can be achieved by using the *signalGroup* attribute of the *Movement State* (SPaT message) and *Connection* (MAP message) entities. Each *Movement State* should have a unique value of *signalGroup* and all the *Connections* with the same value belong to this *Movement State*, i.e. traffic light signal.

1.3.3 Existing GLOSA Implementations

There have been several successful realizations of GLOSA systems in a real world environment. Most of these projects were implemented as proofs of concept, usually to demonstrate abilities of the cooperative ITS technology and its potential for the future. Most of them are also part of ongoing research

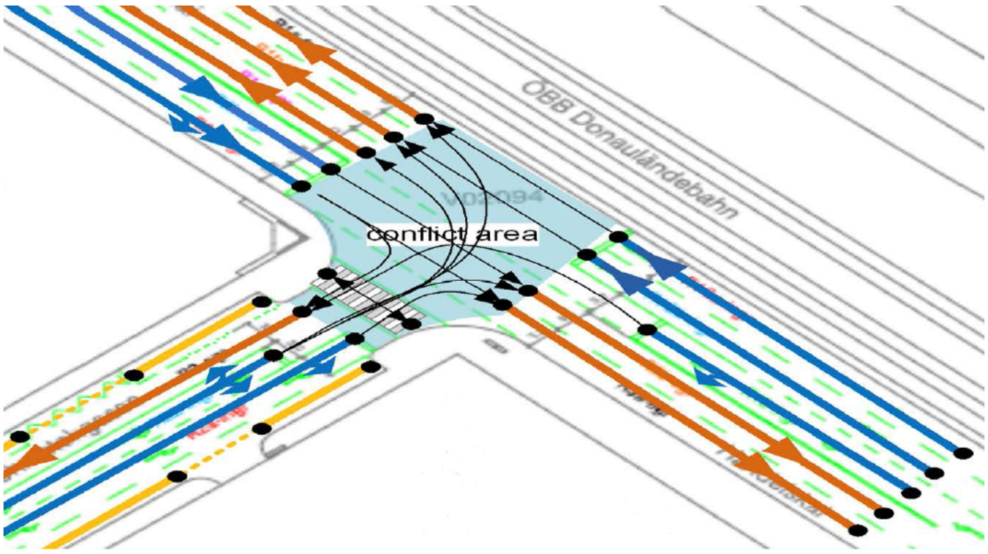


Figure 1.10: Intersection Example (MAP) [31]

activities of major car manufacturers. Some projects even do not satisfy the requirements for GLOSA as defined by ETSI (see 1.3.1). This section presents three interesting examples of successful GLOSA realizations.

Audi Travolution is a collaboration between Audi and many of its partners (City of Ingolstadt, Scheidt & Bachmann GmbH, TaxiFunk Ingolstadt, ADAC etc.). One part of this project is a GLOSA system deployed at 25 intersections in Ingolstadt, Germany. 15 of these intersections use car-to-infrastructure WLAN networks allowing for a direct communication between cars and the intersections. The remaining intersections send data to a backend server located in Ingolstadt city centre. It is then distributed to vehicles using 3G (UMTS) mobile network. The transmitted messages are similar to SPaT and MAP described in 1.3.2. Besides displaying a recommended speed to the driver, two of the test cars have been equipped with an adaptive cruise control connected to the GLOSA system. In this setup, car speed is automatically adjusted to match the recommended speed [32].

Another interesting example is a recent C-ITS project funded by the European Union called *Compass4D* (*Cooperative Mobility Pilot on Safety and Sustainability Services for Deployment*). Particularly important is the *Energy Efficient Intersection* service, designed and deployed in Verona, Italy. It includes a GLOSA system using the ITS-G5 (IEEE 802.11p) access technology and SPaT/MAP messages to communicate with passing vehicles. The project involves 25 intersections, 10 buses and 30 cars equipped with the corresponding C-ITS technology. In addition to the WLAN approach, some intersections use 4G (LTE) network as a possible alternative to the ITS-G5 [33].

A large GLOSA implementation has been carried out by Swarco and Audi in Berlin. It involves over 800 intersections. However, they do not use any form of a direct car-to-infrastructure communication. Data are sent to a central server and then distributed to vehicles using 3G/4G mobile networks. It uses SPaT and MAP messages as standardized in SAE J2735. An interesting additional feature is a connection with the start-stop system in Audi vehicles. When the time to green is below a predefined threshold, the engine is not stopped [34].

Chapter **2**

Application Implementing GLOSA (*GreenLight*)

2.1 Detailed Description of *GreenLight* Application

GreenLight is a mobile application being developed by the *Intens Corporation* [35] in co-operation with *e4t electronics for transportation* [36]. Since *e4t* is a member of *Volkswagen Group*, it benefits from its strong internal research and can be a reliable partner for application development. The current version (released in 2015) can still be considered as a tech-alpha and a few more years are expected until the production phase is reached. The application is currently available only for mobile devices with the *Google Android* operating system.

The *GreenLight* application provides a user-faced car-side implementation of a GLOSA system. It periodically receives data from the intersection, calculates an optimal driving speed and displays it to the driver. An important feature is an integration into the car infotainment system through the *MirrorLink*^{TM} technology described in section 2.2. The application is designed universally and can be used with any ITS system satisfying the GLOSA requirements as defined by ETSI (see 1.3.1). This allows for an extensive deployment once the ITS technology becomes common. A sample photo of the running application integrated in the car infotainment system is shown as figure 2.1.

2.1.1 System Architecture

The overall communication architecture is illustrated in figure 2.2. The intersection controller (RSU unit) broadcasts SPaT and MAP messages as described in section 1.3. The exact communication schemes at the RSU side may vary and can be neglected here. The broadcasted messages are received by an

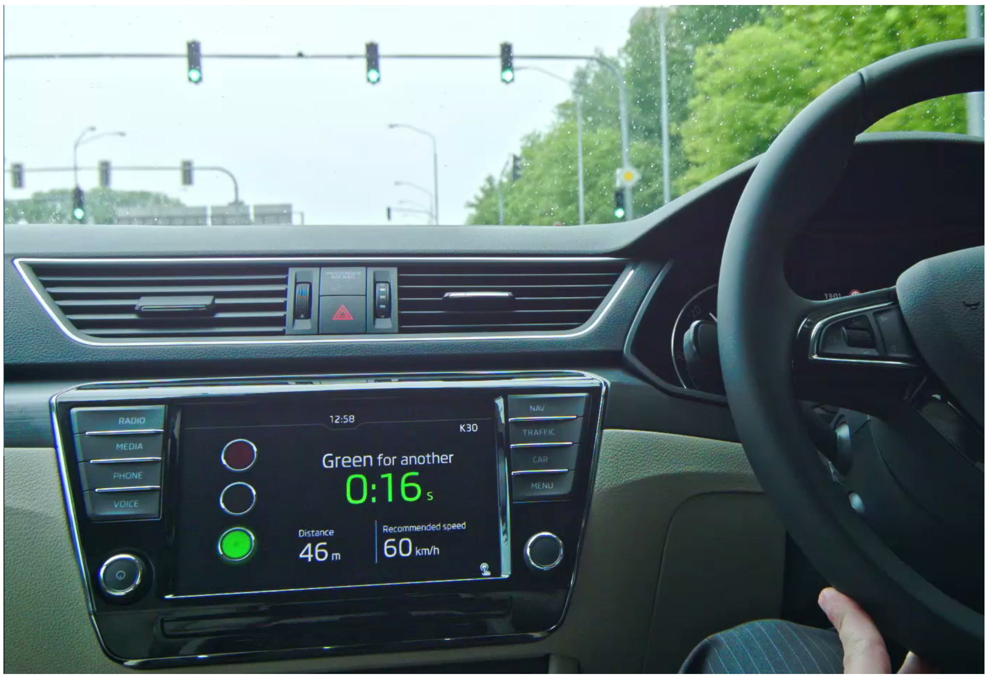


Figure 2.1: Running *GreenLight* Application

ITS communication unit within the car (OBU unit) which further provides an interface (API) for higher-level applications. *GreenLight* connects to the OBU unit by the *Bluetooth* protocol and obtains SPaT and MAP data using its API.

The application can also access information about the current state of the car. This includes car speed, acceleration or breaking force and blinking state. It uses a proprietary technology called *Škoda SmartGate*^{TM} and thus is only relevant for *Škoda Auto* cars. The connection is based on an in-car *Wi-Fi* network controlled by the *SmartGate*^{TM} unit [37].

To achieve a better integration with the car infotainment system, *GreenLight* uses the *MirrorLink*^{TM} technology. It allows the application to be hosted and run on a smartphone, but the user interface is mirrored to the car head-unit display (HUD). In addition, the driver can fully interact with the application using the HUD human-machine interface (HMI) instead of the smartphone itself. This is further described in section 2.2.

2.1.2 On-board ITS Unit

The *GreenLight* application does not handle the network communication by itself since that is controlled by a dedicated ITS communication unit in the car. Therefore, it is not required to use any specific communication technology

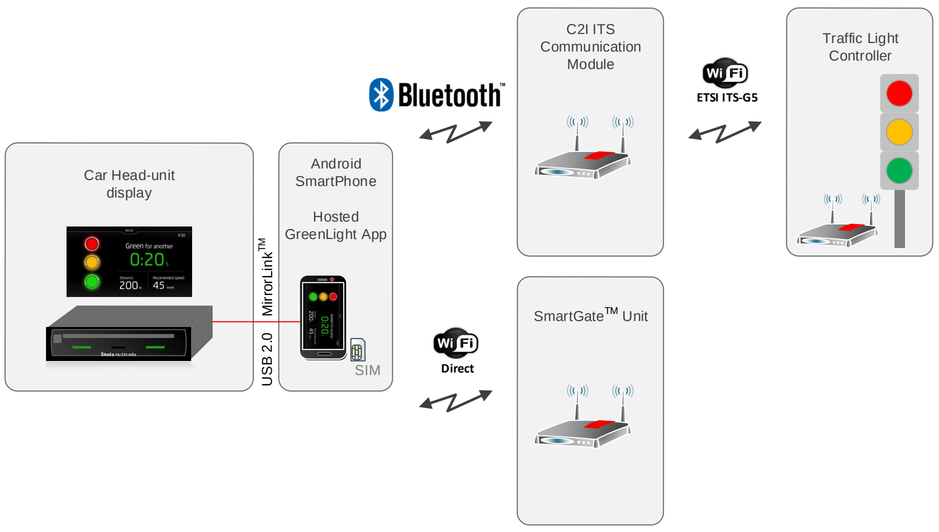


Figure 2.2: *GreenLight* Communication Architecture

between the car and an intersection controller. This increases a versatility of the application but requires to use an appropriate ITS unit.

Unfortunately, most of the existing ITS units provide proprietary and non-standardized application interfaces for communication with higher-level applications. This means *GreenLight* must have been designed with respect to a specific ITS unit type. In case it was needed to change the ITS unit, it would be necessary to adjust *GreenLight* accordingly or at least implement an appropriate wrapper library.

On-board ITS units supporting the ETSI protocol stack are developed by several different manufacturers. So far, *GreenLight* supports and has been tested with units produced by *Commsignia*, a company specialized in research, development and manufacturing of various C2X hardware and software products [38]. Another major player in the field of ITS unit development is *Kapsch*, an Austrian road telematics company [39].

2.1.3 *GreenLight* Function Explained

Displaying an optimal recommended speed is a fundamental and seemingly the only important goal of the application. However, it is not a trivial task and there are many particular features to achieve this functionality. The basic function and data processing flow within the *GreenLight* application is depicted in flowchart 2.3. The exact architecture of the application is an intellectual property of the *Intens Corporation* and cannot be described in this thesis.

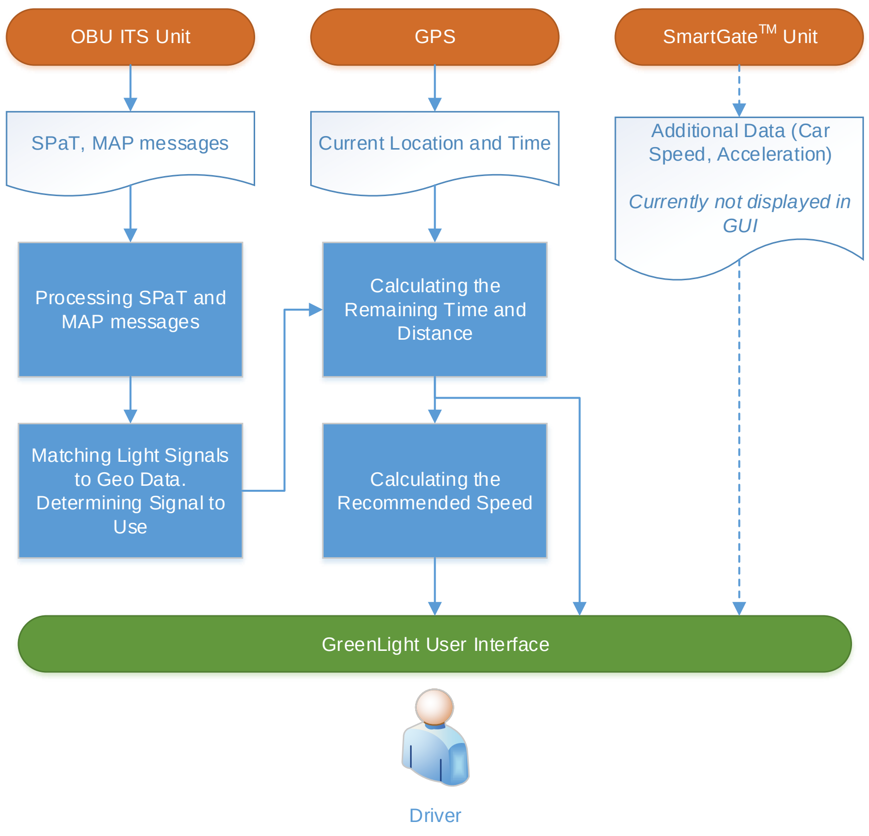

OBU ITS Unit
GPS
SmartGate™ Unit
SPaT, MAP messages
Current Location and Time
Additional Data (Car Speed, Acceleration)
Currently not displayed in GUI
Processing SPaT and MAP messages
Calculating the Remaining Time and Distance
Matching Light Signals to Geo Data. Determining Signal to Use
Calculating the Recommended Speed
GreenLight User Interface
Driver

Figure 2.3: *GreenLight* Data Processing Flow

The simplified data processing flow is the following:

1. Obtaining, parsing and processing the input data. These are SPaT and MAP messages received from an OBU ITS unit, current location and time obtained through the phone GPS sensor, car speed, blinking, acceleration and braking force available from the *SmartGate*^{TM} unit.

2. Matching the signal phase information (SPaT) to the geographical location data (MAP).

3. Determining the traffic light signal belonging to the straight direction.

4. Calculating the remaining distance to the intersection and the time until the signal phase is changed. This is based on SPaT data of the light signal determined in the previous step and the current time and location obtained from GPS.

5. Calculating an optimal approaching speed based on the remaining time and distance.

6. Displaying the current signal phase, remaining time and the recommended speed in the application user interface (mirrored to the car HUD).

The first, second and fourth functions have been implemented already and perform quite well. On the other hand, the remaining functions still need major improvements before the production phase can be reached. Only the traffic light corresponding to the intended driving direction should be displayed instead of the straight direction signal. This is further analyzed in chapter 4. Furthermore, performance and reliability of the current algorithm for calculating the recommended speed is insufficient and thus a completely new calculation is proposed, implemented and tested in chapters 5-6. Finally, the application user interface will possibly need to be adjusted to reflect all the implemented improvements in the application, but this is not an objective of this thesis.

2.1.4 *GreenLight* User Interface

Figure 2.4 shows the relatively simple user interface of the *GreenLight* application. The traffic light picture on the left side symbolizes the current state of the actual traffic light. In this situation, only the green light should be on, although all the lights are shown in the picture. The “08:20” value at the top shows the current time, but it is just an additional information for the driver and has nothing to the with the GLOSA system itself. On the other hand, the “K30” at the upper right corner signifies an intersection name as received from the intersection controller. In the future, this identification is expected to be used automatically by the car infotainment system for any location-related services such as navigation. Purpose of the remaining fields is self-explanatory. All the values are updated every second including recalculation of the recommended speed.

2.2 *MirrorLink^{TM}* – Technology Used in the Application

MirrorLink^{TM} is a car interoperability standard that started as a research project at *Nokia Research Center* in Palo Alto, USA. It is currently standardized by the *Car Connectivity Consortium* (CCC) consisting of many companies from different areas. Some of the most important members are the following [40]:

- **Car Manufacturers:** *Volkswagen Group*, *Daimler*, *General Motors*

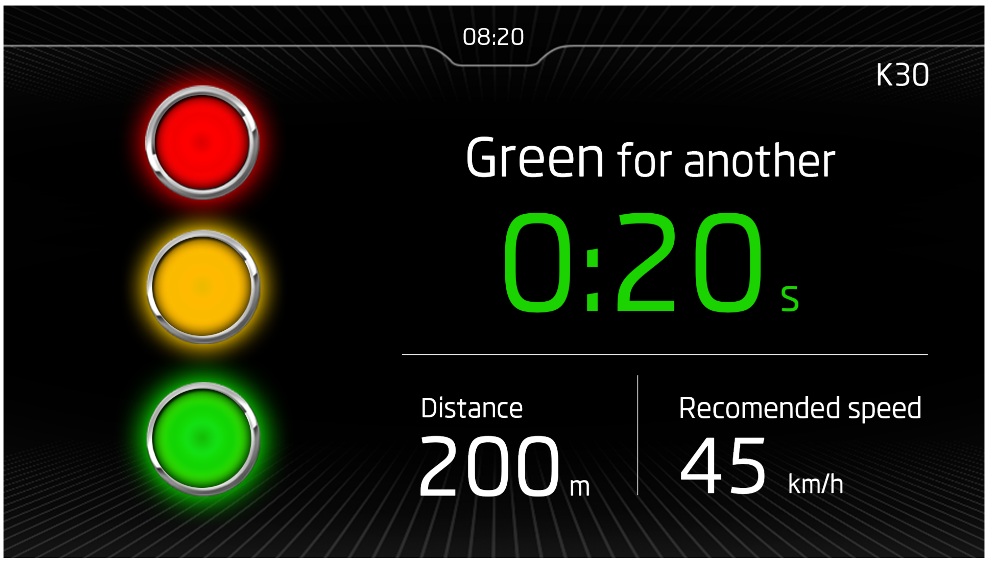


Figure 2.4: *GreenLight* User Interface

- **Phone Manufacturers:** *HTC*, *LG*, *Samsung*, *Sony*, *Microsoft*
- **Infotainment System Manufacturers:** *Panasonic*, *Pioneer*, *Bosch*
- **Other Partners:** *Garmin*, *TomTom*, *QNX*

MirrorLink^{TM} is a technology for connecting a smartphone to the car infotainment system. An application can be hosted and run on the smartphone but its user interface is mirrored to the car head-unit display. In addition, the driver can fully interact with the application using the HUD HMI instead of the smartphone itself.

2.2.1 Certification of *MirrorLink*^{TM} Applications

It is not possible to use the *MirrorLink*^{TM} technology with an arbitrary mobile application. The application needs to satisfy predefined criteria in order to get certified by CCC for the *MirrorLink*^{TM} usage. There are several certification levels as depicted in figure 2.5. Naturally, the higher levels request stricter requirements (and include the requirements of lower levels), but the application is then allowed to be used in more situations [41].

- **Regular Application:** Any mobile application
- ***MirrorLink*^{TM} Aware Application:** Application with implemented *MirrorLink*^{TM} services but without a valid certification. This includes applications with expired or revoked certificates.

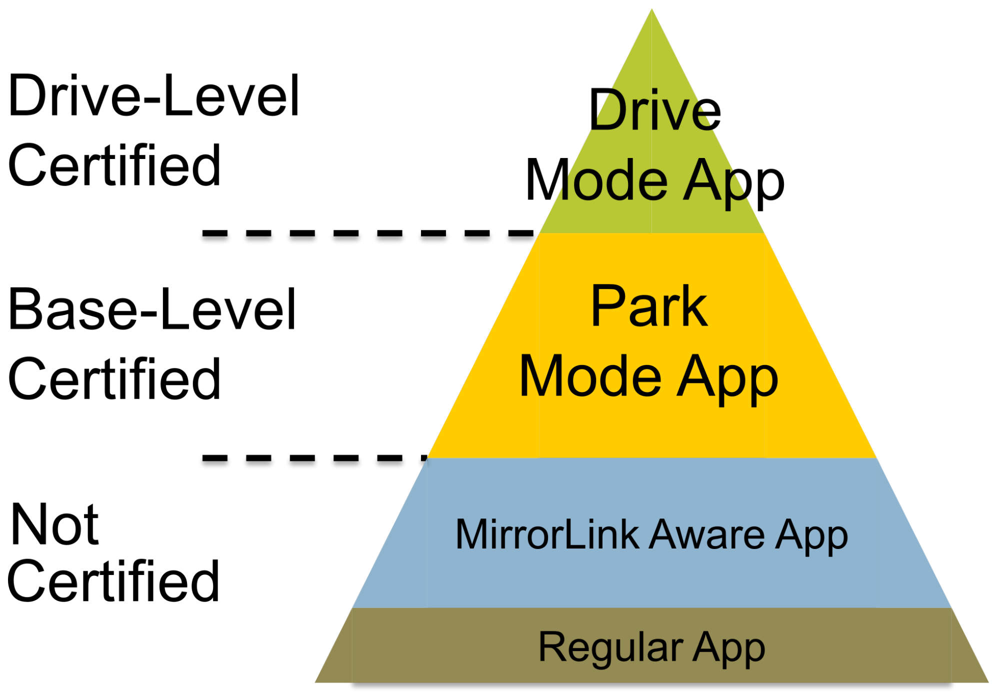


Figure 2.5: *MirrorLink^{TM}* Certification Levels [41]

- **Base-Level Certified Application:** *MirrorLink^{TM}* services are available when the car is not moving.
- **Drive-Level Certified Application:** *MirrorLink^{TM}* services are available also when driving.

Aim of the base-level certification requirements is to make sure that the application is fully compatible with the *MirrorLink^{TM}* technology. This includes display compatibility (resolution, landscape mode support), adequate control support (rotary-knobs, voice commands, single-touch events) and all the technical requirements described further in the next section.

A drive-level certified application is required to minimize any possible driver distraction. Since the driver distraction is usually limited and regulated by legal authorities, the exact requirements may differ according to the region (EU, North America, Asia-Pacific). However, CCC has defined a general guideline that is usable globally. Some of the major points are the following:

- **Restricted Content:** No video, animations, flashing or automatic scrolling text

- **Visual Accessibility:** High color contrast, text legibility
- **Control Accessibility:** No two-handed operations (e.g. multi-touch), forbidden keyboard usage
- **Pace of Interaction:** Input retention, responsiveness, no notifications

From the technical point of view, a *MirrorLink*^{TM} application certificate is an X.509 public key infrastructure certificate issued and digitally signed by the CCC certification authority (CA). CCC CA is an implicitly trusted CA for all *MirrorLink*^{TM} devices. A certificate contains information about the application (unique app ID), issuer and validity period. Management of these certificates is thus very similar to any other public key certificates, such as common personal or SSL certificates.

2.2.2 Technical Background

The *MirrorLink*^{TM} technology is based on the client-server communication model as illustrated in figure 2.6. A *MirrorLink*^{TM} server is a device (usually smartphone or tablet) running certified applications. It connects to a *MirrorLink*^{TM} client which is an infotainment display in the car, usually the head-unit display. Figure 2.7 overviews the protocol suite used for the client-server communication. It can be divided into three basic layers: connectivity, data services and *Common API* [42].

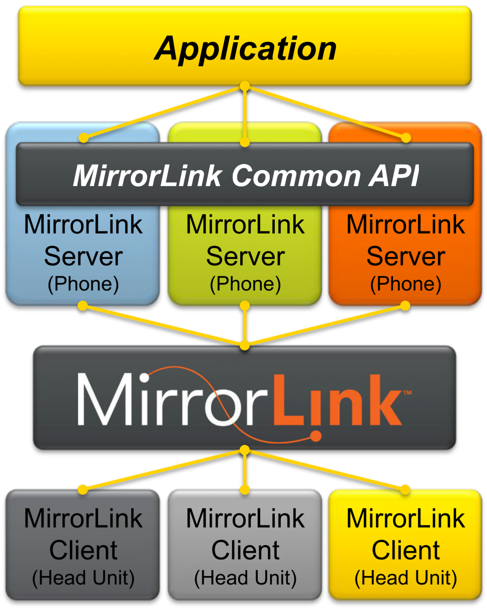

Application
MirrorLink Common API
MirrorLink Server (Phone)
MirrorLink Server (Phone)
MirrorLink Server (Phone)
MirrorLink
MirrorLink Client (Head Unit)
MirrorLink Client (Head Unit)
MirrorLink Client (Head Unit)

Figure 2.6: *MirrorLink*^{TM} Communication Architecture [41]

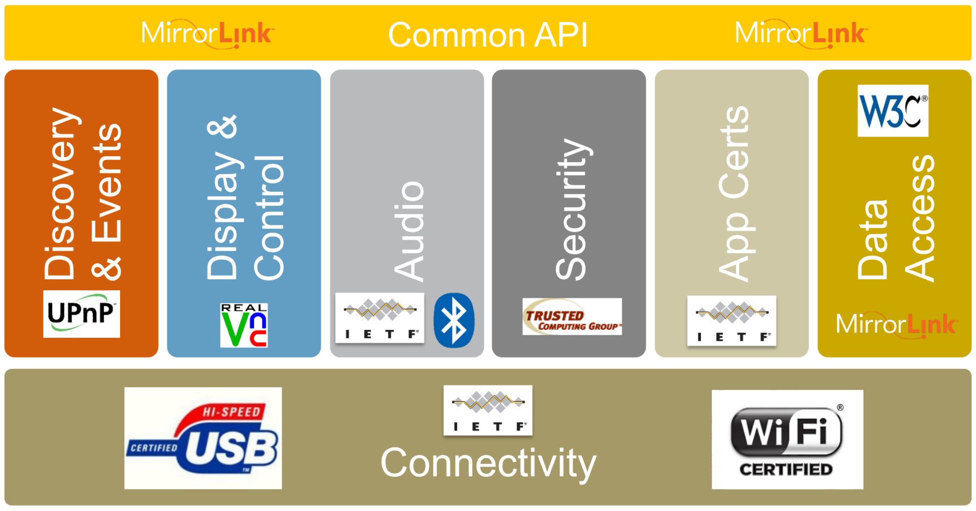


Figure 2.7: *MirrorLink^{TM}* Protocol Suite [42]

The connectivity layer enables a general data transfer between connected devices. This can be achieved by one of the two supported protocols: *Wi-Fi* or USB 2.0 (also used by *GreenLight*). Since each of these protocols use different addressing scheme and frame format, there needs to be an extra abstraction layer providing a common packet format and addressing schema. This is achieved by using the IPv4 protocol running on the top of the connectivity layer [43].

The data services layer offers the core *MirrorLink^{TM}* services. A high level device discovery, addressing and control event notifications (e.g. play, pause, volume) are handled by the *Universal Plug and Play* (UPnP) protocol. The message format used by UPnP is XML. Remote access to the application user interface is managed by *RealVNC* service using the *Remote Frame Buffer protocol*. Audio can be forwarded using the *Real-time Transport Protocol* (RTP) with support of the DVD quality, i.e. 48 kHz sampling rate, 16 bit depth, stereo. Certificate management and security services are also provided on this layer. These are based on asymmetric cryptography (data encryption and digital signatures) using the PKI certificates as described in the previous section.

At the server side, *Common API* is the top layer of the *MirrorLink^{TM}* protocol suite. It is a standardized interface that enables applications to access information related to the current *MirrorLink^{TM}* session. Implementation of this API is platform-specific, currently available only for *Android* devices. Some of the information accessible through *Common API* are the following [44][45]:

- ***MirrorLink^{TM}* Device Info:** This includes, for example, information about the client device manufacturer and model number which can be used to adjust the application settings with e.g. a predefined profile for the given device.

- **Certification Information:** This can be used to check whether the application certificate is valid and successfully verified.
- **Display Information:** This provides information about the client device display such as its size, resolution and scaling support.

Chapter **3**

Security Threats to C2I-based GLOSA Systems

3.1 Motivation to the Threat Analysis

Nowadays, security is an important aspect of any software or hardware system and it should be considered during entire design and implementation phases. It is especially crucial for ITS systems that use C2X communication, including C2I-based GLOSA implementations. There are several reasons for this:

- GLOSA influences an overall traffic flow in a specific area including one or multiple road intersections. A successful attack may have critical consequences including major accidents and traffic collapses.
- C2I-based GLOSA involves a wireless communication between cars and roadside infrastructure. The communication medium (air) is implicitly shared, and hence an attacker does not need to have a physical access to the network infrastructure to carry out a successful attack.
- C2I systems are based on an ad-hoc network design where centralized authentication and security services might be limited or completely unavailable. It may be easier for an attacker to legitimately join such network and perform malicious actions.

This chapter presents an analysis of security threats to general C2I-based GLOSA systems and more specifically to the *GreenLight* application. A security threat can be defined as a potential cause of an incident that may breach security and cause harm [46]. A concrete example of a threat is leakage of a credit card information to unauthorized parties when paying at an online shop. Identification and analysis of relevant threats is an important precondition for security design of any hardware or software system. Without the analysis it

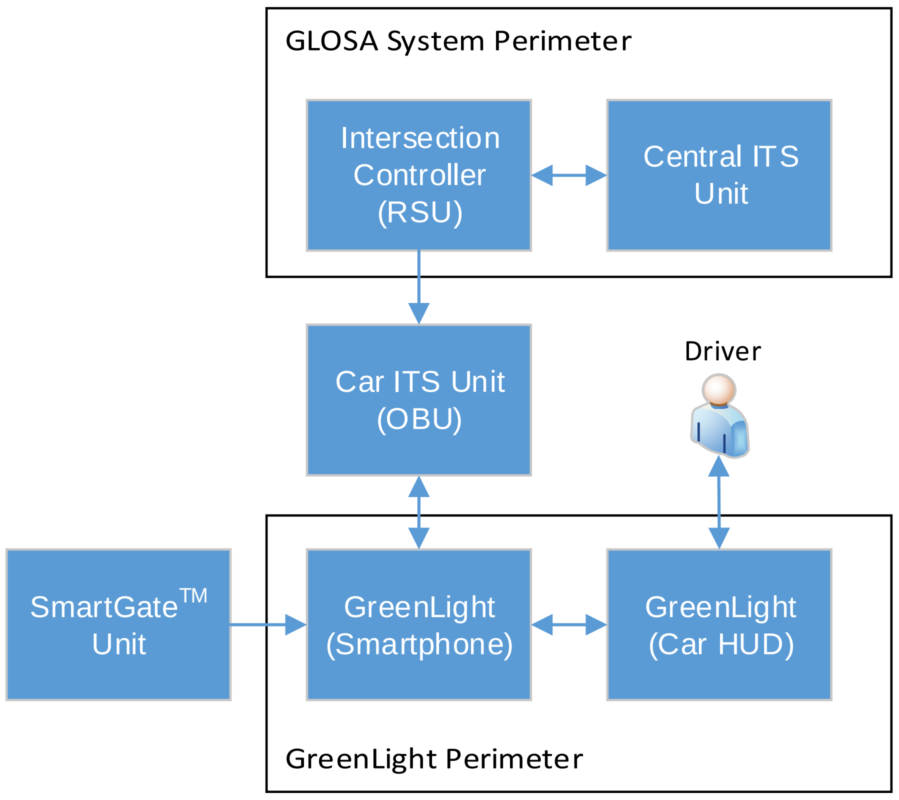


Figure 3.1: GLOSA Security Perimeters

would not even be possible to determine whether the system is secure or not since there are no identified threats and possible dangers.

3.2 Analyzed Security Perimeters

Figure 3.1 shows a C2I-based GLOSA system decomposed into its main components. In this case, it includes the *GreenLight* application as a car-side user-facing part of the system. The connectors represent the used communication interfaces and the arrows determine the expected communication flows. For a reference, a slightly different diagram illustrating the same architecture was presented in the previous chapter in figure 2.2.

Figure 3.1 identifies two main security perimeters that are included in this threat analysis. The *GLOSA System Perimeter* represents a general GLOSA system using a C2I wireless communication between RSU and OBU units. Although the OBU unit itself can also be considered as part of this perimeter, the main focus is given to the road-side part of the system and especially to the wireless communication channel. The *GreenLight Perimeter* includes the *GreenLight* application consisting of the smartphone and the HUD part connected with the *MirrorLink*^{TM} protocol. It also covers interfaces to the driver (GUI), *SmartGate*^{TM} and the OBU unit.

3.3 Threat Classification and Evaluation Method

The threats presented in the following sections are classified according to the *STRIDE* threat modeling method developed and promoted by the *Microsoft Corporation* [47]. *STRIDE* is a mnemonic representing the initial letters of the following six threat categories:

- **Spoofing identity:** Attacker pretends to be someone else.
- **Tampering with data:** Data intended for other parties is maliciously modified.
- **Repudiation:** Action is repudiated by its performer and nobody can prove otherwise.
- **Information disclosure:** Confidential information leaks to unauthorized individuals.
- **Denial of service:** Service is denied to legit users due to an attack or system fault.
- **Elevation of privilege:** Attacker gains (and abuses) privileges which is not entitled to.

To evaluate an overall security risk of each identified threat, another method called *DREAD* is used. This was also created by *Microsoft* and was proposed to be used together with *STRIDE* [48]. The following five ratings are used by *DREAD*:

- **Damage:** Damage potential of a successful attack.
- **Reproducibility:** How difficult it is to reproduce (simulate or repeat) a successfully performed attack.
- **Exploitability:** How difficult it is to create an exploit and perform a successful attack.
- **Affected Users:** Amount or ratio of users affected by a successful attack.
- **Discoverability:** How difficult it is to discover a vulnerability related to this threat. The category is related to an actual system implementation, and hence it is not used in this generally-oriented threat analysis.

The *DREAD* method rates each category (D, R, E, A, D) with a value of 0-10 where a higher number means a higher risk, e.g. higher damage or more affected users. An overal risk score is then determined as an average value of all the ratings. Therefore, this analysis calculates the *DREAD* score with the following expression. Note that the *Discoverability* rating is skipped as mentioned above:

$$Risk_{DREAD} = \frac{D + R + E + A}{4} \tag{3.1}$$

3.4 Identified Threats in the Relevant Security Perimeters

3.4.1 GLOSA System Perimeter

The identified threats in the *GLOSA System Perimeter* are presented in tables 3.1–3.10. These are mostly general threats common to any GLOSA system based on C2I communication. Therefore, the *DREAD* ratings are very approximate and intended to provide only a rough idea about severity of the identified threats.

The threats can be organized in a hierarchic structure as shown in figure 3.2. A parent threat can be viewed as a prerequisite for its child threats. This relation is represented by an arrow from the parent to the child.

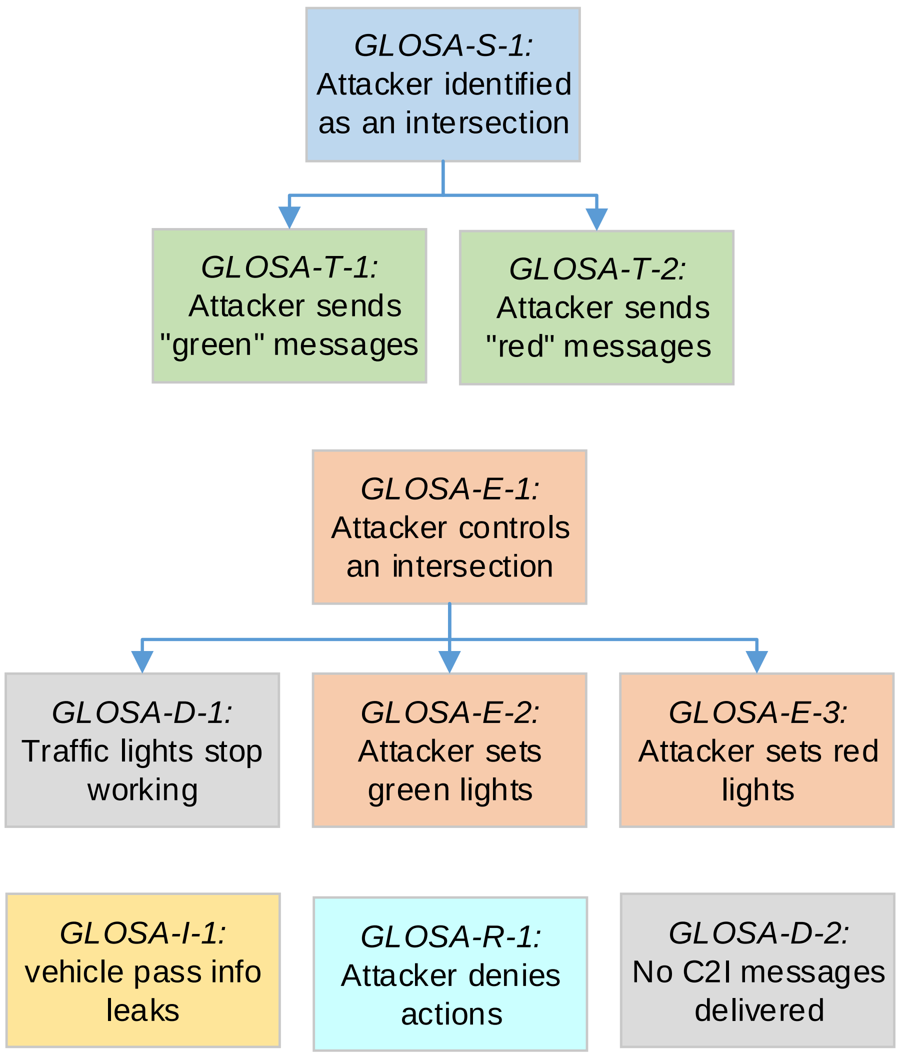

GLOSA-S-1: Attacker identified as an intersection
GLOSA-T-1: Attacker sends "green" messages
GLOSA-T-2: Attacker sends "red" messages
GLOSA-E-1: Attacker controls an intersection
GLOSA-D-1: Traffic lights stop working
GLOSA-E-2: Attacker sets green lights
GLOSA-E-3: Attacker sets red lights
GLOSA-I-1: vehicle pass info leaks
GLOSA-R-1: Attacker denies actions
GLOSA-D-2: No C2I messages delivered

Figure 3.2: Threat Structure – GLOSA System Perimeter

ID	GLOSA-S-1
Category	Spoofing Identity
Threat	Attacker identified as an intersection.
Description	Attacker is able to transmit spoofed C2I messages with a valid identity of a specific intersection or entire group of intersections. This is a general threat that may lead to many subsequent threats (attacks) described separately.
Attacker Motivation	Described separately for subsequent threats.
DREAD Ratings	
Damage	**6** – Seemingly valid (but spoofed) C2I messages advertising arbitrary traffic light states can be delivered to users (cars). This may cause further harm as described for threats *GLOSA-T-1* and *GLOSA-T-2*.
Reproducibility	**6** – Requires advanced technical knowledge to spoof C2I messages but no specific user privileges (e.g. system administration rights) are needed to reproduce the attack.
Exploitability	**4** – Hard to find an exploit if the communication is handled properly, i.e. encrypted and digitally signed messages (see 1.2.6). On the other hand, no physical access to the intersection controller is required, the attacker can e.g. sit in a car near the intersection.
Affected Users	**5** – Theoretically, all the GLOSA users approaching the intersection might be affected. It can be, however, expected that some users still receive legit C2I messages. Exact amount of affected users depends on ability of the attacker.
DREAD Score	**5.3**

Table 3.1: Threat Details – GLOSA-S-1

ID	GLOSA-T-1
Category	Tampering with Data
Threat	Attacker sends “green light” messages for all/specific lanes.
Description	Attacker transmits spoofed messages advertising a green light state for all or specific intersection approach lanes even if the actual light state is red. This becomes more dangerous when the GLOSA system can directly influence a car control (e.g. control of autonomous cars). Such car may not stop although the traffic light is red.
Attacker Motivation	• Attacker wants to cause a general traffic accident (e.g. terrorism, sabotage). • Attacker wants a specific vehicle to crash (e.g. police car, business partner).
DREAD Ratings	
Damage	**7** – In an extreme case, a major traffic accident can happen. However, this is unlikely unless there are autonomous cars relying on the information received from the GLOSA system.
Reproducibility	**6** – Same as *GLOSA-S-1*
Exploitability	**4** – Same as *GLOSA-S-1*
Affected Users	**5** – Same as *GLOSA-S-1*
DREAD Score	**5.5**

Table 3.2: Threat Details – GLOSA-T-1

ID	GLOSA-T-2
Category	Tampering with Data
Threat	Attacker sends “red light” messages for all/specific lanes.
Description	Attacker transmits spoofed messages advertising a red light state for all or specific intersection approach lanes even if the actual light state is green. Analogously to threat *GLOSA-T-1*, a car may stop although the traffic light state is green.
Attacker Motivation	• Attacker wants to block all the traffic (e.g. terrorism, sabotage). • Attacker wants to block a specific car (e.g. police, emergency). • Attacker wants to block a specific direction to make another direction free (e.g. to have a free escape way).
DREAD Ratings	
Damage	**4** – In an extreme case, all the intersection approaches can be blocked by the stopped cars resulting in a traffic collapse. However, this is unlikely unless there are autonomous cars relying on information received from the GLOSA system. Furthermore, a blocked traffic is considered to be less severe in comparison with a major traffic accident.
Reproducibility	**6** – Same as *GLOSA-S-1*
Exploitability	**4** – Same as *GLOSA-S-1*
Affected Users	**5** – Same as *GLOSA-S-1*
DREAD Score	**4.8**

Table 3.3: Threat Details – GLOSA-T-2

ID	GLOSA-R-1
Category	Repudiation
Threat	Attacker denies their previous actions.
Description	Attacker successfully conceals that they performed certain malicious actions (represented by any other threat). It might not be possible to prove identity of the attacker which is needed for e.g. legal purposes.
Attacker Motivation	• Attacker does not want to be identified and accused. • Attacker wants someone else to be accused.
DREAD Ratings	
Damage	**7** – If the attacker cannot be identified, they can easily continue with malicious activities and cause further problems. A potential damage is hence considered to be quite high.
Reproducibility	**N/A** – Cannot be determined for this general threat.
Exploitability	**5** – Since a physical access to the infrastructure is not required for most of the attacks, the attacker can easily stay anonymous. This can be partially mitigated with a proper security design of the system, e.g. by using ETSI security services as described in 1.2.6 and a good security logging.
Affected Users	**N/A** – Only the attacker is directly affected by this threat but potential future attacks can affect any number of users. Therefore, the rating is skipped for this threat.
DREAD Score	**6.0**

Table 3.4: Threat Details – GLOSA-R-1

ID	GLOSA-I-1
Category	Information Disclosure
Threat	Information about vehicle movement is leaked.
Description	A GLOSA system might be designed in such way that all the vehicles passing an intersection are monitored and logged, e.g. for specific safety purposes. The collected data might be stored at each intersection separately or more likely at a central ITS unit if available. If an attacker is able to access such information, they may be able to monitor movements of all the vehicles included in the database.
Attacker Motivation	• Attacker wants to monitor movements of certain vehicles (e.g. police cars to keep track of their common routes).
DREAD Ratings	
Damage	**3** – Leaking information about car movement is considered to be less dangerous than possibility of a traffic accident or other serious threat consequences presented in this section.
Reproducibility	**2** – It can be expected that collected data is stored at a well-secured central ITS unit (server). Therefore, high privileges and access rights are required to reproduce a successful attack.
Exploitability	**1** – It is expected to be much harder to compromise a well-secured central server in comparison with attacks to the wireless communication channel between RSU and OBU units.
Affected Users	**10** – Information about all the cars passing through a group of intersections (connected to a common central server) can possibly be leaked.
DREAD Score	**4.0**

Table 3.5: Threat Details – GLOSA-I-1

ID	GLOSA-D-1
Category	Denial of Service
Threat	Traffic lights switched to flashing yellow for all lanes.
Description	Attacker shuts down an intersection signaling and the intersection goes to the out-of-order state (flashing yellow). Traffic flow is disrupted or it may even collapse (e.g. during peak periods).
Attacker Motivation	• Attacker wants to cause a traffic accident (e.g. terrorism, sabotage). • Attacker wants to block or disrupt the traffic flow.
DREAD Ratings	
Damage	**7** – Traffic flow is disrupted and can possibly collapse, especially during peak periods. Moreover, probability of a traffic accident is increased.
Reproducibility	**2** – Same as *GLOSA-E-1*
Exploitability	**1** – Same as *GLOSA-E-1*
Affected Users	**8** – Same as *GLOSA-E-1*
DREAD Score	**4.5**

Table 3.6: Threat Details – GLOSA-D-1

ID	GLOSA-D-2
Category	Denial of Service
Threat	No C2I messages from an intersection are delivered.
Description	Attacker blocks all the C2I messages coming from an intersection. The intersection then appears to be a normal non-ITS infrastructure unit. This may become a serious issue when there are cars (e.g. autonomous) that assume an ITS-equipped intersection and rely on the received information.
Attacker Motivation	• Attacker wants to cause a traffic accident (e.g. terrorism, sabotage). • Attacker wants to block or disrupt the traffic.
DREAD Ratings	
Damage	**3** – Traffic flow might be disrupted, but most of the cars should be able to easily cope with a non-ITS intersection (which they would consider it to be).
Reproducibility	**9** – No specific privileges or access rights are required to be able to e.g. run a *Wi-Fi* jamming device near the intersection.
Exploitability	**8** – Since the C2I communication is based on a variant of the *Wi-Fi* protocol (see 1.2.1), the threat is easily exploitable e.g. by running a *Wi-Fi* jamming device near the intersection.
Affected Users	**5** – All the GLOSA users approaching the intersection can possibly be affected. However, the exact amount of affected users depends on efficiency and magnitude of the attack.
DREAD Score	**6.3**

Table 3.7: Threat Details – GLOSA-D-2

ID	GLOSA-E-1
Category	Elevation of Privilege
Threat	Attacker controls an intersection.
Description	Attacker gains privileges to control an intersection, change its signaling plan or set invalid signals (e.g. green/red for all directions). This is a general threat that may lead to many subsequent threats (attacks) described separately.
Attacker Motivation	Described separately for subsequent threats.
DREAD Ratings	
Damage	**9** – Ability to control an intersection is much more serious than just spoofing C2I messages. This threat is not only a GLOSA-related but also a general intersection-related threat. This may yield in subsequent threats such as *GLOSA-D-1*, *GLOSA-E-2* and *GLOSA-E-3*.
Reproducibility	**2** – High privileges and access rights are generally required to control an intersection.
Exploitability	**1** – It is expected to be much harder to completely take over an intersection control in comparison with attacks to the wireless communication channel between RSU and OBU units. Furthermore, a physical access to the intersection controller might be required.
Affected Users	**8** – All the cars passing through the intersection are affected, not only the GLOSA users.
DREAD Score	**5.0**

Table 3.8: Threat Details – GLOSA-E-1

ID	GLOSA-E-2
Category	Elevation of Privilege
Threat	Attacker sets "green light" for all/specific lanes
Description	Attacker is able to control an intersection and to set a green light state for all or specific intersection approach lanes. Similarly to threat *GLOSA-T-1*, a car may not stop even if the original state of the traffic light was red.
Attacker Motivation	• Attacker wants to cause a general traffic accident (e.g. terrorism, sabotage). • Attacker wants a specific vehicle to crash (e.g. police car, business partner).
DREAD Ratings	
Damage	**10** – A traffic accident can happen with a high probability. This is considered to be the most dangerous state of an intersection.
Reproducibility	**2** – Same as *GLOSA-E-1*
Exploitability	**1** – Same as *GLOSA-E-1*
Affected Users	**8** – Same as *GLOSA-E-1*
DREAD Score	**5.3**

Table 3.9: Threat Details – GLOSA-E-2

ID	GLOSA-E-3
Category	Elevation of Privilege
Threat	Attacker sets “red light” for all/specific lanes
Description	Attacker is able to control an intersection and to set a red light state for all or specific intersection approach lanes. Similarly to threat *GLOSA-T-2*, a car may stop even if the original state of the traffic light state was green.
Attacker Motivation	• Attacker wants to block all the traffic (e.g. terrorism, sabotage). • Attacker wants to block a specific car (e.g. police, emergency). • Attacker wants to block a specific directions to make another directions free (e.g. to have a free escape way).
DREAD Ratings	
Damage	**8** – All the intersection approaches are likely to be blocked by the stopped cars causing the traffic to collapse.
Reproducibility	**2** – Same as *GLOSA-E-1*
Exploitability	**1** – Same as *GLOSA-E-1*
Affected Users	**8** – Same as *GLOSA-E-1*
DREAD Score	**4.8**

Table 3.10: Threat Details – GLOSA-E-3

3.4.2 *GreenLight* Perimeter

The identified threats in the *GreenLight Perimeter* are presented in tables 3.11–3.16. Unlike the general GLOSA threats, these are predominately focused on the present situation and do not consider for example autonomous cars controlled by GLOSA systems.

Similarly to the *Glosa System Perimeter* threats, the *GreenLight Perimeter* threats can also be organized in a hierarchic structure as shown in figure 3.3.

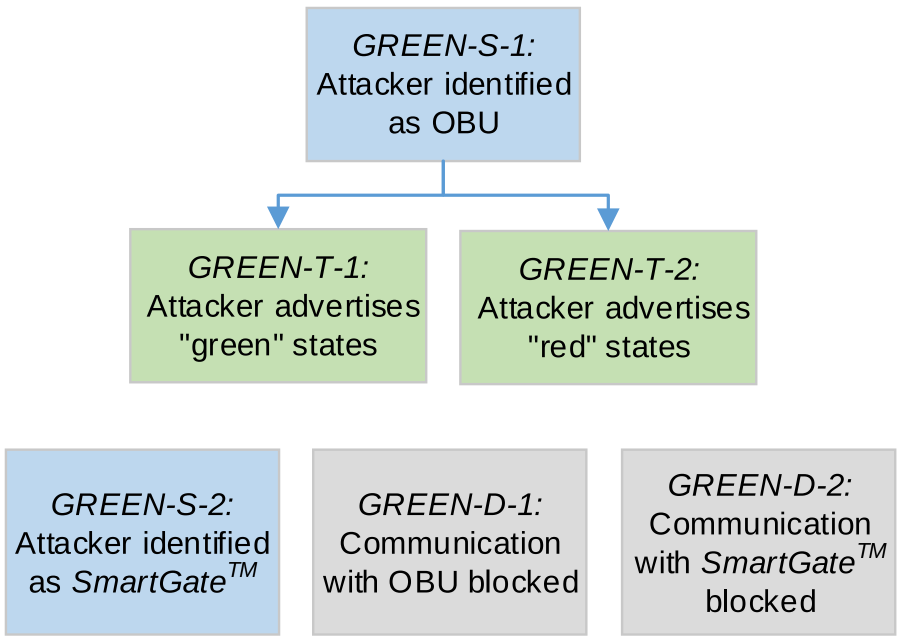

GREEN-S-1: Attacker identified as OBU
GREEN-T-1: Attacker advertises "green" states
GREEN-T-2: Attacker advertises "red" states
GREEN-S-2: Attacker identified as SmartGate™
GREEN-D-1: Communication with OBU blocked
GREEN-D-2: Communication with SmartGate™ blocked

Figure 3.3: *GreenLight* Perimeter

ID	GREEN-S-1
Category	Spoofing Identity
Threat	Attacker identified as an on-board ITS unit.
Description	Attacker is able to deliver spoofed messages with a valid identity of a car on-board ITS unit to the *GreenLight* application. This is a general threat that may lead to several subsequent threats (attacks) described separately.
Attacker Motivation	Described separately for subsequent threats.
DREAD Ratings	
Damage	**6** – Seemingly valid (but spoofed) OBU messages advertising arbitrary traffic light states can be delivered to the *GreenLight* application. This may cause further harm as described for threats *GREEN-T-1* and *GREEN-T-2*.
Reproducibility	**4** – Requires advanced technical knowledge to spoof messages from OBU. No specific user privileges (e.g. system administration rights) are needed to reproduce the attack. However, a physical access to the car might be necessary.
Exploitability	**5** – Connection between the *GreenLight* application and the on-board unit uses the *Bluetooth* protocol. A man-in-the-middle attack is possible under certain circumstances as presented in [49]. Since the car is moving and *Bluetooth* has a limited communication range, a reasonable attack needs to be performed from a distance less than 10 meters, ideally right from the attacked car.
Affected Users	**1** – Only one car is directly affected.
DREAD Score	**4.0**

Table 3.11: Threat Details – GREEN-S-1

ID	GREEN-S-2
Category	Spoofing Identity
Threat	Attacker identified as a *SmartGate*^{TM} unit
Description	Attacker is able to deliver spoofed messages with a valid identity of a *SmartGate*^{TM} unit to the *GreenLight* application. This is a general threat that may lead to several subsequent threats (attacks). Spoofing *SmartGate*^{TM} messages is however evaluated as a low-risk threat (see below) and thus the subsequent attacks are not individually described in this analysis.
Attacker Motivation	• Attacker wants to confuse the driver with an incorrect speed recommendation. The confused driver might make a driving mistake with an increased probability.
DREAD Ratings	
Damage	**3** – The speed calculation can be affected by the spoofed messages resulting in an incorrect speed recommendation. However, the displayed signal phase and timing information remain correct.
Reproducibility	**4** – Requires advanced technical knowledge to spoof *SmartGate*^{TM} messages, but no specific user privileges (e.g. system administration rights) are needed to reproduce the attack. However, a physical access to the car might be necessary.
Exploitability	**6** – Connection between the *GreenLight* application and the *SmartGate*^{TM} unit uses the *Wi-Fi Direct* protocol. There is no additional security layer involved and it is relatively easy to perform a successful attack. Security of *SmartGate*^{TM} is briefly evaluated in [50]. Similarly to threat *GREEN-S-1*, a reasonable attack needs to be performed from a short distance of about 20 meters as mentioned in [50].
Affected Users	**1** – Only one car is directly affected.
DREAD Score	**3.5**

Table 3.12: Threat Details – GREEN-S-2

ID	GREEN-T-1
Category	Tampering with Data
Threat	Attacker advertises green light phases to the *GreenLight* application
Description	Attacker delivers spoofed messages from OBU to the *GreenLight* application advertising a green light state regardless the actual light state at an upcoming intersection.
Attacker Motivation	• Attacker wants to confuse the driver by displaying an incorrect signal phase, timing and recommended speed. The confused driver might make a driving mistake with an increased probability. • Attacker wants the affected vehicle to crash (extreme case, low probability).
DREAD Ratings	
Damage	**7** – Incorrect signal phase, timing and recommended speed may be displayed to the driver. Advertising a green light phase is considered to be more dangerous than a red phase.
Reproducibility	**4** – Same as *GREEN-S-1*
Exploitability	**5** – Same as *GREEN-S-1*
Affected Users	**1** – Same as *GREEN-S-1*
DREAD Score	**4.3**

Table 3.13: Threat Details – GREEN-T-1

ID	GREEN-T-2
Category	Tampering with Data
Threat	Attacker advertises red light phases to the *GreenLight* application
Description	Attacker delivers spoofed messages from OBU to the *GreenLight* application advertising a red light state regardless the actual light state at an upcoming intersection.
Attacker Motivation	• Attacker wants to confuse the driver by displaying an incorrect signal phase, timing and recommended speed. The confused driver might make a driving mistake with an increased probability. • Attacker wants the affected vehicle to stop at the intersection and possibly to disrupt the traffic (extreme case, low probability).
DREAD Ratings	
Damage	**5** – Incorrect signal phase, timing and recommended speed may be displayed to the driver. Advertising a red light phase is considered to be less dangerous than a green phase.
Reproducibility	**4** – Same as *GREEN-S-1*
Exploitability	**5** – Same as *GREEN-S-1*
Affected Users	**1** – Same as *GREEN-S-1*
DREAD Score	**3.8**

Table 3.14: Threat Details – GREEN-T-2

ID	GREEN-D-1
Category	Denial of Service
Threat	No messages from an on-board ITS unit are delivered to the *GreenLight* application
Description	Attacker blocks all the communication between an on-board ITS unit and the *GreenLight* application.
Attacker Motivation	• Attacker wants to confuse the driver by pretending that the intersection does not broadcast any C2I messages. The confused driver might make a driving mistake with an increased probability.
DREAD Ratings	
Damage	**3** – It is not possible to calculate a recommended speed since the needed input data is not available to the *GreenLight* application. The driver perceives the intersection as a regular non-GLOSA infrastructure unit.
Reproducibility	**6** – No specific privileges or access rights are required to jam a *Bluetooth* network. However, a physical access to the car might be necessary.
Exploitability	**8** – Both *Wi-Fi* and *Bluetooth* use the same frequency band (2.4 GHz) and it is quite easy to jam either of these networks. For the best results, a jamming device should be installed on the car itself.
Affected Users	**3** – It is possible to affect more than one vehicle with a single jamming device (depending on its power).
DREAD Score	**5.0**

Table 3.15: Threat Details – GREEN-D-1

ID	GREEN-D-2
Category	Denial of Service
Threat	No messages from a *SmartGate*^{TM} unit are delivered to the *GreenLight* application
Description	Attacker blocks all the communication between a *SmartGate*^{TM} unit and the *GreenLight* application.
Attacker Motivation	• Attacker wants to confuse the driver with an incorrect speed recommendation. The confused driver might make a driving mistake with an increased probability.
DREAD Ratings	
Damage	**2** – The speed calculation can be disrupted resulting in an incorrect speed recommendation. However, the displayed signal phase and timing information remain correct.
Reproducibility	**6** – No specific privileges or access rights are required to jam a *Wi-Fi* network. However, a physical access to the car might be necessary.
Exploitability	**8** – Both *Wi-Fi* and *Bluetooth* use the same frequency band (2.4 GHz) and it is quite easy to jam either of these networks. For the best results, a jamming device should be installed on the car itself.
Affected Users	**3** – It is possible to affect more than one vehicles with a single jamming device (depending on its power).
DREAD Score	**4.8**

Table 3.16: Threat Details – GREEN-D-2

3.4.3 Summary of the Identified Threats

All the threats presented in sections 3.4.1 and 3.4.2 are summarized in table 3.17. Column chart 3.4 shows the same threats in order of their *DREAD* scores.

Threat ID	**Threat Description**	**DREAD Score**
GLOSA-S-1	Attacker identified as an intersection	5.3
GLOSA-T-1	Attacker sends “green” messages	5.5
GLOSA-T-2	Attacker sends “red” messages	4.8
GLOSA-R-1	Attacker denies actions	6.0
GLOSA-I-1	Vehicle pass info leaks	4.0
GLOSA-D-1	Traffic lights stop working	4.5
GLOSA-D-2	No C2I messages delivered	6.3
GLOSA-E-1	Attacker controls an intersection	5.0
GLOSA-E-2	Attacker sets green lights	5.3
GLOSA-E-3	Attacker sets red lights	4.8
GREEN-S-1	Attacker identified as OBU	4.0
GREEN-S-2	Attacker identified as *SmartGate*^{TM}	3.5
GREEN-T-1	Attacker advertises “green” state	4.3
GREEN-T-2	Attacker advertises “red” state	3.8
GREEN-D-1	Communication with OBU blocked	5.0
GREEN-D-2	Communication with *SmartGate*^{TM} blocked	4.8

Table 3.17: Summary of GLOSA and *GreenLight* Threats

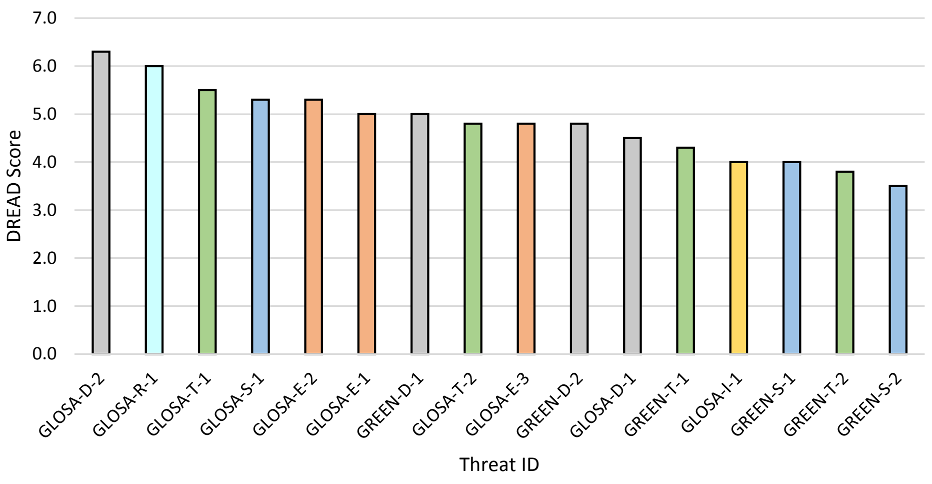


Figure 3.4: *DREAD* Scores of GLOSA and *GreenLight* Threats

3.5 Proposed Security Measures for the *GreenLight* Application

This section proposes suitable security measures for the *GreenLight* application. These are based on the threat analysis and should be used as a security guideline for future development of the application.

3.5.1 Secure Connection to the On-board Unit

Connection between the application and the on-board unit should be always secured. This requirement can be further divided into two sub-parts:

- **Secure hardware protocols:** Wireless protocols are generally more vulnerable to network attacks in comparison with the wired networks. The *GreenLight* application currently uses *Bluetooth* to connect to an on-board unit. Unfortunately, *Bluetooth* is potentially vulnerable to man-in-the-middle (MiM) attacks as presented in [49]. It is advised to consider other protocols that are commonly provided by on-board ITS units:
  - **IEEE 802.11 (*Wi-Fi*):** Generally provides a level of security comparable to *Bluetooth*. Security services are, however, standardized as IEEE 802.11 amendments such as IEEE 802.11i-2004 known as *Wi-Fi Protected Access II* (WPA2) [51]. Although WPA2 would effectively mitigate MiM attacks, the protocol would still remain vulnerable to denial-of-service attacks caused by e.g. a signal jamming. This vulnerability is the same for *Bluetooth*.
  - **USB:** Implicitly more secure since the connection is wired. However, it might be infeasible to connect the smartphone directly to the on-board unit using a USB cable (depending on the on-board unit location and accessibility).
  - **CAN:** Seems reasonable to connect the on-board unit directly to the car network and access it using a car infotainment system. This would, however, require a further effort from car manufacturers to integrate and support on-board units in their car networks.
- **High-level software security:** Even if an insecure hardware protocol is used, the required security services can be provided at higher network layers. However, this depends primarily on the protocols supported by the on-board unit. Therefore, it is advised to prefer on-board units that provide secure connection methods.

3.5.2 Limited Dependency on ***SmartGate^{TM}***

SmartGate^{TM} unit uses the *Wi-Fi Direct* technology which is vulnerable to network attacks as presented in [50]. Therefore, the *GreenLight* application should not be fully dependent on data received from *SmartGate^{TM}*. This means the application should remain operating and be able to handle situations when no *SmartGate^{TM}* data is available or the obtained data is invalid. Furthermore, *SmartGate^{TM}* data should have a limited impact on the recommended speed calculation. This would prevent an attacker from completely controlling the speed calculation.

3.5.3 Input Data From Multiple Sources

Certain input data might be available from multiple independent sources. An example is speed of the car that can be obtained from both *SmartGate^{TM}* unit and phone GPS sensor. It is advised to always obtain data from all the available sources. Benefits of this approach are evident:

- **Increased security:** If the obtained values differ significantly, it is possible that one or more of the data sources or communication channels have been attacked and some of the values are spoofed. In that case, a warning should be displayed to the driver or any other predefined action can be taken. It is, however, possible that the input data could not be obtained because of a technology-related fault such as a lost GPS signal.

- **Increased accuracy:** All the obtained values can be averaged to increase a resulting accuracy. Furthermore, a different weight can be assigned to each input channel to favor accurate/reliable/secure data sources.

3.5.4 Detection of Unexpected Input Data Changes

Most of the input values are obtained periodically, e.g. once per second when a recommended speed calculation is triggered. Each obtained value should naturally correspond to the previous values of the same type. This means it is, for example, extremely unlikely that the car accelerates from 20 to 90 km/h in one second. Or that a traffic light in the middle of a red phase would switch directly to green skipping a yellow phase.

For an increased security, implementation of a simple intrusion detection system can be considered. It should evaluate all the received input values based on a predefined set of rules and change thresholds. If a value is evaluated as “suspicious”, a predefined action can be taken (e.g. warning displayed).

Chapter **4**

Proper Light Signal Determination Analysis

4.1 Motivation for Determining the Proper Signal

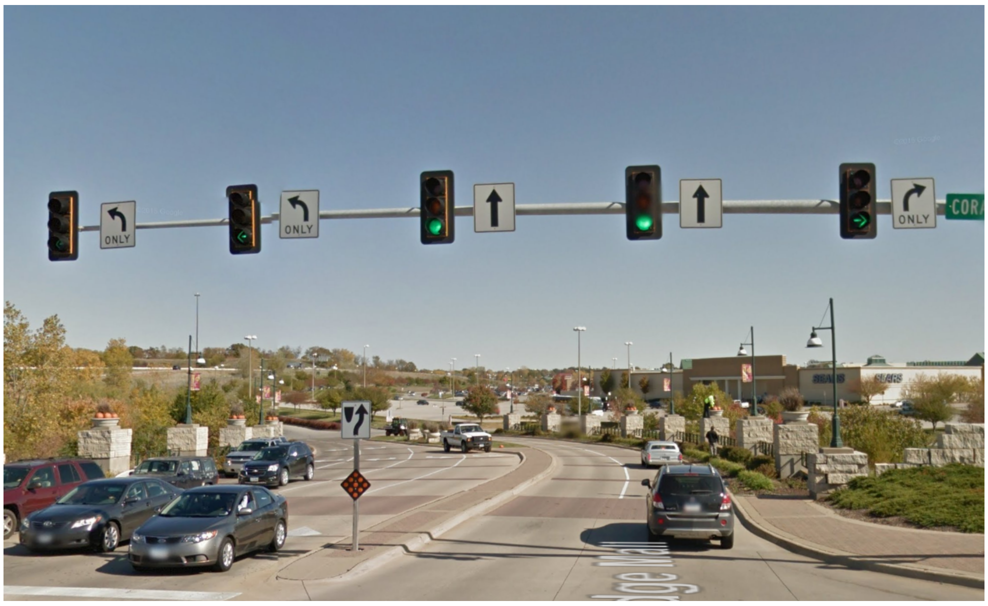


Figure 4.1: Intersection in Coralville in Iowa, USA [52]

Intersection topologies may differ as well as their lane designs. There is often more than one light signal for a given intersection approach. An example of such intersection is shown in photo 4.1 taken in Coralville near Iowa City, USA. Imagine this is an ITS-equipped intersection and a car with the *GreenLight* application is approaching. The driver might want to either go straight or take a left or right turn. In either case a different traffic light is relevant for them and they does not care about the remaining ones. Then the question is: which traffic light should *GreenLight* consider for recommending an optimal speed? There are a few possible answers:

- **Straight Direction Only:** This was an initial approach used in *GreenLight*. It can be assumed that a relative frequency of going straight is higher than frequency of left or right turns. Unfortunately, in case the driver wants to turn, the displayed information (remaining time and recommended speed) is useless. In the worst scenario it may even confuse a driver who forgets to ignore the application when turning.

- **All the Signals:** The relevant information is always displayed but the driver needs to pay an extra attention to distinguish the proper signal according to their driving direction. This could be considered as an unacceptable driver distraction and may lead to a problem, for example, with obtaining a *MirrorLink*^{TM} certification (see 2.2.1).

- **The Proper Signal:** An ideal solution. However, it is not a trivial task to automatically choose the proper light signal. The application needs to determine in which direction the driver intends to go. This can be achieved using several different methods. These are analyzed further in this chapter.

The current version of *GreenLight* calculates a remaining time and recommended speed only for the straight direction signal and this is clearly not sufficient for a production version of the application. At this point, a theoretical analysis of possible methods for determining the intended driving direction (and consequently the proper traffic light) needs to be done. An actual implementation of this feature is planned for Q4 2016 and it should be based on the analysis provided in this thesis.

4.2 Overview of Considered Selection Methods

As mentioned in the previous section, the goal is to determine a particular direction in which the driver intends to go through an intersection, i.e. straight, left, right. This section outlines potentially usable methods and sources of input data that could be used to cope with the task. Most of the presented methods consider only intersections with up to three different driving directions possible (straight, left, right). Since this covers most of the existing intersections, the limitation has been considered acceptable.

4.2.1 Car Navigation

When a car navigation system is active, information about the planned route may be used as a hint for determining the driving direction. There is a high chance that the driver will follow the suggested route. Many drivers however do not use navigation on daily basis, especially for short and repetitive routes such as to and from their work.

Effort to implement: Medium. In theory, it is only needed to allow the *GreenLight* application to access data about the planned route. In fact, this would require a lot of effort and support from the car manufacturer or developer of the navigation system used. This might be a bit easier if the navigation is running right in the smartphone instead of a car built-in navigation system.

4.2.2 Turn Indicators

Turn indicator (blinker) is another quite reliable source of information. When a left/right indicator is active, there is a high probability for turning to the corresponding direction. However, in some cases the driver may decide not to use the indicators or forget to do so. It is also possible that the driver just wants to change lanes before the intersection and uses a turn indicator.

Effort to implement: Low for *Škoda Auto* cars. Information about the active turn indicator is already accessible through the *SmartGate*^{TM} unit (see 2.1.1).

4.2.3 GPS Data

Location data can be obtained easily with a phone GPS sensor. The idea is to use this information to determine the current driving lane and match it to a *Generic Lane* entity received in a MAP message. The MAP message includes all the possible connecting lanes for each approach lane. Therefore, this possibly determines or at least limits the number of possible driving directions. Usability of such approach depends heavily on an intersection lane topology though. Moreover, the driver may change lanes when approaching the intersection and thus a new calculation would be necessary.

According to the official GPS specification, a guaranteed free-space horizontal accuracy is 7.8 meters at 95% confidence interval [53]. There are several augmentation systems such as the *European Geostationary Navigation Overlay Service* (EGNOS) used in Europe and *Wide Area Augmentation System* (WAAS) used in North America providing accuracy of about 3 meters [54][55]. Unfortunately, these are usually not utilized by phone GPS sensors. The only common augmentation technology for smartphones is called *Assisted GPS* (A-GPS). Besides the GPS data itself, it considers location of cell phone towers and *Wi-Fi* access points to improve the accuracy. Even still, a real-world achievable accuracy of A-GPS phones is around 3-8 meters as shown in [56], [57], [58].

Since a typical lane width is 2-4 meters [59] it is not possible to reliably determine the current driving lane with a phone GPS sensor. On the other hand, it is expected that the accuracy of phone GPS sensors will keep improving with new models and the described method may become feasible soon. For now, it can be used as an additional factor with a low confidentiality.

Effort to implement: Low. GPS data is already used within the *GreenLight* application. However, it needs further effort to implement the location matching with MAP message data.

4.2.4 Road Marking Recognition

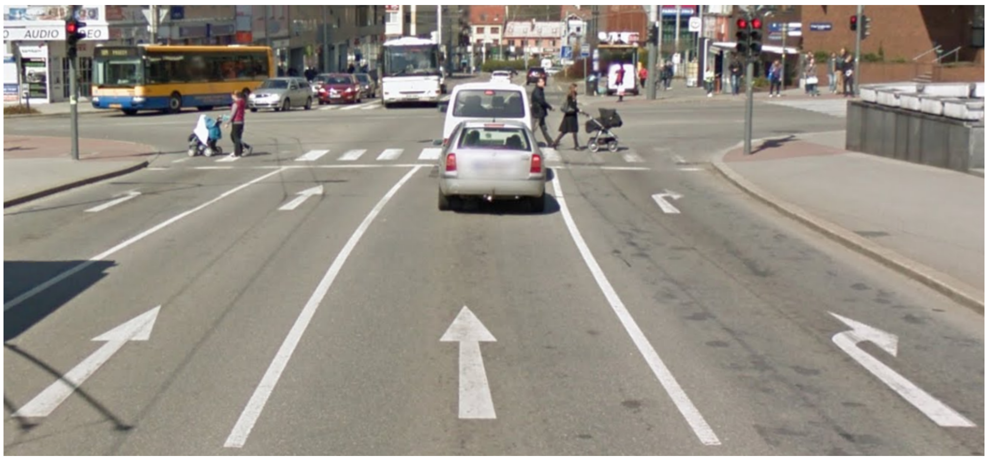


Figure 4.2: Horizontal Road Marking in Zlin, Czech Republic [52]

Recognizing horizontal lane marks (as illustrated in figure 4.2) gives the allowed driving directions from the current lane. This can be viewed as another method for determining the driving lane. However, this approach might be problematic in certain cases: there are no direction-related road marks at some intersections. And similarly to the previous method (GPS data), the driver may change the lane when approaching the intersection, i.e. a new calculation would be necessary.

Effort to implement: Very high. Camera-based road marking recognition is currently used only for specific applications such as lane departure warning systems. This is for example implemented as the *Lane Assist*^{TM} technology for selected *Škoda Auto* models [60]. Recognizing direction-related road marks and providing this data to external application (i.e. *GreenLight*) would require a lot of effort from car manufacturers including implementation of a completely new technology.

4.2.5 Route Memory

If the application knew the previous routes, it could statistically determine a driving direction with the highest probability. This could be especially beneficial for repetitive routes such as to and from work. For example, the driver turns right at a certain intersection every morning. Thus, it can be assumed that the next morning they will turn right again. This can be further enhanced with advanced methods such as machine learning or pattern recognition. Nevertheless, this approach is naturally not usable for unique and completely new routes.

Effort to implement: Medium. It is quite an easy task to log the taken routes using e.g. the phone GPS sensor. On the other hand, a big challenge would be an effective data management including storing, accessing and searching through a high amount of collected data. It should be sufficient to store a list of passing times and taken directions for each considered intersection. This can still be problematic for phones with a very limited memory though and an external storage such as cloud services would need to be considered too. The complexity of the direction determination itself is entirely dependent on the used decision method and can range from a simple statistical calculation to very complex machine learning algorithms.

4.2.6 Personal Schedule

If the application could access a driver's personal calendar, it may consider an upcoming planned meeting as a possible driving destination. Using this location, it can internally calculate the optimal route and use it as a decision factor similarly to the *Car Navigation* method (see 4.2.1). This method is perhaps usable only for a small set of drivers, especially those who often travel for business purposes.

Effort to implement: Medium. The phone calendar data can be accessed easily from the *GreenLight* application. Calculating the route to an upcoming meeting is not a trivial task and it would most likely need to be delegated to an external navigation software.

4.2.7 Summary of the Proposed Methods

All the selection methods presented in sections 4.2.1–4.2.6 are summarized in table 4.1. The table lists main advantages and disadvantages of each method. A quantitative comparison of the methods is then presented in the following section.

Selection Method	**Advantages**	**Disadvantages**
Car Navigation	High reliability	Usable only when car navigation is active
Turn Indicators	Easy to implement for *Škoda Auto* cars, fair reliability	Implementation for non-*Škoda* cars can be problematic
GPS Data	Good potential for the future, easy to implement	Low reliability with current GPS sensors
Road Marking	Good potential for the future, fair reliability	Almost impossible to implement unless road marking recognition is widely available
Route Memory	Fair reliability	Usable only for driver's frequent routes
Personal Schedule	Fair reliability	Usable only in very specific cases (driving to a calendar event)

Table 4.1: (Dis)Advantages of Proposed Selection Methods

4.3 Proposal of a Method to Determine the Proper Light Signal

Each method proposed in the previous section can be implemented separately and independently of the other ones. It is expected that only one or a very few methods will be implemented in the initial phase. More methods can be added later to improve the determination accuracy. Therefore, it does not really make sense to come up with an exact decision algorithm at this point.

This section proposes relative weights of the methods outlined in the previous section. Supposing each method selects one or more possible driving directions, the proposed weights can be used to compare these results and to determine the final decision. Since the weights are strictly relative, it is sufficient to count "points" for each possible direction. These points are granted by the selection methods where number of points is the weight of the corresponding method. This can be formalized with the following simple expression:

$$d_i = \sum_{j=1}^{n} (w_j \cdot m_j) \tag{4.1}$$

where d_i is the i-th driving direction (or consequently the i-th light signal), n is number of utilized selection methods, w_j is relative weight of the j-th method and m_j is either 1 or 0 depending on whether the j-th method selects the i-th direction or not. Naturally, direction with the highest amount of points is determined as the final decision.

4.3.1 Relative Comparison of the Proposed Methods

A relative comparison of the proposed methods is presented in table 4.2. This is just an initial estimation which should be later re-evaluated depending on the implemented methods and their exact abilities. There are three different columns in the table: *Weight* is a method relative weight as explained above. It represents a relative performance of the method (higher is better). *Effort* represents time and resources needed for a successful implementation of the method (lower is better). Finally, *W/E Ratio* is simply the weight divided by the effort and it stands for a relative benefit of implementing the given method (equivalently to price/performance ratio, higher is better). All the methods are also shown in column chart 4.3 in order of their *W/E Ratios*.

Selection Method	**Weight**	**Effort**	**W/E Ratio**
Car Navigation	6	4	1.5
Turn Indicators	4	1	4
GPS Data	2	2	1
Road Marking	3	10	0.3
Route Memory	3	5	0.6
Personal Schedule	2	5	0.4

Table 4.2: Relative Comparison of Proposed Selection Methods

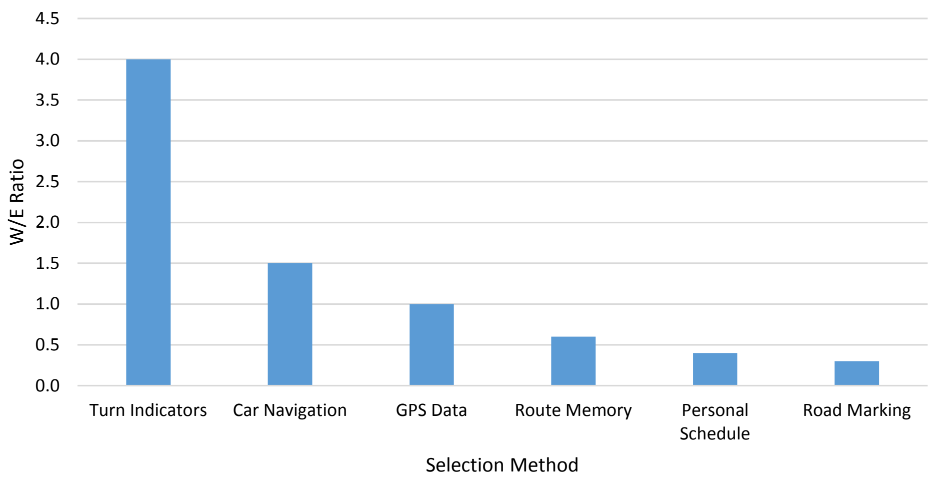


Figure 4.3: *W/E Ratios* of Proposed Selection Methods

In general, implementing the *Turn Indicator* method seems to be the best choice at this point in time. Although its performance is expected to be slightly worse than the *Car Navigation* method, it benefits especially from the lowest implementation effort. Implementation of both *Car Navigation* and *GPS Data* looks to be reasonable too, but it strongly depends on external factors such as accuracy of a phone GPS sensor and ability to access a car navigation data.

All the remaining methods (*Route Memory*, *Road Marking*, *Personal Schedule*) currently appear as not suitable for an implementation. However, this will definitely change in the future once the technology needed for a successful deployment becomes widely available. For example, it can be expected that a camera-based road marking recognition is going to be a standard car equipment in the near future. Then the *Road Marking* method will become significantly easier to implement. Therefore, it is not a good idea to deprecate this method now.

Chapter **5**

Enhanced Calculation of an Advised Speed for the *GreenLight* Application

5.1 Speed Calculation Procedure

The overall procedure of determining a recommended speed can be seen in figure 5.1 in a slightly simplified form. It reflects the present state of the *GreenLight* application, but there is no plan to dramatically change this high-level logic in the future. The process can be decomposed into two main parts labeled as *Inner Calculation* and *Outer Loop*.

The *Inner Calculation* performs the speed calculation itself. Its important property is that it does not know anything about the intersection signal timing, including unawareness of a current signal phase (current traffic light color). Its inputs are simply time, distance and possibly additional information such as current speed of the car. The speed is calculated to match exactly the input time, i.e. to reach the intersections exactly at the requested input time, not earlier or later. In certain cases, such speed cannot be calculated and an error value is returned. This however depends on exact design of the *Inner Calculation* algorithm as described further in this chapter.

The *Outer Loop* is responsible for determining a recommended speed using the *Inner Calculation* algorithm. It handles all the logic related to signal phases and their timings. It means it can, for example, behave differently and query the *Inner Calculation* with different inputs depending on the current signal phase. Naturally, when the traffic light is green, the calculation needs to be different in comparison with a red light. Nevertheless, the *Inner Calculation* part remains the same, only its inputs may be different.

Sections 5.2 and 5.3 deal solely with the *Inner Calculation* part. Moreover, when referring to “recommended speed calculation” or “algorithm for calcu-

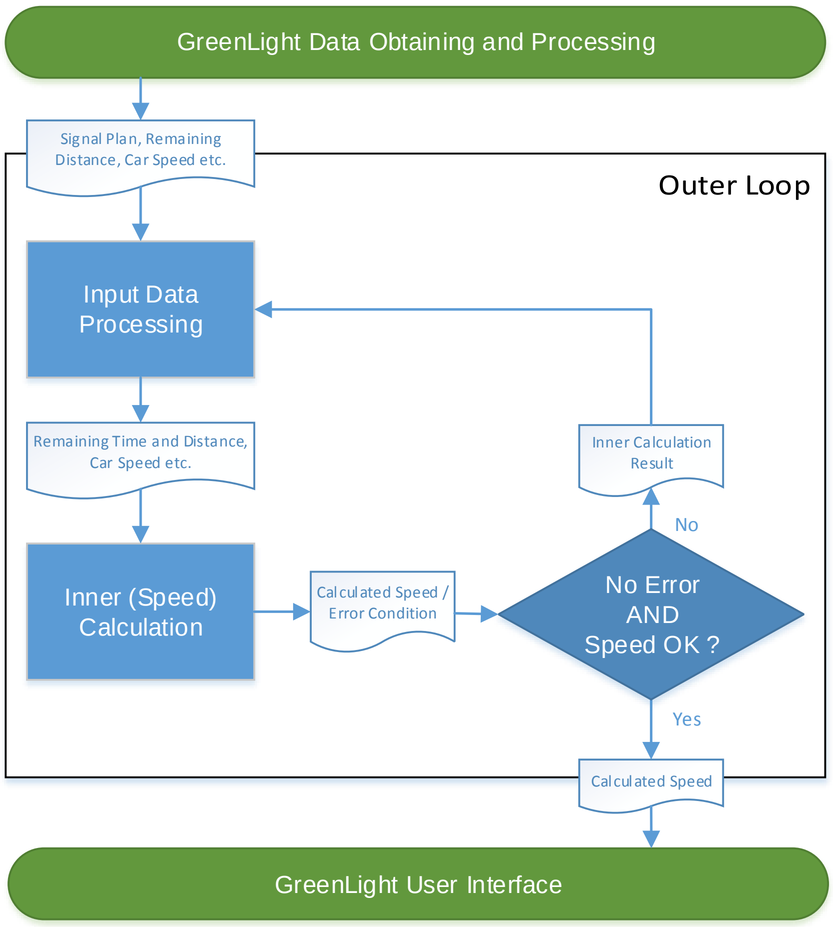

GreenLight Data Obtaining and Processing
Signal Plan, Remaining Distance, Car Speed etc.
Outer Loop
Input Data Processing
Remaining Time and Distance, Car Speed etc.
Inner Calculation Result
No
Inner (Speed) Calculation
Calculated Speed / Error Condition
No Error AND Speed OK ?
Yes
Calculated Speed
GreenLight User Interface

Figure 5.1: Recommended Speed Calculation Procedure

lating a recommended speed", it predominantly means the *Inner Calculation* part. The *Outer Loop* is analyzed in detail in section 5.4.

5.2 Reasons for Proposing a New Algorithm

An algorithm for calculating the recommended speed (*Inner Calculation* as defined above) is a crucial part of any GLOSA system. It can be viewed as a core component strongly influencing an overall performance of the system. There are many factors that can possibly affect a theoretically ideal recommendation. This makes the calculation a non-trivial task and an appropriate attention should be paid to its design and implementation. The current version of the *GreenLight* application uses a very simple algorithm providing an insufficient performance. Its exact drawbacks are outlined in the following section.

5.2.1 Pros and Cons of the Original Calculation

The originally implemented calculation is based on the basic velocity equation $v_{adv} = \frac{s}{t}$ where v_{adv} is a resulting recommended (advised) speed, s is a distance to the intersection and t is the target time. Advantages and drawbacks of this calculations are the following:
Advantages:

- The calculation is very simple and fast, only one single division is required.
- Calculation succeed always when $t > 0$. It does not make sense to calculate the speed for a zero time anyway.
- It does not need any extra input data beyond time and distance. This makes the implementation very simple and calculation results predictable.

Drawbacks:

- It is supposed that the driver is able to achieve the recommended speed in zero time. However, this is not true since a non-zero time is needed to (de)accelerate to the final speed.
- Similarly to the previous point, it takes some time until the driver notices the recommended speed and reacts accordingly.

Although there are only two main drawbacks identified, they seem to be important for performance and reliability of the calculation algorithm. Thus, the rest of this chapter deals with a proposal of a new enhanced algorithm designed to eliminate the listed issues. Theoretical and experimental evaluation and comparison of both algorithms is presented in chapter 6.

5.3 Proposal of a New Speed Calculation

5.3.1 Additionally Considered Input Data

Besides the distance and time, the newly designed algorithm will use three new input values to enhance the calculation. These are listed below:

- **Current Speed of the Car:** Real car speed at the time of the advised speed calculation. This can be easily obtained from a phone GPS sensor or the *SmartGate*^{TM} unit.

- **Car Acceleration Speed:** Average car acceleration speed when (de)accelerating to the recommended speed. It strongly depends on the car model and driver's driving style. It is usually also different for acceleration and deceleration and depends on an actual speed range (e.g. 10 to 20 km/h vs. 50 to 60 km/h). However, it is considered to be a constant number (e.g. 5 ms^{-2}) for now. This is clearly not ideal and it is further discussed in section 5.5.

- **Driver Reaction Time:** Average time interval between the recommended speed is displayed and the driver actually starts (de)accelerating. Similarly to the car acceleration speed, this value is also considered to be constant for now (e.g. 3 seconds).

5.3.2 Idea behind the Calculation

Assume a car to intersection distance of s and a requested target time of t. At this point, a speed calculation takes place and the resulting recommended speed is displayed to the driver. Then it takes t_{react} time until the driver starts to act. This means that the first t_{react} time after the advised speed is displayed, the car keeps going with the initial speed v_0. Possible initial (de)acceleration is neglected here, but it can be considered as a possible future improvement, see section 5.5 for more information. During the first t_{react} time, a corresponding distance of s_{react} is traveled.

After t_{react} time, the driver starts to (de)accelerate with a constant acceleration speed of a (simplification mentioned in 5.3.1). They keep (de)accelerating until the displayed advised speed v_{adv} is reached. It takes t_{acc} time and a distance of s_{acc} is traveled during the (de)acceleration phase. After the final speed is reached, the car continues with the v_{adv} speed until it passes the intersection. This takes t_{rest} time and s_{rest} distance. Another simplification is that this calculation neglects any traffic-related situations when the driver is forced to unexpectedly change the speed during the intersection approach.

Timing of this entire process is examined in figure 5.2. A graph with an x axis displaying distance instead of time would be analogous. However, for any particular intersection approach, both graphs would look a bit differently since the time and distance progresses would not be necessarily the same. Furthermore, in the displayed graph, $v_0 > v_{adv}$ and thus the driver needs to decelerate to the final speed. Analogous graphs could also be drawn for the $v_0 < v_{adv}$ and $v_0 = v_{adv}$ cases. However, when $v_0 = v_{adv}$, the calculation is trivial since there is no speed change needed.

The main idea behind the calculation is the following: Using the known variables s, t, a, v_0 and t_{react}, it is possible to mathematically calculate v_{adv}. This calculation is described in detail in the next section.

There is one more important note. In some cases, there is no $v_{adv} > 0$ satisfying the process illustrated in graph 5.2. This happens when there is not

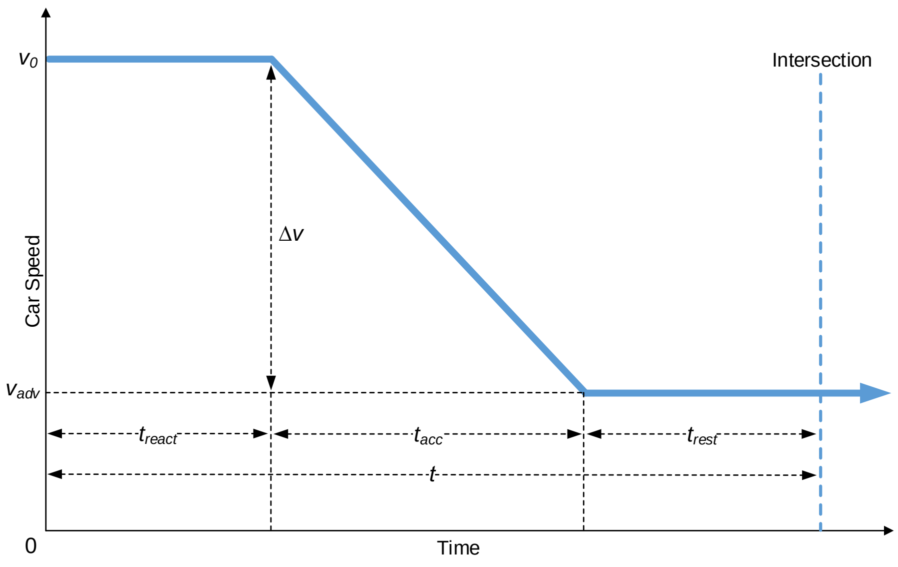


Figure 5.2: Advised Speed Calculation

enough time and/or distance to accelerate or decelerate to v_{adv} before passing the intersection. This means it is not possible to reach the intersection exactly at the requested time. In such case, a reserved error value needs to be returned and the error indicated to the *Outer Loop* which is responsible for handling the situation.

5.3.3 Deduction of the Speed Expression

This section describes step-by-step a deduction of an expression to calculate v_{adv}. The entire calculation is also available as an interactive *Wolfram Mathematica* notebook included on the enclosed CD and attached as appendix C.1 to this thesis. It demonstrates the expression deduction performed self-automatically by the *Mathematica* computation engine. Moreover, second part of the notebook includes dynamic graphs demonstrating a possible usage of the calculated expression. It is highly recommended to evaluate the notebook as well.

The calculation is based on the basic distance equation. Total distance to the intersection (s) is a sum of the following sub-distances:

$$s = s_{react} + s_{acc} + s_{rest} \tag{5.1}$$

Each sub-distance from (5.1) is then expanded using the well-know velocity and acceleration formulas. Conventionally, s represents a distance, t is a time, a is an acceleration, v is a speed and v_0 is an initial speed:

- **Distance:** $s = v \cdot t$
- **Acceleration Distance:** $s = v_0 t + \frac{a \cdot t^2}{2}$
- **Acceleration Speed:** $v = v_0 + a \cdot t$

The expansion of s_{react} is obvious. The car goes with a constant initial speed v_0 for t_{react} time and hence:

$$s_{react} = v_0 \cdot t_{react} \tag{5.2}$$

A similarly easy approach can be used for s_{acc}. It is modeled as a linear motion with a constant acceleration a. The speed is changed from v_0 to v_{adv}. Note that deceleration is the same as acceleration but $a < 0$.

$$s_{acc} = v_0 \cdot t_{acc} + \frac{a \cdot t_{acc}^2}{2} \tag{5.3}$$

Things get a bit trickier with s_{rest}. Analogously to equation (5.2), the car goes with a constant final speed v_{adv} for the remaining time t_{rest} and hence $s_{rest} = v_{adv} \cdot t_{rest}$. To avoid using an extra unknown variable t_{rest}, it can be expressed as $t - t_{react} - t_{acc}$. Furthermore, v_{adv} can be expressed using the "Acceleration Speed" formula mentioned above. Combining all these into a single equation results in the following:

$$s_{rest} = (v_0 + t_{acc} \cdot a) \cdot (t - t_{react} - t_{acc}) \tag{5.4}$$

(5.1) is then expanded using expressions (5.2), (5.3) and (5.4). This results in an equation with a single unknown variable t_{acc}. After some basic algebraic simplifications, the following equation can be obtained:

$$2 \cdot s + a \cdot t_{acc} \cdot (-2 \cdot t + t_{acc} + 2 \cdot t_{react}) = 2 \cdot t \cdot v_0 \tag{5.5}$$

(5.5) is a quadratic equation, and hence it generally has two complex solutions. It means there might be two different t_{acc} satisfying the basic distance equation (5.1) for a given set of input values s, t, a, t_{react}, v_0. This is clearly not possible as follows from section 5.3.2, especially diagram 5.2. To always obtain only a valid solution, equation (5.5) needs to be solved with two additional conditions (5.6) and (5.7):

$$t_{acc} \geq 0 \tag{5.6}$$

$$t_{acc} + t_{react} \leq t \tag{5.7}$$

In case $t_{react} > t$ there might be a seemingly valid solution with $t_{acc} < 0$. Condition (5.6) eliminates such solutions. Similarly, in any other case when there is not enough time to (de)accelerate to the final speed v_{adv}, there might be a solution with t_{acc} being too long, i.e. v_{adv} is reached after the target time. In this case $t_{acc} + t_{react} > t$ and such solutions are eliminated by condition (5.7).

(5.5) is then solved together with (5.6) and (5.7) as a system of three (in)equations over real numbers. 5.8 is the resulting expression for t_{acc}:

$$t_{acc} = t - t_{react} - \sqrt{(t - t_{react})^2 - \frac{2 \cdot (s - t \cdot v_0)}{a}} \tag{5.8}$$

From the “Acceleration Speed” formula mentioned above, it follows that v_{adv} can be simply expressed as $v_0 + a \cdot t_{acc}$. This is also valid for the deceleration case since in that case $a < 0$. The resulting speed expression (equation) is then the following:

$$v_{adv} = v_0 + a \cdot (t - t_{react} - \sqrt{(t - t_{react})^2 - \frac{2 \cdot (s - t \cdot v_0)}{a}}) \tag{5.9}$$

5.3.4 Conditions of a Valid Solution Existence

The speed expression (5.9) is valid only when certain conditions are satisfied. A trivial example of such condition is $a \neq 0$ since a is a fraction denominator. If at least one of those conditions is broken, then there is no valid solution to the speed equation. It means that the intersection cannot be reached exactly at the target time. Contrarily, a valid solution guarantees that it is theoretically possible to reach the intersection at the target time. Note that these solution-existence conditions are different from the conditions eliminating seemingly valid solutions as mentioned in 5.3.3, this section deals solely with the solution-existence conditions.

The conditions strongly depend on possible values of the speed expression inputs (s, t, a, t_{react}, v_0). However, they can be simplified using the following list of assumptions. Some of the assumptions are common for both acceleration and deceleration cases, but some of them are different:

- **Common Assumptions:** $s > 0$, $t > 0$, $v_0 \geq 0$, $t_{react} \geq 0$
- **Acceleration Assumptions:** $a > 0$, $\frac{s}{v_0} > t$
- **Deceleration Assumptions:** $a < 0$, $\frac{s}{v_0} < t$

Most of the assumptions listed above are self-evident. It does not really make sense to calculate a recommended speed for zero or negative remaining time or distance. On the other hand, a zero v_0 is theoretically possible as well as a zero t_{react} for e.g. an autonomous car where the reaction time might be negligible.

A notable assumption is $\frac{s}{v_0} > t$ which is always true for the acceleration case. This compares a hypothetical time in which the car would arrive at the intersection if continued with the initial speed v_0. If this arrival time is greater than the target time t, then the car needs to accelerate to arrive at the intersection exactly at the target time. Condition $\frac{s}{v_0} < t$ for the deceleration case is analogous.

Finally, there is a valid solution to the speed equation (i.e. valid v_{adv}) if and only if all the following conditions are satisfied. These were determined when mathematically deducting the speed equation and simplified using the assumptions listed above.

- **Common Conditions:** $t > t_{react}$
- **Acceleration Conditions:** $a > \frac{2 \cdot (s - t \cdot v_0)}{(t - t_{react})^2}$
- **Deceleration Conditions:** $a < \frac{2 \cdot (s - t \cdot v_0)}{(t - t_{react})^2}$, $v_{adv} > 0$

5.3.5 Algorithm for Calculating the Advised Speed

From the mathematical point of view, calculation of the recommended speed is complete. However, the speed expression and its conditions need to be transformed to an actual algorithm which is described in this section. Implementation-related aspects of the proposed algorithm are then examined in chapter 6.

Figure 5.3 illustrates the speed calculation algorithm and related data flows. The algorithm can be decomposed into five main steps:

1. **Acceleration / Deceleration Determination:** First, it needs to be determined whether the car should accelerate or decelerate. It can be done easily by calculating $\frac{s}{v_0}$ which is a hypothetical time in which the car would arrive at the intersection if continued with a constant initial speed v_0. Three different situations may occur:
   - $\frac{s}{v_0} > t$**:** The car would reach the intersection after the target time. Hence, the driver needs to accelerate to arrive exactly at the target time.
   - $\frac{s}{v_0} < t$**:** The car would reach the intersection before the target time. Hence, the driver needs to decelerate to arrive exactly at the target time.
   - $\frac{s}{v_0} = t$**:** The car would reach the intersection exactly at the target time. In this case, no speed change is needed and $v_{adv} = v_0$. All the remaining steps of the algorithm can be skipped.
2. **Acceleration Conversion:** The deceleration case is assumed to be the same as the acceleration but with $a < 0$. However, a is currently a fixed positive real number and thus it needs to be inverted if and only if the car is expected to decelerate. This can be done easily by calculating

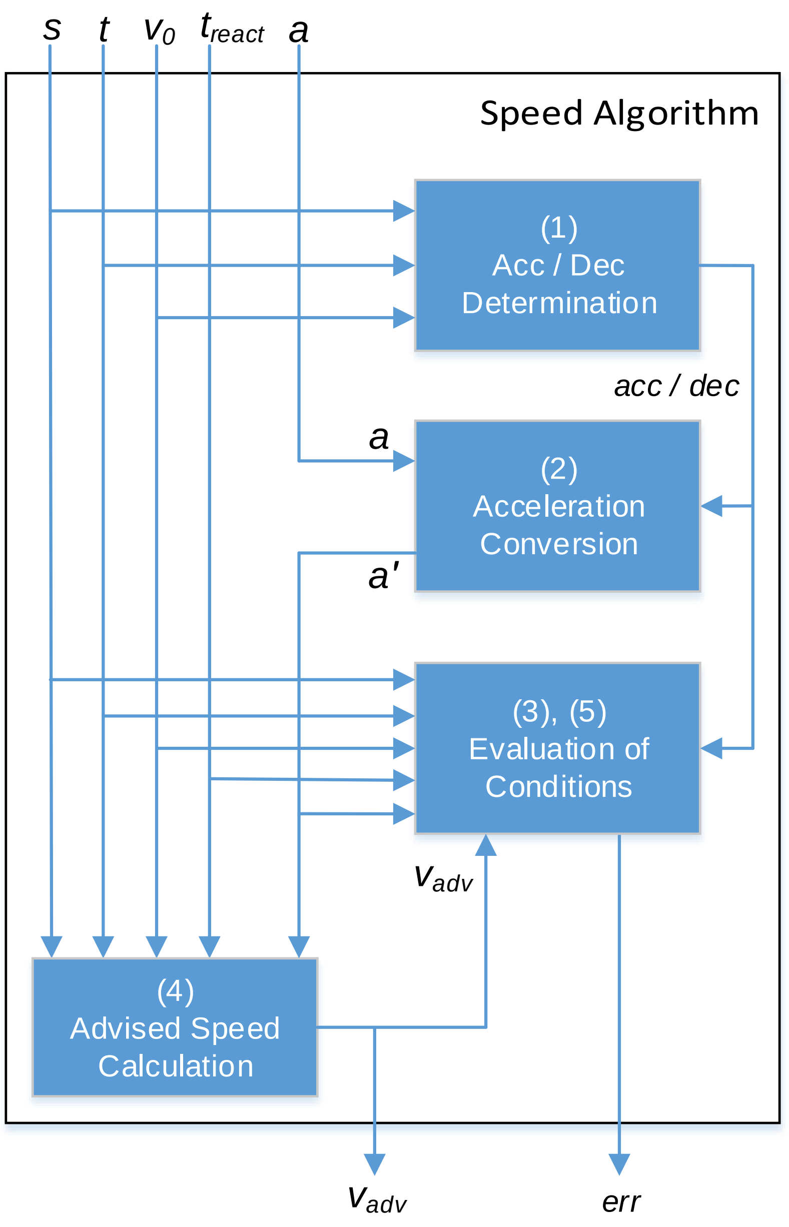

s
t
v0
treact
a
Speed Algorithm
(1) Acc / Dec Determination
acc / dec
a
(2) Acceleration Conversion
a'
(3), (5) Evaluation of Conditions
vadv
(4) Advised Speed Calculation
vadv
err

Figure 5.3: Algorithm for Calculating the Advised Speed

$a' = (-1) \cdot a$. From this point, the algorithm would work only with the converted value a' instead of the original a.

3. **Evaluation of Conditions:** All the relevant solution-existence conditions are evaluated. The only exception is the condition $v_{adv} > 0$ which

cannot be evaluated at this step since v_{adv} is not available yet. Note that the conditions are different for acceleration and deceleration cases. If at least one the conditions is not satisfied, an error is indicated and no actual calculation is performed. Otherwise, the algorithm continues with the next step.

4. **Advised Speed Calculation:** Finally, the speed expression is evaluated with the input values s, t, a', t_{react}, v_0, resulting in a recommended speed v_{adv}. As already discussed in 5.1, this is not necessarily a final value to be displayed to the driver but only a result of the *Inner Calculation* part.

5. **Evaluation of Calculated Speed:** In certain cases, it is possible that $v_{adv} \leq 0$. Therefore, an extra condition check is required. This is relevant only for the deceleration case. If $v_{adv} \leq 0$ then an error is indicated. Otherwise, v_{adv} is returned. The algorithm ends after this step.

5.4 Algorithm Usage in Various Traffic Scenarios

The algorithm proposed in the previous section acts as the *Inner Calculation* as defined in 5.1. It can be seen as a black-box function with five input values (s, t, a, t_{react}, v_0) and two outputs (v_{adv}, err). This section describes its usage for determining an actual recommended speed to be displayed to the driver. This process represents the *Outer Loop* part of the calculation.

The *Outer Loop* algorithm is based on an original *GreenLight* implementation done by the *Intens Corporation.* Since there have been no problems identified with this part, only a few minor adjustments and improvements are proposed for the new application update. The main difference is that the *Inner Calculation* changes from the simple s/t expression to the complex algorithm proposed in 5.3. Thus, the *Outer Loop* needs to be modified to support this enhanced calculation. First, it might be useful to describe the original implementation.

5.4.1 Original Implementation of ***Outer Loop***

As already described in section 1.3.2, a single SPaT message includes a signal plan and timing for the current and up to 15 future signal phases. The main idea of the *Outer Loop* is to iteratively query the *Inner Calculation* with time values t until a reasonable recommended speed v_{adv} is obtained. Each input time t corresponds to a start or end time of one of the signal phases obtained in the SPaT message.

Furthermore, a MAP message may optionally include a maximum allowed speed (regulatory speed limit) for the relevant intersection approach. Each obtained v_{adv} is compared with this limit and rejected if it is higher. However, a speed limit may not be available in the received MAP message. In that case, a fixed default limit is used (e.g. 50 km/h).

The originally implemented *Outer Loop* algorithm is visualized in flowcharts 5.4 and 5.5 in slightly simplified forms. The processing depends on the current phase of the light signal. For an easy understanding, there is one flowchart for the green phase and another for the remaining phases (red or yellow).

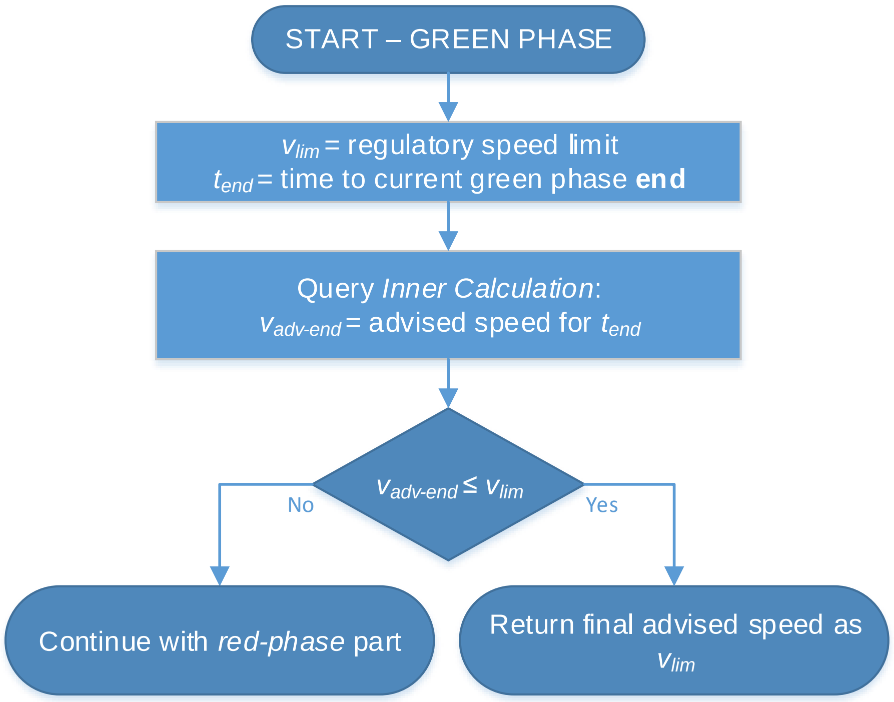

START – GREEN PHASE
v_{lim} = regulatory speed limit
t_{end} = time to current green phase **end**
Query *Inner Calculation*:
$v_{adv\text{-}end}$ = advised speed for t_{end}
$v_{adv\text{-}end} \leq v_{lim}$
No
Yes
Continue with *red-phase* part
Return final advised speed as v_{lim}

Figure 5.4: Original *Outer Loop* – Green Phase

If the current signal phase is green, the algorithm is quite straightforward. A speed to reach the intersection at the end of the current (green) phase is calculated. If it is higher than the speed limit, it means the intersection cannot be passed during the current signal phase. Hence, the processing continues with the algorithm for red or yellow phase in which the future green phases are considered. Otherwise, the final recommended speed is set as the speed limit since there is a preference of recommending the maximum possible speed (see below).

For a red or yellow signal phase, the algorithm is a bit more complicated. It iterates over the future green phases until there is a phase for which the intersection can be passed with a lower or the same speed as the speed limit.

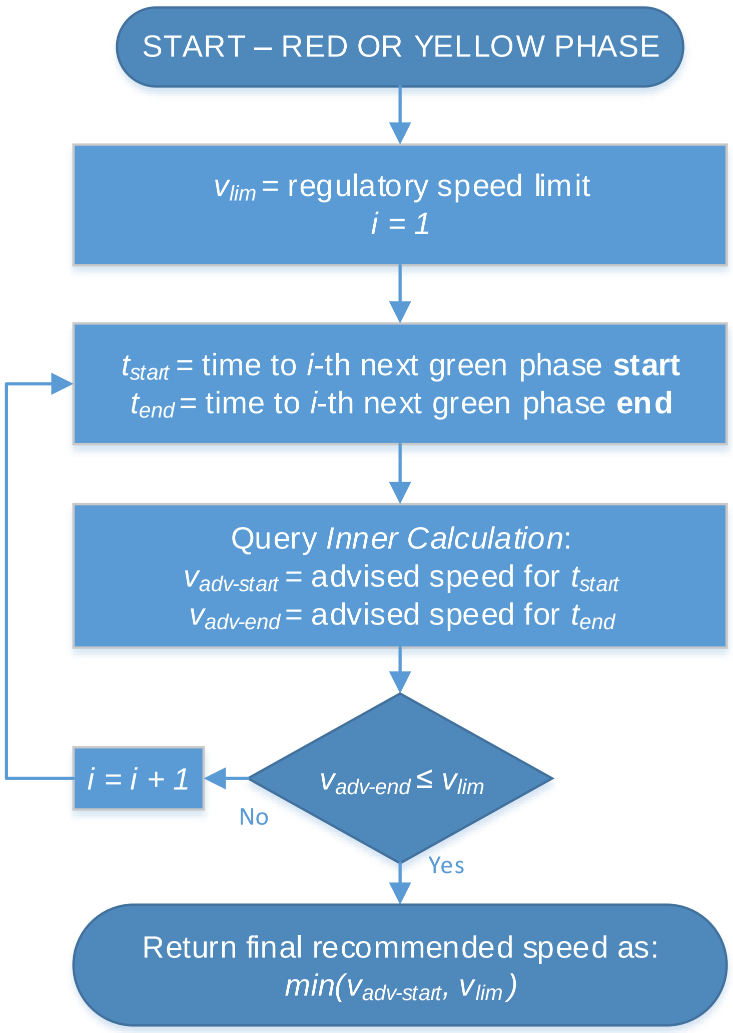

START – RED OR YELLOW PHASE
v_{lim} = regulatory speed limit
$i = 1$
t_{start} = time to i-th next green phase **start**
t_{end} = time to i-th next green phase **end**
Query *Inner Calculation*:
$v_{adv\text{-}start}$ = advised speed for t_{start}
$v_{adv\text{-}end}$ = advised speed for t_{end}
$v_{adv\text{-}end} \leq v_{lim}$
$i = i + 1$
No
Yes
Return final recommended speed as:
$min(v_{adv\text{-}start}, v_{lim})$

Figure 5.5: Original *Outer Loop* – Red or Yellow Phase

Since this phase begins in the future, it is needed to also consider the maximum speed that guarantees passing the intersection during this phase. The maximum passing speed is calculated as an advised speed to reach the intersection at the beginning of that particular phase. Then, a minimum of the speed limit and the calculated maximum passing speed is set as the final recommended speed to be displayed to the driver.

A notable attribute of both algorithms is a preference of recommending the highest possible speed (up to a speed limit). According to *Intens Corporation*, this strategy was chosen to improve a traffic flow continuity. However, the driver might be forced to accelerate even if it is not necessary.

5.4.2 Proposed Improvements to *Outer Loop*

The original *Outer Loop* algorithm is considered to be satisfactory and there is no intention to replace it with a completely new algorithm. There are, however, a few improvements and modifications proposed for the upcoming *GrenLight* release. These are mainly aimed towards a full support of the new *Inner Calculation* algorithm and to enhance the general speed recommendation performance. Some of the modifications are only implementation-related and thus neglected in this section.

The main proposed change is to always recommend a speed range instead of a single speed. This gives the driver a better information and allows some flexibility in choosing an actual driving speed. For any considered green phase (current or future), the recommended speed range bounds are determined as follows:

- **Lower bound:** Calculated as a speed to reach the intersection at the end of the phase. In certain cases, it may not be possible to reach the intersection after the end of the phase according to the calculation. This can happen, for example, when the current phase is green and speed of the car is too hight to decelerate enough to pass the intersection after the phase end. In such cases, the lower bound (minimum) speed is set to 0.

- **Upper bound:** Calculated as a speed to reach the intersection at the start of the phase. In certain cases, it may not be possible to reach the intersection before the start of the phase according to the calculation. For example, this is true when the considered green phase is the current phase. In such cases, the upper bound (maximum) speed is set to the speed limit.

Difference between both recommendation variants is demonstrated in examples shown in figure 5.6. The original algorithm prefers recommending the highest possible speed. This is especially imperfect in cases shown in the figure. For example, suppose that the car is approaching the intersection with an initial speed of 30 km/h, current signal phase is green and the intersection can be passed smoothly with a speed of at least 30 km/h. The original algorithm recommends a speed of 50 km/h which is a speed limit in this example. The driver does not know if they can pass smoothly with any lower speed, making them to accelerate to 50 km/h which is actually not necessary in this case. In contrast, the newly proposed algorithm recommends a speed range of 30-50 km/h allowing the driver to keep their initial speed. Moreover, the driver knows that they can accelerate if they prefer to.

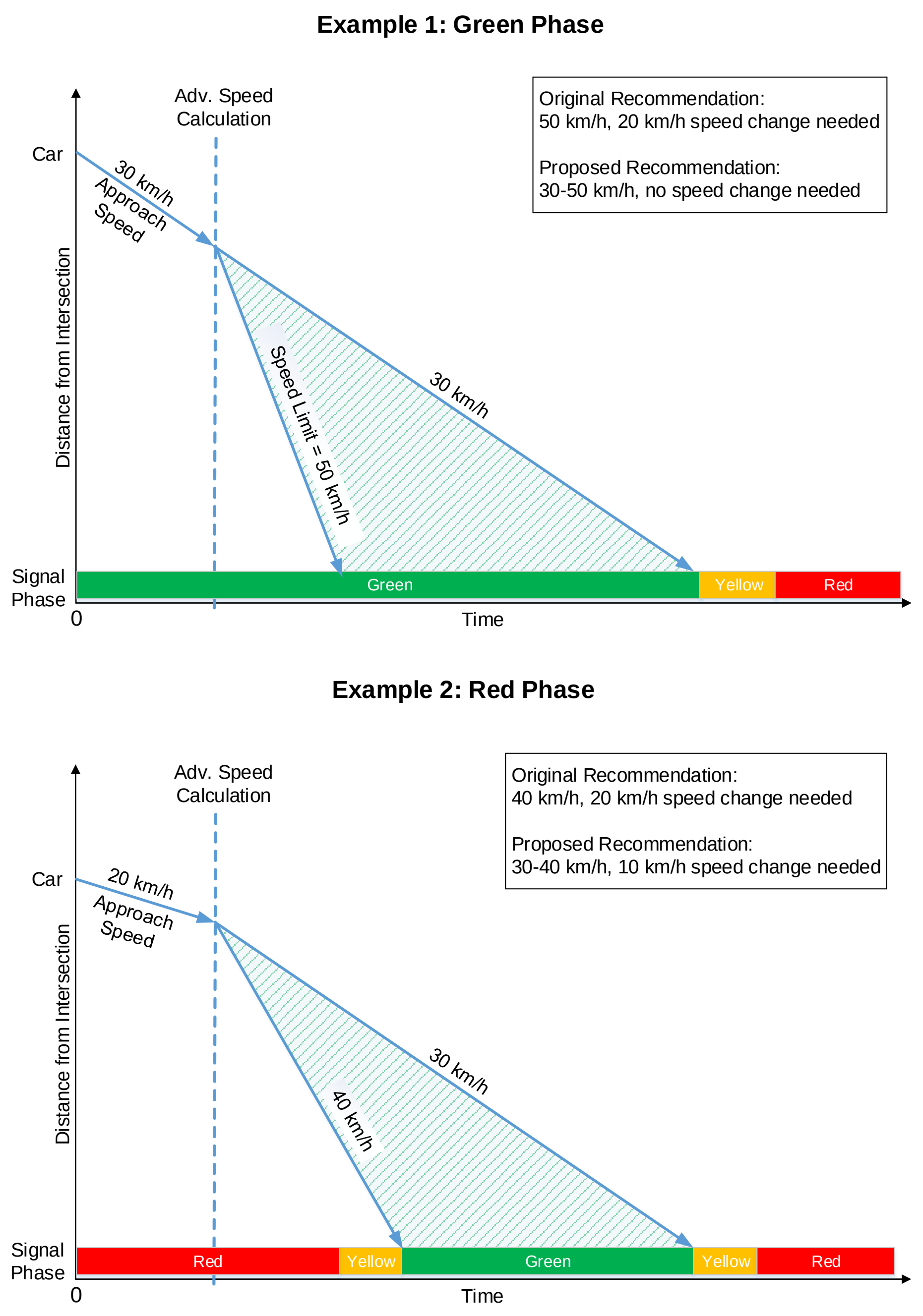


Figure 5.6: Difference Between Speed Recommendation Methods

5.5 Possible Further Improvements

The speed algorithm presented in this chapter uses a few simplifications that could be possibly considered problematic for a production deployment. On the other hand, the *GreenLight* application is still in the proof-of-concept development phase, and all these simplifications have been evaluated as acceptable. Nevertheless, there is a big room for improvements that could significantly increase performance of the proposed algorithm. Some of them are discussed in this section.

5.5.1 Adaptive Reaction Time

The proposed algorithm uses a reaction time of the driver as one of the new inputs for the calculation. It is currently designed to be a constant value set in a configuration file (see 6.1.3 for implementation details). This is not an ideal solution since a real reaction time varies depending on many factors. These include a level of driver's attention to the application, current traffic situation and others.

A possible method to improve accuracy of the reaction time estimation is to use an adaptive algorithm. The saved reaction time value would automatically keep adjusting depending on real reaction times measured in previous situations. In its simplest form, it can be just an arithmetic mean of reaction times observed in last n situations, where n is a reasonably small positive integer. This is quite easy to implement, but the biggest problem would be to reliably measure the actual reaction time.

Reaction time is intuitively a time between the recommended speed is displayed and the driver starts to (de)accelerate. This period can be problematic to measure since the car can be accelerating due to traffic conditions instead of the displayed recommended speed. Moreover, it is not guaranteed (nor probable) that the car has a zero acceleration at the beginning of the calculation.

5.5.2 Variable Acceleration Speed

Similarly to the reaction time, acceleration speed is a calculation input currently designed as a constant value set in a configuration file. The same method (adaptive learning algorithm) as described for the reaction time can also be used for the acceleration time, but there is even more variability factors involved:

- **Acceleration vs. deceleration:** Acceleration speed (absolute value) is likely different for acceleration and braking.

- **Acceleration curve:** Acceleration speed depends on speed the vehicle is moving with. It is determined by vehicle construction as well as other factors such as aerodynamics, current gear etc.
- **Speed Difference:** Driver can react differently depending on the needed speed change. For example, they would probably accelerate faster from 20 to 50 km/h in comparison with a slight speed adjustment from 45 to 50 km/h.
- **Driving style:** Each driver can have a slightly different driving style. Some prefer a dynamic style with faster (de)acceleration while others can be more calm and steady.
- **Traffic conditions:** Possible speed change might be limited by current traffic conditions. For example, it could be difficult or even impossible to rapidly (de)accelerate in a heavy traffic.

The factors listed above show that the acceleration speed is rather a function of multiple input variables (factors) than a constant number. Moreover, it can even vary during a single calculation. This would require a detailed standalone analysis to be able to choose a proper model and method for estimating the acceleration speed.

5.5.3 Initial Acceleration

The proposed algorithm supposes that the car has a constant speed with a zero acceleration at the moment of the recommended speed calculation. This is, however, not guaranteed and a possible initial acceleration should be taken into account. Technically, it is not a problem to obtain the acceleration speed information since it is available from the *SmartGate*^{TM} unit and possibly also from a phone GPS sensor. However, this would complicate the calculation, hence it is currently neglected.

5.5.4 Traffic Conditions

In certain traffic situations, it is not possible to follow the recommended speed. Example might be a dense, slowly moving or completely congested traffic. Such situations are currently not considered and the algorithm keeps calculating an advised speed regardless the traffic conditions. It would be better if the application could react accordingly and e.g. modify the calculation to reflect the traffic situation. However, this is another complex topic which should be analysed separately.

Chapter 6

Implementation and Verification of the Proposed Speed Algorithm

6.1 Implementation of the Algorithm

The newly proposed algorithm to calculate a recommended speed was presented in chapter 5. This section discusses its implementation and a successful integration in the *GreenLight* application which has been performed as part of this thesis project.

6.1.1 Technologies Used for the Implementation

As already described in section 2.1, *GreenLight* is a mobile application for devices with the *Google Android* operating system. It is being developed in the *C#* programming language using the *Xamarin* platform.

Xamarin is a *Microsoft*-owned company known for their cross-platform implementations of *.NET Framework* called *Mono*. *GreenLight* uses the *Xamarin.Android* platform (formerly called *Mono for Android*) that allows to build native *Android* applications in *C#* using *.NET Framework* libraries. Additionally, *Xamarin.iOS* is an analogous product for *Apple iOS* [61]. This means that a possible future port of the *GreenLight* application for *iOS* devices is certainly possible since most of the code-base would be fully compatible. On the other hand, *iOS* is currently not supported by the *MirrorLink*^{TM} technology, refer to section 2.2 for more information.

GreenLight is currently being developed and maintained by the *Intens Corporation.* Due to legal issues, the application source code could not be completely available for the purposes of this thesis project. Nevertheless, *Intens* is actively supporting the project and they agreed to adjust the application to bypass this limitation. It is fortunately possible to dynamically load an external application module (assembly) from a running *.NET* application. *Intens* provided a compiled underlying application logic in a form of an *Android* .apk package and a template of an external assembly to implement the speed calculation. More specifically, the external assembly was intended to include entire *Outer Loop* as described generally in 5.1 and concretely in 5.4.

6.1.2 Architecture of the Speed Calculation Component

Composition of the speed calculation component can be seen in figure 6.1 in form of a UML class diagram. *SampleExternalRecommendedSpeedCalculator* is a class responsible for the calculation. Its settings are loaded as a *SampleExternalRecommendedSpeedCalculatorSettings* instance during the application initialization. These are the acceleration speed and reaction time values, as discussed in the previous chapter.

The speed calculation itself is performed by the *GetRecommendedSpeed()* method. It is called periodically every time the calculation is triggered, i.e. once per second when the car is approaching a GLOSA-equipped intersection. Input data for the calculation are passed as an *ExternalRecommendedSpeedCalculatorArgs instance.* It includes a current speed of the car, allowed maximum speed and all the available information about the intersection, especially its distance and complete signaling plan. A calculated recommended speed is then returned as a *RecommendedSpeed* instance. It either contains the calculated speed range or it can be marked as an invalid recommendation in case of any calculation-related error such as a wrong input data.

6.1.3 XML Configuration Files

Configuration of the application including the speed calculation component is loaded from external XML files during the application initialization. It includes the acceleration speed and reaction time values that are mapped to the *SampleExternalRecommendedSpeedCalculatorSettings* instance as mentioned above. The rest of the configuration files is mostly related to the underlying application logic and it is not important to describe it here.

Nevertheless, one notable feature of the application is a possibility to completely define one or more "virtual" intersections in the configuration files. All the details including intersection location, approach lanes, signaling plan and speed limits are configurable. It means the specified intersection would appear as a real GLOSA-equipped intersection to the application. This approach can be very beneficial for various debugging and testing purposes.

6.2 Algorithm Testing in Real Traffic

After the successful implementation, the only remaining task was to verify the performance and reliability of the newly implemented speed algorithm. This

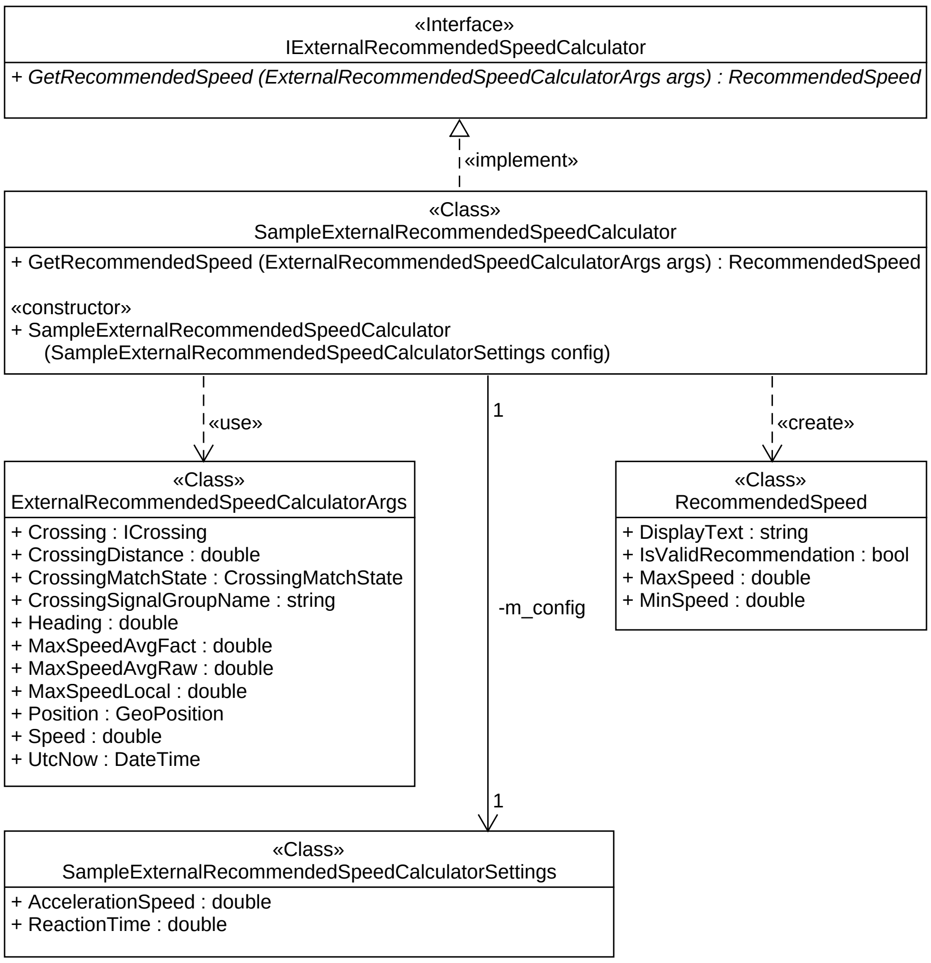

«Interface»
IExternalRecommendedSpeedCalculator
+ GetRecommendedSpeed (ExternalRecommendedSpeedCalculatorArgs args) : RecommendedSpeed
«implement»
«Class»
SampleExternalRecommendedSpeedCalculator
+ GetRecommendedSpeed (ExternalRecommendedSpeedCalculatorArgs args) : RecommendedSpeed
«constructor»
+ SampleExternalRecommendedSpeedCalculator
(SampleExternalRecommendedSpeedCalculatorSettings config)
«use»
1
«create»
«Class»
ExternalRecommendedSpeedCalculatorArgs
+ Crossing : ICrossing
+ CrossingDistance : double
+ CrossingMatchState : CrossingMatchState
+ CrossingSignalGroupName : string
+ Heading : double
+ MaxSpeedAvgFact : double
+ MaxSpeedAvgRaw : double
+ MaxSpeedLocal : double
+ Position : GeoPosition
+ Speed : double
+ UtcNow : DateTime
-m_config
«Class»
RecommendedSpeed
+ DisplayText : string
+ IsValidRecommendation : bool
+ MaxSpeed : double
+ MinSpeed : double
1
«Class»
SampleExternalRecommendedSpeedCalculatorSettings
+ AccelerationSpeed : double
+ ReactionTime : double

Figure 6.1: UML Class Diagram of Speed Calculation Component

section presents an experimental verification in real traffic conditions.

6.2.1 Testing Environment

The original idea was to use an existing GLOSA-equipped intersection to test the updated *GreenLight* application. Unfortunately, there are currently no intersections permanently equipped with compatible ITS technology in a reasonable distance from Prague, Czech Republic in which this thesis project is based. Furthermore, due to a complicated administrative process and related legal issues, it was not even possible to temporarily equip an intersection with relevant ITS technology.

As an alternative approach, we used the method described in section 6.1.3.

A virtual intersection and its timing were defined in the *GreenLight* configuration file instead of actual receiving data from the intersection using C2I communication. This is, in fact, not considered to be a problem since the test objective is the speed calculation algorithm and not the C2I communication itself. To be able to test it in a real traffic, location and signaling plan of the virtual intersection was set to location and signaling plan of an existing intersection.

This required to find an existing intersection with a static signaling plan, i.e. length of a signal period (e.g. green to green) is fixed as well as length of each green, red and yellow phase. Most of the modern intersections, however, change their signaling plan dynamically depending on factors such as current traffic situation, waiting pedestrians and passing public transport vehicles. Such intersections are generally not ideal for GLOSA systems since the signaling plan and consequently also the recommended speed can change frequently. Figure 6.2 shows a statically timed intersection at street Radlická in Prague. This one was chosen and used for the testing.

The *GreenLight* application with the new speed algorithm was loaded in a *Sony Xperia Z3 Compact* tablet (running *Android* 5.1.1) and placed in a testing car. Unfortunately, it was not possible to use a *MirrorLink*^{TM} integration since the originally certified application has been modified and thus needs a new *MirrorLink*^{TM} certification. This process takes further time and could not be finished before the testing session. The infotainment display was used to display a current speed of the car instead. The used test setup can be seen in figure 6.3.

6.2.2 Test Results

The testing itself consisted of multiple passes through the intersection described in the previous section. The testing driver always tried to follow the speed recommendation. As expected, this was not always possible due to a dense traffic or other traffic related events such as crossing pedestrians or a slow vehicle ahead.

Results of the tests are summarized in table 6.1. Each row in the table represents one of the monitored metrics. The *Count* and *Percentage* fields represent the number and ratio of intersection passes for which the corresponding metric was satisfied. Meanings of individual metrics are the following:

- **Total pass count:** Total number of passes through the intersection that were performed during the testing session.
- **Application not failed:** Number of passes when the application was working properly. Any unexpected application behavior noticed by the driver or by another tester would be considered as a failure for this metric.

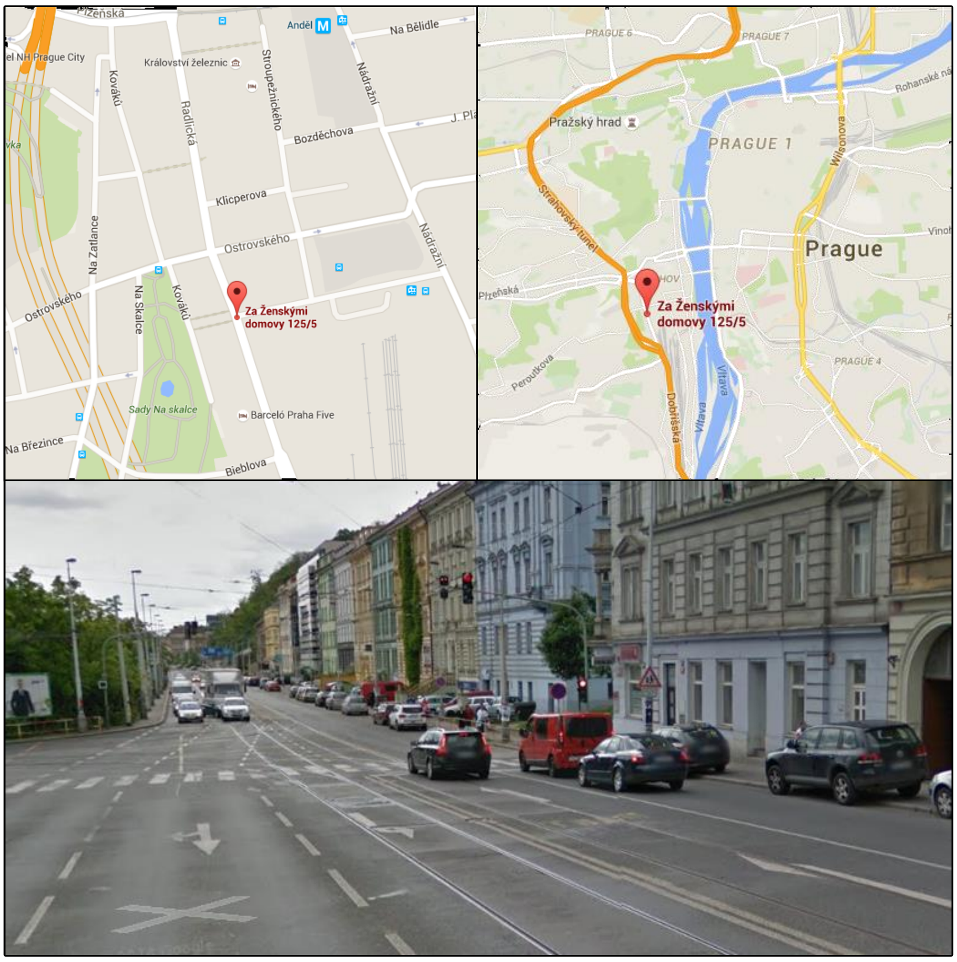


Figure 6.2: Statically Timed Intersection in Prague, Czech Republic [52]

- **Advised speed reasonable:** Number of passes when the recommended speed seemed to be "reasonable", i.e. it seemed possible to pass the intersection during a green phase with the displayed recommended speed. This is a subjective metric relying on a manual evaluation from the tester. The only reason to include it is to detect cases when no application error had been observed, but the recommended speed was clearly incorrect.

- **Advised speed could be followed:** Number of passes when the driver was able to follow the speed recommendation. As mentioned above, this was not always possible due to a dense traffic, crossing pedestrians, slow vehicle ahead or a stopped vehicle of public transport. This includes cases when following the recommended speed would disrupt the traffic

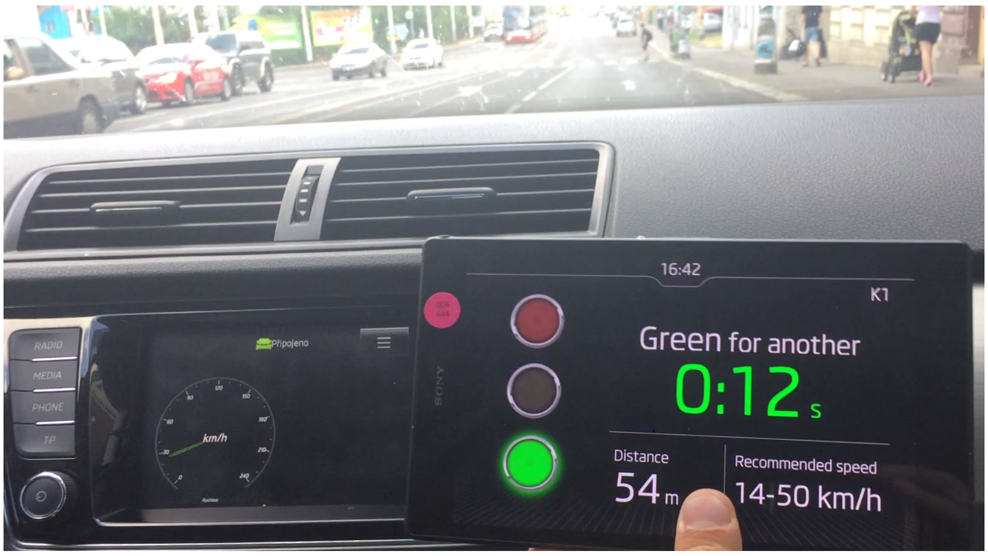


Figure 6.3: *GreenLight* Test Setup

flow, e.g. by going much slower than other vehicles in a dense traffic.

- **Passed as advised:** Number of passes when the recommended speed was followed and the intersection was passed during a green phase in accordance with the speed recommendation.

Metric	**Count**	**Percentage**
Total pass count	20	100%
Application not failed	20	100%
Advised speed reasonable	20	100%
Advised speed could be followed	8	40%
Passed as advised	8	40%

Table 6.1: Results of Speed Algorithm Testing

As can be seen in table 6.1, the application worked properly during the entire testing session. Nonetheless, in 60% cases it was not possible to fully follow the speed recommendation since the testing was conducted at a busy intersection in a dense traffic. Unfortunately, there were limited options for selecting a feasible intersection since a statically timed intersection was required (refer to section 6.2.1 for more details). In all the remaining cases, the speed recommendation had been successfully followed and the intersection was smoothly passed during a green phase.

Although the scope of the testing and number of performed tests were quite limited, it was demonstrated that the newly implemented speed algorithm is reliable and performs with good results. Nevertheless, an extensive and much more detailed testing should be definitely performed before reaching the production phase of the application. This is, however, beyond the scope of this thesis project.

A short video documenting selected parts of the testing session is included on the CD that is enclosed to this thesis.

6.3 Comparison with the Previously Implemented Algorithm

Although the new enhanced algorithm has been tested in real traffic situations, it is important to compare it with the original simple calculation which is described in section 5.2. Due to limited opportunities for a real-traffic testing, a theoretical comparison is presented in this section.

6.3.1 Statistical Evaluation Using Randomly Generated Data

Statistical comparison using randomly generated data has been selected as a suitable method to compare both algorithms. The evaluation has been implemented as a notebook for *Wolfram Mathematica* and it is included on the enclosed CD and attached as appendix C.2 to this thesis. The notebook also describes the entire comparison in detail, including determination of feasible parameters, generating input data, querying the algorithms and obtaining the needed statistic metrics.

The main idea behind the method is to generate a large set of input vectors such that the same vectors can be used as inputs for either of the algorithms. Each vector represents a random car approaching a hypothetical intersection. It consists of a distance to the intersection, light signal timing, current speed of the car, driver's reaction delay, car acceleration speed and regulatory speed limit. The last three parameters are fixed for entire evaluation in order to match the values used for the real-traffic testing described in section 6.2. The remaining parameters are generated as random numbers from predefined ranges. More specifically, these are random variates from corresponding uniform distributions. The only exception is the current speed of the car which is a random variate from normal distribution shown in figure 6.4. This was chosen to approximate speed distribution of cars approaching the intersection.

After the set of input vectors is generated, both algorithms are queried with all this data. The resulting recommended speeds are stored in pairs such that each pair contains calculated speeds corresponding to one of the vectors but obtained with different algorithms. All the comparisons are then performed pair-wise, i.e. only speeds obtained with the same input vector are compared together.

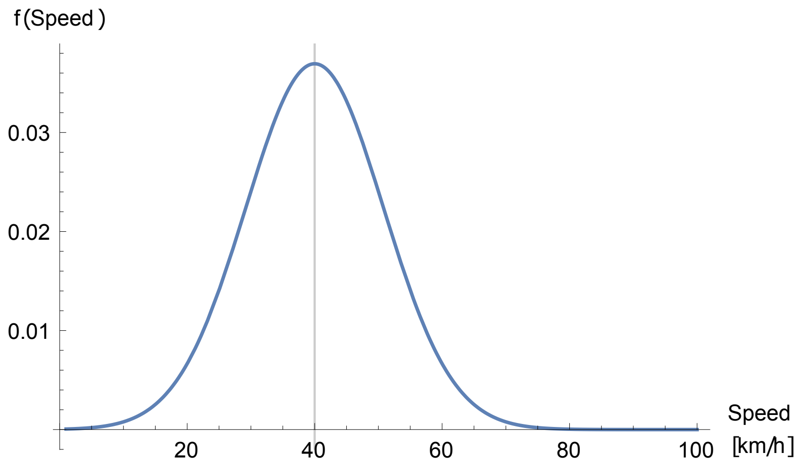


Figure 6.4: Probability Distribution of Initial Speed Parameter

f() stands for probability density function

Prior description of the comparison and evaluation methods themselves, please recall how the recommended speeds are determined by each of the algorithms in the following situations (refer to chapter 5 for more details):

- **Considering current green phase**
  - **Original algorithm:** If a calculated minimum speed is not higher than the regulatory speed limit (v_{lim}), then the value of v_{lim} is recommended. Otherwise, the intersection cannot be passed during the current phase.
  - **Newly implemented algorithm:** If a calculated minimum speed ($v_{adv-end}$) is not higher than the speed limit (v_{lim}), then a speed range of $[v_{adv-end}, v_{lim}]$ is recommended. Otherwise, the intersection cannot be passed during the current phase.
- **Considering future green phase**
  - **Original algorithm:** If a calculated minimum speed ($v_{adv-end}$) is not higher than the regulatory speed limit (v_{lim}), then the value of $min(v_{adv-start}, v_{lim})$ is recommended. $v_{adv-start}$ stands for a calculated maximum speed. Otherwise, the intersection cannot be passed during the considered phase.
  - **Newly implemented algorithm:** If a calculated minimum speed ($v_{adv-end}$) is not higher than the speed limit (v_{lim}), then a speed range of $[v_{adv-end}, min(v_{adv-start}, v_{lim})]$ is recommended.
    $v_{adv-start}$ stands for a calculated maximum speed. Otherwise, the intersection cannot be passed during the considered phase.

Regarding the comparisons, calculated speeds are compared together in each pair and evaluated whether they differ or not. In some cases, a degree of difference is also quantified (see below). Entire evaluation is divided in two parts processed separately:

- **Green Phase Part:** In this part, it is assumed that the current state of the signal is green. The signal timing parameter consists only of a remaining time to the phase end. All the calculations are performed with respect to this phase, no future phases are considered. The following situations may occur:
  - If both algorithms recommend valid speeds, then the speed recommended by the original algorithm is always within the range recommended by the new algorithm. Thus, the results are evaluated as not differing.
  - If only one of the algorithms recommends a valid speed, the results are evaluated as differing.
  - If neither of the algorithms recommends a valid speed, the results are ignored. These cases are mostly caused by input vectors for which it is not realistic to calculate a reasonable speed, e.g. 5 seconds to red and a distance of 500 meters. Such cases do not indicate anything about relative qualities of the algorithms.
- **Red Phase Part:** In this part, it is assumed that the current state of the signal is red. The signal timing parameter consists of times to the start and the end of the next green phase. All the calculations are performed with respect to that phase, no other future phases are considered. The following situations may occur:
  - If both algorithms recommend valid speeds, the evaluation depends on whether the speed recommended by the original algorithm is within the range recommended by the new algorithm or not. If it is, then the results are evaluated as not differing. If it is not, then the results are evaluated as differing and distance from the closest range bound is interpreted as a degree of difference.
  - If only one of the algorithms recommends a valid speed, the results are evaluated as differing.
  - If neither of the algorithms recommends a valid speed, the results are ignored (analogously to the green phase case).

6.3.2 Results of the Statistical Evaluation

Results presented in this section were obtained using 1,000,000 of randomly generated input vectors for each evaluation part. It means each algorithm was queried 2,000,000 times and 2,000,000 result pairs were compared in total. The *Mathematica* notebook attached to this thesis (on the CD and as appendix C.2) is configured to use only 10,000 input vectors and the evaluation should take no more then a few seconds. The vectors were generated with the following configuration of parameter ranges:

- Distance to the intersection: 1–500 m
- Time to current green end: 1–60 s
- Time to next green start: 1–60 s
- Time to next green end: Time to next green start + 15–60 s
- Car speed: random variate from $\mathcal{N}(40\ km/h,\ 11.1\ km/h)$, limited to 0–100 km/h
- Reaction delay: 3 s
- Acceleration speed: 5 $m.s^{-2}$
- Speed limit: 50 km/h

The results are listed in table 6.2.

Metric	**Count**	**Percentage**
Green Phase Part		
Number of pairs w/o ignored	712,059	100%
Not differing pairs	698,494	98.1%
Differing pairs	13,565	1.9%
Red Phase Part		
Number of pairs w/o ignored	986,016	100%
Not differing pairs (in range)	380,391	38.6%
Differing pairs (total)	605,625	61.4%
Differing pairs (out of range)	603,800	61.2%
Differing pairs (no recommendation)	1,825	0.2%

Table 6.2: Statistical Comparison of Speed Algorithms

Regarding the *Green Phase Part*, the algorithms differ in only 1.9%. This is caused by the recommendation logic of the original algorithm, more specifically its preference of recommending the highest possible speed (i.e. speed limit). If both algorithms recommend a valid speed, then the speed recommended by the original algorithm is always equal to upper bound of the speed range recommended by the new algorithm. Therefore, both algorithms are evaluated as not differing even though the new algorithm provides driver with a better information and allows flexibility in choosing an actual driving speed. This was already discussed in section 5.4.

Regarding the *Red Phase Part*, the algorithms differ in 61.4%. Only in 0.2%, one of the algorithms was not able to calculate a reasonable recommended speed while the other one was. Therefore, the difference is caused predominately (61.2%) by missing the speed range.

Besides this simple metric, it is also important to examine how "far" from range bounds the recommended speeds are. This is visualized in figure 6.5 which is a histogram of distances to the closest bound of the speed range. Only the differing results are displayed. Width of each bin (interval) is 0.1 km/h. Positive values are the cases where the recommended speed is above the upper bound of the range. Analogously, negative values would stand for cases where the recommended speed is below the lower bound of the range. In fact, this is very unlikely since the lower bound of the range is calculated as minimum speed to pass the intersection during the phase, while the original algorithm calculates the speed as maximum possible speed.

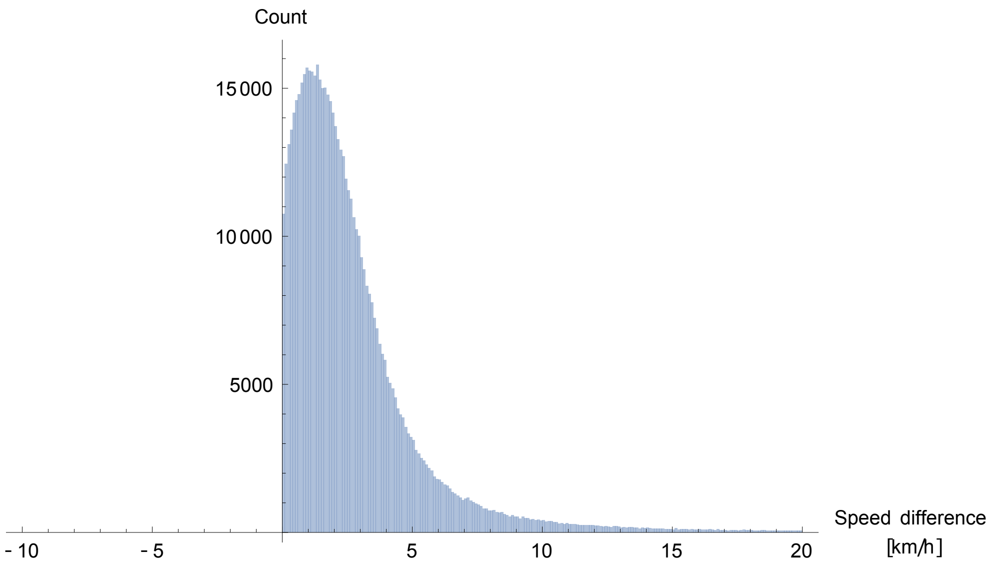


Figure 6.5: Histogram of Speed Differences

It can be seen in the histogram that the differing speeds recommended by the original algorithm are close to the corresponding upper range bounds calculated by the new algorithm. Point estimate of expected value of the difference is 2.9 *km/h* and standard deviation 3.4 *km/h*. Nevertheless, there is a non-negligible amount of results for which the difference is more than 10 km/h. In such cases, both recommendations can clearly be considered as significantly different and most likely would lead to different outcomes for the driver.

Conclusion

First part of the thesis introduced car-to-infrastructure (C2I) networks, an important wireless communication scheme used within *Intelligent Transportation Systems*. Main focus was given to *Green Light Optimal Speed Advisory* (GLOSA), one of the main applications of C2I networks. It helps drivers to avoid any unnecessary stops at traffic intersections and thus it increases an overall traffic flow continuity. *GreenLight* is a mobile application providing a driver-side part of GLOSA. When the car is approaching a GLOSA-equipped intersection, the application displays the current state of the traffic light signal and a recommended approaching speed. Furthermore, *GreenLight* can be integrated in a car infotainment system using the *MirrorLink*^{TM} technology.

Security is an important aspect of any technical system involving wireless communication. This is especially true for intelligent transportation systems where successful attacks may have serious consequences for all traffic participants. The analysis of possible security threats presented ten general threats for C2I-based GLOSA systems and six more specific threats for the *GreenLight* application. The threats were categorized and evaluated using the *STRIDE* and *DREAD* methods. On top of this analysis, several suitable security measures for the *GreenLight* application were identified and will be considered for implementation.

Inability to determine direction in which the driver intends to go through an intersection is considered to be a current drawback of *GreenLight*. The application is unable to show the proper light signal and considers only the straight direction. The thesis proposed and discussed six promising methods to determine the intended direction. A method following states of turn indicators currently seems to be the best choice for implementation. In addition, using car navigation to follow the planned route and also using GPS data to determine the current driving lane were evaluated as reasonable choices as well. Most of the remaining methods were recommended to be postponed until the time when the required technology, such as road marking recognition, is widely available. First method is planned to be implemented and deployed by the end of 2016.

Previous version of the *GreenLight* application contained a very simple algorithm for calculation of the recommended speed to be displayed to the driver. It was based on the $v = s/t$ velocity equation. A new enhanced algorithm was designed, implemented and integrated in the application. Besides the distance and remaining time, it takes into account also the needed (de)acceleration and reaction delay of the driver. Furthermore, it recommends a speed range instead of a single speed. This is especially beneficial when the car is approaching intersection during a red light phase and thus both minimum and maximum speeds can be important.

The enhanced algorithm was successfully tested in real traffic. Although the testing was limited by the available resources currently allocated in the project, the updated application worked exactly as expected in all the tests. Moreover, when the driver followed the recommended speed, they always passed the intersection as advised. It can be concluded that the new algorithm provides better results than the original one and hence it will remain as the main speed calculation algorithm in the official *GreenLight* development branch.

Besides the limited real-traffic testing, a simulated comparison with the original algorithm was performed. Both algorithms were queried with 2,000,000 randomly generated input vectors in order to compare and statistically evaluate the resulting speeds. Difference between the algorithms was especially significant in test where a red light phase was simulated and the speed was calculated for the next green phase. In this scenario, speed recommended by the original algorithm did not comply with the new algorithm in 61.4% cases, i.e. speed recommended by the original algorithm was not within the range calculated by the new algorithm.

After all, I consider this thesis project successful and contributive to the area of intelligent transportation systems. I believe the new enhanced algorithm is the most important outcome and will significantly improve performance of the *GreenLight* application.

Bibliography

[1] ETSI. *Intelligent Transport Systems* [online]. © 2016, [cit. 2016-03-20]. Available from: `http://www.etsi.org/technologies-clusters/technologies/intelligent-transport`

[2] SAE J3016. *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems.* January 16th 2014. Available from: `http://standards.sae.org/j3016_201401/`

[3] ADAPTIVE CONSORTIUM. *Deliverable D2.1: System Classification and Glossary, v1.2.* February 6th 2015, [cit. 2016-05-23]. Available from: `https://www.adaptive-ip.eu/index.php/deliverables_papers.html?file=files/adaptive/content/downloads/Deliverables%20%26%20papers/AdaptIVe-SP2-v12-DL-D2.1%20System%20Classification.pdf`

[4] BMW GROUP. *BMW Innovations at the 2015 Consumer Electronics Show (CES) in Las Vegas. 360-degree collision avoidance and fully-automated parking in multi-storey car parks* [online]. December 15th 2014, [cit. 2016-05-23]. Available from: `https://www.press.bmwgroup.com/global/article/detail/T0198231EN/bmw-innovations-at-the-2015-consumer-electronics-show-ces-in-las-vegas-360-degree-collision-avoidance`

[5] ETSI. *Cooperative ITS* [online]. © 2016, [cit. 2016-03-20]. Available from: `http://www.etsi.org/technologies-clusters/technologies/intelligent-transport/cooperative-its`

[6] CAR 2 CAR COMMUNICATION CONSORTIUM. *Green Light Optimal Speed Advisory* [online]. © 2016, [cit. 2016-03-20]. Available from: `https://www.car-2-car.org/index.php?id=197`

[7] GREENBERG, Andy. Hackers Remotely Kill a Jeep on the Highway. In: *Wired.com* [online], July 21st 2015, [cit. 2016-03-20]. Available from: `http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/`

[8] FRODIGH M., P. JOHANSSON and P. LASRSSON. *Wireless ad hoc networking – The art of networking without a network.* 2000, [cit. 2016-04-07]. Available from: `http://www.ericsson.com/ericsson/corpinfo/publications/review/2000_04/files/2000046.pdf`

[9] DRIVE C2X. *Functions* [online]. © 2016, [cit. 2016-04-07]. Available from: `http://www.drive-c2x.eu/use-cases`

[10] AMSTERDAM GROUP. *Roadmap between automotive industry and infrastructure organisations on initial deployment of Cooperative ITS in Europe, v1.0.* June 7th 2013. Available from: `https://amsterdamgroup.mett.nl/Downloads/downloads_getfilem.aspx?id=492324`

[11] ETSI EN 302 665. *Intelligent Transport Systems (ITS); Communications Architecture*, v1.1.1. September 2010. Available from: `http://www.etsi.org/deliver/etsi_en/302600_302699/302665/01.01.01_60/en_302665v010101p.pdf`

[12] ETSI EN 302 663. *Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*, v1.2.1. July 2013. Available from: `http://www.etsi.org/deliver/etsi_en/302600_302699/302663/01.02.01_60/en_302663v010201p.pdf`

[13] SHAGDAR, Oyunchimeg and Thierry ERNST. *ITS Standardization.* May 29th 2013, [cit. 2016-06-25]. Available from: `http://bentley.u-bourgogne.fr/jnct2013/proceedings/ITS-Standards-IJCT.pdf`

[14] IEEE 802.11p-2010. *IEEE Standard for Information technology – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments.* July 15th 2010. Available from: `http://standards.ieee.org/findstds/standard/802.11p-2010.html`

[15] NAGEL, Carsten. *Design and Failure Mode and Effects Analysis (FMEA) of a Vehicular Speed Advisory System.* Master's thesis, Hamburg University of Technology, Institute of Communication Networks, Hamburg, Germany, 2015. Available from: `http://lsbg.hamburg.de/contentblob/4575458/data/smart-city-projekte-masterarbeit-carsten-nagel.pdf`

[16] STIBOR L., Y. ZANG and H. REUMERMAN. Evaluation of communication distance of broadcast messages in a vehicular ad-hoc network using IEEE 802.11p. In: *IEEE Wireless Communications and Networking Conference*, March 2007. Available from: `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4224298`

[17] ETSI EN 302 636-1. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1: Requirements*, v1.2.1. April 2014. Available from: `http://www.etsi.org/deliver/etsi_en/302600_302699/30263601/01.02.01_60/en_30263601v010201p.pdf`

[18] ETSI EN 302 636-5-1. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol*, v1.2.1. August 2014. Available from: `http://www.etsi.org/deliver/etsi_en/302600_302699/3026360501/01.02.01_60/en_3026360501v010201p.pdf`

[19] ETSI TS 102 637-1. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 1: Functional Requirements*, v1.1.1. September 2010. Available from: `http://www.etsi.org/deliver/etsi_ts/102600_102699/10263701/01.01.01_60/ts_10263701v010101p.pdf`

[20] ETSI EN 302 637-2. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, v1.3.2. November 2014. Available from: `http://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.02_60/en_30263702v010302p.pdf`

[21] ETSI EN 302 637-3. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*, v1.2.2. November 2014. Available from: `http://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.02.02_60/en_30263703v010202p.pdf`

[22] ETSI EN 302 895. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM)*, v1.1.1. September 2014. Available from: `http://www.etsi.org/deliver/etsi_en/302800_302899/302895/01.01.01_60/en_302895v010101p.pdf`

[23] ETSI TS 102 723-2. *Intelligent Transport Systems (ITS); OSI cross-layer topics; Part 2: Management information base*, v1.1.1. November 2012. Available from: `http://www.etsi.org/deliver/etsi_ts/102700_102799/10272302/01.01.01_60/ts_10272302v010101p.pdf`

[24] ETSI TS 103 175. *Intelligent Transport Systems (ITS); Cross Layer DCC Management Entity for operation in the ITS G5A and ITS G5B medium*, v1.1.1. June 2015. Available from: `http://www.etsi.org/deliver/etsi_ts/103100_103199/103175/01.01.01_60/ts_103175v010101p.pdf`

[25] ETSI TS 102 940. *Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management*, v1.1.1. June 2012. Available from: `http://www.etsi.org/deliver/etsi_ts/102900_102999/102940/01.01.01_60/ts_102940v010101p.pdf`

[26] ETSI TS 102 731. *Intelligent Transport Systems (ITS); Security; Security Services and Architecture*, v1.1.1. September 2010. Available from: `http://www.etsi.org/deliver/etsi_ts/102700_102799/102731/01.01.01_60/ts_102731v010101p.pdf`

[27] NORTH CAROLINA DEPARTMENT OF TRANSPORTATION. *Drive green, save green* [online]. [cit. 2016-04-07]. Available from: `http://www.ncdot.gov/travel/drivegreen/`

[28] SAE J2735. *Dedicated Short Range Communications (DSRC) Message Set Dictionary.* March 30th 2016. Available from: `http://standards.sae.org/j2735_201603/`

[29] ISO/AWI TS 19091. *Intelligent transport systems – Cooperative ITS – Using V2I and I2V communications for applications related to signalized intersections* [under development]. [cit. 2016-04-07]. Available from: `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=69897`

[30] SAE INTERNATIONAL. *Dedicated Short Range Communications (DSRC): European Union Region Support Page* [online]. © 2016, [cit. 2016-04-07]. Available from: `http://www.sae.org/standardsdev/dsrc/eu`

[31] SWARCO, WEINGART J. Standardisation of SPaT and MAP. In: *Amsterdam INTERTRAFFIC*, 2014, [cit. 2016-04-07]. Available from: `https://www.collaborative-team.eu/downloads/get/SWARCO%20standardisation%20activities%20in%20cooperative%20systems.pdf`

[32] ABUELSAMID, Sam. Audi demonstrating Travolution vehicle to infrastructure communications. In: *autoblog.com* [online], June 3rd 2010, [cit. 2016-03-20]. Available from: `http://www.autoblog.com/2010/06/03/audi-demonstrating-travolution-vehicle-to-infrastructure-communi/`

[33] COMPASS4D C/O ERTICO - ITS EUROPE. *Compass4D: Verona* [online]. [cit. 2016-04-07]. Available from: `http://www.compass4d.eu/en/cities_02/verona/verona.htm`

[34] SOUKUP, Pavel and Petr KUMPOŠT. Kooperativní systémy optimalizace rychlosti vozidel na světelných křižovatkách. In: *Young Engineers Transportation Conference*, September 30th 2015. Available from: `http://docplayer.cz/6387950-Kooperativni-systemy-optimalizace-rychlosti-vozidel-na-svetelnych-krizovatkach.html`

[35] INTENS Corporation s.r.o. [cit. 2016-04-24]. Available from: `http://www.intens.cz/en/index.html`

[36] e4t electronics for transportation s.r.o. [cit. 2016-04-24]. Available from: `http://www.e4t.cz/default.aspx?lang=en-US`

[37] SKODA AUTO. *SmartGate* [online]. [cit. 2016-04-24]. Available from: `http://www.skoda-auto.com/en/experience/product-features/connectivity/#TilesWebPart`

[38] COMMSIGNIA LTD. *V2X Hardware* [online]. © 2016, [cit. 2016-05-24]. Available from: `http://www.commsignia.com/hardware/`

[39] KAPSCH TRAFFICCOM. *Kapsch V2X Cooperative Systems.* [cit. 2016-04-07]. Available from: `https://www.kapsch.net/ktc/downloads/brochures/Kapsch-KTC-BR-V2X-EN?lang=en-US`

[40] CAR CONNECTIVITY CONSORTIUM. *MirrorLink Members List* [online]. [cit. 2016-04-24]. Available from: `http://mirrorlink.com/about%20ccc/members`

[41] CAR CONNECTIVITY CONSORTIUM, PICHON E. *Certifying Applications for MirrorLink.* [cit. 2016-04-24]. Available from: `http://www.carconnectivity.org/public/files/files/Ed-Slides.pptx`

[42] CAR CONNECTIVITY CONSORTIUM, BRAKENSIEK J. *Technical Considerations of Application Certification.* [cit. 2016-04-24]. Available from: `http://carconnectivity.org/public/files/files/CCC-Summit-Krakow-TechnicalConsiderationOfAppCertification-Jorg.pptx`

[43] CAR CONNECTIVITY CONSORTIUM, BRAKENSIEK J. *MirrorLink In a Technical Nutshell: Protocol, Interoperability and Roadmap.* 2012, [cit. 2016-04-24]. Available from: `http://carconnectivity.org/public/files/files/Summit_Template_Jorg_Brakensiek_English.pdf`

[44] CAR CONNECTIVITY CONSORTIUM, PICHON E. *MirrorLink API Overview*. May 18th 2015, [cit. 2016-04-24]. Available from: http://www.slideshare.net/bemyapp/mirrorlink-hackathon-workshop-presentation-by-ed-pichon

[45] CCC-TS-038. *MirrorLink Common API, v1.1.8*. March 24th 2016.

[46] RFC 2828. *Internet Security Glossary*. May 2000. Available from: https://tools.ietf.org/html/rfc2828

[47] MICROSOFT CORPORATION. *The STRIDE Threat Model* [online]. © 2005, [cit. 2016-06-16]. Available from: https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx

[48] MICROSOFT CORPORATION. *Improving Web Application Security: Threats and Countermeasures: Chapter 3 – Threat Modeling* [online]. June 2003, [cit. 2016-06-16]. Available from: https://msdn.microsoft.com/en-us/library/ff648644.aspx

[49] HAATAJA, Keijo and Pekka TOIVANEN. *Two Practical Man-In-The-Middle Attacks on Bluetooth Secure Simple Pairing and Countermeasures*. January 8th 2010, [cit. 2016-06-16]. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5374082

[50] LINK, Rainer. *Is Your Car Broadcasting Too Much Information?* [online]. July 28th 2015, [cit. 2016-06-17]. Available from: http://blog.trendmicro.com/trendlabs-security-intelligence/is-your-car-broadcasting-too-much-information/

[51] IEEE 802.11i-2004. *IEEE Standard for information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements*. July 23rd 2004. Available from: http://standards.ieee.org/findstds/standard/802.11i-2004.html

[52] GOOGLE INC. *Google Street View* [software][online]. 2007, [cit. 2016-05-03]. Available from: https://www.google.com/intl/en_ALL/maps/streetview/

[53] GPS SPS PS. *Global Positioning System, Standard Positioning Service, Performance Standard, 4th Edition*. September 2008. Available from: http://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf

[54] EUROPEAN GNSS AGENCY. *EGNOS Performance Overview* [online]. [cit. 2016-05-03]. Available from: `http://www.egnos-portal.eu/discover-egnos/services/performance-overview`

[55] GPS WAAS PS. *Global Positioning System, Wide Area Augmentation System (WAAS), Performance Standard, 1st Edition.* October 2008. Available from: `http://www.gps.gov/technical/ps/2008-WAAS-performance-standard.pdf`

[56] JOYCE, Ken. *How accurate is your smartphone?* [online]. March 27th 2015, [cit. 2016-05-03]. Available from: `https://www.linkedin.com/pulse/how-accurate-your-smartphone-ken-joyce`

[57] ZANDBERGEN, Paul and Sean BARBEAU. Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones. In: *The Journal of Navigation, vol. 64*, July 2011. Available from: `http://journals.cambridge.org/repo_A82eaJIy`

[58] ZANDBERGEN, Paul. Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning. In: *Transactions in GIS, vol. 13*, June 26th 2009. Available from: `http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9671.2009.01152.x/full`

[59] WEGMAN, Fred and Marinus SLOP. *Safety Effects of Road Design Standards in Europe.* Available from: `http://onlinepubs.trb.org/onlinepubs/circulars/ec003/ch39.pdf`

[60] SKODA AUTO. *Lane Assist* [online]. [cit. 2016-05-03]. Available from: `http://www.skoda-auto.com/en/models/new-octavia/assistants/#ExtendableModuleWebPart_1_1_1`

[61] XAMARIN INC. *Xamarin Platform* [online]. © 2016, [cit. 2016-06-21]. Available from: `https://www.xamarin.com/platform`

Appendix A

Acronyms

3G Third Generation of Mobile Telecommunications Technology

4G Fourth Generation of Mobile Telecommunications Technology

A-GPS Assisted Global Positioning System

ADAS Advanced Driver Assistance Systems

AG Aktiengesellschaft

API Application Programming Interface

.apk Android Application Package (file format)

App Application

ARIB Association of Radio Industries and Businesses

BMW Bayerische Motoren Werke AG

BSA Basic Set of Applications

BSS Basic Service Set

BTP Basic Transport Protocol

C-ITS Cooperative Intelligent Transportation System

C2C Car-to-Car

C2I Car-to-Infrastructure

C2X Car-to-X (Car-to-Anything)

CA Certification Authority

CAM Co-Operative Awareness Message

CAN Controller Area Network

CCC Car Connectivity Consortium

CD Compact Disc

CEN European Committee for Standardization

CO2 Carbon Dioxide

Compass4D Cooperative Mobility Pilot on Safety and Sustainability Services for Deployment

CTU Czech Technical University in Prague

DCC Decentralized Congestion Control

DEN Decentralized Environmental Notification

DREAD Damage, Reproducibility, Exploitability, Affected Users, Discoverability

DVD Digital Video Disc

e4t electronics for transportation

EGNOS European Geostationary Navigation Overlay Service

Err Error

ETSI European Telecommunications Standards Institute

EU European Union

FA Interface between Facility Layer and ITS-S applications

FIT Faculty of Information Technology (CTU Prague)

GHz GigaHertz

GIS Geographic Information System

GLOSA Green Light Optimized Speed Advisory

GmbH Gesellschaft mit Beschränkter Haftung

GPS Global Positioning System

GUI Graphical User Interface

h Hour

HMI	Human Machine Interface
HUD	Head-unit Display
Hz	Hertz
IEEE	Institute of Electrical and Electronics Engineers
ID	Identifier
IN	Interface between Access Layer and Networking & Transport Layer
iOS	iPhone Operating System
IP	Internet Protocol
IPv4	Internet Protocol version 4
ISO	International Organization for Standardization
ISM	Industrial, Scientific and Medical Radio Bands
ITS	Intelligent Transportation System
ITU-R	Radiocommunication Sector of International Telecommunication Union
kHz	Kilohertz
km	Kilometer
L1-7	Layer 1-7
LDM	Local Dynamic Map
LTE	Long-Term Evolution
MA	Interface between Management Entity and ITS-S applications
MAC	Media Access Control
MAP	Map Message
MF	Interface between Management Entity and Facility Layer
MI	Interface between Management Entity and Access Layer
MIB	Management Information Base
MiM	Man-in-the-middle Attack
MN	Interface between Management Entity and Networking & Transport Layer

MS Interface between Management Entity and Security Entity

N/A Not applicable

NF Interface between Networking & Transport Layer and Facility Layer

OBU On-Board Unit

OSI Open Systems Interconnection

PKI Public Key Infrastructure

Q1-4 Year Quarter 1-4

QoS Quality of Service

RFC Request for Comments

RSU Roadside Unit

RTP Real-time Transport Protocol

SA Interface between Security Entity and ITS-S applications

SAE Society of Automotive Engineers

SF Interface between Security Entity and Facility Layer

SI Interface between Security Entity Access Layer

SN Interface between Security Entity and Networking & Transport Layer

SNMP Simple Network Management Protocol

SPaT Signal Phase and Timing Message

SSL Secure Sockets Layer

STRIDE Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege

TCP Transmission Control Protocol

TM Trademark

TS Technical Standard

UDP User Datagram Protocol

UML Unified Modeling Language

UPnP Universal Plug and Play

UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
USA	United States of America
VANET	Vehicular Ad Hoc Network
VNC	Virtual Network Computing
W/E	Weight/Effort Ratio
w/o	Without
WAAS	Wide Area Augmentation System
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access II
XML	Extensible Markup Language

Appendix B

Contents of enclosed CD

```
Readme.txt ............................ Brief summary of CD content
Thesis
├── DP_Beran_Jan_2016.pdf ....................... This thesis in PDF
└── LaTeX ............................ LaTeX source files of this thesis
    ├── DP_Beran_Jan_2016.tex ............ Text of the thesis in LaTeX
    ├── ref.bib ..................................... Bibliography file
    ├── FITthesis.cls ................... Thesis template of FIT CTU
    ├── iso690.bst ..................... Bibliography format definition
    ├── cvut-logo-bw.pdf ....................... Logo of CTU Prague
    ├── Assignment_Beran_Jan.pdf ................. Thesis Assignment
    ├── images ......................... Images needed for compilation
    │   └── ...
    └── resources .............. Other resources needed for compilation
        └── ...
Source
└── SampleExternalRecommendedSpeedCalculator.cs
    ........................ C# source file of the new speed algorithm
Mathematica ....................... Wolfram Mathematica notebooks
├── Advised_Speed_Calculation.nb ... Recommended speed calculation
└── Speed_Algorithms_Comparison.nb
    ..................... Statistical evaluation of the speed algorithms
Videos
├── GreenLight_Testing_PRG.mp4
│   ................. Moments from testing session of the new algorithm
└── GreenLight_Old_HK.mp4
    ............... Promotional video of original GreenLight application
```

Appendix C

Wolfram Mathematica Notebooks

C.1 Advised Speed Calculation

GLOSA - Advised Speed Calculation

This Wolfram Mathematica notebook describes a method (algorithm) for calculating an advised driving speed to be used within a GLOSA system. This is part of the "Car-to-Infrastructure Communication in the Context of Intelligent Traffic Intersections" Diploma thesis (Jan Beran, FIT CTU, Prague 2016).

Speed Expression

```
In[1]:= (* Delete all symbols and set global options *)
        ClearAll["Global`*"];
        SetOptions[Manipulator, Appearance → "Labeled"];
```

Overview
The simplest possible method for calculating an advised speed is to use the basic velocity equation $v = s/t$. However, several important factors would be ignored: car initial speed, time/distance needed for (de)acceleration and driver's reaction delay. The calculation described in this notebook takes all these factors into account. This is the same calculation as the calculation described in chapter 5 of the thesis, but this notebook presents a strictly theoretical approach not applied to any concrete GLOSA system or application. Therefore, the explanation might be slightly different in both documents.

Inputs
s: Distance to the intersection [m].
t: Time to a light signal change [s], e.g. green to yellow (to red).
v0: Initial car travel speed $[m.s^{-1}]$, assuming the car is not (de)accelerating at the beginning.
tReact: driver's reaction delay [s], i.e. time when the car starts to (de)accelerate after the advised speed is displayed.
a: Acceleration speed $[m.s^{-2}]$, i.e: how fast the car's speed changes when (de)accelerating (constant value for simplification, same for both (de)acceleration).
SpeedLimit: Allowed speed limit $[m.s^{-1}]$, advised speed cannot be greater than this limit.

Output
vAdv: Advised driving speed $[m.s^{-1}]$, might be further converted to km.h^{-1} and displayed to the driver.

Notes
This *Mathematica* notebook has been created and tested under Wolfram *Mathematica* 10.0.1.0. It is recomended to evaluate entire notebook at once (Evaluation -> Evaluate Notebook). In case of any problems related to this notebook, please contact Jan Beran (beranj29@fit.cvut.cz).

Let's start with the basic distance equation (*distEq*). Total distance to the intersection (*s*) is a sum of the following sub-distances:
sReact: distance travelled during driver's reaction time [m]
sAcc: distance travelled during (de)acceleration to the resulting (advised) speed [m]
sRest: remaining distance to the intersection [m]

```
In[3]:= distEq = s == sReact + sAcc + sRest;
```

Next, we expand *distEq* with the well-know velocity/acceleration formulas:
distance formula: $s = v * t$
acceleration distance: $s = v_0 * t + a * t^2/2$
acceleration speed: $\Delta v = \Delta t * a$, $\Delta v = v - v_0$ => $v = v_0 + a * t$

There is also one more variable used:
tAcc: time needed for (de)acceleration to the final (advised) speed

```
In[4]:= sReact = v0 * tReact;
        sAcc = (v0 * tAcc) + a * tAcc^2 / 2;
        sRest = (v0 + tAcc * a) * (t - tReact - tAcc); (* sRest=vAdv*tRest, vAdv=v0+tAcc*a *)
```

This gives us a single equation with one unknown variable (*tAcc*):

```
In[7]:= Simplify[distEq]
Out[7]= 2 s + a tAcc (-2 t + tAcc + 2 tReact) == 2 t v0
```

All the equations presented above are valid in case the car is accelerating. Fortunately, the same equations might also be used for the deceleration case, but *a* must be inverted ($a := a * (-1)$). Thus, the calculation is divided in two cases solved separately. Whether the car is (de)accelerating is determined as $s / v0 > t$. I.e. Remaining time is compared with the needed travel time (to the intersection) at initial speed. If the needed travel time is greater than the remaining time, the car needs to accelerate (it would pass the intersection after the signal changes otherwise).

We solve the equation for *tAcc,* with additional conditions that $tAcc \geq 0$ and $(tAcc + tReact) \leq t$. Without these conditions, we could obtain more solutions satisfying *distEq,* but only one (or zero) would be correct anyway. Incorrect solutions are formally eliminated by these conditions.

To simplify the solution, we add a list of assumptions such as all the times, distances and speeds are > 0.

```
In[8]:= tAccRuleAcc = Simplify[Solve[{distEq, (tAcc + tReact) ≤ t, tAcc ≥ 0}, tAcc, Reals],
          {a > 0, s > 0, t > 0, v0 ≥ 0, tReact ≥ 0, s / v0 > t}]
        tAccRuleDec = Simplify[Solve[{distEq, (tAcc + tReact) ≤ t, tAcc ≥ 0}, tAcc, Reals],
          {a < 0, s > 0, t > 0, v0 ≥ 0, tReact ≥ 0, s / v0 < t}]
```

Out[8]= $\left\{\left\{\text{tAcc} \to \text{ConditionalExpression}\left[t - \text{tReact} - \sqrt{(t - \text{tReact})^2 - \frac{2 (s - t\, v0)}{a}},\ t > \text{tReact} \,\&\&\, a > \frac{2 (s - t\, v0)}{(t - \text{tReact})^2}\right]\right\}\right\}$

Out[9]= $\left\{\left\{\text{tAcc} \to \text{ConditionalExpression}\left[t - \text{tReact} - \sqrt{(t - \text{tReact})^2 - \frac{2 (s - t\, v0)}{a}},\ t > \text{tReact} \,\&\&\, a < \frac{2 (s - t\, v0)}{(t - \text{tReact})^2}\right]\right\}\right\}$

The resulting expression is the same for both (de)acceleration cases. However, the conditions (for a valid solution to exist) are slightly different. Since we have an expression for the (de)acceleration time (*tAcc*), we can calculate the advised speed (*vAdv*) easily: $vAdv = v0 + tAcc * a$

We also need to invert *a* in the deceleration condition since negative *a* is assumed in the condition expression, but we will always get a positive *a* as an input.

```
In[10]:= vAdvExpr = tAccRuleAcc[[1, 1, 2, 1]] * a + v0
         tAccCondAcc = tAccRuleAcc[[1, 1, 2, 2]]
         tAccCondDec = tAccRuleDec[[1, 1, 2, 2]] /. a → -a
```

Out[10]= $v0 + a\left(t - tReact - \sqrt{(t - tReact)^2 - \frac{2\,(s - t\,v0)}{a}}\right)$

Out[11]= $t > tReact \,\&\&\, a > \frac{2\,(s - t\,v0)}{(t - tReact)^2}$

Out[12]= $t > tReact \,\&\&\, -a < \frac{2\,(s - t\,v0)}{(t - tReact)^2}$

Now we can just define the advised speed function using the previously computed expression and conditions. For the deceleration case, we add a condition that the resulting speed must be > 0 (which might not be satisfied in some cases)

```
In[13]:= getAdvSpeedIn[s_, t_, v0_, tReact_, a_] := Evaluate[vAdvExpr];
         getAdvSpeed[s_, t_, v0_, tReact_, a_] := Which[
           s / v0 == t, v0,
           s / v0 > t, If[t > tReact && a > 2 (s - t v0) / (t - tReact)^2, getAdvSpeedIn[s, t, v0, tReact, a], -1],
           s / v0 < t, If[t > tReact && -a < 2 (s - t v0) / (t - tReact)^2 && (getAdvSpeedIn[s, t, v0, tReact, -a] > 0),
            getAdvSpeedIn[s, t, v0, tReact, -a], -2]
         ]
```

Demonstration - Interactive Graphs

In this section, possible usage of the speed expression (determined in the previous section) is demonstrated on two interactive graphs. There are two basic cases that can occur in a real traffic:

1. Light signal is green (or yellow before green): In this case, we calculate a minimum speed to pass the intersection within the current green phase. We neglect the "yellow before green" phase and consider it to be part of the green phase. Calculated advised speed might be limited by an allowed speed limit (if available).
2. Light signal is red (or yellow before red): In this case, we calculate a speed range - a maximum speed to pass the intersection after the signal changes to green and a minimum speed to pass during the next green phase (coming after the current red). "Yellow before red" phase is considered to be part of the red phase.

Note: The algorithm described in chapter 5 of the thesis is more general and considers up to 15 future green phases in a single calculation. This is neglected in this notebook in order to keep the demostration as simple and understandable as possible.

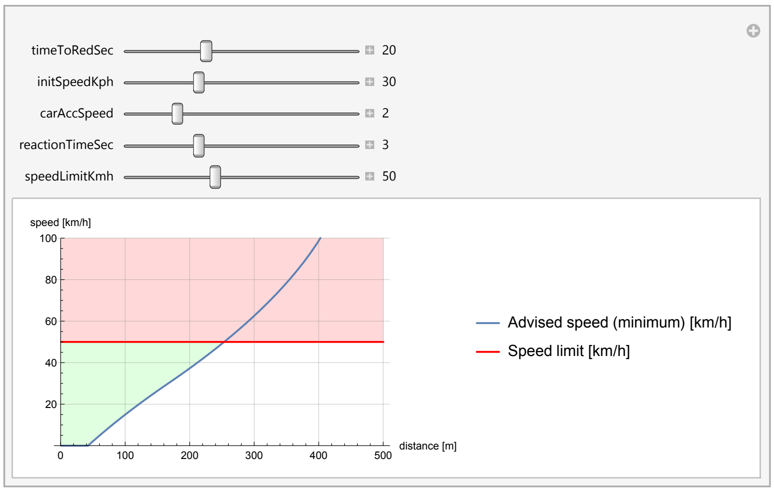
All the *Mathematica* code below is just to create reasonable interactive graphs. Since the graphs themselves should be self-explanatory, there is no need for further comments. The manipulable parameters are the following:
timeToRedSec / timeToGreenSec / timeToNextRedSec: Remaining time to a corresponding signal change [s]
initSpeedKph: Initial car speed [km.h^{-1}]
carAccSpeed: Car average acceleration speed [$m.s^{-2}$]
reactionTimeSec: driver's reaction delay [s]
speedLimitKmh: Allowed speed limit [km.h^{-1}]

1. Signal is green (or yellow before green)

```
In[15]:= plotYrange = 100;
         getAdvSpeedGreen[s_, t_, v0_, tReact_, a_] := Switch[getAdvSpeed[s, t, v0, tReact, a],
           -1, plotYrange,
           -2, 0,
           _, getAdvSpeed[s, t, v0, tReact, a]
          ]
```

```
In[17]:= (* calculating speed to pass the intersection on (current) green *)
         Manipulate[
          Plot[
           {getAdvSpeedGreen[distance, timeToRedSec, initSpeedKph / 3.6, reactionTimeSec, carAccSpeed] * 3.6,
            speedLimitKmh},
           {distance, 1, 500},
           AxesLabel → {"distance [m]", "speed [km/h]"},
           PlotRange → {0, plotYrange},
           GridLines → Automatic,
           Method → {"GridLinesInFront" → True},
           PlotStyle → {{}, Red},
           PlotLegends → {"Advised speed (minimum) [km/h]", "Speed limit [km/h]"},
           Filling → {1 → {{2}, {LightGreen, None}}, 2 → {Top, LightRed}}
          ],
          {{timeToRedSec, 20}, 0.001, 60},
          {{initSpeedKph, 30}, 0.001, 100},
          {{carAccSpeed, 2}, 0.001, 10},
          {{reactionTimeSec, 3}, 0.001, 10},
          {{speedLimitKmh, 50}, 20, plotYrange}
         ]
```

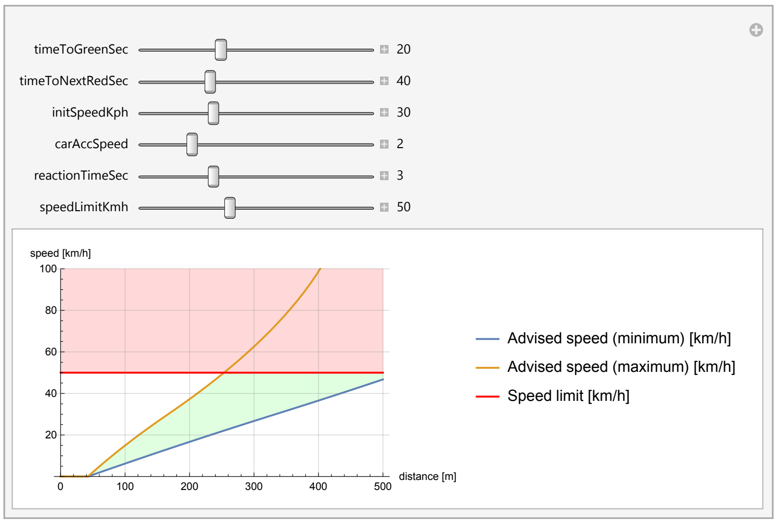
Out[17]=



```
timeToRedSec      20
initSpeedKph      30
carAccSpeed       2
reactionTimeSec   3
speedLimitKmh     50
```

2. Signal is red (or yellow before red)

```
In[18]:= plotYrange = 100;
         getAdvSpeedRed[s_?NumericQ, t_?NumericQ, v0_?NumericQ, tReact_?NumericQ, a_?NumericQ] :=
          Switch[getAdvSpeed[s, t, v0, tReact, a],
           -1, plotYrange,
           -2, 0,
           _, getAdvSpeed[s, t, v0, tReact, a]
          ]

In[20]:= (* calculating speed range to pass the intersection on the next green *)
         Manipulate[
          Plot[
           {getAdvSpeedRed[distance, timeToNextRedSec,
              initSpeedKph / 3.6, reactionTimeSec, carAccSpeed] * 3.6,
            getAdvSpeedRed[distance, timeToGreenSec, initSpeedKph / 3.6, reactionTimeSec, carAccSpeed] * 3.6,
            speedLimitKmh,
            Min[getAdvSpeedRed[distance, timeToGreenSec,
               initSpeedKph / 3.6, reactionTimeSec, carAccSpeed] * 3.6, speedLimitKmh]},
           {distance, 0, 500},
           AxesLabel → {"distance [m]", "speed [km/h]"},
           PlotRange → {0, plotYrange},
           GridLines → Automatic,
           Method → {"GridLinesInFront" → True},
           PlotStyle → {{}, {}, Red, None},
           Filling → {1 → {{4}, {LightGreen, None}}, 3 → {Top, LightRed}},
           PlotLegends →
            {"Advised speed (minimum) [km/h]", "Advised speed (maximum) [km/h]", "Speed limit [km/h]"}
          ],
          {{timeToGreenSec, 20}, 0.001, 60},
          {{timeToNextRedSec, 40}, timeToGreenSec, 90},
          {{initSpeedKph, 30}, 0.001, 100},
          {{carAccSpeed, 2}, 0.001, 10},
          {{reactionTimeSec, 3}, 0.001, 10},
          {{speedLimitKmh, 50}, 20, plotYrange}
         ]

Out[20]=
```



```
timeToGreenSec     20
timeToNextRedSec   40
initSpeedKph       30
carAccSpeed        2
reactionTimeSec    3
speedLimitKmh      50
```

C.2 Comparison of Speed Algorithms

GLOSA - Statistical Comparison of Advised Speed Algorithms

This Wolfram Mathematica notebook deals with a statistical comparison of two algorithms for calculating a recommended speed in a GLOSA system. This is part of the "Car-to-Infrastructure Communication in the Context of Intelligent Traffic Intersections" Diploma thesis (Jan Beran, FIT CTU, Prague 2016).

Overview
This document compares the newly proposed enhanced speed algorithm with the originally implemented simple algorithm. Please refer to chapter 5 of the thesis for a detailed information about both algorithms. Idea of this comparison is to generate a large set of random input vectors for both algorithms, each vector simulates a random car approaching the intersection. All these vectors are evaluated by both algorithms and the results are then compared. This should provide an approximate information about difference between the original and the newly implemented algorithm.

Notes
This *Mathematica* notebook has been created and tested under Wolfram *Mathematica* 10.0.1.0. It is recommended to evaluate entire notebook at once (Evaluation -> Evaluate Notebook). In case of any problems related to this notebook, please contact Jan Beran (beranj29@fit.cvut.cz).

Definition of the Speed Algorithms to Compare

```
In[1]:= (* Delete all previously defined symbols *)
        ClearAll["Global`*"];
```

Function *getAdvSpeed[]* represents the newly implemented speed algorithm. It has been copied from the *Advised_Speed_Calculation.nb.* Please refer to that notebook for a detailed description:

```
In[2]:= getAdvSpeedIn[s_, t_, v0_, tReact_, a_] := v0 + a (t - tReact - √((t - tReact)^2 - 2 (s - t v0)/a));

        getAdvSpeed[s_, t_, v0_, tReact_, a_] := Which[
           s / v0 == t, v0,
           s / v0 > t, If[t > tReact && a > 2 (s - t v0)/(t - tReact)^2,
            getAdvSpeedIn[s, t, v0, tReact, a], Infinity], (* acceleration *)
           s / v0 < t, If[t > tReact && -a < 2 (s - t v0)/(t - tReact)^2 && (getAdvSpeedIn[s, t, v0, tReact, -a] > 0),
            getAdvSpeedIn[s, t, v0, tReact, -a], 0] (* deceleration *)
          ];
```

Function *getAdvSpeedSimple[]* represents the originally implemented simple calculation based on the basic velocity equation:

```
In[4]:= getAdvSpeedSimple[s_, t_, v0_, tReact_, a_] := s / t; (* v = s/t *)
```

Input Parameters for the Algorithms

Values of the following parameters will be fixed for entire comparison.

carAcceleration $[m.s^{-2}]$: Approximate acceleration speed of the car.
driverReactTime [*s*]: Approximate reaction delay of the driver.
speedLimit $[km.h^{-1}]$: Regulatory allowed maximum speed.

```
In[5]:= carAcceleration = 5;
        driverReactTime = 3;
        speedLimit = 50;
```

Values of the following parameters will be randomly generated from predefined ranges of allowed values.

distMin, distMax [*m*]: Distance to the intersection
initSpeedMin, initSpeedMax $[m.s^{-1}]$: Initial speed of the car
timeGreenEndMin, timeGreenEndMax [*s*]: Time to the end of the current light phase. Only relevant when assuming that the current light phase is green.
timeNextGreenStartMin, timeNextGreenStartMax [*s*]: Time to the start of the next green phase. Only relevant when assuming that the current light phase is red.
nextGreenLengthMin, nextGreenLengthMax [s]: Length of the next green phase. Only relevant when assuming that the current light phase is red.

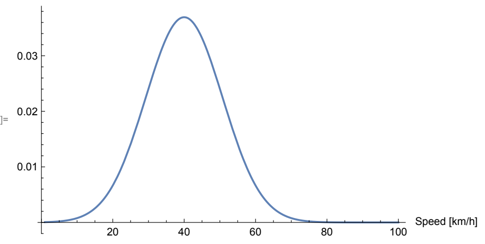
All the values will be generated as random numbers from corresponding uniform distributions *unif(<min_value>, <max_value>)*. Only the Initial speed will be generated from a normal distribution *norm(40 km/h, 10.8 km/h)*. This should provide a better approximation of speed distribution of approaching cars at the intersection (e.g. probability of 40km/h is usually higher than 5 km/h).

```
(* distance 1-500m *)
distMin = 1;
distMax = 500;

(* init speed: 1-100 km/h generated from norm(40 km/h, 10.8 km/h) *)
(* Values stored in m/s *)
initSpeedMin = 1 / 3.6;
initSpeedMax = 100 / 3.6;
initSpeedMean = 40 / 3.6;
initSpeedVar = 3;

(* current green end: 1-60s *)
timeGreenEndMin = 1;
timeGreenEndMax = 60;

(* next green start: 1-60s *)
timeNextGreenStartMin = 1;
timeNextGreenStartMax = 60;

(* next green length: 15-60s *)
nextGreenLengthMin = 15;
nextGreenLengthMax = 60;
```

The normal distribution for the initial speed values is displayed in the following plot of its probability density function (speeds are displayed in km/h):

In[20]:=
```
Plot[PDF[NormalDistribution[initSpeedMean * 3.6, initSpeedVar * 3.6], x],
 {x, 1, 100}, AxesLabel → {"Speed [km/h]"}]
```

Out[20]=



For a basic comparison, it should be enough to generate 10 000 random input vectors . This number can be decreased to achieve a faster calculation with less resources or increased to achieve a better precision.

In[21]:=
```
sampleCnt = 10 000;
```

Case 1 - Signal is Green

Data Preparation

First, let's generate a set of *sampleCnt* random input vectors. The same set will be used by both algorithms.

In[22]:=
```
getRandomInputs[count_] := Table[
   {RandomReal[{distMin, distMax}],
    RandomReal[{timeGreenEndMin, timeGreenEndMax}],
    (While [tmp = RandomVariate[NormalDistribution[initSpeedMean, initSpeedVar]];
      tmp < initSpeedMin || tmp > initSpeedMax]; tmp),
    driverReactTime,
    carAcceleration},
   {count}
  ];
randData = getRandomInputs[sampleCnt];
```

Now we can use the generated data as input vectors for the speed algorithms. The obtained resulting speeds are converted to km/h:

In[24]:=
```
advSpeedsNewAlg = Apply[getAdvSpeed, randData, {1}] * 3.6;
advSpeedsOldAlg = Apply[getAdvSpeedSimple, randData, {1}] * 3.6;
```

We obtained two lists of calculated speeds to reach the intersection at the end of the current green phase. These are, however, "raw" values that do not consider the speed limit. Both algorithms differ in the way they handle the speed limit. For a detailed explanation, please refer to chapter 5 of the thesis. Nevertheless, the main difference between both algorithms is, in our case, the following:

Original algorithm:
If the calculated speed is not higher than the speed limit, then the final recommended speed is set to the speed limit (preference of the highest possible speed). Otherwise, it is indicated that reaching the intersection during the current phase is not possible.

New algorithm:
If the calculated speed is not higher than the speed limit, then a speed range from the calculated speed to the speed limit is recommended. Otherwise, it is indicated that reaching the intersection during the current phase is not possible.

The “raw” calculated speeds need to be transformed to actual recommended speeds in accordance with the considered algorithms as described above. This is easily achieved with the following two replace rules:

applySpeedLimit1: All the speeds higher than the speed limit are replaced by *Infinity*. This is a special value reserved for cases when it is not possible to pass the intersection during the considered phase. This rule is applied to both lists.

applySpeedLimit2: All the speeds not higher than the speed limit are replaced by the speed limit. This rule is applied only to the list corresponding to the original algorithm.

After applying these replace rules, *advSpeedsOldAlg* holds the speeds recommended by the original algorithm and *advSpeedsNewAlg* holds the speeds recommended by the new algorithm. In both lists, a value of *Infinity* means that it is not possible to reach the intersection during the current green phase.

```
In[26]:= applySpeedLimit1[list_, speedLimit_] := Replace[list, n_?(# > speedLimit &) → Infinity, {1}];
         applySpeedLimit2[list_, speedLimit_] := Replace[list, n_?(# < speedLimit &) → speedLimit, {1}];

         advSpeedsNewAlg = applySpeedLimit1[advSpeedsNewAlg, speedLimit];
         advSpeedsOldAlg = applySpeedLimit1[advSpeedsOldAlg, speedLimit];
         advSpeedsOldAlg = applySpeedLimit2[advSpeedsOldAlg, speedLimit];
```

Statistical Comparison of Both Algorithms

The goal is to statistically compare the obtained recommended speeds with respect to the considered algorithms. For an easier manipulation with the results, the lists can be merged to a common list of result pairs. Both elements in each pair were obtained with the same input vector but with different algorithms. Thus, the resulting list looks like:
{
  {<*oldAlg_input1*>,<*newAlg_input1*>},
  {<*oldAlg_input2*>,<*oldAlg_input2*>},
  ...
}

Furthermore, we are not interested in cases for which neither of the algorithms could recommend a reasonable speed. These are mostly caused by input vectors for which it is simply not realistic to calculate a reasonable speed. Such pairs cannot help us to compare the algorithms and thus they can be safely removed.

```
In[31]:= advSpeedsPairs = Transpose[{advSpeedsOldAlg, advSpeedsNewAlg}];
         advSpeedsPairs = DeleteCases[advSpeedsPairs, x_ /; x[[1]] == Infinity && x[[2]] == Infinity];
```

The result pairs can be now divided in three separate groups:

bothValid: Both algorithms recommend valid speeds for a given input vector.
    The new algorithm recommends a speed range of <*calculated_speed*> - <*speed_limit*>.
    The original algorithm recommends a single speed of <*speed_limit*>
oldValid: Only the original algorithm recommends a valid speed for a given input vector.
newValid: Only the new algorithm recommends a valid speed (range) for a given input vector.

```
In[33]:= bothValid = DeleteCases[advSpeedsPairs, x_ /; MemberQ[x, Infinity]];
         oldValid = Cases[advSpeedsPairs, x_ /; x[[2]] == Infinity];
         newValid = Cases[advSpeedsPairs, x_ /; x[[1]] == Infinity];
```

Percentages of results in each group are the following:

```
In[36]:= (Length[bothValid] / Length[advSpeedsPairs]) * 100 // N
         (Length[oldValid] / Length[advSpeedsPairs]) * 100 // N
         (Length[newValid] / Length[advSpeedsPairs]) * 100 // N

Out[36]= 98.1396

Out[37]= 1.73451

Out[38]= 0.125892
```

In the *bothValid* group, all the speeds recommended by the original algorithm are within the corresponding speed ranges recommended by the new algorithm. Thus, in this case, the results do not differ.

In both *oldValid* and *newValid* group, the algorithms provided different results for each input vector. Thus, in this case, the results do differ. The total percentage of the differing results is:

```
In[39]:= (1 - (Length[bothValid] / Length[advSpeedsPairs])) * 100 // N
Out[39]= 1.8604
```

Case 2 - Signal is Red

Data Preparation

Again, let's generate a set of *sampleCnt* random input vectors. The time included in each of these vectors represents start of the next green phase. Then copy all the generated vectors and adjust the time to represent the end of the next green phase instead of its start.

```
In[40]:= getRandomInputs[count_] := Table[
            {RandomReal[{distMin, distMax}],
             RandomReal[{timeGreenEndMin, timeGreenEndMax}],
             (While [tmp = RandomVariate[NormalDistribution[initSpeedMean, initSpeedVar]];
               tmp < initSpeedMin || tmp > initSpeedMax]; tmp),
             driverReactTime,
             carAcceleration},
            {count}
          ];
         getRandomPhaseLengths[count_] := Table[
            {0,
             RandomReal[{nextGreenLengthMin, nextGreenLengthMax}],
             0,
             0,
             0},
            {count}
          ];
         randDataPhaseStart = getRandomInputs[sampleCnt];
         randPhaseLengths = getRandomPhaseLengths[sampleCnt];
         randDataPhaseEnd = randDataPhaseStart + randPhaseLengths;
```

We can use the generated data as input vectors for the speed algorithms. For an easier manipulation, the calculated phase-end and phase-start speeds are grouped together.

```
In[45]:= advSpeedsNewAlgPhaseStart = Apply[getAdvSpeed, randDataPhaseStart, {1}] * 3.6;
         advSpeedsNewAlgPhaseEnd = Apply[getAdvSpeed, randDataPhaseEnd, {1}] * 3.6;
         advSpeedsOldAlgPhaseStart = Apply[getAdvSpeedSimple, randDataPhaseStart, {1}] * 3.6;
         advSpeedsOldAlgPhaseEnd = Apply[getAdvSpeedSimple, randDataPhaseEnd, {1}] * 3.6;

         advSpeedsNewAlg = Transpose[{advSpeedsNewAlgPhaseEnd, advSpeedsNewAlgPhaseStart}];
         advSpeedsOldAlg = Transpose[{advSpeedsOldAlgPhaseEnd, advSpeedsOldAlgPhaseStart}];
```

We obtained two lists (one for each algorithm) of calculated minimum (phase-end) and maximum (phase-start) speed pairs to reach the intersection during the next green phase. These are, however, “raw” values that do not consider the speed limit. Both algorithms differ in the way they handle the speed limit. For a detailed explanation, please refer to chapter 5 of the thesis. Nevertheless, the main difference between both algorithms is, in our case, the following:

Original algorithm:
If the calculated phase-end speed is not higher than the speed limit, then the final recommended speed is *min(<speed_limit>, <calculated_speed_phase_start>).* Otherwise, it is indicated that reaching the intersection during the next green phase is not possible.

New algorithm:
If the calculated phase-end speed is not higher than the speed limit, then a speed range from *<calculated_speed_phase_end>* to *min(<speed_limit>, <calculated_speed_phase_start>)* is recommended. Otherwise, it is indicated that reaching the intersection during the next green phase is not possible.

The “raw” calculated speeds need to be transformed to actual recommended speeds in accordance with the considered algorithms as described above. This is easily achieved with the following two transformations:

limitMaxSpeed*:* All the phase-start speeds higher than the speed limit are replaced by the speed limit. This transformation is applied to both lists.

applySpeedLimit*:* All the results with phase-end speeds higher than the speed limit are replaced by an {*Infinity, Infinity*} pair. This is a special value reserved for cases when it is not possible to pass the intersection during the considered phase. This transformation is applied to both lists.

One extra transformation is performed with the *advSpeedsOldAlg* list. All the phase-end-speeds are discarded and the phase-start-speeds become the final recommendations. After the transformations, *advSpeedsOldAlg* holds the speeds recommended by the original algorithm and *advSpeedsNewAlg* holds the speed ranges recommended by the new algorithm. In both lists, a value of *Infinity* means that it is not possible to reach the intersection during the next green phase.

```
In[51]:= limitMaxSpeed[pair_] := If[pair[[2]] > speedLimit, {pair[[1]], speedLimit}, pair];
         applySpeedLimit[pair_] := If[pair[[1]] > speedLimit, {Infinity, Infinity}, pair];

         advSpeedsNewAlg = Map[limitMaxSpeed, advSpeedsNewAlg];
         advSpeedsNewAlg = Map[applySpeedLimit, advSpeedsNewAlg];

         advSpeedsOldAlg = Map[limitMaxSpeed, advSpeedsOldAlg];
         advSpeedsOldAlg = Map[applySpeedLimit, advSpeedsOldAlg];
         advSpeedsOldAlg = advSpeedsOldAlg[[All, 2]];
```

Statistical Comparison of Both Algorithms

The goal is to statistically compare the obtained recommended speeds with respect to the considered algorithms. For an easier manipulation with the results, the lists can be merged to a common list of result “pairs”. Both elements in each pair were obtained with the same input data but with different algorithms. Note that each element corresponding to the new algorithm is a speed range instead of a single speed. Thus, the resulting list looks like:

```
{
   {<oldAlg_input1>, {<newAlg_input1_min>,<newAlg_input1_max>}},
   {<oldAlg_input2>, {<newAlg_input2_min>,<newAlg_input2_max>}},
   ...
}
```

Furthermore, we are not interested in cases for which neither of the algorithms could recommend a reasonable speed. These are mostly caused by input vectors for which it is simply not realistic to calculate a reasonable speed. Such results cannot help to compare the algorithms and thus they can be safely removed.

```
In[58]:= advSpeedsPairs = Transpose[{advSpeedsOldAlg, advSpeedsNewAlg}];
         advSpeedsPairs = DeleteCases[advSpeedsPairs, x_ /; x[[1]] == Infinity && x[[2, 1]] == Infinity];
```

The obtained results can be now divided in three separate groups:

The new algorithm recommends a speed range
from *<calculated_speed_phase_end>* to *min(<speed_limit>, <calculated_speed_phase_start>)*
The original algorithm recommends a single speed
of *min(<speed_limit>, <calculated_speed_phase_start>)*
oldValid: Only the original algorithm recommends a valid speed for a given input vector.
newValid: Only the new algorithm recommends a valid speed (range) for a given input vector.

```
In[60]:= bothValid = DeleteCases[advSpeedsPairs, x_ /; MemberQ[x, Infinity, 2]];
         oldValid = Cases[advSpeedsPairs, x_ /; x[[2, 1]] == Infinity];
         newValid = Cases[advSpeedsPairs, x_ /; x[[1]] == Infinity];
```

Percentages of results in each group are the following:

```
In[63]:= (Length[bothValid] / Length[advSpeedsPairs]) * 100 // N
         (Length[oldValid] / Length[advSpeedsPairs]) * 100 // N
         (Length[newValid] / Length[advSpeedsPairs]) * 100 // N

Out[63]= 99.8479

Out[64]= 0.152145

Out[65]= 0.
```

The *bothValid* group can be further divided in the following three sub-groups.

inRange: For any given input vector, the speed recommended by the original algorithm is within the range of the new algorithm result
belowRange: For any given input vector, the speed recommended by the original algorithm is below the lower bound of the new algorithm result
aboveRange: For any given input vector, the speed recommended by the original algorithm is above the upper bound of the new algorithm result

```
In[66]:= inRange = Cases[bothValid, x_ /; x[[1]] ≥ x[[2, 1]] && x[[1]] ≤ x[[2, 2]]];
         belowRange = Cases[bothValid, x_ /; x[[1]] < x[[2, 1]]];
         aboveRange = Cases[bothValid, x_ /; x[[1]] > x[[2, 2]]];
```

Percentages of results in each sub-group are the following:

```
In[69]:= (Length[inRange] / Length[bothValid]) * 100 // N
         (Length[belowRange] / Length[bothValid]) * 100 // N
         (Length[aboveRange] / Length[bothValid]) * 100 // N

Out[69]= 37.8403

Out[70]= 0.

Out[71]= 62.1597
```

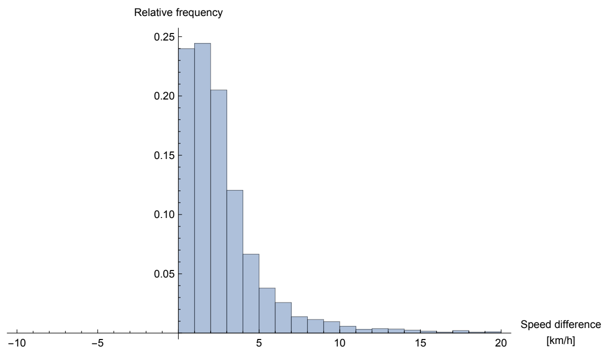
The *belowRange* and *aboveRange* groups can be examined further. It might be interesting to determine “how far” are the speeds calculated by the original algorithm from the corresponding range bounds calculated by the new algorithm.

```
In[72]:= subLow[a_] := a[[1]] - a[[2, 1]];
         subHigh[a_] := a[[1]] - a[[2, 2]];
         belowRangeDiffs = Map[subLow, belowRange];
         aboveRangeDiffs = Map[subHigh, aboveRange];
```

The differences can be visualized in a histogram. Positive speed differences are the *aboveRange* results, negative are the *belowRange* results. All the differences are from the closest speed range bound calculated by the new algorithm:

```
In[76]:= Histogram[{belowRangeDiffs, aboveRangeDiffs}, {-10, 20, 1}, "PDF",
          AxesOrigin → {0, Automatic}, AxesLabel → {"Speed difference\n[km/h]", "Relative frequency"}]
```

Out[76]=



Only the results from the *inRange* group can be considered as not differing**.** Thus, the total percentage of the differing results is:

```
In[77]:= (1 - (Length[inRange] / Length[advSpeedsPairs])) * 100 // N
Out[77]= 62.2173
```