POLITECNICO DI TORINO

Master Degree in Communications and Computer Networks Engineering

Master Thesis

Implementation of a MEC Application for a Traffic Control Center

Academic supervisor:
Prof. Claudio Ettore Casetti

Candidate:
Marta Lamanna

External Supervisors
Ing. Daniele Brevi
Ing. Edoardo Bonetto

October 2019

Abstract

Multi-Access Edge Computing (MEC) is a completely new paradigm in application development, since it has the purpose of offering cloud computing services at the edge of the network and closer to the users, introducing a near server between the client and the traditional remote server. The benefits of this new technique are especially the ultra-low latency, high bandwidth, real-time access to radio network, location awareness and a flexible and extendable framework for the services.
In this thesis an implementation project is presented, where the MEC environment is used for the development of an Intelligent Transport System application: a traffic monitoring system able to connect the vehicles to a Traffic Control Center (TCC) that allows drivers to report a road obstruction detection directly, exploiting a crowdsourcing approach. Moreover, the application is able to offer to the drivers a traffic rerouting service, providing a customized response that contains an alternative route advice personalized for each user: the MEC combines its contextual information knowledge and scalability with the global traffic conditions awareness provided by the TCC server, in order to compute a smart traffic redirection able to take into account the whole traffic demand of the users in the area, the current road conditions and preventing further congestion.
The project focuses on the implementation of all the communication chain from the users to the TCC, with a particular attention to the MEC Application and to the MEC Location API, the standardized location service for MEC.
For the information exchange with the TCC, the protocols that have been studied and used for the creation of the data messages are DATEX II and S.I.MO.NE..
In the last part of the thesis, SUMO (Simulation for Urban Mobility) is used to compute the traffic rerouting for the drivers and to analyze the behavior of a macroscopic routing algorithm on a realistic map of Turin.

List of Figures

List of Tables

Contents

Acronyms

3GPP	3rd Generation Partnership Project
API	Application Programming Interface
AR/VR	Augmented Reality and Virtual Reality
BTS	Base Transceiver Station
BWMS	Bandwidth Management service
DNS	Domain Name System
ETSI	European Telecommunications Standards Institute
GPS	Global Positioning System
IoT	Internet of Things
ISG	Industry Specification Group
ITS	Intelligent Transport Systems
KPI	Key Performance Indicator
LS	Location Service
ME	Mobile Edge
MEC	Multi-Access Edge Computing
MEO	Mobile Edge Orchestrator
MEPM	MEC platform manager

NFV	Network Functions Virtualization
OMA	Open Mobile Alliance
OSS	Operations Support System
REST	REpresentational State Transfer
RNI	Radio Network Information Service
SUMO	Simulation of Urban MObility
TCC	Traffic Control Center
UALCMP	User Application Lifecycle Management Proxy
UE	User Equipment
VIM	Virtualisation Infrastructure Manager
VM	Virtual Machine

Chapter 1

Introduction

In the last years the evolution of computing and communications technologies has led the ICT researchers to be interested in one of the most important sectors concerning our society: transport systems. Indeed, the development of the Smart City allowed the creation of real time flows of information and data among vehicles and traffic control centers, bringing innovation in this field.
In particular there are some important aspects that need to be enhanced in the context of surface transportation such as safety, efficiency and convenience, and these are the main reasons why the Intelligent Transport Systems (ITS) are born and widely used today. In fact the main purposes of the ITS are to decrease the risks and the probability of car accidents, to improve the traffic conditions and public services and to increase traffic efficiency and travel comfort for both drivers and pedestrians. Moreover these systems allow to optimize the infrastructure utilization in order to better exploit the yet existing roads and traffic services, to reduce environmental pollution and to promote energetic efficiency [1].
The introduction of 5G Networks contributes to this development and makes ITS products and services more and more sophisticated and reliable, improving performances and user experience: indeed, the new generation of mobile networks leverages on some techniques as Multi-Access Edge Computing (MEC) and Network Functions Virtualization (NFV) that make the infrastructure very flexible and adaptable to new different kind of services, such as vehicular communication and Internet of Things (IoT). In particular this thesis will focus on MEC, that is a technology that creates a new distributed computing software development model able to provide cloud-computing capabilities at the edge of the network and that

is characterized by an extreme user proximity, ultra-low latency, high bandwidth availability, real time access to Radio network information and location awareness.[2] In this thesis a simulated use of ITS will be presented, developed in collaboration with Links Foundation and 5T in the context of the Smart Road project, a convention made by Città di Torino that involves different companies in order to develop and test new innovative products, technologies and services in a real context in the field of autonomous driving and connected vehicles. The project consists in the implementation of a communication system between a Traffic Control Center (provided by 5T) and users (android devices) through the use of a MEC application, in order to build a real-time viability framework in which users can signal local events and the Traffic Control Center can reduce their impact providing a feedback. The system is a Vehicular Rerouting Advisor: it will be able through users signalling to detect an event (such as an accident, a road obstruction due to environmental causes or a system fault) and communicate it to the MEC application, that will notify the Traffic Control Center specifying the entity of the event and its position known thanks to Location API, that is the MEC interface that provides location information about the users; then the TCC will return an answer that will be sent to all the involved devices, providing a redirection advice. The basic idea is to optimize the viability computing a new traffic routing, finding a new best path that avoids the detected obstacle, but also in order to avoid congestion in other roads; so the purpose of 5T answer is to provide information data about current traffic conditions to the MEC Application, that will offer a personalized alternative to each involved vehicle.
This project is also implemented as a basic use case of a traffic monitoring system, that can be summarized in the following steps:

- Acquisition: the system needs to know which and how many vehicles there are in each location area and it collects their information data. In case of obstacle or accident, the event signal is notified.
- Elaboration: the system receives the alert recording the event and provides the information about all the possible vehicles and users involved exploiting the MEC Location API. In the next phase the system, with the help of the 5T server, will compute and provide re-routing information.
- Communication: the system creates messages according to the proper standards and sends them to all the users and allows the MEC application and the 5T

server exchange information.

Acquisition in traffic monitoring is usually done by means of sensors, that sometimes can cover only limited areas; so in our thesis a new approach similar to crowdsourcing is used: each user can instantaneously signal by himself the event collaborating as an active part of the system thanks to Internet-connected geolocated smart devices (in particular in our simulation simple android smartphones are used). Indeed this technique is becoming very common in IoT applications because in general it brings many enhancements such as the use of low-cost and yet operating resources, real-time acquisition of data and the exploitation of the direct human-users experience in every possible position or area to provide information.
The re-routing computation required after the event signalling is made by the MEC Application, that knowing the topology of the road network and providing as input the 5T response containing traffic information with travel times, is able to exploit SUMO routing tools to compute a smart traffic balancing assigning a different and customized alternative route to each vehicle.

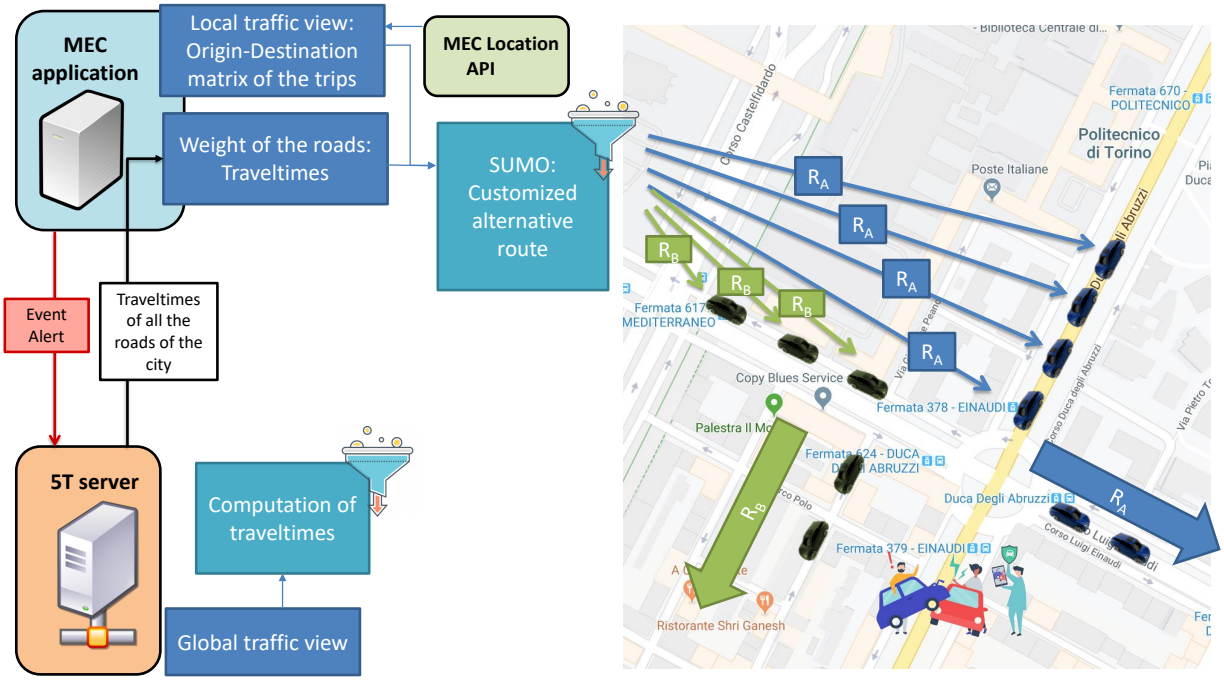


Figure 1.1. Vehicular Rerouting System: overview

The thesis is organized as follows:

- In Chapter 2 Multi-Access Edge Computing is presented: the use-cases, the framework and architecture, the standardized services and the current usages and implementations.
- Chapter 3 illustrates all the system operations and the implementation details of each block: user interface, MEC Application and MEC Location API.
- Chapter 4 presents the uses and the modelling of the two protocols studied in this thesis to interact with the Traffic Control Center, DATEX II and S.I.MO.NE..
- Chapter 5 describes the analysis of the macroscopic router used by the system to advise alternative routes to the users, exploiting some SUMO simulations, first made with a Manhattan topology network and then with the road network of Turin.

Chapter 2

Multi-Access Edge Computing (MEC)

In this chapter a focus on the ETSI MEC technology is presented; the first section describes what is MEC and which are its main purposes; section 2.2 briefly presents the primary use cases of MEC, whereas sections 2.3, 2.4, 2.5, 2.6 and 2.7 introduce the MEC architecture and the standardized interfaces that allow applications to interact with MEC.
Section 2.8 describes how MEC is deployed in view of future 5G networks and section 2.9 presents the current available implementations.

2.1 MEC overview

Some of the most important innovations introduced by the 5G Networks are the latency reduction, high bandwidth and real-time access to radio network information that can be exploited by users [2]. For these purposes and for other important improvements in networks performances and user experience, the European Telecommunications Standards Institute (ETSI) launched in September 2014 an Industry Specification Group (ISG) with the aim of creating the standard of the Multi-Access Edge Computing (MEC) in order to offer cloud-computing capabilities at the edge of the network that might enable the demanding Key Performance Indicators (KPIs) of 5G; when the project started the original name was Mobile Edge Computing, but due to the necessity of making the standard available not only for mobile networks operators but also for authorized third-parties such as vertical segments, the name changed, while the acronym still holds.
MEC can be defined as a set of techniques based on the idea of having applications

and services not hosted on the remote server but closer to the user, providing a highly distributed computing environment: this is a new approach that modifies the traditional networks architecture model composed of just a client and a server and that introduces a new element between them. Indeed MEC servers are the new intermediate elements integrated in the network, creating a new development paradigm composed of Client, Near Server (MEC server) and Far Server (the traditional cloud server); this new model, combined with the Network Function Virtualization (NFV), allows 5G applications to reach the new requirements and enhancements that make them suitable for all the new technologies as IoT or vehicular communications (V2X) and to offer new personalized services. In fact, in the context of Smart City, collecting and managing data and information is a current issue that can be solved moving all the computing and elaboration near the sources; this solution makes the latency lower and makes the remote cloud not be congested or overloaded. Another interesting feature of MEC is its local perspective respect to the whole network topology: in this way service providers can exploit the geographical distribution to offer personalized services and functionalities that depend on the peculiarities of the covered area, such as linguistic and cultural characteristics [2].

2.2 MEC use cases

MEC enables a lot of new key applications and the main use cases currently identified are [2]:

- Video stream analytics and video surveillance: the video management application can transcode and store captured video streams from cameras, while the video analytics application can process the video data to detect and notify specific configurable events. This kind of service can be used for safety and public security.

- Augmented Reality and Virtual Reality (AR/VR), Video Content Delivery Optimization: MEC offers Content Delivery to applications that need high bandwidth and low latency and moreover it can offer local object tracking and local AR content caching.

- Enterprise applications enablement, for example for government institutions,

fleet management or for enterprises that require employees to access an enterprise application within the physical location of the corporation.

- Applications with critical communication needs such as traffic safety and control, autonomous cars, connected vehicles, industrial IoT and Healthcare
- Smart City and IoT applications and Gateway, such as AWS IoT Greengrass
- Active Device Location Tracking: MEC can provide location services for applications where GPS coverage is not available
- RAN-aware Content Optimization: in this use case, the application exposes accurate cell and subscriber radio interface information (cell load, link quality) to the content optimizer, enabling dynamic content optimization, improving QoE and network efficiency.
- Content Cashing [3] for different types of services that aim to gain a better user experience

2.3 Framework and reference architecture

Networks infrastructures have always been following the hour-glass model based on IP protocol: the applications and the networks are always interoperable but without knowing the reciprocal specifications; now with the advent of MEC applications the model is still worth despite networks and applications are now converging into the new Edge component. Indeed now the application can get some information useful to adapt its behaviour to the environmental conditions on-the-fly, and this makes the network functioning less unpredictable and more efficient. With the MEC technology every application can be supported, but only the MEC-aware ones can take advantages exploiting the new additional services made available [2]; in order to manage and verify this availability the Mobile Edge Platform Application Enablement API has the task of giving "instructions" and starting the proper procedures to enable the MEC services.

As described in [4] the MEC reference architecture consists of four main parts:

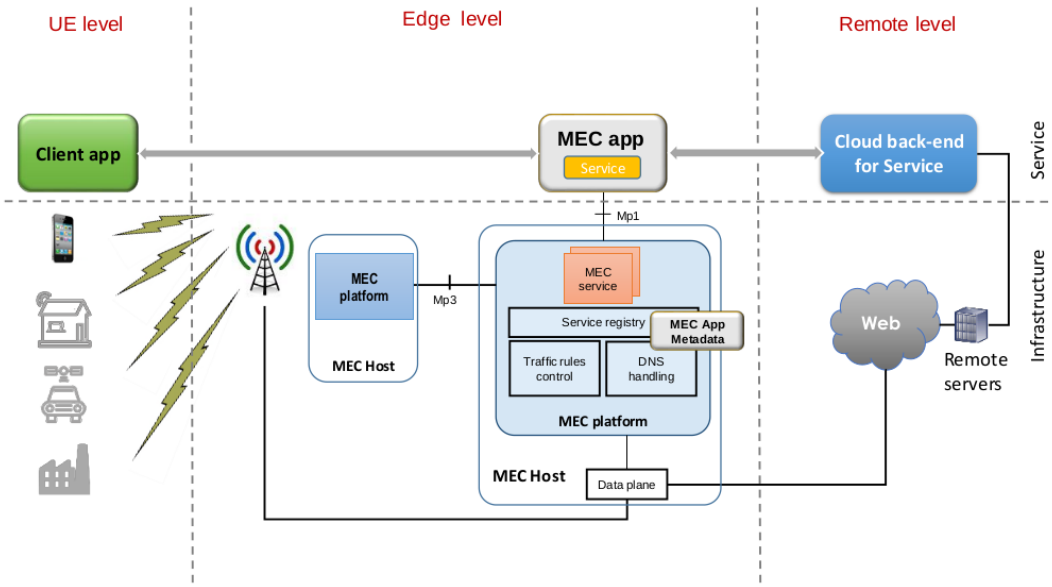


Figure 2.1. New application development paradigm introduced by MEC [2]

- **Mobile Edge Host**: it consists of mobile edge platform, mobile edge applications and virtualisation infrastructure, and it is used for providing resources able to run Mobile Edge (ME) applications. The **mobile edge platform** offers an environment where mobile edge applications can leverage mobile edge services, receives traffic rules providing consequently the proper data plane, receives DNS records from the mobile edge platform manager and configures a DNS proxy/server accordingly, hosts ME services and provides access to persistent storage and time of day information. The **MEC application** leverages Virtual Machines (VMs) or containers to store data or doing computation and, as better explained in 2.6, exploits RESTful APIs to offer services and interact with the customers applications. The **virtualisation infrastructure** includes a data plane that executes the traffic rules received by the ME platform, and routes the traffic among applications, services, DNS server/proxy, 3GPP network, local networks and external networks; this component runs ME applications as VMs and provides ME services interacting with the ME platform.

- **ME System Level Management**: it manages mobile edge hosts in order to run ME applications within an operator network. System Level Management includes **Mobile Edge Orchestrator (MEO)**, that is responsible for making

decisions about which ME hosts are suitable for application instantiation (checking constraints like latency and available resources) or about applications terminations and for maintaining an overall view of the ME system, available ME services and topology. Other components are the **Operations Support System (OSS)**, that receives requests for instantiation or termination of applications, decides to admit them or not, and eventually forwarding them to the MEO, and the **User Application lifecycle management proxy** that interacts with the UE applications and the OSS.

- **ME host level management**: it manages the mobile edge specific functionality of a particular ME host and the applications running on it; it is composed of the **MEC platform manager (MEPM)**, that has the function of managing the life cycle, rules and requirements of the applications, and the **Virtualisation Infrastructure Manager (VIM)**, that is responsible for allocating and managing virtualised resources and for preparing the virtualisation infrastructure to run a software image.

- **ME Networks Level**: it includes external related entities.

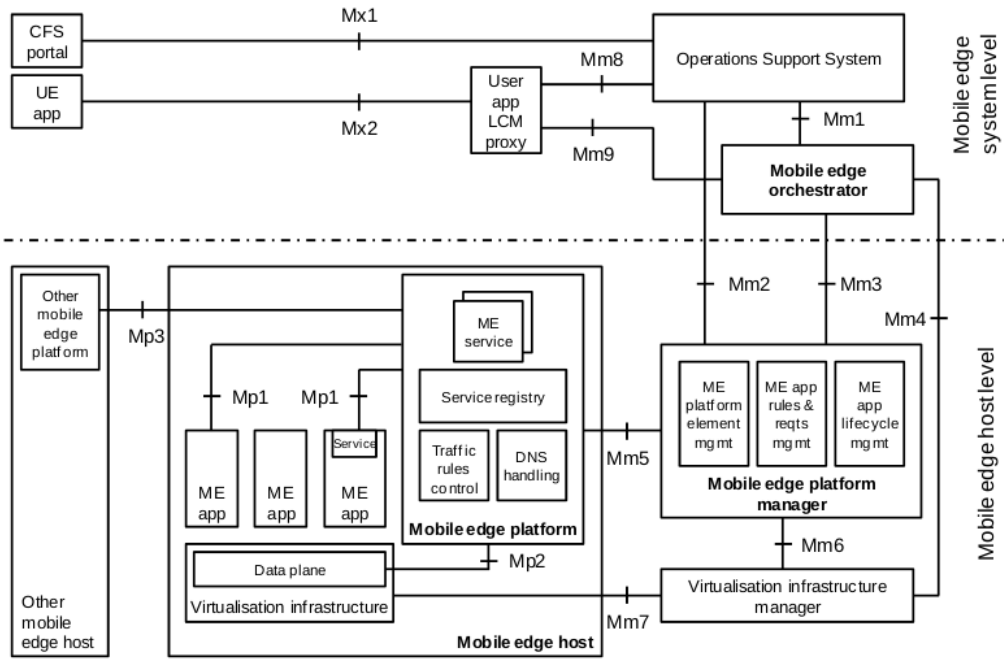


Figure 2.2. Mobile Edge system reference architecture [4]

As shown in figure 2.2, between the system entities there are three groups of reference points:

- **Mp**: reference points regarding the ME platform
- **Mm**: management reference points
- **Mx**: reference points connecting to external entities

2.4 Architecting and Developing MEC Applications

Building software for MEC should comply with the MEC framework and consequently designing a MEC Application requires some steps that can be summarised in four phases [2].
The first phase is the **MEC Application packaging and on-boarding**: this step is made by application developers and the applications are usually set up as a VM or Container and must respect the MEC platform's requirements and configurations. The OSS receives requests for managing of applications and it decides whether to accept them or not and eventually send them to the MEO for further processing. The MEO has the task of onboarding the applications into MEC systems, of checking the integrity and authenticity of the signed packages and validating application rules. Then the MEO assigns an application package ID and supplies the MEPM with the location of the application image. The MEPM prepares the VIMs selected by the MEO for application instantiation, by providing the necessary infrastructure configuration information and sending the application images, which are then stored by the VIM. After been on-boarded, the application package is in "Enabled, Not in use" state.
The second phase is the **MEC application instance instantiation and operation**; the instantiation procedure refers to launching an application in its virtual machine or container in the virtualization environment of a MEC host. The application instantiation can be triggered from a device, and in this case the device application that works as a client should support Mx2 API, enabled by the User Application Lifecycle Management Proxy (UALCMP), or from OSS, and with this option the developer interacts directly with the MEC operator. The instantiation request contains information, rules and requirements about the application to run. First MEPM sends lifecycle requests to the VIMs for allocating virtualized resources and

for the instantiation, and has the responsibility of receiving virtualized resources fault reports and performance measurements from VIM. Once set up, the application is enabled to interact with MEC Platform over Mp1 in order to manage some settings and procedures of the lifecycle of the application. After the instantiation, the application lifecycle management APIs can perform the "start" operation, in order to instruct the application instance to run and provide the service, and the "stop" operation, in order to instruct the application instance to stop running.
The third phase is **client-side application and MEC application communication**: each end user device should have a client application installed, which is the one that sends the request for services from the MEC application. The device application is used to invoke user application lifecycle management operations on MEC system's management plane. The client application can connect with a MEC application instance with two option: the first one is that the developer/service provider takes responsibility for making the address of the MEC application making it known to the client applications; the second option is that the client application discovers the MEC application instance via DNS look-up, but the developer needs to correctly make the domain name available on the DNS server through the MEC Platform services.
The fourth phase is the **usage of the MEC platform and services**: the MEC application is now up and running and gets operational, so it can exploit MEC services and MEC APIs produced by the MEC platform or by another MEC application. The accurate description of the APIs is presented in the next sections.

2.5 Mobile Edge Platform Application Enablement

In order to be enabled, the applications can interact with the mobile edge system through Mp1 reference point, that connecting ME applications with ME platform is able to provide some functionalities such as service registration, service discovery, communication support for services, traffic rules, DNS rules activation and time of the day [5].
The flow of ME application startup is shown in 2.3.
In the ME application startup procedure, first ME application is instantiated and it informs the ME platform that it is up-running; authorization and authentication of the application may be applied depending on operator's policies or third party service providers' requirements and then ME application instance may request the

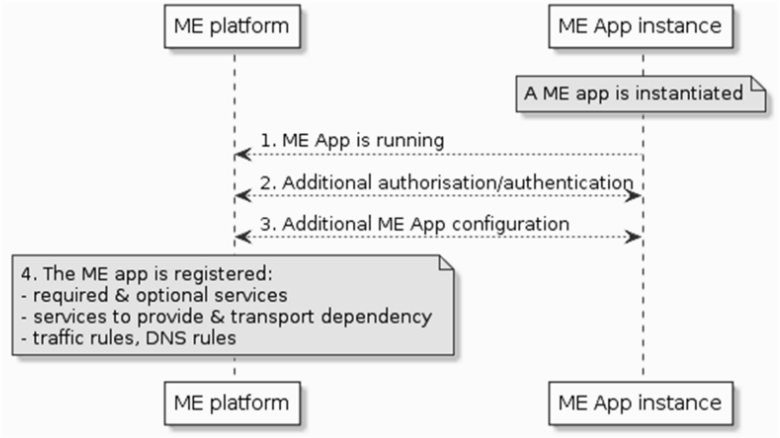


Figure 2.3. Flow of Mobile Edge application startup [5]

ME platform to activate or deactivate a traffic rule or to update the parameters of an existing one. Finally ME platform has the task of configuring the ME application, registering it internally and storing the related information about application instance, such as required and optional services, services to be offered by this application and the associated transport dependency, traffic rules and DNS rules associated. This procedure is set up by the Mobile Edge Platform Application Enablement API.

2.6 Service Application Programming Interfaces

ISG provided all the normative specifications and standardized APIs in order to offer services and interact with the customers applications.
In order to offer services RESTful APIs are exploited. A RESTful API is an Application Programming Interface (API) that uses the HTTP protocol as a tunnel or transfer mechanism for interaction between remote entities. RESTful refers to a stateless design through REpresentational State Transfer (REST), an architectural style often used in web service development [6].
The main MEC services are Radio Network Interface, Location, Bandwidth Management and UE Identity and they are available for the MEC platform and authorized MEC applications.
The **Radio Network Information Service (RNI)** as described in detail in ETSI GS MEC 012 [7] is a service that provides to authorized ME applications and to ME platforms radio network related information, that is very important in an environment

characterized by proximity, low latency and high bandwidth. The granularity of the radio network information may be adjusted based on parameters such as information per cell, per User Equipment, per QCI class or it may be requested over period of time. This API is usually used to provide up-to-date radio network information regarding radio network conditions, measurement information related to the user plane based on 3GPP specifications and information about UEs connected to the radio nodes associated with the mobile edge host, their UE context and the related radio access bearers.
Therefore this API is used to enhance all the applications that need to know radio conditions to provide new services; a typical example of these requirements is video throughput guidance, that thanks to this API can dispose of real-time indications on the throughput estimated to be available at the radio downlink interface in the next time instant and data transmission is consequently adapted. Radio Network Information may be also used by the ME platform to optimize the mobility procedures required to support service continuity. In some ME application requests a simple request-response model typical of RESTful methods can be used, but in other cases where multiple notifications are requested and a large amount of information is needed, RNI may be provided over the message broker of ME platform.
The **UE Identity API** is described in ETSI GS MEC 014 [8] and it is used to allow UE specific traffic rules in the ME system. In fact MEC platform offers the service of registering a ME application with an externally defined tag or a list of tags and this association with the UE Identity helps the user to protect its privacy identity information at both mobile and enterprise network. So a MEC application can register or deregister a tag (representing a UE) or a list of tags in the MEC platform, that will apply traffic filters rules corresponding to that tag. In this way authorized UEs can have their user plane traffic routed directly to the corresponding network without passing through the MEC application [2]. The tag-based traffic filter rules are handled in the ME platform and described in [5].
The **Bandwidth Manager API**, as defined in ETSI GS MEC 015 [9], works as central bandwidth resource allocator service on the ME platform in order to allocate in an optimized way the bandwidth resources among application sessions that are competing over them. In fact the main objective of the Bandwidth Management service (BWMS), offered by ME platform, is to provide a fair distribution of bandwidth resources between the applications according to their requirements [2].

The API has a RESTful design and each ME application session is identified by a set of filters within the resource request. Another interesting aspect of this API is that the BWMS design might interface with the Radio Network Information Service to use available Layer 2 and QoS information to facilitate the traffic arbitration and to obtain a network optimization [2].
The last main API is **Location API**, on which a better focus will be presented in the next section.

2.7 Location API

The **Location API** is described in ETSI GS MEC 013 [10] and is used to provide the location related information to the ME platform or authorized applications. This is the most important API for this thesis project since, as described in section 3.4, it will be implemented in a simulated version to offer location information to the main MEC application.
The Location Service (LS) is enabled over the Mp1 reference point defined in ETSI GS MEC 003 [4] and supports different types of location information: the location information of specific UEs, all UEs or eventually a certain category of UEs currently served by the radio nodes associated with the ME host, a whole list of UEs in a certain location area, the specific UEs that move in or out of a particular location area and information about the location of all radio nodes currently associated with the ME host. Moreover Location Service is able to support the location retrieval mechanism (location is reported only once for each location information request), location subscribe mechanism (location is reported multiple times for each information request, periodically or based on specific events) and anonymous location report (for example, in case of statistics collection). LS can offer information based on geolocation, such as geographical coordinates, and logical location, such as cell ID [10].
These services are accessible through an API previously defined in the Open Mobile Alliance (OMA) specification "RESTful Network API for Zonal Presence" [11]. This API uses the concept of "zone", as a way to groupe all the radio nodes associated to one or more ME hosts; this service is called Zonal Presence and provides to an application information about a zone, users connected inside a zone, access points and

so on. Furthermore, the OMA Zonal Presence API gives the possibility to authorized applications of subscribing to a notification mechanism that reports about activities and information of users within a certain zone.
All the main procedures of this API are described in detail with the corresponding diagrams in [10]: UE Location Lookup, UE Information Lookup, UE Location Subscribe, UE Information Subscribe, Subscribe Cancellation and Radio Node Location Lookup.
Moreover in the specification document data modelling is described: data types are the same of those described in OMA specifications [11] and in particular, as described in the next chapter, the data types used in this project in the Location API implementation are UserList (type containing list of users) and UserInfo (type containing user information).

2.8 MEC Deployments in 5G

Despite MEC is considered a 5G feature and since MEC reference architecture is flexible and agnostic to the mobile network evolution, it has been enabled also in yet existing 4G networks but still in sight of the future deployment for 5G services and of a feasible transition through a simple software update [12]. The standard does not provide specifications about the underlying radio infrastructure and for this reason it is a highly flexible element in the communications networks, giving the possibility to service providers of trying applications suitable for 5G in a virtual retail space using 4G edge test bed and allowing a smooth transition among the network evolution. Furthermore MEC allows to re-use the existing deployed systems in the process thanks to its virtualized characteristics [12].
So MEC, even if born as a standard independent of the mobile network infrastructure, can easily benefit from the edge computing enablers of the 5G system specification as well as 3GPP ecosystem can benefit from the MEC system and its APIs as a set of complementary capabilities to enable applications and services environments in the edge of mobile networks [13]. The design approach taken by 3GPP allows the mapping of MEC onto Application Functions (AF) that can use the services and information offered by other 3GPP network functions based on the configured policies [13].

2.9 Proposals for MEC solutions

Currently different solutions have been provided by some companies in order to support MEC. For example Nokia provided a product called Airframe Open Edge Server, a x86 architecture compatible with edge cloud deployments and able to implement distributed computing. This server is efficient in terms of memory consuming, is compatible with the Open Platform for NFV Project (OPNFV) and above all it is characterized by low latency and other features essential to provide real-time services using MEC and Cloud RAN.
Also another important technology company as Intel is developing database-centric infrastructure that can be used for optimized edge solutions, distinguishing on-premise edge, consisting of edge platforms included in enterprise premises and hosted by customer itself, from network edge, hosted in network nodes and composed of the edge platforms in the wireless access or cable access. In order to satisfy these needs Intel created the Open Network Edge Services Software (openNESS), an open source reference toolkit aimed at easily deploying applications and services at the edge and making easy to IoT developers to deal with the new infrastructure of 5G and the adoption of edge computing. In particular this software platform enables different functionalities such as traffic steering, service authentication or telemetry, exploits standardized APIs exposed to an Open Source community and allows secure on-boarding and management of applications with a web-based GUI.
Other companies that are implementing different solutions for MEC are Saguna, that focuses also on Ultra Reliable and Low Latency Communication (URLLC), Athonet and Huawei.

Chapter 3

System description and implementation

In this chapter the overall project framework is presented in a detailed way: after the general description in section 3.1, section 3.2 describes the implementation of the user interface, specifying in subsection 3.2.1 the user signaling of a detected event and in subsection 3.2.2 the feedback that the system can provide. Section 3.3 explains the implementation of the MEC application and section 3.4 the implementation of the MEC Location API.

3.1 System overview

The Vehicular Rerouting Advisor is designed to work as an ITS that monitors traffic and viability in an urban mobility scenario, building an efficient communication system between the vehicles traveling in a certain traffic zone and the Traffic Control Center (TCC). As previously described in chapter 1, the elements that compose the informative chain of a traffic monitoring system are the acquisition, the elaboration and the communication of the data. The acquisition is made by the interaction with the user interface: the system gives the possibility to the users of signalling an obstruction encountered along the road to the central part of the system, that is the MEC Application, so that it knows the entity of the problem and requires its position from the Location API; then it sends in turn the alert to the central server. After the obstruction reporting, the central server provides the elaboration of the data exploiting its global knowledge of the traffic conditions and computes each real-time travel time that must be employed to take a road of the area.

At the last step, the server sends the traffic information message containing the travel times to the MEC Application, so that it is able to calculate through a macroscopic traffic routing algorithm a personalized alternative path for each user, taking into account not only the length and the duration of the routes, but also the real current traffic of the involved urban area and the entire traffic demand, comprised of a origin-destination matrix of the users trips.
Therefore the framework, as shown in figure 3.1, consists of four main parts: the users application, the MEC application, the MEC Location API and the 5T server.

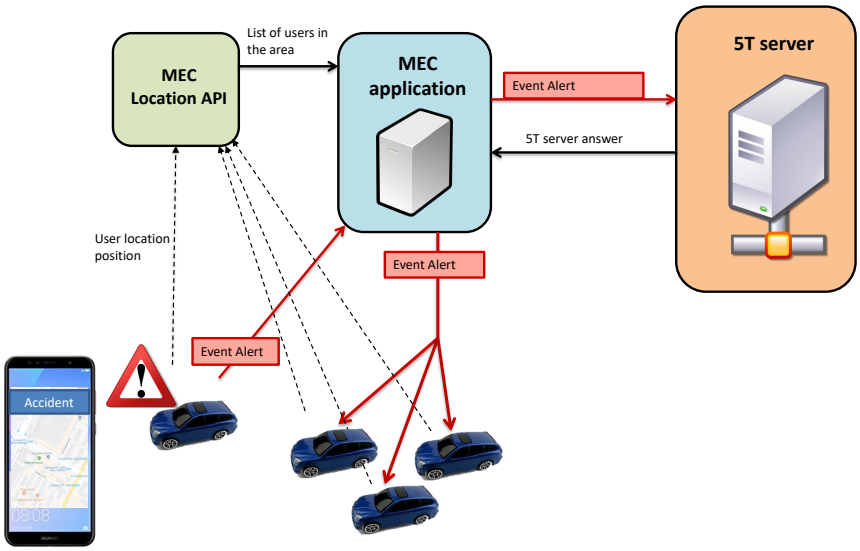


Figure 3.1. System overview: graphical representation

From the architectural point of view, since the Location API is a MEC service, it can be considered as a component belonging to the MEC Application, and the end-to-end service can be divided into the three main elements typical of the MEC environment:

- Terminal Device Components: User Application
- Edge component: MEC Application
- Remote Component: 5T Server

Figure 3.2 represents the overall system project specifying the elements implemented and described in this project:

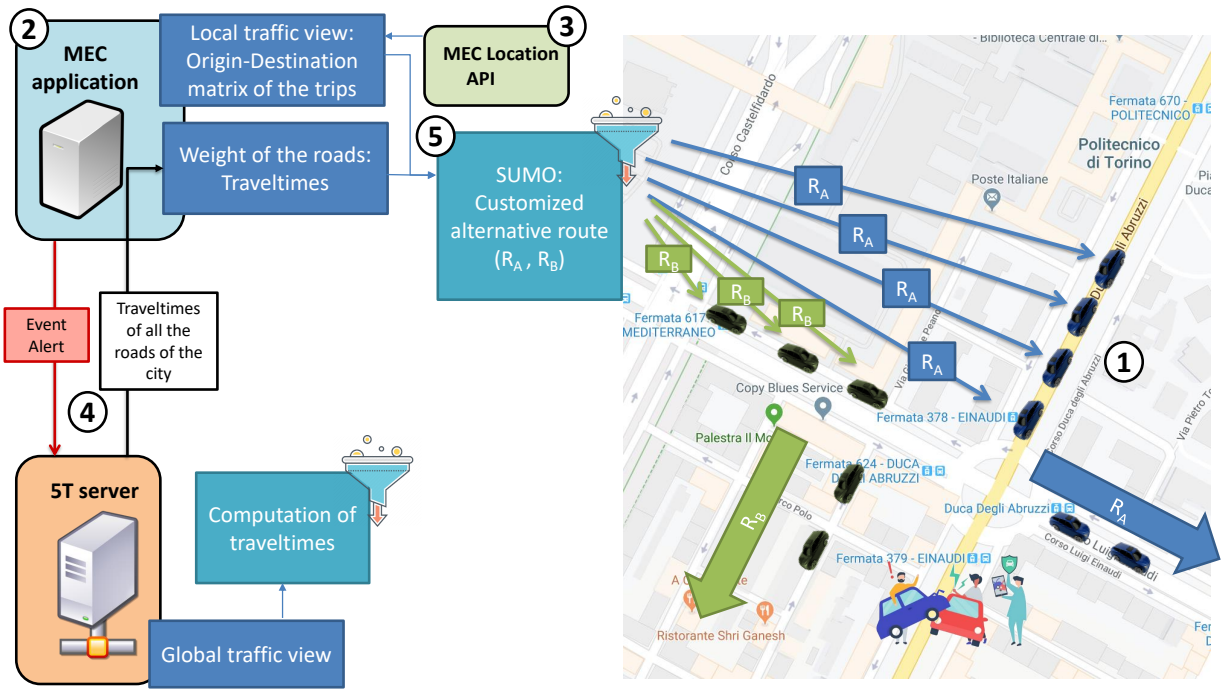


Figure 3.2. Vehicular Rerouting System: implemented blocks

- 1: User interface, described in section 3.2
- 2: MEC Application, described in section 3.3
- 3: MEC Location API, described in section 3.4
- 4: Interaction with 5T server, described in chapter 4
- 5: Smart rerouting with SUMO, described in chapter 5

All these blocks have been implemented and made operational, except for block 4: the interaction with the 5T Server results incomplete because, even if the implementation of the creation of the messages according to the TCC standards is complete, currently there is no web service active to enable the communication with the server.

3.2 User application

3.2.1 Event reporting from the user

The first step of our system is the acquisition phase: the users are the drivers of the vehicles traveling in the location area involved by the TCC; they are meant to have a connected-device such as a smartphone and an application installed that can send the alert to the MEC Application in real-time with a command.
In this experimental phase of the project a basic method through the HTTP protocol is used, but a more specific protocol for vehicular communication can be used in some future project enhancements. The used physical interface during the tests is a hotspot WiFi, whose interface is set with a class A private subnet (10.0.0.1/24) and all the used android devices have been connected to the same network (10.0.0.0/24). For the event report a basic application on the android user device is used and in particular in the first experimental tests of the project the employed application is Rest Client [14].
The system is implemented supposing that the user signals the event instantaneously as soon as he detects it on the road, by sending an URL to the MEC application whose IP address is known (10.0.0.1), specifying also the type of event as a parameter of the HTTP request (`inc`), that in the current implementation is coded as:

- **1: Accident**: an accident between cars or other vehicles
- **2: Environmental obstruction**: a road obstruction due to a natural or environmental problem, such as the fall of a tree, river flooding or animal presence
- **3: System fault**: a malfunction of the road equipment, such as a broken traffic light or unusable rail crossing

In each case the event implies that the carriageway is obstructed and consequently closed, so our system will identify the point of the obstruction as a Black Spot.
Other parameters sent by the same URL through the HTTP request are the latitude and longitude coordinates of the destination of the user, and these have been added to the implementation because it is essential for the MEC Application to compute the O/D (Origin/Destination) file of the trips of the involved vehicles, used to correctly recalculate the alternative route through the simulations made with SUMO, as described in chapter 5.

The position of the event is not included in the event alert, since it is provided by the MEC Location API.
An example of URL sent from the user application taken from the tests is reported below:

```
http://10.0.0.1:8080/event_alert?inc=1&latdest=45.065&londest7.659
```

The example reports an event alert where a user, that is traveling towards a destination located at the geographic coordinates (45.065, 7.659), detects a road accident, identified by the code `inc=1`.

3.2.2 Smart Traffic Redirection Response

The feedback provided by the Vehicular Rerouting Advisor to the user is a personalized alternative route advice computed by MEC Application; indeed the output of the rerouting algorithm provides the more convenient path from a macroscopic point of view assigning a route to each user, as explained in detail in chapter 5; subsequently the MEC Application extrapolates each route and converts it in a sequence of points in geographic coordinates, in order to provide a response in a format suitable for a GPS referenced application. Afterwards this sequence is saved in a string that is sent to the corresponding user through a HTTP GET Request.
In this thesis project implementation the smart traffic redirection system works only for the users that signaled the event, because in this case we can obtain the destination position of the user's trip; the system could be improved with the development of an application that allows the user to subscribe to the service and set the destination of the next route: then the user application sends destination and current position of the user to the MEC Application that will have the task of providing the best path computed through the smart routing algorithm. The application will be running for all the trip duration and in case of a change of the traffic conditions that alter the cost of the current path (such as an accident or a road congestion), it will recompute and advise to the user an alternative route in real-time. In this way not only the users that signal an obstruction would take benefits from the system.

3.3 MEC application

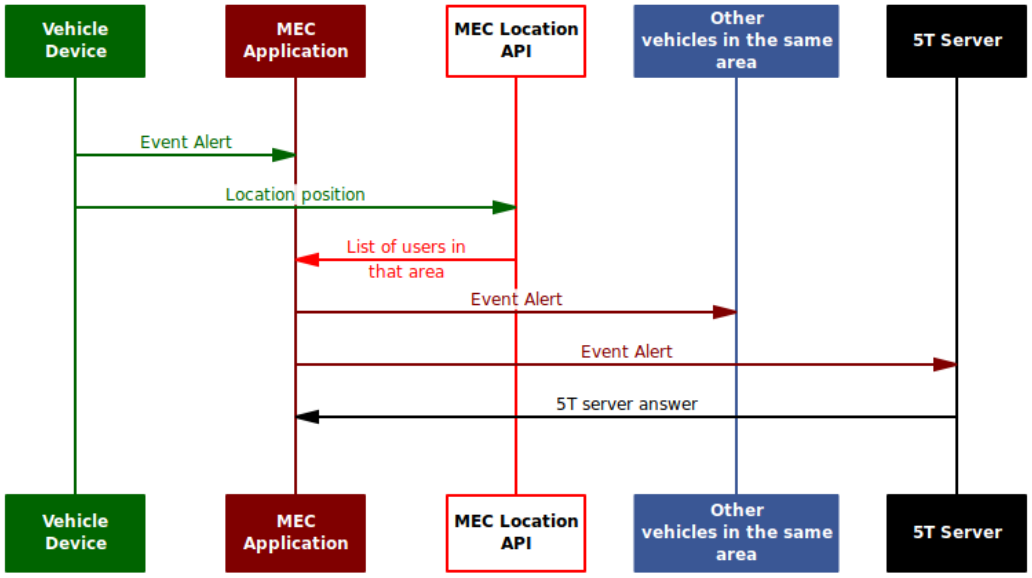


Figure 3.3. Workflow of the system

The MEC application is the core part of the thesis, indeed it is the central element of the entire system. This application is implemented with a C++ code that can be virtualized on a Virtual Machine or a container, but since currently there are no MEC platforms available and configurable for the tests, the simulation is executed locally. The peculiarity of this application is that it is supposed to run on a MEC architecture, that establishes an environment able to guarantee low latency, efficient network, data management and storage and in our case the system can take advantage from the MEC geographical awareness to know which are the involved vehicles and offer to them customized information. Moreover, the traffic monitoring system used in a real scenario will collect a huge amount of data, that can be hardly managed by one single server, so the addition of an intermediate server that has the purpose of handling a portion of this data makes the system more scalable and efficient, in terms of both latency and computational performances. Figure 3.3 represents the workflow of the overall system and in particular the behavior of the MEC application: first it has the task of saving in an internal structure all the received event alerts (*Events List*) and in parallel of interacting with the Location API, that, as better described in

section 3.4, provides some information useful for the system as the location of the user that sent the alert and the whole list of the users that recorded their position in the same location area (*Users List*). Subsequently the application builds a message for 5T server with S.I.MO.NE or Datex II, the protocols used in 5T Traffic Control Centers, described in chapter 4, and sends it to the server. The last task of the application is to send the more convenient route to each one of all the users present in the users list provided by Location API and whose destination location is known.
The table 3.1 shows how the main data structures of the application are organized, in particular the *Event report*, that describes a single event alert, and the *user* that describes a single user present in the location area. The *Event report*s are collected in the structure *Events List*, whereas the *User*s are collected in the structure *Users List*.

Structure	Fields	Description
Event report	IP address	IP address of the user that sent the alert
	accessPoint_ID	Field used by Location API
	zone_ID	Field used by Location API
	event_type	Code [1,2,3] used to identify the type of the event
	situation_ID	Field used by Datex II or S.I.MO.NE.
	latitude	Used to define the position of the event
	longitude	Used to define the position of the event
	accuracy	Accuracy of the measure of the event position
	timestamp	Human readable timestamp
	unix_timestamp	Unix timestamp
User	address	IP address
	accessPoint_ID	Field used by Location API
	zone_ID	Field used by Location API
	resource_URL	Field used by Location API
	destination	Field used for the rerouting computation

Table 3.1. Main data structures of the MEC Application

All the implemented steps can be summarized in the following list;
the MEC Application:

1. provides a HTTP server in order to receive the users alerts through the HTTP GET request from the URL `http://10.0.0.1:8080/event_alert`

2. records the event and saves it into the Events List.

3. sends a UserInfo request (as described in chapter 3.4.1) to the MEC Location API in order to know the geographic coordinates of the signaling user and consequently the position of the accident.

4. creates and sends a message to signal the event to the 5T TCC server in the proper standard, DATEX II or S.I.MO.NE. (details about this step are presented in chapter 4).

5. sends a UserList request to the MEC Location API (described in chapter 3.4.1) in order to know the full list of the IP addresses corresponding to the users in the involved location area.

6. receives as a response from the 5T TCC server a message containing the travel times related to all the roads of the location area.

7. creates input files (trips with origins and destinations, travel times) for the rerouting computation for the traffic redirection and starts the simulation with the simulator (SUMO).

8. elaborates the routes obtained from the output of the simulation and sends the alternative path to each user present in the Users List whose destination is known.

3.4 Implementation of the Location API

In this thesis the MEC Location API has three main functions:

- To receive and to read data sent from the GPS location application
- To save the list of users and positions in an internal data structure
- To interact with the MEC application and provide the location information about a user or the full list of the users present in a zone.

As described in section 2.7 the MEC Location API is the standardized RESTful API that offers Location Service (LS) for MEC applications, exploiting the Zonal Presence Service described by Small Cell Forum in [15] and in [16] and supports queries

and subscriptions used over the RESTful APIs originally defined by Open Mobile Alliance (OMA) in [11]. The API provides the user positions retrieving information from the BTS; since currently there is no active implementation of this service, in the Vehicular Rerouting System it has been implemented through a temporary workaround based on an explicit user reporting, that makes the MEC Application work with the simulated version in accordance with the MEC Location API standard; with this solution the MEC Application can work through the same implementation even when the real standard RESTful API will be enabled.
The workflow of the interactions with the user device and MEC Application is shown in figure 3.4: the user sends its position through the vehicle device, afterwards the MEC Application sends a MEC UserInfo with subsequent Location API response, then it sends a request for UserList and gets a second response. The picture shows just one user position flow for the sake of simplicity, but there are as many flows as users; in the same way, just one userInfo request is represented but there are as many requests as event reports.

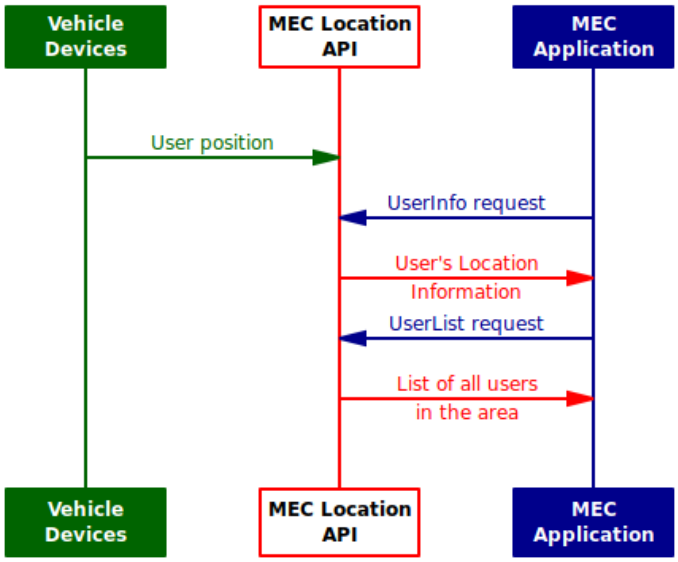


Figure 3.4. Interaction between Location API and MEC application

3.4.1 Location API: specifications

The main procedures used to interact with Location API presented in [10] are:

- **UE Location Lookup**: API client sends a request for location information about

one or more UEs to the corresponding resource; at each request corresponds one lookup result containing the location information of the UE.

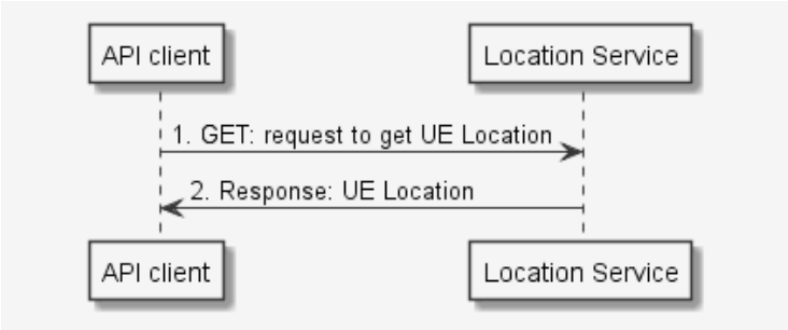


Figure 3.5. Flow of UE Location lookup [10]

- **UE Information Lookup**: API client sends a request in order to receive information of a list of UEs in a particular location; also in this case at each request corresponds one lookup result, that reports the list of UEs in the location area.

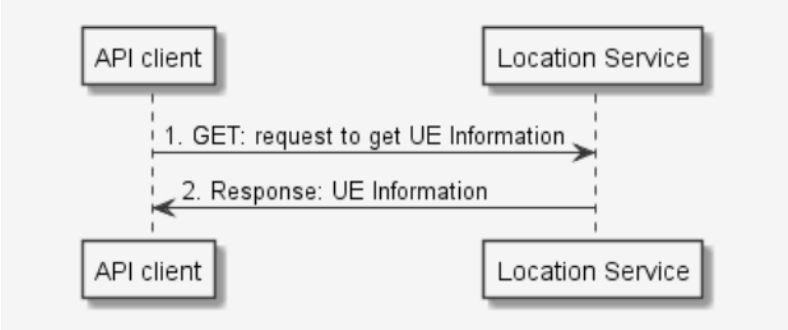


Figure 3.6. Flow of UE Information lookup [10]

- **UE Location Subscribe**: in this case ME application needs up-to-data location information of one or more UEs for a time lapse, so it subscribes to UE location notification by requesting the creation of a resource containing the subscription details, that contains UE identifier and a callback URL; then the LS returns a response which includes a resource URI with a subscriptionId; finally LS provides through UE Location notification the updated information, sending it to the callback URL. The notification service will last until the subscription is cancelled.

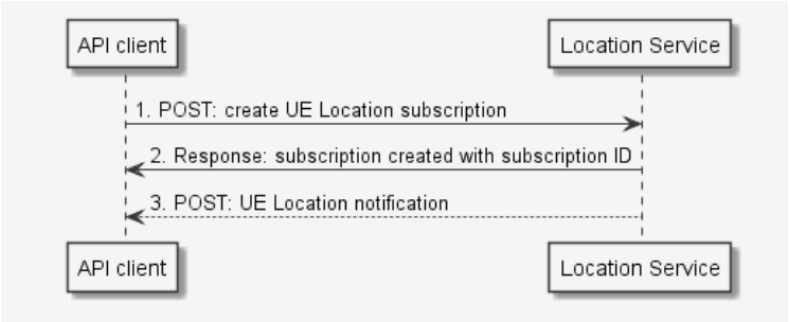


Figure 3.7. Flow of UE Location Subscribe [10]

- **UE Information Subscribe**: this procedure is used by applications to receive notifications of UE Information updates for the list of UEs in a particular location. Similarly to the UE Location Subscribe, the ME application asks for a notification service requesting the creation of a resource with subscription details (location area information and callback URL); then the LS returns a response which includes a resource URI with a subscriptionId; finally LS provides through UE Information notification the updated information, sending it to the callback URL. Also in this case, the notification service will last until the subscription is cancelled.

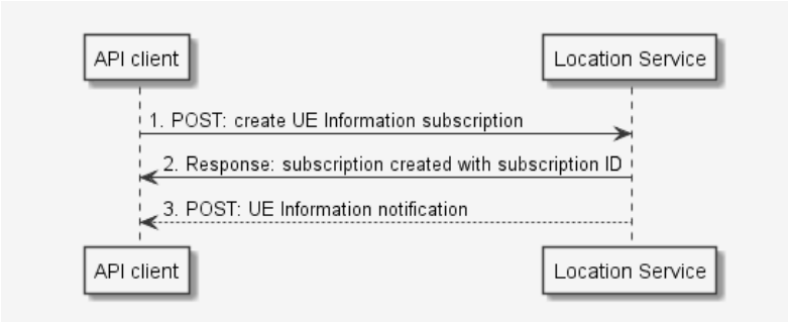


Figure 3.8. Flow of UE Information Subscribe [10]

- **Subscribe Cancellation**: the ME application stops the notification service by sending a request to delete the resource URI containing the subscriptionId; then LS returns a response to confirm the subscription cancellation.

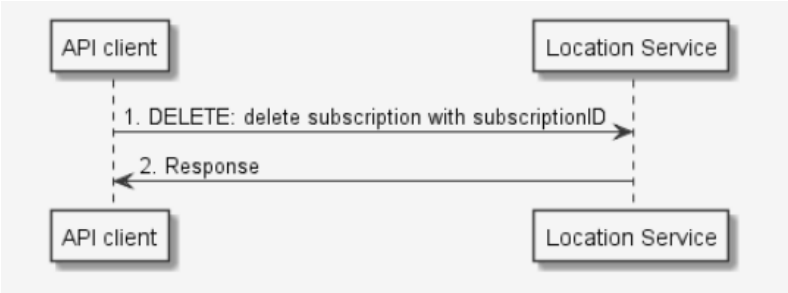


Figure 3.9. Flow of Location Subscribe Cancellation [10]

- **Radio Node Location Lookup**: this procedure enables the ME application to make a location enquiry about the radio nodes currently associated with the Mobile edge host; so LS returns a response with message body including the requested list of radio nodes and the location information of each radio node.

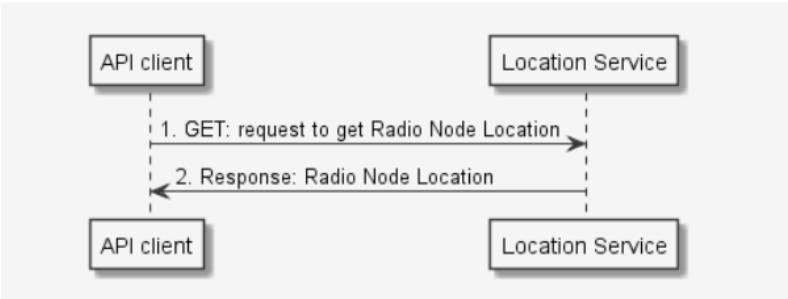


Figure 3.10. Radio Node Location Lookup [10]

For this reason, in the implementation of the MEC Location API, the used procedures are UE Location Lookup and UE Information Lookup.
Another important aspect of Location API specifications is the Data Model, that also in this case complies with the OMA Zonal Presence data types [11]. The data structures are [10]:

- **ZoneList**: type that contains list of zones
- **ZoneInfo**: type that contains zone information
- **AccessPointList**: type that contains list of access points
- **AccessPointInfo**: type that contains access point information
- **UserList**: type that contains list of users

- **UserInfo**: type that contains user information

3.4.2 Location API: implementation description

Since the TCC system is enabled, the Location API starts working and before the interaction with MEC application it has another main task: to acquire and save users positions. In the implementation of this thesis project the acquisition of the geographical position of each user is made through a location android application that leverages the Global Positioning System (GPS) receiver; in the tests the application GPSLogger [17] is used, since it has all the basic GPS positioning functions, it is very battery efficient and it gives the possibility of saving the positions measurements in a URL that we set as the URL resource of the MEC Location API.
Also in this case in order to test the functioning of the system the used physical interface is the hotspot WiFi created from the host and the android devices of the vehicles are connected to this network. After the settings GPSLogger is activated and it starts logging the location, composed of Latitude, Longitude, Accuracy and Timestamp; then this measurement is sent to the custom URL over HTTP GET request.
For the implementation of the data types, the OpenAPI Specification (OAS) compliant descriptions [18] published by ISG MEC and Platform Application Enablement (Mp1) API specifications on the ETSI hosted Forge site [19] are used. Indeed the OAS offers an open source framework for defining and creating RESTful APIs. YAML (or JSON) interface files compliant to the OAS are both human and machine-readable providing the means to describe, produce, consume and visualize RESTful web services [2]. Therefore these files have been the main reference for the realization of the Location API responses to the MEC Application.
When the MEC application receives the event alert, it immediately sends a UserInfo request about the signalling user, and the Location API replies with the geographical coordinates WGS84 (latitude and longitude) and the accuracy of the position measurement.

UserInfo.yaml 511 Bytes

```yaml
description: A type containing user information.
type: object
required:
- address
- accessPointId
- zoneId
- resourceURL
properties:
  address:
    $ref: '#/definitions/Address'
  accessPointId:
    $ref: '#/definitions/AccessPointId'
  zoneId:
    $ref: '#/definitions/ZoneId'
  resourceURL:
    $ref: '#/definitions/ResourceURL'
  locationInfo:
    $ref: '#/definitions/LocationInfo'
  contextLocationInfo:
    $ref: '#/definitions/ContextLocationInfo'
  ancillaryInfo:
    $ref: '#/definitions/AncillaryInfo'
```

Figure 3.11. UserInfo description [19]

LocationInfo.yaml 489 Bytes

```yaml
description: A type containing location information with latitude, longitude and altitude,
type: object
required:
- latitude
- longitude
- accuracy
properties:
  latitude:
    type: number
    format: float
    example: "80.123"
  longitude:
    type: number
    format: float
    example: "70.123"
  altitude:
    type: number
    format: float
    example: "10.0"
  accuracy:
    type: integer
    format: int32
    example: "10"
```

Figure 3.12. LocationInfo description [19]

The figure 3.11 represents the YAML description of the UserInfo, composed of

address, accessPointId, zoneId, resource URL as required and mandatory fields, and some optional fields as locationInfo, contextLocationInfo and ancillaryInfo, but in the thesis project implementation only locationInfo has been used.
The locationInfo type, shown in figure 3.12, requires latitude, longitude and accuracy in float number format.

An example of the Location API UserInfo response is reported below:


```
{
  "userInfo": {
    "address": "acr:10.0.0.2",
    "accessPointId": "00101000000000000000000000000001",
    "zoneId": "zone01",
    "resourceURL": "http://10.0.0.1:8081/location/v1/users/acr%3A10.0.0.2",
    "locationInfo": {
      "latitude": "45.112",
      "longitude": "7.619",
      "altitude": "10.0",
      "accuracy": "15"
    }
  }
}
```


The second request that Location API receives from MEC Application is the UserList, in order to know IP addresses of all the users. This type is composed of the resource URL and the full list of connected users, as described in the YAML description in figure 3.13.

UserList.yaml 277 Bytes

```yaml
description: A type containing list of users.
type: object
required:
- resourceURL
properties:
  user:
    description: Collection of the zone information list.
    type: array
    items:
      $ref: '#/definitions/UserInfo'
  resourceURL:
    $ref: '#/definitions/ResourceURL'
```

Figure 3.13. UserList description [19]

An example of the Location API UserList response is reported below:


```
{
  "userList": {
    "user": [{
        "address": "acr:10.0.0.2",
        "accessPointId": "00101000000000000000000000000001",
        "zoneId": "zone01",
        "resourceURL": "http://10.0.0.1:8081/location/v1/users/acr%3A10.0.0.2"
      }, {
        "address": "acr:10.0.0.5",
        "accessPointId": "00101000000000000000000000000001",
        "zoneId": "zone01",
        "resourceURL": "http://10.0.0.1:8081/location/v1/users/acr%3A10.0.0.5"
      }, {
        "address": "acr:192.0.0.19",
        "accessPointId": "00101000000000000000000000000010",
        "zoneId": "zone01",
        "resourceURL": "http://10.0.0.1:8081/location/v1/users/acr%3A10.0.0.19"
      }
    ],
    "resourceURL": "http://10.0.0.1:8081/location/v1/users"
  }
}
```

Chapter 4

Protocols for the information exchange in Traffic Control Centers

As outlined in chapter 3.3, the MEC Application needs to interact with 5T TCC Server: first because it has the purpose of reporting the road obstruction event to the TCC and subsequently because it requires a response from the server, that has a more global and accurate view about the traffic and roads conditions, in order to compute the rerouting for traffic redirection. For this exchange of information, the chosen communication protocols suggested and already used by 5T TCC are DATEX II and S.I.MO.NE.
During the first implementation and in the first tests of the system, the messages created by the MEC Application followed the DATEX II specifications and syntax, but since the granularity of the map used in this protocol (in the version of the Italian profile) could not satisfy the requirements of the system, that is thought to work in an urban environment, the choice of the standard has switched to S.I.MO.NE.. Indeed the second one is more suitable to map all the roads present in an urban context and it is used by 5T to handle the communication between its TCCs in Turin.
Therefore the MEC Application has first been implemented in order to create DATEX II files, but at a later stage other messages have been created to comply with S.I.MO.NE. protocol; since currently no real interaction with 5T server is available, both the standards are implemented in the application, so a future stakeholder of the project can have the choice on which one to adopt.
The next chapter will present a description with the detailed specifications of DATEX II in section 4.1 and of S.I.MO.NE. in section 4.2.

4.1 DATEX II

4.1.1 DATEX II: background

DATEX II (DATa EXchange) is a standard developed by the European Commission for the exchange of traffic information and traffic data between traffic centers, service providers, traffic operators and media partners [20]. The first standard developed for these purposes, DATEX, has been replaced by DATEX II since it had too many limitations to be used in modern ITS and it was hardly suitable in case of several heterogeneous data sources. Indeed one of the main features of this standard project is the interoperability: it has the task of creating a seamless data exchange across boundaries and among different European countries and types of platforms. Currently DATEX II is a multi-part standard maintained by CEN Technical Committee 278, Road Transport and Traffic Telematics [21].
In addition to the data exchange, another main design characteristic of DATEX II is the possibility to separate concerns, in fact it can be used in different contexts and applications, and to allow users to create their own profile through a well defined but extensible modelling of data [22].
The main DATEX II use cases are the ITS that require a dynamic management of data and real-time traffic information: it can be used for linking traffic management and traffic information systems, for rerouting, network management and traffic management planning, lane control systems and related applications like dynamic speed limits and overtaking control; moreover it can be useful for different kinds of applications employed for road safety or that need information exchange between individual vehicles and traffic management (Car-to-infrastructure systems) or between management systems for different modes (multi-modal information systems) [20]. Other typical examples of DATEX II applications are Variable Message Signs (VMS), used mostly in motorways, and Parking Publications, especially used for secure truck parking services.
All the DATEX II usages concern the employment by different European countries in the motorway and highway context. Each country defines its own profile adapting the standard to the current needs and innovative proposals; for example, the Austrian motorway operator corporation ASFINAG created different DATEX II profiles with the project ECo-AT, born to create harmonized and standardized cooperative ITS

applications jointly with partners in Germany and in the Netherlands. In particular the main use cases involved in this project are: Dedicated Short Range Communications (DSRC) Protected Zones, In-Vehicle Infotainment (IVI), Other DENM (Decentrilized Environmental Notification Message) and Road Works Warning (RWW).

4.1.2 DATEX II: content and modelling

In order to enable its key objectives, DATEX II covers a wide range of content in the road traffic and transport domain. Indeed it has been created to be used all over Europe and in different contexts, aiming at becoming the leading reference model for information exchange. Therefore the model covers: level of service on the network, both in terms of messages for specific situations or as an overall status on the network, travel times, be it on short network links or for long distance travel itineraries, all types of accidents, road works, road infrastructure status, closures, blockages and obstructions, road weather, all kinds of traffic related measurements, public events with impact on traffic, current settings of variable message signs [20].
The last version of DATEX II is the v3 but since in the Italian profile the used version is v2.3, all the reference documents used in this thesis are v2.3.
The modelling approach used is Unified Modelling Language (UML), which is a stable environment for system specification and defines the contents of DATEX II Traffic and Travel publications independent of the exchange mechanism or implementation technology [23]. Indeed this standard development makes a clear distinction between data and exchange mechanisms, and a further distinction between platform independent modelling aspects (PIM), described by UML, and platform specific modelling aspects (PSM). Separating traffic domain data modelling and data exchange specifications (referring to information and communication technology) makes the application of the standard easier from both users and developers point of view.
In order to make the data modelling and usage more intuitive and readable for the user, DATEX II provides a document called "Data Dictionary" [24] that describes the meaning of each field in the data modelling.
Data models are divided into levels: level A, level B and level C. Level A represents the basic usage of DATEX II and so it includes a very rich set of models. If some applications need to use some data concepts that are not present in level A data

modelling, they can add some model extensions through level B data; in this case the standard guarantees the interoperability between the two levels. On the contrary, in case of applications that require a completely different new kind of data modelling not available in level A and level B, they can still create their own data models with level C, but they lose the interoperability benefit with other levels data contents; even if not compliant with other data models, level C data should use common modelling rules and common exchange protocols [23].
The data modelling can be mapped to several platforms, but currently the mapping for the exchange message syntax uses XML schema, that is designed to fit a certain area of applications. The schema is the tool to understand the content of the exchanged data [23].

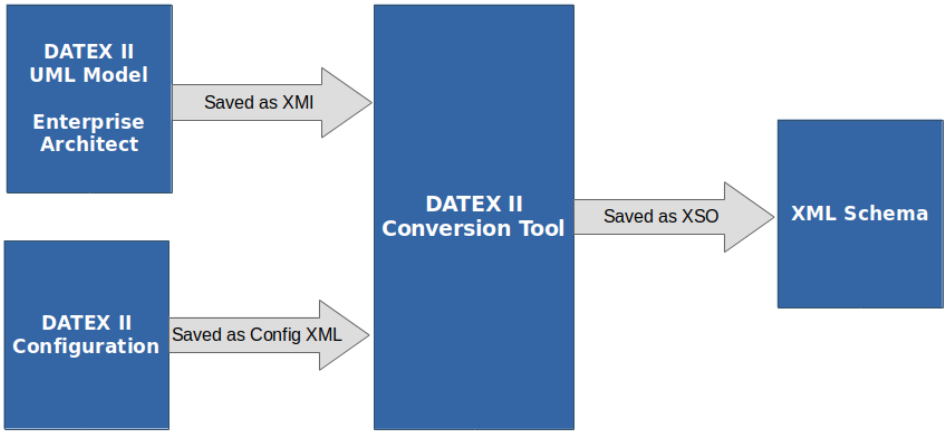


Figure 4.1. Conversion from UML to XML schema [23]

The workflow in figure 4.1 shows the procedure to obtain the XML schema from the data models: first the UML model is exported from the modelling tool Enterprise Architect into an XMI (XML Metadata Interchange) file, then the conversion tool, available on DATEX II website [24], converts it to an XML Schema based on the DATEX II configuration file.

Since the data modelling is very detailed to be suitable for different kinds of usages, the best way for a user to define which are the used data models and operating modes is to create a profile, in order to define a customized subset of options offered by a standard, adding or removing functionalities depending on their needs.

The Italian profile has been created for the Italian motorway operators and is described in the document [25], made by the technical group Mare Nostrum and presented in Aiscat (Associazione italiana società concessionarie autostrade e trafori), for the management of the variable message signs (PMV). The specifications include the data exchange made just through HTTP and XML v1.0, called Low Cost Profile (Simple HTTP Pull) and exchange through Web Service, with Push mode.
In the Italian profile these types of publications have been analysed:

- Situation: total events
- VmsTable: PMV Registry
- Vms: PMV State and Messages
- MeasurementSites: Sensors registry and control units
- MeasuredData: Data and Sensors status
- ElaboratedData: Travel times, elaborated data
- Generic: Extensions

In the implemented MEC Application, since the purpose is to signal an event, the used type of publication is *Situation*. The *Situation* type is modelled as shown in figure 4.2.

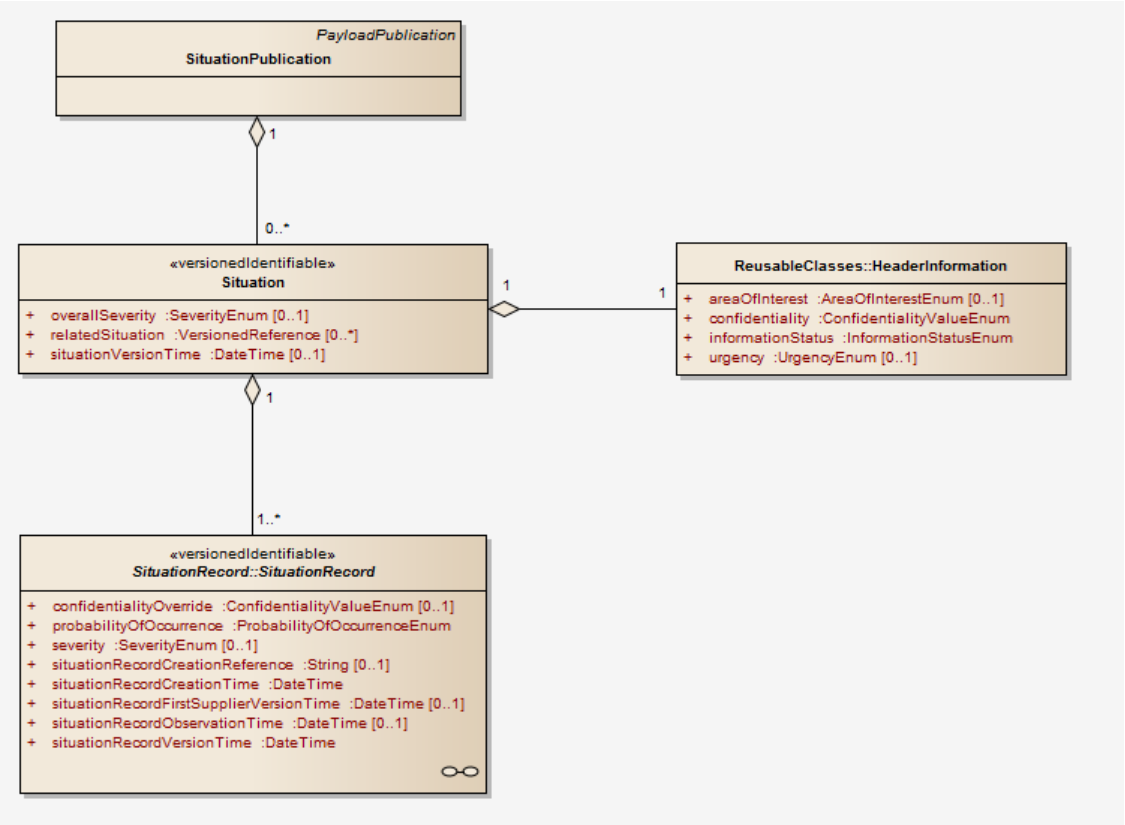


Figure 4.2. UML model of Situation type [21]

The DATEX II *Situation* messages are created by the MEC Application following the data format provided by 5T. Therefore only the useful types will be shown in this thesis. Since in the UML the models have a hierarchical structure, in the next tables the types are shown with a tree structure, where each type is a child of the type on the top-left, if present.

The message is first composed of a header called *Exchange* that includes the country and the unique national name of the supplier, as shown in 4.1:

Exchange		
supplierIdentification	country:	it
	nationalIdentifier:	CinqueT

Table 4.1. Exchange description

The second part of the message is the information content of the *Payload Publication*: in particular the used type is *Situation Publication*, composed of *publication Time*, *publication Creator* and *Situation*, whose types and corresponding hierarchies are shown in table 4.2:

Situation	
overallSeverity	
situationVersionTime	
headerInformation	confidentiality informationStatus urgency
SituationRecord	*see table 4.3*

Table 4.2. Situation description

SituationRecord		
situationRecordCreationTime		
situationRecordVersionTime		
confidentialityOverride		
probabilityOfOccurrence		
severity		
validity	validityStatus validityTimeSpecification	overallStartTime
groupOfLocations	alertCPoint	AlertCLocationCountryCode AlertCLocationTableNumber AlertCLocationTableVersion AlertCDirection AlertCMethod2PrimaryPointLocation
accidentType		

Table 4.3. SituationRecord description

Some of the more meaningful parameters present in *Situation Record*, described in

table 4.3, that should be highlighted are:

- **confidentialityOverride**: specifies the recipients that can receive the information; some typical values are for example *internalUse*, *no Restriction* or *restricted To Authorities*.
- **probabilityOfOccurrence**: represents the probability the event is verified and it can take only the values *certain*, if the event is certified by official information sources, *probable*, if the event is not verified but have a probability of happening greater than zero, or *risk of*, if the event is not certified by official sources but have been reported by other sources.
- **validity**: is used to specify the temporal validity of the reported event; the mandatory attribute *validityStatus* can take the values **active**, in case of a valid and active event, **suspended**, if the information is temporarily suspended, and **overrunning** if the event has a fixed date of start and a date that indicates the end of the event. This last value is not suitable for the implementation of the MEC Application since it is not applicable on accidental events.
- **GroupOfLocations**: indicates one or more physically separate locations and can also be used to state an itinerary. Alert C is the coding used to reference geographical points, defined by a location code that identifies a mapped geographical point and one offset (if describes only one point) or more offsets (in case of a road), that indicate the distance from *alert C Point* in a direction specified by *Alert C Direction*. The coding used for the location code is TMC (Traffic Message Channel).

The *Situation Record* type can be distinguished by three different types, and the selected ones for the implementation in the MEC Application are: *Accident, Obstruction* and *Equipment Or System Fault*. The three types of messages have a similar syntax with some exceptions: in the *Obstruction* type a field *Environmental Obstruction* that specifies the type of obstruction (for example *fallenTrees*) is added, while in the *Equipment Or System Fault* two new fields are added, *equipment Or System Fault Type* (such as *not Working*) and *faulty Equipment Or System Type* (such as *level Crossing*), that describe the entity of the system fault.

4.2 S.I.MO.NE.

4.2.1 S.I.MO.NE.: background

After some evaluations about the experiments and simulations of the MEC Application system project, the best protocol solution results to be S.I.MO.NE. (Sistema Innovativo di gestione della MObilità per le aree metropolitaNE), that is the most used one for the information exchange between TCC and sensors for the traffic in 5T and furthermore it has a more accurate reference road database. Indeed S.I.MO.NE. is a standard protocol that allows the communication between floats control centers, Vehicles Services Centers (CSV) and Local Control Centers of Mobility [26].
The project was born in 2012 with a deal of the cities of Turin, Bologna, Cagliari, Florence and Genoa. The main objective of this standard is to offer real-time traffic monitoring, traffic lights management, atmospheric pollution measurements and mobility services for users, exploiting available local platforms. Moreover the design is based on the concept of using floating cars for the data acquisition (Floating Car Data) in order to extend the capillarity of the existing systems for the information acquisition, reducing the necessary infrastructures and improving the user experience [27].

4.2.2 S.I.MO.NE.: content and modelling

Figure 4.3 shows the framework of S.I.MO.NE. project. In particular the communication protocol is applied at 1 and 3.

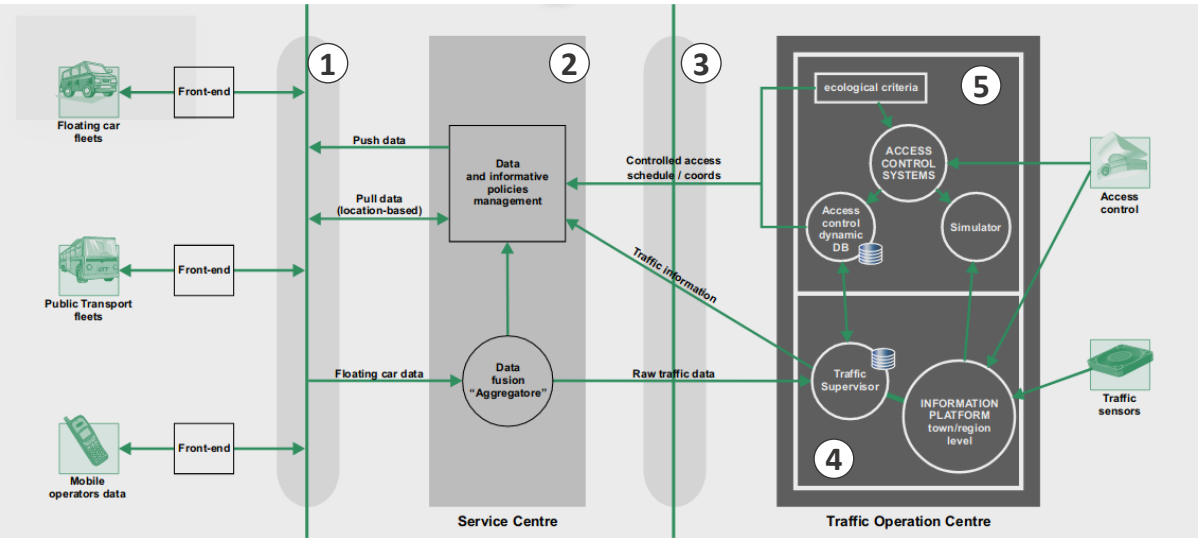


Figure 4.3. S.I.MO.NE.: project framework [26]

Communication in 1 is a bidirectional communication where floating cars center (front-end) transfers Floating Cars Data (FCD) towards the Vehicles Services Center, that sends a response with elaborated traffic data, traffic events and ZTL (Limited Traffic Zone) information. Communication in 3 is a bidirectional communication where Vehicles Services Center sends elaborated traffic data from FCD (such as travel times) and Traffic and Mobility Center provides events, traffic information and ZTL information.

Data types refer all to quantities or characteristics of the road network, therefore a geographic reference graph needs to be specified. 5T refers to a graph defined on purpose for its TCC (graph RR).
For the Raw Data and ZTL management the most common reference system is the one that leverages geographic coordinates, referenced with World Geodetic System 1984 (WGS84), that defines the reference ellipsoid used in GPS system for the position calculation [26].

Data types provided by S.I.MO.NE. [26] are:

- RD (Raw Data)

- MRD (Map-matched Raw Data)
- TDP (Tempi di percorrenza - Travel Times)
- OD (Trips Origin/Destination matrices)
- EdT (Eventi di Traffico - Traffic Events)
- ZTL (Zona a Traffico Limitato - Limited Traffic Zone)
- FdT (Flussi di Traffico - Traffic Flows)
- PK (Parkings)
- TLight (Traffic Lights Data)
- TState (Traffic State)

In the thesis system, the MEC Application uses two data types for the interaction with 5T Server: for reporting the event the data type EdT is used, whereas for the 5T response the data type TDP is used.
EdT type, based on a XML schema (XSD) called `traffic_info`, is defined in table 4.4.

Name	Description
SOURCE	ID of the event provider that confirmed and inserted the event
SITUATION_ID	Event situation identifier
EVT_ID	Event identifier
ROAD	ID and description of the road corresponding to the event location (TMC coding)
LOCATION	Location of the event (a single point or two points in case of road lane, TMC coding)

Table 4.4. Traffic Event (EdT) description

TDP type, based on a XML schema (XSD) called `traffic_data`, is defined in table 4.5.

Name	Description
lcd1	Location code LCD of the first point of the road
lcd2	Location code LCD of the last point of the road
start_time	timestamp of the starting time of the measurements
end_time	timestamp of the ending time of the measurements
time	average time of the travel time [s]
speed	average speed of the road travel [km/h]
n_vehicles	number of vehicles detected for the measurements
std_dev	standard deviation of the samples distribution
accuracy	percentage accuracy of the measurement
estimated_speed	estimated speed [km/h]
q_idx	estimate quality index (1=min, 5=max)
vehicle_type	vehicle type (AU = motor vehicle)

Table 4.5. Travel Time (TDP) description

The protocol definition assumes that between the client and the service provider the data types, the reference system and the exchange mode have been agreed in a previous phase. In order to reduce elaboration times and resources, the exchange mechanism is the Push Mode, where the service provider sends periodically the data to the client, without an explicit request. If the client needs to send a data request, a second option is provided consisting of a Pull Mode [26].
With the Push Mode the service provider sends the data complying with the predetermined data types established with the client. The used method is HTTP POST, using a web service identified by a URL such as:

`http://<provider_host>/post_traffic_data`

With the Pull Mode the service provider prepares the data complying with the predetermined data types, then the client sends an explicit request through a HTTP GET request, invoking a web service identified by a URL such as:

`http://<provider_host>/get_traffic_data`

Chapter 5

Traffic rerouting with SUMO: analysis and results

The last part of the thesis focuses on the last important step of the MEC Application operations: data processing and elaboration.
The Vehicular Routing Advisor requires an evaluation of the customized alternative route for each user in the MEC Location area, after having received as response of 5T Server the file containing all the travel times of the urban roads.
In order to apply to the system an efficient rerouting algorithm, the MEC Application has been integrated with Simulation of Urban MObility (SUMO) package; indeed this simulator allows to manually create or edit a network (through the tool `netedit`) or to load a detailed road network of a city.
At a preliminary stage, different simulations have been made in order to choose the best SUMO algorithm for routing, that results to be MAROUTER, since it is the only one that performs a macroscopic routing, taking into account not only the shortest path for each flow, but the whole traffic demand. The router is performed with an incremental assignment method, described in section 5.1.
The simulation requires a file containing all the routes of each vehicle, that is generated with the MAROUTER command:

```
marouter --weight-files wfile.xml --route-files TRIPS/trips_10.xml
--net-file network.net.xml -o man.rou.xml
```

In SUMO each network is defined as a graph, where the roads correspond to

the edges and the junctions to the vertices; the weight file (`wfile.xml`) represents the cost of each edge belonging to the network (`.net.xml`) in terms of travel time of the road in seconds, while the route file `trips_10.xml` is the file that describes all the flows of the simulation, specifying the id, the departure time, the origin and the destination of each vehicle.
In the Vehicular Rerouting Advisor, this file is generated by the MEC Application setting as origin the position provided by the Location API and as destination the position provided by the user application. The weight file is generated exploiting the response of 5T server with all the updated roads travel times and setting the highest possible travel time to the edge in which the obstruction has been indicated.
The MAROUTER command generates a new route file `.rou.xml` that contains all the routes that each flow can choose, the corresponding probability and all the edges that should be traveled for each route.
The simulation is started with the command `sumo` or `sumo-gui` to open the graphical user interface, providing as input the configuration file `.sumocfg`, that must be set with the network file `.net.xml` and the route file `.rou.xml`.
All the simulations have been made with a different number of vehicles in order to increase the traffic load: different files `trips_`x_i`.xml` have been created, where x_i represents the number of vehicles present in each file and is defined as $x_i = x_{i-1} + 10$, with $0 < i \leq 20$ and $X_0 = 0$.

The rest of the chapter is organized as follows: section 5.1 illustrates the SUMO macroscopic traffic assignment and routing algorithm; section 5.2 presents the simulations made in ideal conditions and with a simple Manhattan topology; in section 5.3 some ideal cases on the Turin road network used by the MEC Application are proposed. These experiments aim at showing the behavior of the MAROUTER algorithm.

5.1 MAROUTER: Incremental assignment

MAROUTER is the macroscopic traffic assignment router implemented in SUMO; it generates routes receiving as input an Origin-Destination file of the flows. The two currently implemented methods in SUMO are Incremental assignment and SUE (Stochastic User Equilibrium) assignment. This one has not be chosen for the

implementation of the Vehicular Rerouting Advisor since it takes into account travel times perceptions among the vehicles and not the weights set into the travel times file, making difficult to change the weight of the edges dynamically, as in case of the MEC Application use [28].
Therefore the used method is Incremental, that assigns the given O-D matrix with a certain proportion and iteratively (the default number of iterations is 20). The algorithm operates assigning at each iteration a fraction of the total traffic demand to the fastest route with an all-or-nothing assignment; then the travel times of the edges are consequently updated, using a hard-coded capacity-constraint function that depends also on the speed limits and the link volumes.
As described in the results shown later in 5.2 and in 5.3, this traffic assignment does not converge to an user equilibrium solution, due also to the fact that, once a route has been assigned to a flow, it cannot be removed [29], but even if it does not produce a balance solution, it is useful to find one or more alternative routes in case of congestion.
Moreover the SUMO options allow to select among different routing algorithms; in the following experiments the used one is Dijkstra algorithm.

5.2 Manhattan road network

For the sake of simplicity, the first analysis has been made on a regular topology, that has been built through the command:

```
netgenerate -g --grid.number=5 --grid.length=100 -o manhattan.net.xml
```

The generated network is a 5x5 Manhattan topology, in which each road has a length of 100 meters and a maximum allowed speed of 50 km/h (standard travel time: 7.2 s).
The preliminary simulations are made on this network, generating trips that have the same starting point (S), the same destination (D) and the same departure time. Since all the vehicles have the same starting and ending positions, the simulation environment simulates an initial queue where the departure time is set later than the expected one; this randomness is the reason why each simulation presented in this

chapter has been repeated 50 times with different seeds.
The results of these preliminary tests revealed that since the vehicles have all the same single lane as destination, a congestion is generated for the vehicles arriving from the left routes, compromising the analysis. For this reason, the topology of the road network has been modified, duplicating the starting lane and the arriving one. Then a third lane is added to the arrival edge in order that each flow can exploit a lane at the maximum capacity, without creating congestion due to road priorities.

5.2.1 Scenario *1a*: standard case with equal weight on all the edges

In the first experiment the weight file is set in order to have the same cost in terms of travel time for each road of the network, with a value equal to 7.2 seconds.
Routes chosen by MAROUTER algorithm are highlighted in green.

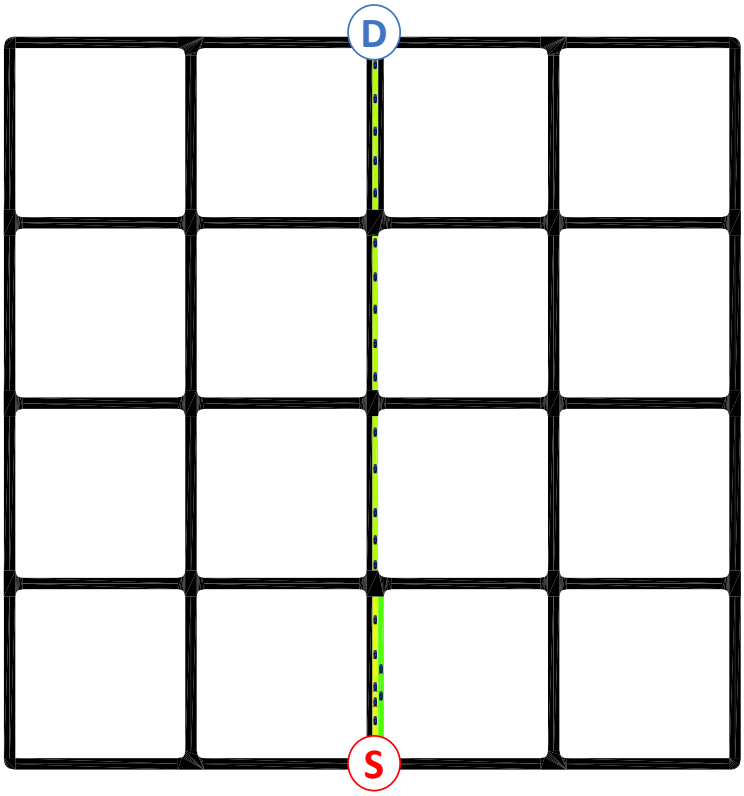


Figure 5.1. Scenario *1a*: Routing

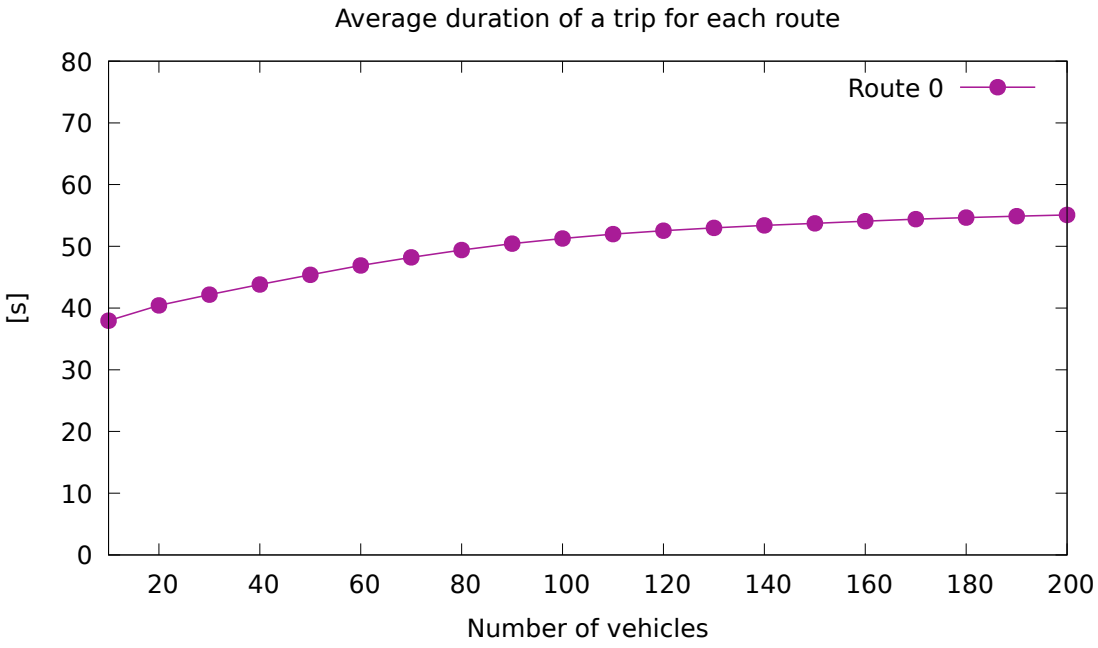


Figure 5.2. Scenario *1a*: Average duration of the trips

Figure 5.1 shows that the algorithm chooses as best path the shortest route between the starting point and the destination. Even increasing the number of the vehicles, the selection of route 0, corresponding to the straight path, does not change and the probability of the chosen route remains constant to 1, since no other routes are selected. The average duration of the trips, obtained computing the difference between the arrival time and the departure time of each vehicle, is represented in figure 5.2; the duration slightly increases in function of the increasing traffic load, reaching the value of 56 seconds.

5.2.2 Scenario *1b*: closed edge with equal weight on all the open edges, one single lane per edge

In the next scenario one of the four edges required to travel the direct route between S and D is closed in order to simulate a road obstructed by an accident, as in the use-case of the MEC Application of the Vehicular Rerouting System, and the weight file is set to all equal values (equal to 7.2 seconds) except for the closed edge, that is set to 1000 seconds in order to impose the route choice on other routes; in this case the algorithm changes the routing and is able to find four new best paths, all with the same cost (figure 5.3):

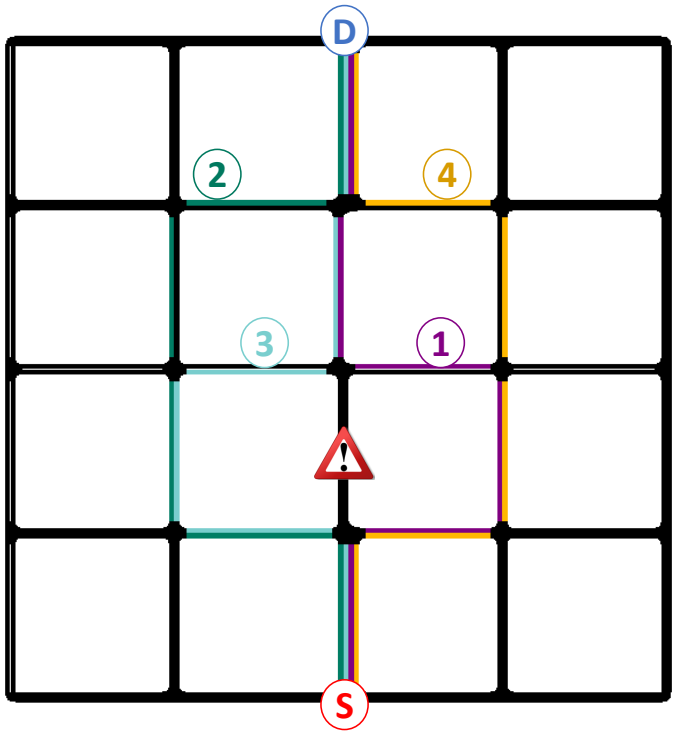


Figure 5.3. Scenario *1b*: Routing with a closed edge and four best equal routes

Figure 5.3 shows the alternative routes found by the algorithm: four routes of equal length; since the edges have all the same cost, also the four selected routes have the

same cost. Nevertheless the probability of the routes choice is not equal for all the four alternatives as expected. The probability of each route selected in this scenario is shown in the table 5.1:

Route	*Probability*
Route 1	0.2
Route 2	0.3
Route 3	0.2
Route 4	0.3

Table 5.1. Scenario *1b*: Probability of route selection

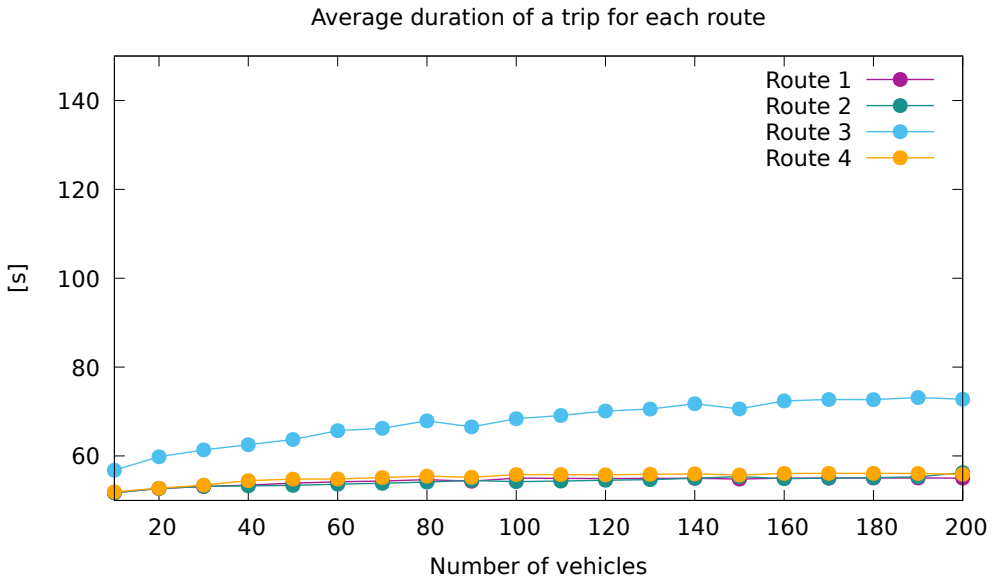


Figure 5.4. Scenario *1b*: Average duration of the trips, edge closed

The difference between the probabilities can explained by the fact that the two central routes 1 and 3 share one edge, which consequently slows down the viability of the vehicles; this is the reason why the router assigns to them a lower probability (0.2). The graph of the average duration of the trips in figure 5.4 shows that route 3 has a longer duration trip; this difference with the other three routes increases with the traffic load: this is due to the fact that routes 1 and 3 have an edge in common and to the presence of the priority to the right, in which drivers are required to give way to

vehicles approaching from the right at the junctions; therefore the waiting times of the drivers approaching from the left increase with a higher number of vehicles and in this case, since the topology provides only one available lane for two flows in the central shared road, a waiting queue is generated creating congestion (figure 5.5).

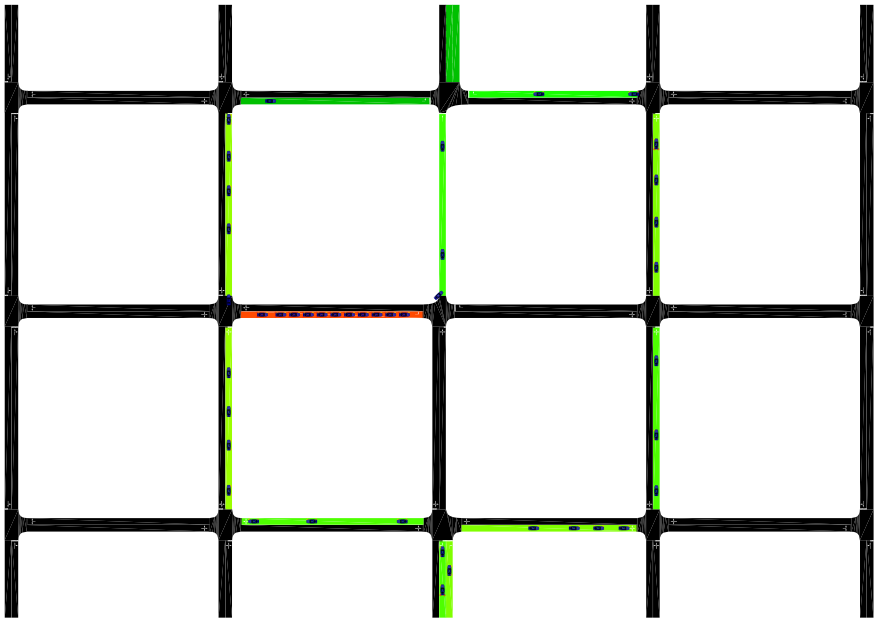


Figure 5.5. Scenario *1b*: waiting queue (red edge) generated by priority to the right

5.2.3 Scenario *1c*: closed edge with equal weight on all the open edges, two lanes in the common edge

In order to avoid the queue described in 5.2.2 and remove the advantage of the routes coming from the right direction due to priority, in this scenario the network topology has been modified through `netedit`: the edge in common between routes 1 and 3 is now composite of two lanes.

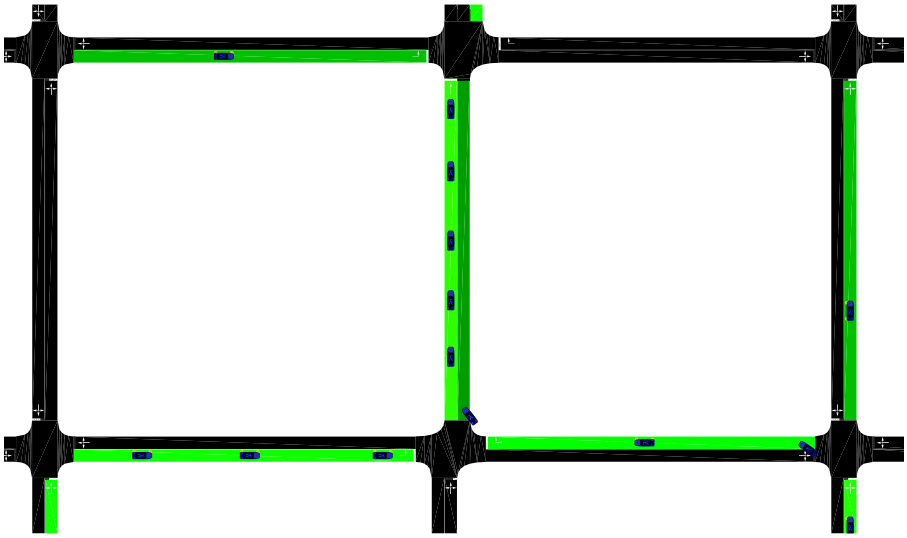


Figure 5.6. Scenario *1c*: zoom of the network, two lanes per edge

In this case the results in terms of average duration of the trips are more predictable and in accordance with the equal costs of the routes. The probabilities of each route selection are changed respect to the previous scenario:

Route	*Probability*
Route 1	0.25
Route 2	0.25
Route 3	0.25
Route 4	0.25

Table 5.2. Scenario *1c*: Probability of route selection

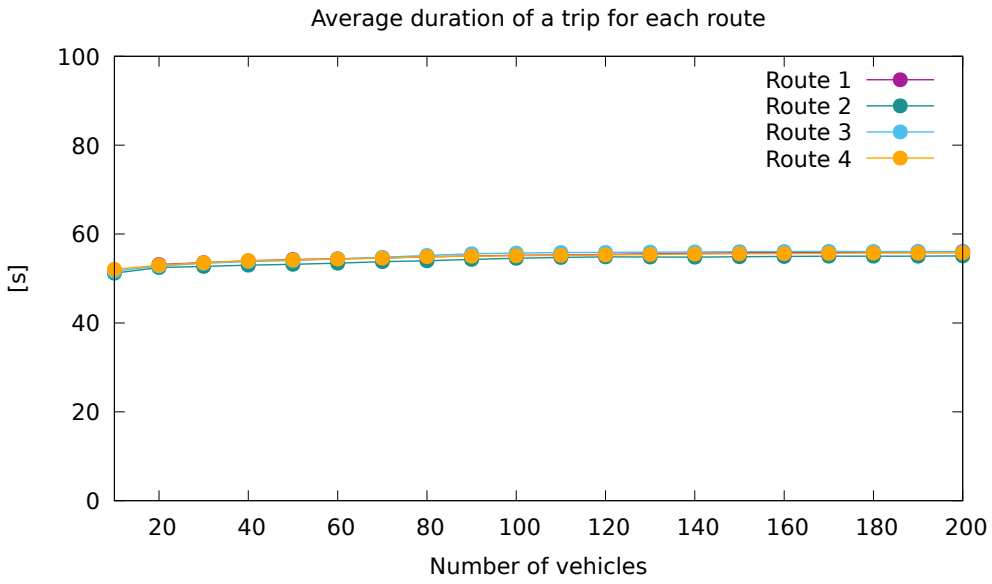


Figure 5.7. Scenario *1c*: Average duration of the trips, edge closed and double lane

As illustrated in table 5.2 and in figure 5.7, now the routes have all the same probability and the same average duration of the trips.
This experiment reveals also that in case of high traffic load the value of the average duration of a trip of the alternative routes (1, 2, 3 and 4) is similar to the value of the shortest one (route 0). On the contrary, in case of low traffic load, the shortest path is always the more convenient.

5.2.4 Scenario *1d*: closed edge with different weight and two shortest routes

For a clearer analysis, the weight file is then modified in order to have only two best routes, decreasing the maximum allowed speed of the more central roads and consequently increasing their travel times. The algorithm selects the two fastest paths route 2 and route 4, that in this case have the same travel cost and conditions.
The resulting scenario is shown in figure 5.8:

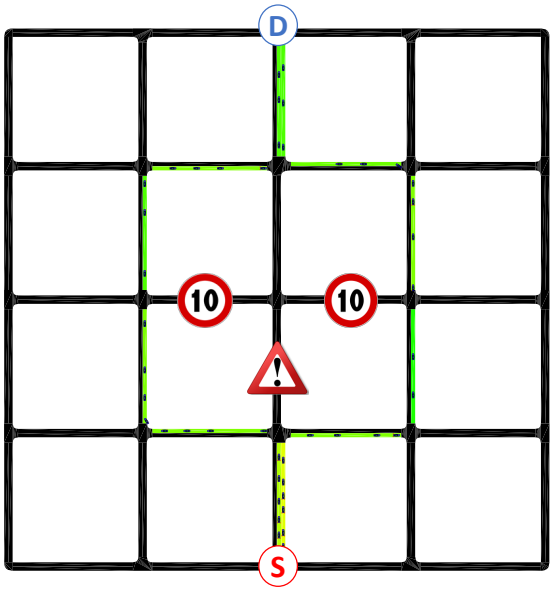


Figure 5.8. Scenario *1d*: Routing with a closed edge and two best equal routes

In this case, since the two paths have equal cost and equal length, both the probability of selecting each one of the two routes and the average duration of the trips are equal, independently from the number of vehicles set in the simulation.

Route	*Probability*
Route 1	0
Route 2	0.5
Route 3	0
Route 4	0.5

Table 5.3. Scenario *1d*: Probability of route selection

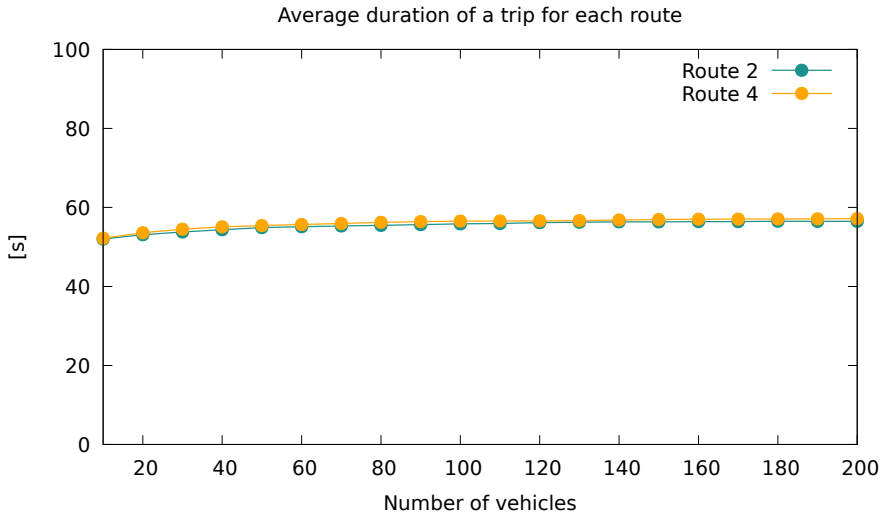


Figure 5.9. Scenario *1d*: Average duration of trips, closed edge and two best equal routes

5.2.5 Scenario *1e*: closed edge with different weight on all the edges

The last simulation made on the Manhattan topology is obtained modifying the weight file, making the right route (route 4) have a less cost respect to the left route (route 2), modifying also in this case the maximum allowed speed, setting an edge to 46.8 km/h, and the travel times of the route 4, setting the modified edge at a travel time of 7.7 seconds. Therefore the difference in terms of travel times between route 2 (7.2 seconds per edge) and route 4 (7.7 seconds on one edge, 7.2 seconds on all the others) is lower respect to the difference with the other routes costs (12 seconds on all the other edges).

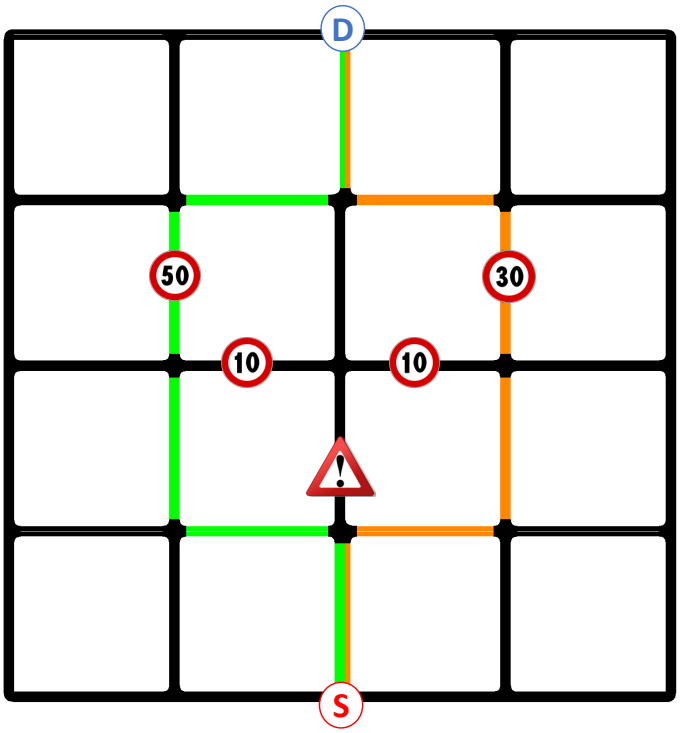


Figure 5.10. Scenario *1e*: Routing with a closed edge and two best different routes

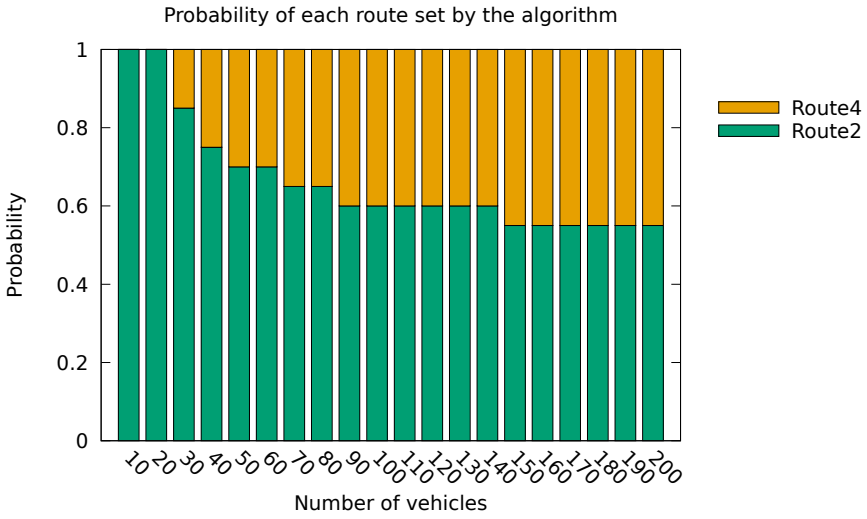


Figure 5.11. Scenario *1d*: Probability of route selection, closed edge and different travel times (7.7s on route 4)

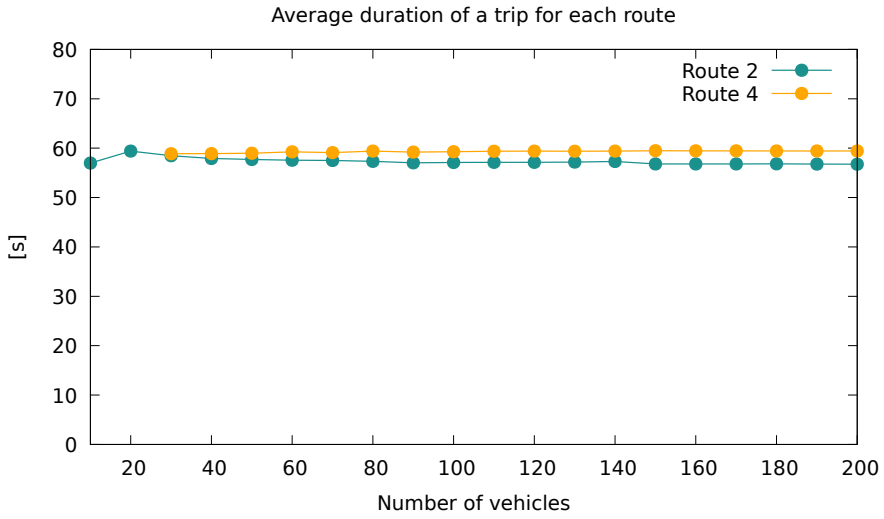
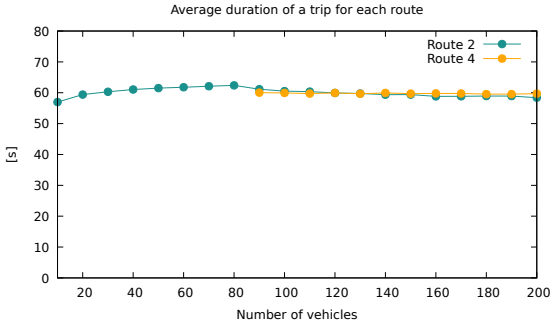


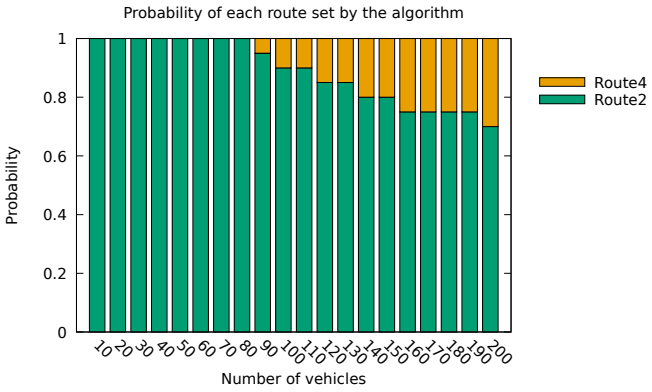
Figure 5.12. Scenario *1e*: Average duration of trips, closed edge and different travel times (7.7s on route 4)

Histogram in figure 5.11 shows the results of the simulation corresponding to this scenario: up to 20 vehicles scheduled in the traffic demand, the chosen route is the fastest in terms of travel time (route 2), but increasing the traffic load the probability of choosing the second best route (route 4) increases, up to reach a probability of 0.58 for route 2 and 0.42 for route 4.
The average duration of the trips (figure 5.12) of route 2 increases until the MAROUTER chooses the alternative route 4: when the alternative path is added to the routing the average duration of the trips of route 2 becomes lower; moreover it becomes almost constant with an increasing traffic load for both the routes; as expected, route 2 still results faster than the alternative even with a higher traffic load.

Other experiments with the same scenario are made, modifying the maximum allowed speed of the modified edge of route 4.
The graphs 5.13 and 5.14 represent the behavior of the MAROUTER for different values of maximum speed of route 4: decreasing the maximum allowed speed of an edge, the algorithm assigns to the corresponding route a higher cost, decreasing its probability of being selected.

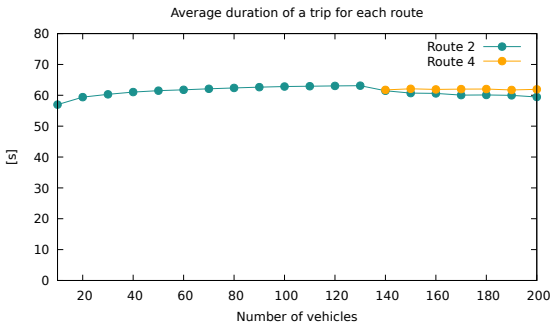


(a) Average duration of the trips

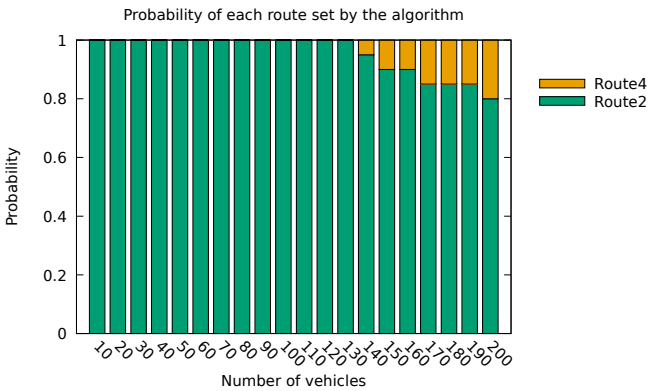


(b) Probability of route selection

Figure 5.13. Edge of route 4 set at 40 km/h



(a) Average duration of the trips

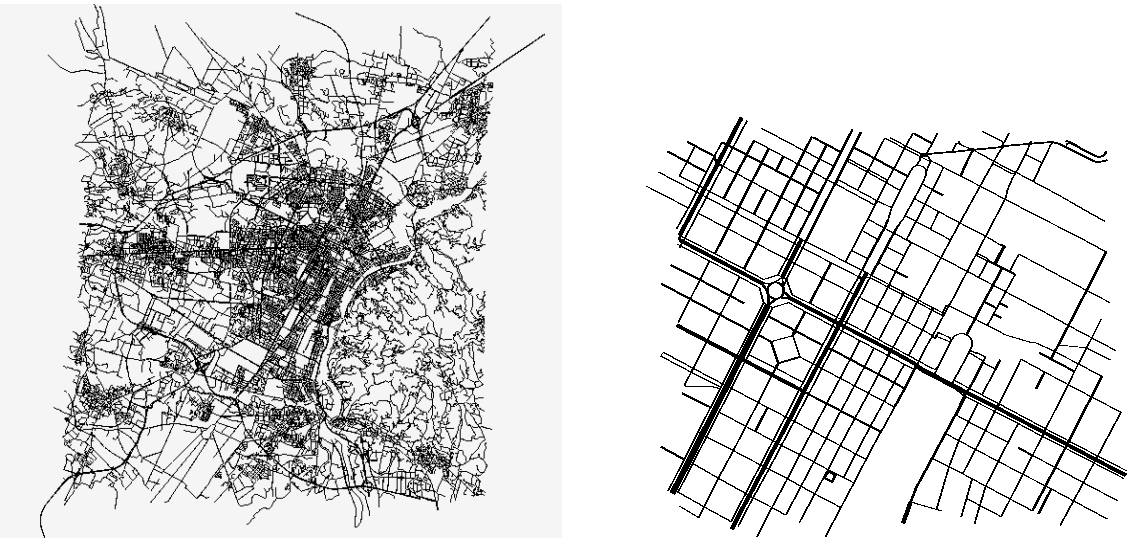


(b) Probability of route selection

Figure 5.14. Edge of route 4 set at 36 km/h

5.3 Turin road network

The second experiment has been made simulating the behavior of the Vehicular Rerouting Advisor in an urban topology: the MEC Application loads a realistic road network of the whole city of Turin, represented in figure 5.15, and converts the geographic coordinates of users positions and destinations into edges of the road network with SUMO tools, creating the `trips.xml` file.
In the next simulations the used weight files (travel times) are realistic static files, obtained by statistical analysis of the road traffic in Turin in one hour, between 7am and 8am; the MEC Application, when the interaction with 5T will be implemented, is supposed to use dynamic data and to change travel times in case of closed edges, consequently to an obstruction detection. Even in this simulations, in order to do a qualitative analysis of the MAROUTER algorithm, the created trips have all the same starting point and the same destination. This traffic demand construction makes the following experiments face some limitations: indeed the travel times are realistic but not the traffic conditions; so the MAROUTER will detect the alternative routes but, since the simulations are made in absence of other traffic flows as it would happen in a real scenario, vehicles can run faster and take less time respect to the predicted travel times, making the duration of the trips not in accordance with the router choices.
Similarly to the experiment in 5.2, the simulations have been made with an increasing traffic load through different files `trips_`x_i`.xml`. For more significant results, a trajectory longer than 15 km has been chosen, in order to highlight the ability of the routing algorithm to select among more possible routes.
Simulations on this topology present high computational time requirements, therefore the following experiments are the result of the mean values of only 4 simulations repetitions with different seeds.



(a) Full map (b) Details

Figure 5.15. Road network of the city of Turin

5.3.1 Scenario 2a: standard case with regular travel times

In the first scenario the experiment studies the behavior of the MAROUTER in standard conditions, with all the edges open and available for the viability of the vehicles.

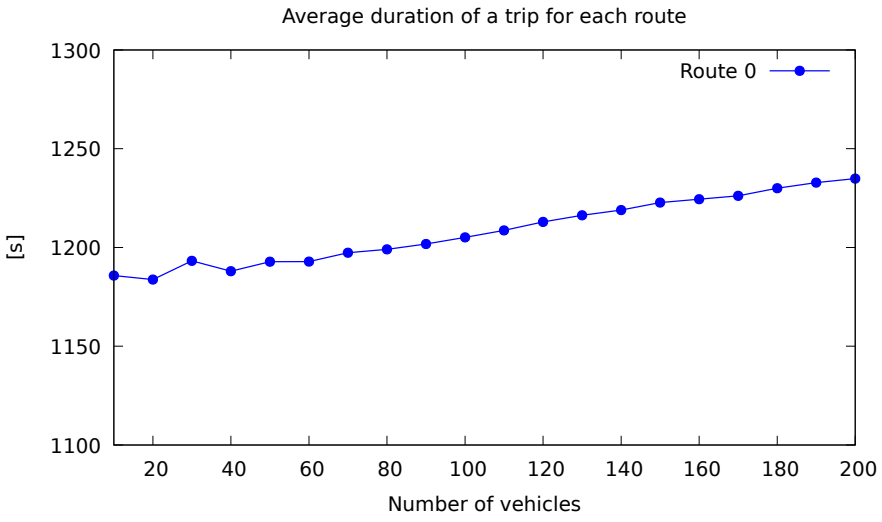


Figure 5.16. Scenario *2a*: Average duration of trips, no closed edge

In this case the probability of selecting the fastest route, corresponding to route 0, remains constant to 1 regardless the traffic load, similarly to scenario 5.2.1: this is due to the fact that route 0 has a higher advantage respect to the other possible alternative routes.
On the contrary, in this scenario the increasing traffic load affects the average duration of the trips, making it increase.

5.3.2 Scenario 2b: edge closed

In the second scenario made with the network of Turin, one of the edges of the route 0 selected in the previous simulation is closed setting its travel time to 1000 seconds, simulating an obstruction detection as in the use-case of the Vehicular Rerouting Advisor.
Graph 5.17 shows the correct functioning of the MAROUTER in a complex urban topology: with an increasing traffic load the router is able to find more and more alternative routes in order to avoid congestion, assigning a probability proportional to the route speed and affordability.
Figure 5.18 reports instead the average duration of the trips, that even if in presence of non-realistic traffic conditions and therefore not in accordance with the travel times, is affected by the number of selected alternative routes: route 1 has an increasing average duration of the trips until an alternative route is detected, while after the introduction of route 2, the duration of route 1 becomes almost constant; some variability in the trend of the graph is due to the randomness of the simulation.
Nevertheless, for the reasons previously outlined, this graph allows to have only a qualitative analysis of the route selection but not of the effective average duration, that for the non-realistic traffic demand construction results to be non reliable: route 5, even if the longest of the selected routes, results to be faster respect to route 3 and route 4 because the low traffic load in this route and the absence of traffic make the vehicles run faster respect to the travel time prediction.

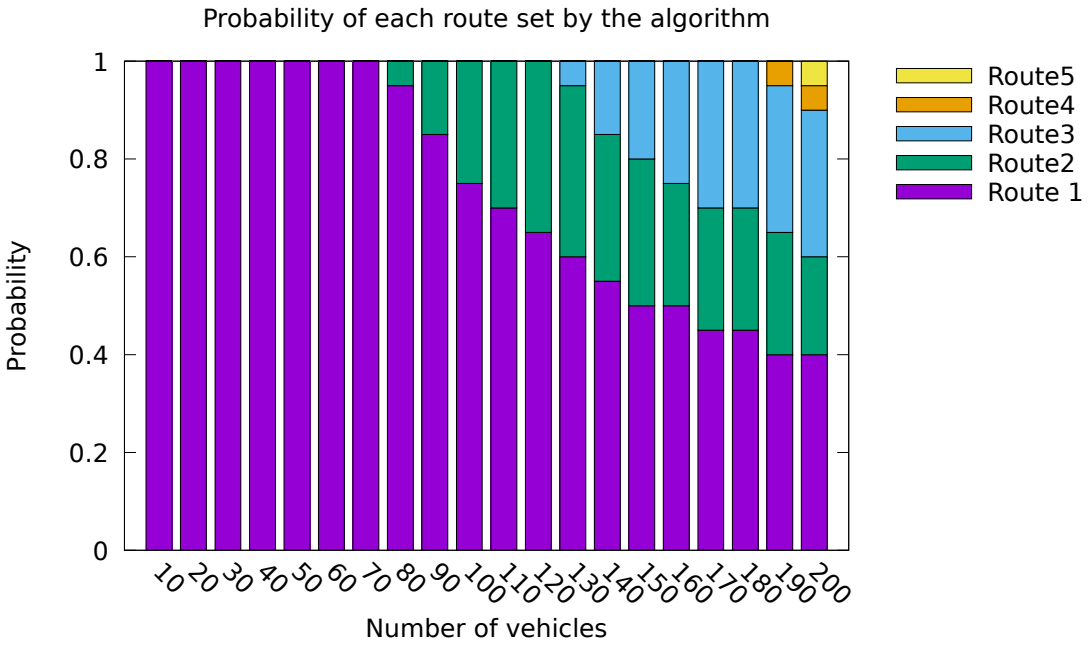


Figure 5.17. Scenario *2b*: Probability of route selection, closed edge

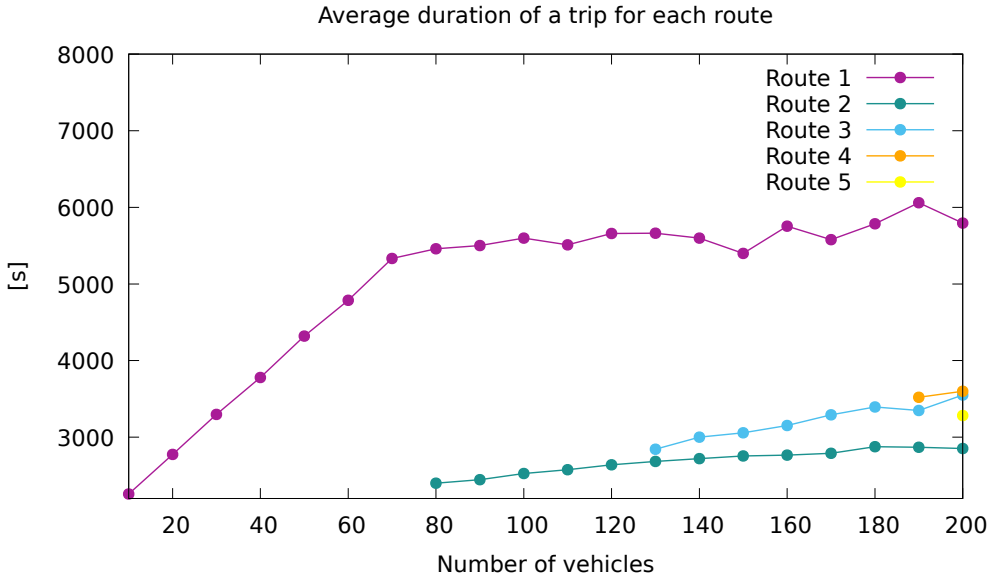


Figure 5.18. Scenario *2b*: Average duration of trips, closed edge

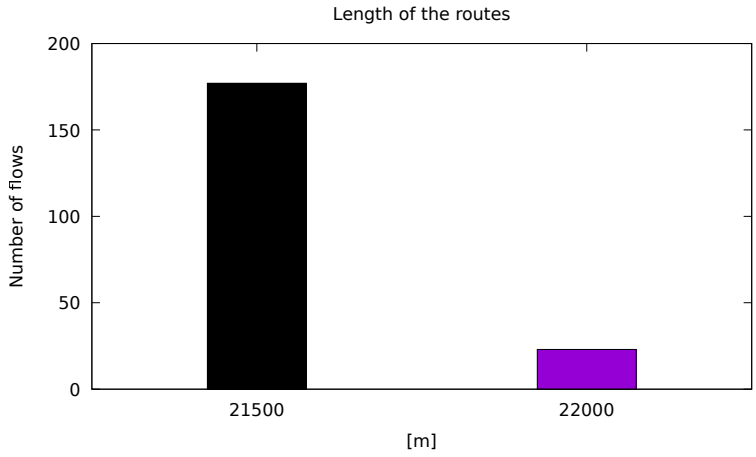


Figure 5.19. Scenario *2b*: Histogram of the route-lengths

The histogram shown in figure 5.19 has been created through the SUMO tool `createStats` and shows the lengths for all routes of the output route file of the simulation made with 200 vehicles: 88% of the flows has a route length of about 21.5 km (routes 1, 2 and 3), while only the 12% has a route length of about 22 km (routes 4 and 5).
The result demonstrates that MAROUTER, even selecting five different routes, is able to keep the route length as shortest as possible; therefore it could be considered efficient for rerouting purposes, when the traffic demand of an urban context is known in advance.

Chapter 6

Conclusion and future work

This thesis proposes the prototype of an innovative traffic monitoring system based on a MEC architecture, called Vehicular Rerouting Advisor, that is able to exchange information about road obstructions between the drivers and the TCC.
The core part of the system on which the work of this thesis project focuses is the MEC Application, that is designed to run at the edge of the network on a MEC Platform. First the user interface has been realized, in order to allow the user to signal a detected event to the MEC Application through a HTTP GET Request. In addition, the MEC Location API has been implemented in parallel according to the MEC standard, covering an essential role in the system, since the peculiarity of the MEC Application is the local awareness and position knowledge of the users in the area; therefore another role of the user interface is to send the geographic coordinates to the Location API continuously on the way through a GPS location application. A future extension of this interface involves the development of a single application able to perform these tasks: the user would subscribe to the system service, that would be responsible of sending with a high frequency over time the user position to the Location API; when the driver is going to start a trip, he should set also the destination location; then, in case of obstruction along the road, he has the possibility of sending the event alert specifying the entity of the problem. Currently the system offers three types of obstruction report (Accident, Environmental Obstruction and System Fault) but integrating the infrastructure with sensors it would be possible to signal any type of location feature, for example a road congestion, the air quality and pollution of the streets or other environmental features. The main service provided by this user application to each subscribed driver would be to receive the answer of the Vehicular

Rerouting Advisor, that indicates the best route to take.
The MEC Application has been implemented creating a HTTP server used to communicate with the user and a HTTP client used for the future interaction with the TCC server.
The data exchange leverages the standards of DATEX II and S.I.MO.NE., since they are the most important protocols used for traffic monitoring and management. As future work, the real-time interaction with the TCC through a Web Service should be activated, in order to test the efficiency of the system.
The last part realized in this project has been the rerouting computation analysis, implemented integrating SUMO package in the MEC Application and setting the static data provided by the TCC as input for the MAROUTER; the results demonstrate that this macroscopic router is able to provide convenient alternative routes in situations where a road is obstructed, making the system efficient; when the real-time interaction with the central server will be enabled, the router will be able to process real-time dynamic data, making the Vehicular Rerouting Advisor fully operational.

Bibliography

[1] G Giannopoulos, E Mitsakis, and JM Salanova. "Overview of Intelligent Transport Systems (ITS) developments in and across transport modes". In: *JRC Scientific and Policy Reports* (2012).

[2] Dario Sabella, Vadim Sukhomlinov, Linh Trang, Sami Kekki, Pietro Paglierani, Ralf Rossbach, Xinhui Li, Yonggang Fang, Dan Druta, Fabio Giust, Luca Cominardi, Walter Featherstone, Bob Pike, Shlomi Hadad, et al. "Developing software for multi-access edge computing". In: *ETSI White Paper* 20 (2019).

[3] T. Subramanya, L. Goratti, S. N. Khan, E. Kafetzakis, I. Giannoulakis, and R. Riggio. "A practical architecture for mobile edge computing". In: *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. 2017, pp. 1–4. DOI: 10.1109/NFV-SDN.2017.8169855.

[4] *ETSI GS MEC 003 V1.1.1 (2016-03), Mobile Edge Computing (MEC); Framework and Reference Architecure*. 2016.

[5] *ETSI GS MEC 011 V1.1.1 (2017-07), Mobile Edge Computing (MEC); Mobile Edge Platform Application Enablement*. 2017.

[6] *RESTful API*. URL: https://restfulapi.net/.

[7] *ETSI GS MEC 012 V1.1.1 (2017-07), Mobile Edge Computing (MEC); Radio Network Information API*. 2017.

[8] *ETSI GS MEC 014 V1.1.1 (2018-02), Mobile Edge Computing (MEC); UE Identity API*. 2018.

[9] *ETSI GS MEC 015 V1.1.1 (2017-10), Mobile Edge Computing (MEC); Bandwidth Management API*. 2017.

[10] *ETSI GS MEC 013 V1.1.1 (2017-07), Mobile Edge Computing (MEC); Location API*. 2017.

[11] *OMA - RESTful Network API for Zonal Presence*. 2016.

[12] Fabio Giust, Gianluca Verin, Kiril Antevski, Joey Chou, Yonggang Fang, Walter Featherstone, Francisco Fontes, Danny Frydman, Alice Li, Antonio Manzalini, et al. "MEC deployments in 4G and evolution towards 5G". In: *ETSI White Paper* 24 (2018), pp. 1–24.

[13] Sami Kekki, Walter Featherstone, Yonggang Fang, Pekka Kuure, Alice Li, Anurag Ranjan, Debashish Purkayastha, Feng Jiangping, Danny Frydman, Gianluca Verin, et al. "MEC in 5G networks". In: *Sophia Antipolis, France, ETSI, White Paper* (2018).

[14] *Rest Client - Test REST API with your phone*. URL: https://play.google.com/store/apps/details?id=com.app.restclient&hl=en_US.

[15] *SCF 084.07.01: "Small cell zone services - RESTful bindings"*.

[16] *SCF 152.07.01: "Small cell services API"*.

[17] *GPSLogger for Android*. URL: https://gpslogger.app/.

[18] *OpenAPI Specification (OAS)*. URL: https://github.com/OAI/OpenAPI-Specification.

[19] *Platform Application Enablement (Mp1) API specifications*. URL: https://forge.etsi.org/.

[20] *EasyWay, DATEX II - The standard for ITS on European Roads*.

[21] *DATEX II (European Commission*. URL: https://datex2.eu/.

[22] I. Ciuciu, C. Debruyne, H. Panetto, G. Weichhart, P. Bollen, A. Fensel, and M.E. Vidal. *On the Move to Meaningful Internet Systems: OTM 2016 Workshops: Confederated International Workshops: EI2N, FBM, ICSP, Meta4eS, and OTMA 2016, Rhodes, Greece, October 24–28, 2016, Revised Selected Papers*. Lecture Notes in Computer Science. Springer International Publishing, 2017.

[23] European Commission. *DATEX II V2.3 User's Guide, Document version: 2.3*. URL: http://d2docs.ndwcloud.nu/_static/data/v2.3/DATEXII-UserGuide.pdf.

[24] *DATEX II (European Commission)*. URL: http://d2docs.ndwcloud.nu/downloads/modelv23.html.

[25] *Profilo d'uso DATEX II per il comparto autostradale italiano*. URL: https://www.retedatex.it/sites/default/files/D_II_Profile_IT_V2.0_IT.pdf.

[26] *Progetto S.I.MO.NE., Definizione del protocollo di comunicazione*. 2014.

[27] *5T, S.I.MO.NE.* URL: http://simone.5t.torino.it/.

[28] Michael Behrisch, Daniel Krajzewicz, and Yun-Pang Wang. "Comparing performance and quality of traffic assignment techniques for microscopic road traffic simulations". In: (2008). URL: http://infoscience.epfl.ch/record/154987.

[29] M. Patriksson. *The Traffic Assignment Problem: Models and Methods*. Dover Publications, 2015. ISBN: 9780486802275. URL: https://books.google.it/books?id=PDhkBgAAQBAJ.

Acknowledgements

I would first like to express my sincere gratitude to professor Casetti and my supervisors Daniele and Edoardo, that have followed the growth of this project with patience, expertise and motivation, always offering me support and advice.
I also would like to thank all the people that have contributed to the creation of this thesis, such as 5T team, in particular Ing. Arneodo and Ing. Gagliardi, and Marco Rapelli.

I would like to offer my special thanks to my parents, to Silvia and to my grandparents, that with their presence and encouragement helped me to face all the difficult moments and that shared with me joy and pain of these years.
I am also very grateful for all the support given by all my friends and by all the people that I met in these unforgettable years and that shared with me this experience.