University POLITEHNICA of Bucharest

Faculty of Automatic Control and Computers,
Computer Science and Engineering Department

BACHELOR THESIS

Distributed Storage and Dissemination Service based on Floating Content

Scientific Adviser:
Conf.Dr.Ing. Ciprian DOBRE

Author:
Mihai CIOCAN

Bucharest, 2014

Contents

Chapter 1

Mobile context aware social applications

1.1 Introduction

Mobile devices have become a staple of our society, with everyone of us owning at least one. They bring significant improvements in the quality of life and most people rely on them for their basic daily activities. The number of smartphones have reached in 2014 around 1.75 billion users and is expected that about 50% of them may use the internet network (1). And they are spreading at a fast pace as the price of mobile continues to drop and the data network expands.

Mobile technology transformed how companies do business. Mobile technology makes possible documents transfer almost anywhere in the world in small amount of time so business is addressed when it encounters an emergency. It keeps employees and business owners connected regardless of their position on the globe and increases the company productivity. It also keeps the customers updated with information about services they use and pay for.

Mobile devices can no longer serve one single purpose as communication, now being able to run rich stand-alone applications like games, navigation apps, ebook readers, music and video players and many more. We have the information in our hand and it has become easy and natural for us to quickly look up for helpful resources for whatever activity we need to do. Mobile application can even anticipate what information we need and present it to us when it is most useful.

Some of the applications are used only in specific contexts e.g. travelling to unfamiliar places may force the user to use navigation apps like Google Maps or Waze. But out of all categories, the social and communication apps are the most used throughout the day. The most popular is the social network Facebook, which enables you to share photos and personal content with your friends. Another one is

Twitter which displays the content in a page similar with a blog.

The evolution of location-aware mobile technology has influenced the mobile application industry offering users context based experiences. The Facebook Messenger provides the users with the location of their communication partners. Moreover the latest feature, Nearby Friends send notifications when a the user comes within a short distance of a friend if they choose to share their location. Many location aware apps have been developed because of the increasing curiosity of people about friends activities. They even enable to interact with strangers around you like Tinder or Skout does.

Prior the worldwide spread of smartphones location was mainly used for space orientation. Drivers used global positioning devices (GPS) to guide themselfs in a journey. The major problem was the little memory space which could carry a a limited area of the geographic map. Now application like Google Maps, Google Earth or Waze has replaced the single-purpose GPS devices and can display geographic proximity based on the position of the user. Although mobiles have much bigger storage sizes, several GB, it is still very limited for storing big amount of map data. The data network services enhance them to provide the user detailed views of any position on the globe.

Recently, navigation services have become social and act as real communities where people share information with certain location importance. Maps support point-of-interest tagging, photo and text description, and even online street navigation. They are able to share information in real time, like Waze does. Drivers can inform other about the driving conditions they experience on the road, and send alerts about traffic jams or accidents.

New problems have emerged from the fact that mobile applications rely on the data network. The next section describes in detail the challenges mobile application may encounter.

1.2 Weaknesses of network-based mobile application

While social network applications are dependent on the infrastructure services in order to overcome distances and to connect people around the world, relying on the infrastructure services for location-aware application may introduce issues regarding content and location relevance and security (12):

- *Location privacy* concerns may arise from the need of the application to provide the user with the exact location, especially for the navigation services like Google Maps or Waze. It needs determining it with a high level of accuracy in order to obtain the right context information.

- *Content privacy* issues occur because the shared information is stored by a private company in a private "central" location and can be easily subject to censorship.

- *Connectivity* to the infrastructure services can be a problem, especially for traveling users who may have to deal with high roaming charges, unavailability of data services, or no network coverage at all.

- *Geographic validity* concerns the location characteristics of the information. Locally relevant shared content may be of little interest to the rest of the world, so storing it in an accessible location may only cause memory waste.

- *Temporal validity* concerns the temporal characteristics of the information. Shared content which is stored in a "central" location is only valid for a limited amount of time and it is rarely associated with an expiry information. This practice leads to content never being deleted or never being read either,thus wasting databases memory.

- *User identification* of some kind is used usually in order to limit the amount of data being shared which creates some sense of responsibility towards the service provider. It can also be a privacy problem because the information shared is associated with the owner and nowadays companies like Facebook or Google are giving access to the records to security agencies for population surveillance (most recent case being PRISM) or for marketing purposes.

The solution to all the above problems is a content sharing service which is entirely dependent on mobile devices in the vicinity, using principles of opportunistic networking. Bringing social media and content sharing into ad-hoc networks seems it is the next frontier in mobile industry. It can be seen as an extension to the Internet infrastructure, by bringing connectivity where the infrastructure is not able to.

1.3 Inter-Vehicle Communication

Vehicle-to-vehicle wireless communication have emerged from the desire to provide more safety, entertainment and comfortable driving to a huge number of individuals that use vehicles on the roads every day.

It is a known fact that many car manufacturers have already introduced wireless communication equipment in their cars. Volvo implementation is called "Volvo on Call" (18) and BMW implementation is called "BMW Assist" (9). Their purpose is to help drivers in case of accidents or collisions. Their current implementation use “central” infrastructure based services based on cell phone technology, existing base stations, long range and centralised servers. Naturally, the next step would be to modify the equipment in order to provide Inter-Vehicle Communication.

Inter-vehicle Communication is a type of ad-hoc network which mainly uses broadcast as a method of information distribution. The only limitations vehicles may have are the available transmission capacity which depends on the rate and the size of the information broadcasted.

As we mentioned, communication between vehicles means adding benefits to driving experience. Some of the relevant information to be shared could be: accident warnings, driving conditions, weather, announcements and even advertisements with audio/video content. They provide useful information related to a limited geographical area, for a limited amount of time. For example, an accident happened a day ago would not be of much interest for the drivers, so the need of limiting the life of information is obvious. The biggest challenge is making the information stay alive and “float” to achieve the main purpose of the sharing process which is to reach a high number of vehicles. In the next chapter we analyzed the feasibility of a distributed storage and dissemination service addressed to overcome all the mentioned before problems.

Chapter 2

Floating Content

This chapter will describe the Floating Content model of the data storage. Our application is based on this analytic model used to simulate and observe the existence behaviour of the content on a real world map. The testing and evaluation is presented detailed in a further chapter.

2.1 Introduction

Floating Content network is a particular type of network that uses intermittent connectivity also known as delay tolerant networks (DTN). Recently, the new term disruption-tolerant networking has gained popularity in the United States. Among the events that may generate disruption, the following can be spotted: the limits of wireless radio range, noise, energy resources and sparsity of mobile nodes (20). This concept of information sharing matches the needs of context-aware applications because it keeps the spatial proximity in a close relation with connectivity.

Context awareness is a characteristic of mobile devices and includes different types such as : identity, activity, time, and the most important for our analysis, location. The application can determine what activities may occur near the entity, the objects and people that are nearby.

A use case scenario of context-aware application could be a tourist guide information sharing. When you are visiting unfamiliar places, having an application that provides you with useful information about the surroundings can be a blessing. The content can include description, history about places you are visiting, hotel and restaurants and their service quality, directions you have to follow to reach a desired place and many more.

2.2 Applications for Floating Content

The features of floating content offer the user exciting opportunities but many problems can emerge. The most useful opportunity enables localized information sharing without using infrastructure services and without central data records. Since there is no remote access, it gives the user some degree of privacy, as he must be present in order to "see" something which is a usual concept in the daily life. The major challenge is that the communications service makes no guarantees that the data will stay around until its lifetime expires. For example during the night the content is expected to disappear. We can make intuitive deductions, which are assisted by the simulations which will be later discussed. If the service is used in an overcrowded place like a market square or a heavy traffic road, there is a high probability that the content will float for some time, even if not all other people use context-aware mobile devices. In the end, any type of communication (event the best-effort) may be better than no communication at all.

People's daily activities cause density fluctuations over the day which may be a problem for the residing information in certain places. Thus, the information is expected to remain available for no more than a few hours. Making predictions that there will be enough people in that location is very limited for now. However, a limited amount of time, let's say one hour of "floating" may be enough for the application to do its job. As a solution, a user can be a permanent seed, staying around and resending updated content.

Multiple use cases can emerge from the floating content concept. One of them can use infrastructure-less local data availability for advertising or selling goods. This type of market could have a dynamic catalog of available merchandise, being able to operate updates on the fly.

Another one is information sharing between tourists and visitors about the local attractions or notifications about good services a certain hotel is offering. Spreading news and keeping it localized, time-bounded and most important anonymous can be another use case of floating content for which best-effort operation perfectly suits the needs.

Overall, floating content can be used in many ways, taking into consideration two important aspects of it. First, floating information is location aware so the developers should consider multiple data-oriented architectures. The second aspect is that the concept is best effort, a problem which the Internet infrastructure solves with repair mechanisms that can recover lost packets. In our case, data that expires is irrecoverable in an area. Taking these into account for future application will produce more interesting use cases.

2.3 Service model

This section describes the floating content design and the environment constraints that the architecture needs in order to work. We assume users to be mobile nodes who are interested in the content generated by all other nodes. We assume that they are using mobile devices with unlimited data memory in order to handle the amount of data exchanged during their participation in ad-hoc network. Also, there is no supporting infrastructure for the system.

We assume that nodes are uniformly distributed and travel independently, with a constant speed. In (13) it is shown that this mobility model preserve the spatial node distribution at all time points.

The devices are equipped with wireless interfaces (Bluetooth or WLAN) to enhance network communication. Analysis of performance for 802.11p standard displayed in (11) have shown that using a bitrate of 6Mbps and a payload of 500 bytes yields a delivery rate of up to 80%. This indicates the acceptable reliability and performance of IEEE 802.11p and confirms the viability of floating up to several megabytes of data (from text messages to photos). This standard is used also in our simulation which will be discussed in a further chapter. Making intuitive judgements, we can determine that contacts cannot last more than several tens of seconds, in vehicle case even less due to their high speeds. Thus there will be no need for the mobile devices to reserve a considerable amount of storage for the floating content.

The devices also need to be equipped with accurate systems which determine their position, e.g. using GPS tracking, cellular base stations, cell tower triangulations using WLAN access points or Wi-fi tracking which is a very popular technology. Each of these methodologies has its own advantages which will make them more appropriate in different use cases. Many factors need to be taken into consideration lieke accuracy percentage needed, battery consumption etc. In order to provide the best location based service, the equipment must acquire the most accurate location coordinates. Finally, nodes need to synchronize their clock time so the users can process exchanged information as anyone else ; it can be done with the help of GPS or cellular networks.

When producing information, the application must tag the information with its geographic origin, validity range and and time-to-live (TTL). Other nodes decide if they store the information and replicate it further. The only requirement is they need to be in the associated anchor zone to be able to receive it. The information is explicitly allowed to disappear providing no guarantees about its availability. If the lifetime of the content expires, it will be discarded by the application. In consequence an item may disappear if there are no nodes (or to few) to replicate it

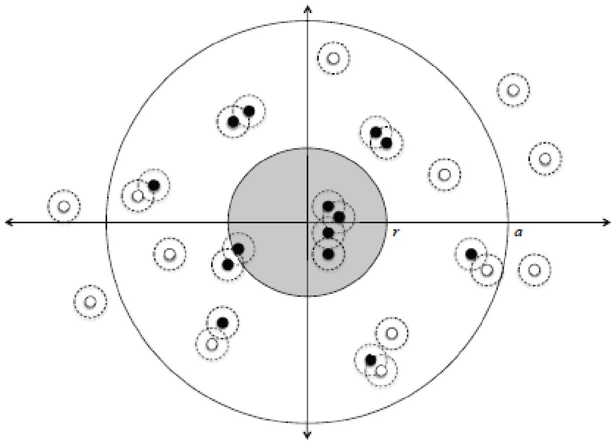


Figure 2.1: Moving nodes inside an anchor zone. Black nodes are information-carrying nodes, white nodes will eventually get the information from the black ones. The probability of a node carrying an item tends to 1 inside the replication zone, and decreases until reaching an availability distance a after which no more copies are found.

in its associated anchor zone, unless the creator is there to re-issue it again.

2.3.1 System operation

As (12) presents, a node generates information I which has a size of $s(I)$ and a defined lifetime (TTL). The information is tagged with the anchor zone also which is defined by its geo-located center P and two radii: r identifies the *replication range*, inside which nodes replicate the information to other nodes they meet on their way and a defines the *availability range* inside which the information is still stored with limited probability. As shown in the 2.1 outside the availability zone there exist no copy of the item in the node data storage.

The idea of the concept is that if two nodes meet in the anchor zone of some information, and one of them doesn't have it stored in the data storage then the other node will immediately send it so both nodes can be able to access it. As a result, every node inside the anchor zone should have a copy of the item while nodes which are leaving the anchor can delete it at their own discretion.

Let's consider two nodes A and B which encounter at some point in time. Node A does have an item I tagged with an anchor zone centered in point P and radii a and r. Let h be the distance of node A from the center P. When node A meets node B, item I gets replicated to B with the probability $p_r(h)$:

$$p_r(h) = \begin{cases} 1 & \text{if } h \leq r \\ R(h) & \text{if } r < h \leq a \\ 0 & \text{otherwise} \end{cases}$$

$R(h)$ is a decreasing function which determines the probability of replication between the outer replication border and the availability border of the anchor zone. The deletion probability $p_d(h)$ is defined as with $D(h)$ in $[0, 1]$:

$$p_d(h) = \begin{cases} 0 & \text{if } h \leq r \\ D(h) & \text{if } r < h \leq a \\ 1 & \text{otherwise} \end{cases}$$

The node preserves the free space due to the deletion function. Many deletion policies can be applied. The elementary first in, first out (FIFO) principle can be used, deleting the oldest items first if there is a need for free space.

The area between *replication range* and *availability range* acts as a buffer zone, that prevents immediate deletion of items. It is advantageous for nodes that leave the *replication range* for a short period of time, preventing them to perform deletion operation. Thus, after returning inside *replication range*, they already have the associated items. Outside the anchor zone, a node could delete specific items when meeting with other nodes, or at a predefined timeout, checking if items are outside their zones or their lifetime expired.

As shown in (12) and because of simplicity and more trackability, in our evaluation there is no buffer zone, i.e. $r = a$, deletion and replication function becoming useless in this case.

2.3.2 Communication Protocol

As described in (12), the floating content protocol, also used in our evaluations, is an efficient and simple method to exchange items between nodes. A message is identified by a message id Id, the anchor point with the attributes described earlier (P, r, a), and the lifetime T. The header of the message is filled with these characteristics, and the message body of size $s(I)$ is filled with the desired payload.

Below is the protocol which has 4 phases:

1. Nodes keep sending neighbor discovery beacons to discover peers.
2. When receiving a discovery beacon, the peer decides to send in return, its list of items that verifies the condition $p_r(h) > 0$, thus valid for replication. In this phase the list contains only the attributes of the items, keeping this message

as compact as possible: Id, $s(I)$, (P, r, a) and lifetime T. If the list doesn't fit into a single message, it will be spread across multiple summary messages in a round-robin fashion.

3. When receiving the list of items from a discovered peer, the node request those items for which $p_r(h)$ suggests that they should be replicated.

4. In the last phase, requested items are exchanged until transfer completion or nodes lose contact. Upon this step, the protocol should remove uncompleted messages and return to phase 2.

It is assumed that nodes can exchange messages fully bidirectional during any step. Moreover, they can exchange messages simultaneously to multiple nodes (even though technology may be limited). The beaconing process can take place while in middle of message exchange in order to keep the discovering process running. Since message exchanging is done in an incremental way, nodes can append the new coming messages to the list, while they are still in transfer.

Deletion of item I occurs immediately the node moves outside of a. In (12) is is suggested that a second possibility could be deletion $upon-encounter$ which discards a message when meeting a new node, stating that this policy is more sensible due to its asynchronous characteristic triggered by an external event. Naturally, deletion takes place before the list of items is sent to other peers.

2.3.3 Security issues and resource management

The presented protocol does not restrict the user in any way, regarding content generation and its context parameters. The single requirement is that the owner must be in the anchor zone at the time of creation. In consequence, this may represent a flaw in the protocol, users being able to insert items in network with an infinite anchor zone. Flooding items represent a threat for the network, because they exhaust the system resources very quickly, especially channel and buffer capacity.

Reputation mechanisms like (3) or accounting could represent solutions to the above spamming issues. But it would be very difficult to implement them without having an central infrastructure to enhance authentication of identities. Also being a best-effort service is another nail in the coffin for the rewarding system. Some simple mechanisms could be added to implementation like prioritizing items taking into consideration their expected storage consumption and the distance from the anchor location. Thus, deleting the farthest or the most largest item may discourage unlimited content distribution.

As shown in (12), the application can smartly manage its resources by giving preference to items with smallest anchor range. Items with very large anchor zones would have low availability due to the poor coverage it may have. Spammers could move around on a large area to create items with small anchor zones and simulate a large anchor zone. There is no prevention mechanism for this, but it is considered that spammers should put a lot of effort to achieve their goal. Also, the anchor zone needs to be periodically revisited due to the ephemerality characteristic of the information. This security mechanism doesn't require infrastructure service or any degree of mutual trust which is an advantage.

2.4 Analytical model

Floating content model has been a subject of analysis in (6) . The most important objective of this work has been finding a pattern to guarantee that a specific information remains in its anchor zone until the expiry of its lifetime with a high probability. It is called the *criticality condition* and it depends on many aspects like mobility patterns and replication policies of the nodes inside the anchor zone.

2.4.1 Criticality Condition

As before, we assume each information is being tagged with a geo-location data of the anchor zone in which nodes keep entering, spend some time and finally exit. Also, we assume that the nodes spend a consistent amount of time inside the anchor and follow a random mobility pattern. The population is assumed to be large and "well mixed". It is important to preserve the proportion between the total number of nodes and the number of ones which have a copy of information. Further the criticality condition at the fluid limit is explained.

While moving inside the anchor zone, a node may come in contact randomly with other nodes. We assume there are only two nodes moving permanently inside the zone. Let υ be the frequency at which they come in contact with each other. Now, assuming the population of nodes in anchor is N, then the total number of pairs is $\frac{1}{2}N(N-1) \approx \frac{1}{2}N^2$ and the total rate of encounters is $\frac{1}{2}N^2\upsilon$. A part of these encounters, more exactly $2p(1-p)$, replicate an item to nodes that doesn't have it yet in the data storage, thus the total rate of such events is $p(1-p)N^2\upsilon$. This rate shows the the type of monotonicity of the size of the population which have the item I. Let $\frac{1}{\mu}$ be the time spent by a node in the anchor zone. It results that the total exit rate of nodes is $N\mu$ and the exit rate of tagged nodes is $Np\mu$. The growth rate is determined by the formula:

$$N\frac{d}{dt}p = N^2p(1-p)\upsilon - Np\mu \tag{2.1}$$

The two terms on the right hand side are equal when in equilibrium leading to the stationary value $p^* = 1 - \mu/(\upsilon N)$. In order to have a positive solution, $p^* > 0$, it requires that,

$$N\frac{\upsilon}{\mu} > 1. \tag{2.2}$$

Equation (2.2) is called *criticality condition*. The left hand side value represents the average number of collisions a randomly chosen node has during its sojourn time. Taking into consideration the sign of the equation (2.1) it can be seen that the solution is stable. If $p > 1-\mu/(\upsilon N)$, it tends to increase, else if $p < 1-\mu/(\upsilon N)$, it tends to decrease. The information disappears (even in the fluid model) when the derivative is everywhere negative leading the solution to p = 0. Moreover, since we need to prevent accidental disappearance of the information carrying population by stochastic fluctuations, $Np = N - \mu/\upsilon$ must be large.

2.4.2 Information evolution during its lifetime

Inspired by the mathematical modeling of the spread of infection diseases (8) we decided to analyze the evolution of information spreading as if it was a virus. Moreover, the protocol proposed uses broadcasting as way of transmission which is known as epidemic routing.

Our approach is very similar with the SIR model: S stands for susceptibles, the nodes interested in information but which don't own a copy yet; I stands for the infected ones, or the ones who already have a copy and can send it further to the neighbours; R stand for the removed, the one which deleted their copy either because their availability time expired, or they are out of the anchor range.

As shown in (8), we define r as the infection rate and the increase number of the infected proportional with the current number of susceptibles and infected: rSI. We define a as the removal rate and the increase number of removed proportional to the infected only: aI.

We are able to compute the number of each class using the following equations:

$$\frac{dS}{dt} = -rSI \tag{2.3}$$

$$\frac{dI}{dt} = rSI - aI \tag{2.4}$$

$$\frac{dR}{dt} = aI \tag{2.5}$$

The equations ensure that the total population $N = S + I + R$ and $S_0 > 0$, $I_0 > 0$ and $R_0 > 0$.

From (2.4), at the beginning of information exchanging, when $t = 0$ the following equation results:

$$\left.\frac{dI}{dt}\right|_{t=0} = I_0(rS_0 - a) \gtrless 0 \; if \; S_0 \gtrless \frac{a}{r} = \mu \tag{2.6}$$

There are two cases which emerge from (2.6): $S_0 < \mu$ implies that the number of infectives drop from I_0 to 0 and no epidemic can occur, $S_0 > \mu$ the number of infectives increases and the information spreads. μ can be considered as a threshold which determines whether the information will live or not. The presented model helps us to understand how the information develops in time, how the number of neighbours and the radius size influence its spread.

2.4.3 Model applicability

It can be seen that the black non-spatial model for content exchange is highly abstract, only capturing the essential elements. The most important assumptions on which the model relies on are the fluid limit approximation and a well-mixed mobility pattern. In this case the *criticality condition* shows that the information "floats" due to the large number of nodes in the anchor zone assumed by the fluid limit approximation. In a real case situation the number of nodes carrying the information may be small, thus having a big probability of information disappearance due to stochastic fluctuations in the system. The well mixed characteristic of the population leads to the fact that all the nodes inside an anchor zone, are equally likely to encounter each other at some point of time, past encounters having no influence on it.

In our simulation, which will be discussed later in the document, the spatial aspects of information exchange is ignored. So the probability of a node carrying information is the same on the entire simulation network (of course without taking in consideration water zones). In addition, the *criticality condition* doesn't take into consideration some important parameters e.g. transmission range which affects directly the encounter rate υ.

The non-spatial model is applied in the simulation performed, because it captures the fundamental elements of the system, and defines the *criticality condition* which is an indicator whether an information item may float or not. As in (12) we compute three key elements and then observe how the *criticality condition* correlates with the road network and the final information life time. The simulation produced similar results, confirming the applicability of the abstract model in real case mobility scenarios.

Chapter 3

Floating Content Implementation

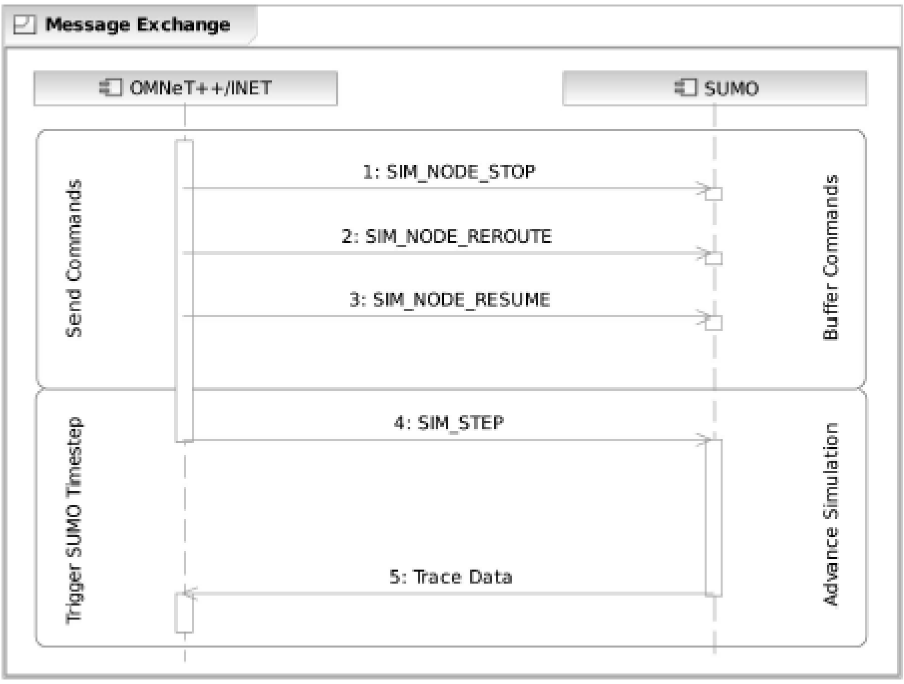
In the previous chapters we presented the theoretical protocol of a data storage service using geo-located information and its advantages over the classic infrastructure based network. We also discussed the criticality condition which is an indicator regarding the probability of item floating which takes into consideration the number of the population, its sojourn time and the exchange frequency. In this chapter we present our implementation of the service using a network simulator and a mobility simulator using vehicles as nodes.

3.1 Omnet++ and Simulation of Urban MObility

Omnet++ is an extensible, modular, component-based, discrete C++ simulation library primarily for building network simulators. In this definition, "network" has a broader sense that includes wired and wireless communication networks, on-chip networks, queueing networks, and many more. It comes with model frameworks, developed as independent projects, which provide support for sensor networks, wireless ad-hoc networks (used by us), Internet protocols, etc.

Omnet++ offers an Eclipse based IDE, a graphical runtime environment, and host of other tools. There are extensions for real-time simulation, network emulation, alternative programming languages such as Java and C#, database integration and several other functions (17). The fundamental ingredient of its infrastructure is the component architecture for simulation models. Models are assembled from reusable components termed *modules*, and can be combined in various ways like LEGO blocks.

As we mentioned earlier, the Omnet++ model consists of multiple modules that communicate with message passing. Simple modules, which are written in C++ using the simulation class library, can be combined to form compound modules. Compound modules can be also combined having no limitation on the number of hierarchy levels.



Messages may contain any sort of data, in addition to usual attributes such as timestamp. They are sent via gates, the in and out interfaces of modules. Connections are created within a single level of module hierarchy, they are not possible across multiple modules as they would obstruct model reuse. Because the hierarchical structure of the model, messages typically travel through a chain of connections.

Given the type of network we need in our implementation, we use an open-source framework for running vehicular network simulations called Veins (16). It uses Simulation of Urban Mobility (SUMO) along with Omnet++, connected through TCP sockets, to perform Inter-Vehicular Communication evaluations.

Simulation of Urban Mobility (SUMO) is a microscopic road traffic simulation package designed to handle large road network (4). Thus, we can perform bidirectionally-coupled simulation of road traffic and network traffic. Movement of nodes in Omnet++ simulation is determined by movement of vehicles in road traffic simulator SUMO. Nodes can then interact with the running road traffic simulation.

3.2 Network Description in Omnet++

The user must describe the structure of the simulation model in a language called NED. NED stands for Network Description and allows the user to declare simple modules, and connect and assemble them into compound modules. It has several features which let it scale well to large projects.

- Hierarchical. Any module which appears to be complicated, can be split into smaller modules, and used as a compound module.

- Component-Based. It provides module reusability, reducing code copying, and more importantly allows component libraries like Veins to exist.

- Interfaces have a special purpose: they replace a module or channel type that would normally be declared in the network description. The concrete module or channel type is determined at network setup time by a parameter. Concrete modules types must implement the interface they substitute. For example, a compound module type named MobileHost contains a mobility submodule of the type IMobility (an interface); the actual type of mobility may be chosen from the module types that implemented IMobility (RandomWalkMobility, TurtleMobility, etc.)

- Inheritance. Modules and channels can be inherited like C++ classes. Extended modules could add new parameters, gates, new submodules or connections. They may initialize some parameters to default values.

- Packages. The NED language has a package structure similar to Java language. Its main purpose is avoid name conflicts between different modules. It has a similar "CLASSPATH" called NEDPATH introduced to make it easier to specify dependencies among simulation modules.

3.3 Architecture

Every simulation in Omnet++ is defined by a configuration file, usually called *omnetpp.ini*. It contains settings that control how the simulation is executed, values for model parameters, etc.

3.3.1 Network

The most important parameter which has to be defined in the configuration file is the network module type (if it is not defined in the configuration file it will be asked for at runtime). Our network is called simply FloatingScenario and extends a simple module called FloatingContent. Networks usually consist of two main important section: first include the nodes that form the network, and second include how the nodes are to be connected. In our particular case the nodes are vehicles which are moving across a map, so they are not static and its defined based on multiple files (we will talk about it later in the document). The module which handles this information is TraCIScenarioManagerLaunchd.

It contains the next submodules:

- ObstacleControl model uses obstacles from poly file resulted from the map export in order to block the radio transmission as in real world. The obstacles refer mainly to buildings which exist on the map, and may block radio propagation of signals between vehicles.

- AnnotationManager manages annotations in Omnet++ canvas. It displays module graphical representation (as icon) or additional representation strings.

- ConnectionManager handles all connection related stuff. It will be discussed in detail later in the document.

- WorldUtility extends basic module BaseWorldUtility which provide utility methods and information used by the whole network as well as simulation wide black board functionality. It collects global parameters like the dimensions of the network (playground), whether is 2D or 3D, anchor range and distance between them.

- TraCIScenarioManagerLaunchd extends TraCIScenarioManager and handles nodes from the network details.

3.3.2 Traffic Simulation

Traffic simulation in *Veins* is done with the help of microscopic road traffic simulation package SUMO. It can perform simulations both running with and without GUI and it is able to import city maps from a variety of file formats including OpenStreetMap types.

SUMO deals with high-performance simulations of huge networks with roads consisting of multiple lanes. Vehicles can move accordingly to a configured timetable, dynamically generated routes, or statically assigned routes (16).

The connection between SUMO and Omnet++ is done with the help of TraCI: Traffic Control Interface. Is an architecture that couples two simulators: a road traffic and network simulator. It is important to know that mobility patterns are not pre-defined as fixed trace files. The traffic and network simulators are connected in real time, enabling the control of mobility attributes of each simulated vehicle. The reason there are no predefined routes is the fact that the mobility of each car can be influenced by the network simulator, based on the information it receives (for example a bad weather warning).

We denote *TraCI-Server* the road simulator and the *TraCI-Client* the network simulator. They are connected over a TCP connection for exchanging messages. So when the connection is established, the network controls the traffic simulator via the data exchange protocol, enhancing the control of the movement for each vehicle.

Data exchanges are controlled by the client that send commands as requests to the traffic simulator. The traffic simulator may respond to each command once or multiple times.

Time synchronization between both simulators is performed by the client which sends periodically, at each simulation step, e.g. 1[s], a message to the road traffic simulator that contains the current simulation time plus one simulation step. The road traffic simulator computes the next step and sends the data to the network simulator. The vehicle positions are converted into linear movements by the network simulator, in order to allow all vehicles to reach their positions at the time specified by traffic simulator. The execution of the atomic mobility commands is delayed by one simulation step, because the road traffic simulator is always one simulation step ahead. This may represent a drawback of the method, but considering the very short simulation time step - for traffic efficiency applications approximately 1[s] and for safety applications - 0.1[s], the drawback is negligible. When the network simulator or road traffic simulator ends, it closes the TCP connection, and causes the other simulator to stop immediately (19).

TraCIScenarioManagerLaunchd is the manager module which handles the Traffic Control Interface. Parameters *host* and *port* indicate Omnet++ the address and the port on which SUMO listens for connections. We can specify the time interval of host's position updates by setting $updateInterval$. Parameter $moduleType$ specifies the module type to be used in the simulation for each managed vehicle.

3.3.3 Connection modeling

Connectivity modelling is a more difficult task to be designed in wireless networks than in the wired one. In the latter case, connectivity is achieved with wires which in Omnet++ are easily modelled with *connections*. The channel between two nodes in a wireless network is basically the air and cannot be represented just by one connection, because is a broadcast medium. $ConnectionManager$ module manages the connectivity in our simulation.

In theory, signals sent out by a node may affect all other node in the simulation. Adding attenuation makes the power of the signal very low for the nodes which are far away from the origin, and thus making it negligible.

As presented in (10), the $maximalinterferencedistance$ is a conservative bound on the maximal distance at which a node can still disturb the communication of a neighbour. In order to reduce the computational complexity, MIXIM connects nodes only if they are within the maximal interference distance.

The objects present in the propagation environment influence the $maximalinterferencedistance$ by reducing it due to the additional attenuation they imply. It may cause disconnection between two nodes, leading to the hidden node problem.

It is important for safety applications to perform IVC simulations as realistic and correct as possible. Thus, a model for modeling buildings and their properties must be used, to accurately simulate the signal propagation. One type of building generated attenuation is shadowing, defined as the fluctuations in received signal power. In (15) is presented a realistic and computationally inexpensive simulation model for IEEE 802.11p radio shadowing in urban environments. It is validated using real world measurements in a city scenario for different types of obstacles. The obstacle model is integrated in the Veins making it suitable for our simulation needs.

In Omnet++ gates are the connection points of modules. There are three types of gates: *input*, *output*, *inout*, the latter having the behaviour of the first ones. Gates can only be connected to other gates via *connections*. These *connections* are found in the network description files (NED) of the modules.

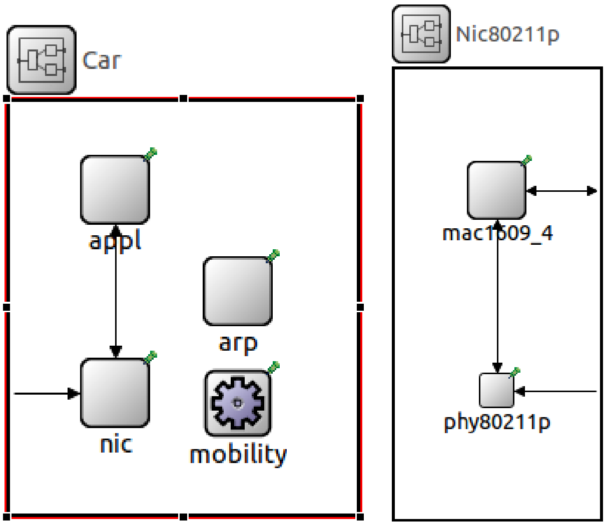
Every node module registers its network interface card (nic) to the *ConnectionManager* along with its position. Every time the node position changes, the manager is notified, and after the calculations taking into consideration the new value and the attenuation, it updates the list of neighbours which the node can establish connections with.

*Nic*s have an *input connection* called *radioIn* which expects messages from other modules. So whenever a node sends out a message, the *ConnectionManager* passes it to the list of neighbours computed as we mentioned above, using *sendDirect*() function. The function receives the module and the gate id as parameters. The neighbours pass the received message further to the upper layers.

3.4 Car modules

Nodes in Omnet++ network are entities desiring to communicate with each other. The Car node module, we are using in our simulation, can be seen in 3.1a. In the left side appears the standard layer according to WAVE protocol, the application layer (appl), the MAC layer and the physical layer (phy). The physical layer takes care of the reception and collision handling. Layers are modules connected by two pairs of Omnet++ “gates”. The first one is used for transferring data up and down as in the real world protocol stacks, the second one is for exchanging messages between the layers and support the control communication, e.g. a request from the MAC layer for PHY layer to perform carrier sensing, or notification of PHY layer that the transmission of data is over.

The MAC and physical layer are grouped into a module called Network Interface



(a) Car module **(b)** Nic module

Figure 3.1: Car and Nic modules

Card (NIC) in Figure 3.1b, because they their design is usually tightly coupled and specific for different communication techniques like bluetooth or GSM. A node can have multiple different NICs.

The mobility module performs movement of a node or object. It will be discussed in a further section. The arp module runs the Address Resolution Protocol which translates the network address into a MAC address. The application module is the top layer and uses the network interface card to send messages. It is the application equivalent in the IP stack.

3.4.1 Car mobility

In our simulation, we use TraCIMobility module which extends BaseMobility and relies on TraCIScenarioManager for state updates. Basically, the TraCIScenarioManager holds a list of nodes updated continuously from the road traffic simulation through the TCP connection. Every simulation step, determined by the *updateInterval*, the module iterates over the entire node list, reads the next node position on the road and updates the fields accordingly.

The BaseMobility class takes care of the graphical representation of the module. So every change of the position is reflected into the on the graphical representation. It publishes the new position so that other modules get informed, e.g the physical layer must update the position with the ConnectionManager. It also takes care of exiting the border.

TraCIMobility contains a field called *statistics* which records generated data during the simulation. Recorded information are: *startTime*, *stopTime*, *totalTime*, *totalDistance*, *minSpeed*, *maxSpeed* and much more could be added.

3.4.2 WAVE: Wireless Access in Vehicular Environment

WAVE is system architecture built by IEEE and its main objective is to facilitate communication between vehicles and road side infrastructure. The reason why other existing standards aren't used is because they don't work efficiently with authentication and update of routing tables with the continuous changing network topology and short amount of time available for connection and communication.

Our NIC module implements an 802.11p interface card, with the physical and medium access control layers based on the IEEE standard, because it is a widely known standard with many manufacturers guaranteeing interoperability. Some features of 802.11p standard are (14):

- *Fast connection setup* is a necessary behaviour when running in a highly dynamic network topology, because the time spent in the communication range of other node is very short. For example the case of two vehicles travelling in opposite directions, on a high speed highway.

- *Random MAC address* ensure the anonymity of the vehicles make them untraceable. It makes it difficult to associate data, collected over the ad-hoc network, with a specific node and track it using this data.

- *Priority Control* helps achieve Quality of Service (QoS) in the applications running over the ad-hoc network. It prioritizes data from different applications and provides critical data access to the channel.

- *Power Control* enables the applications to specify the transmit power used to pass each packet over the channel. Customizing this from the application is useful, because one can modify it according to the surrounding topographical characteristics.

3.4.3 Physical layer

As (10) presents, the physical layer is the core of the wireless node in MIXIM framework. It provides message sending and receiving, collision detection and bit error calculation.

The MIXIM physical layer has three components described in details: the BasePhyLayer which provides the interfaces to the MAC layer, the AnalogueModel which is responsible for simulating the attenuation of received signal and the Decider responsible for evaluation and demodulation of the received messages. The analogue and decider modules are pure C++ classes.

The signal strength is influenced by the environment it travels in order to reach the destination node. It can be modeled with attenuation factors caused by path loss, shadowing and fading, and can be sent with multiple frequencies. Thus, a message can have varying sending power, attenuation, and bit-rate in time, space and frequency. MIXIM attaches to every message a signal object representing sending power, attenuation, and bit-rate in three dimensions time, frequency and space. The receiving node adds the attenuation.

When the physical layer receives a message, it passes the message to the analogue model to compute the attenuation part of the signal. It is also responsible for simulating the propagation and transmission delay of the message. The message is passed at least twice to the decider, at the beginning and the end of it transmission. The decider can request the message any time in-between. After the decider calculates the bit errors, the message reaches the MAC layer. The BasePhyLayer acts as an interface between physical layer messages(AirFrame) and the AnalogueModel and the Decider. The physical layer stores all messages in the ChannelInfo class which can compute the AirFrames intersecting with a given time interval. This is useful for the decider that calculates the signal-to-noise ratio (SNR) of a given message.

MIXIM simulates path loss, shadowing and fading, because Omnet++ doesn't have attenuation. The physical layer supports multiple analogue models, each one representing basically a filter class for signals. Summing up the attenuation from all analogue models, gives the attenuation part of the signal, calculated at the beginning and at the end of the message reception. The decider can calculate then, together with the sending power of the received message, the SNR and the bit errors.

The Decider is responsible with performing three main tasks: must classify incoming messages into noise or valid messages, must calculate the bit errors for a valid message when its reception is over and must provide information about the current state of the channel.

When the physical layer receives a message, it is sent to the decider which can either decide immediately whether the message is noise or not, or it can force the physical layer to resubmit the packet after a certain amount of time. This enable the decider to revise its decision, if a second stronger message arrives in the mean time.

The decider can request the message from the physical layer the latest at the end of message reception process. This moment is also the time for the decider to compute the bit errors for the message. It uses all the intersecting messages from the ChannelInfo in order to calculate the SNR. Then it can calculate bit errors and positions depending on the complexity of the particular decider model.

In the end, the decider must provide information about the channel state,

```
connections:

    radioIn --> phy80211p.radioIn;

    mac1609_4.lowerControlOut --> phy80211p.upperControlIn;
    mac1609_4.lowerLayerOut --> phy80211p.upperLayerIn;
    phy80211p.upperLayerOut --> mac1609_4.lowerLayerIn;
    phy80211p.upperControlOut --> mac1609_4.lowerControlIn;

    mac1609_4.upperControlIn <-- upperControlIn;
    mac1609_4.upperLayerIn <-- upperLayerIn;

    mac1609_4.upperLayerOut --> upperLayerOut;
    mac1609_4.upperControlOut --> upperControlOut;
```

Figure 3.2: Connections between physical and MAC layers.

needed at the MAC layer for Carrier Sense Multiple Access (CSMA) protocols. The MAC layer can request the decider to check the channel and return if it is currently idle or busy.

3.4.4 MAC layer

The Medium Access Control (MAC) protocol has the purpose to handle the medium for communication between nodes in the system. Since we use a wireless system the air represents the shared medium. Thus, the protocol needs to decide the moment when a node should emit signals in order to transmit a message, to avoid interference with signals from other nodes.

The IEEE 1609.4 standard is used which support the following features as shown in (14):

- *Control Channel Monitoring* : nodes have to monitor the channel at a predefined interval.
- *Synchronized Switching*: devices with a single channel cannot monitor the Control Channel and the Service Channel at the same time. Thus, they must synchronize to monitor the Control Channel at the same time, to be able to communicate.
- *Channel Routing* passes the data from the higher layers to the right channel. It also passes the data from the low layer to the correct networking protocol.

The model is discussed in detail in (5) and the implementation is available free and open source for Omnet++ simulator included in Veins framework for IVC simulation.

Figure 3.2 displays the *connections* between the physical module and the MAC module. This section states that all messages from the physical module are passed through the *upperLayerOut* gate and they are received at the *lowerLayerIn* gate

```
connections:
    nic.upperLayerOut --> appl.lowerLayerIn;
    nic.upperLayerIn <-- appl.lowerLayerOut;
    nic.upperControlOut --> appl.lowerControlIn;
    nic.upperControlIn <-- appl.lowerControlOut;

    radioIn --> nic.radioIn;
```

Figure 3.3: Connections between newtork interface card and application layer.

of MAC layer. Analogue connections exist when messages come from the upper layers. The $radioIn$ ensures all messages received are sent to the physical module to simulate the their reception. In fact messages come from $ConnectionManager$ as we stated above.

3.4.5 Application layer

The application layer runs the algorithm which handles the exchange of the messages between the nodes and is the top layer in the IEEE802.11p standard. Our application module is called FloatingContentApp and extends BaseWaveApplLayer. Figure 3.3 shows the pair of connections between the two layers.

The most useful feature our application inherits from the BaseWaveApplLayer is the beaconing system, used in our application for peer discovering. In Omnet++ scheduling future events in order to implement timers, timeouts or delays is performed by letting the simple module to send a message to itself. The messages are called $self - messages$ and are delivered later to the simple module. It does not use the network interface card for its implementation. Self-messages are sent using $scheduleAt()$ function which accepts absolute simulation time and a type of message for event identification as parameters. They are sent as regular messages and can be identified by using $isSelfMessage()$. We use self-messages to schedule beacons at an interval of 300s, a value which is read from the configuration file.

Regular messages are sent down to the lower layer using $sendDelayed()$ which keeps the message for a delay interval and send it afterwards. The message, delay value and gate id are the method arguments.

Messages arrived at a module are handled in $handleMessage()$ method where a the application can decide to which callback to pass it based on the gate it arrived on. This method is specific to $BaseLayer$, a class extended by $BaseWaveApplLayer$ which call $handleSelfMessages()$ for $self-messages$, $handleUpperMsg()$ for messages from upper layers and $handleLowerMsg()$ for messages from the lower layers. $BaseWaveApplLayer$ classifies again the messages received in $handleLowerMsg()$ in "beacons" or "data" and call the methods implemented by FloatingContentApp, $onData()$ and $onBeacon()$, accordingly.

Since our application must be aware of the node localization, there must be a method to send notifications from the road traffic simulator towards the network simulation at every simulation step. As we presented in an earlier section, the TraCIScenarioManager which handles message exchanges between the two simulators, updates the positions for every mobility module specific to a vehicle. Every mobility must notify the application module about the new position. Omnet++ has implemented a publish-subscribe style communication among modules. The advantage is that the two communication modules might not know about each other and possibly there is many-to-many relationship among them. In our implementation, the two modules we talk about are the *mobility* module and the *application* module.

Thus, every position update the network simulator receives, reaches the traffic scenario manager which notifies every mobility module in the network. The mobility module then signals the application module with a signal called *mobilityStateChangedSignal* and with the help of *emit*() function. The function takes the mobility module as argument to pass the new position in the application module.

The application must subscribe to the mobility state change signal in order to start listening for these type of events. The listeners use *receiveSignal*() function to handle the received signals and decide next steps according to their type. In our case the callback function *handlePositionUpdate*() is called, which computes results with the new position.

When a node appears in the simulation the *initialize*() method of each module and its submodule is called to perform data initialization. The application's ancestor class, *BaseWaveApplLayer* schedules the discovering system based on beacon messages.

We stored into a vector container the coordinates of the anchor zones we received information for. Thus, when a peer is discovered i.e. a beacon is received, we encapsulate the list into a wave short message and send it out to the sender of the beacon. Since the message is broadcasted, we simulate unicasting by checking that the message recipient address field is the same with the current node id. Otherwise we discard the message.

3.5 Collecting Simulation Data

Omnet++ has built-in support for recording simulation data, via output vectors and output scalars. Output vectors give the full picture of what is happening during the simulation, while the output scalars are summary results, computed during the simulation and written out when the simulation completes. We use the latter in our simulation in order to compute the *criticality* condition.

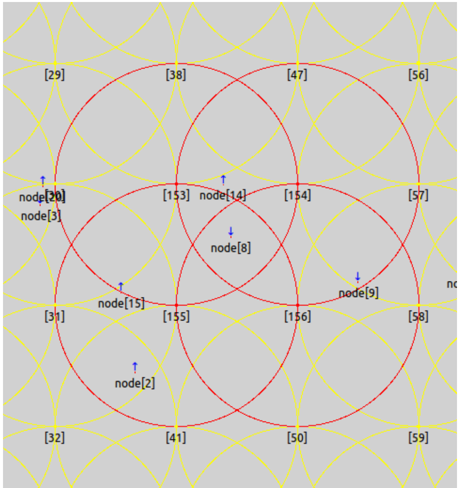


Figure 3.4: Anchor zones in which node number 8 resides, are coloured with red, the other ones are coloured with yellow.

To estimate the feasibility of floating content, we positioned anchor zones every 200m in a grid across the simulation area as shown in the figure **??**. More details about the simulation environment in the next chapter.

Every anchor zone has its own independent module which records multiple statistics during the simulation : the total number of vehicles crossing the anchor, the average sojourn time of a vehicle, the number of contacts of a vehicle during its sojourn time, etc. We decided to hold this data inside the $TraCIScenarioManager$ module due to its importance and the fact that it is unique and provides the same data to all the modules.

With the help of $handlePositionUpdate()$, we can track the vehicles inside an anchor zone. We use different replication ranges for the anchor zones which are bigger than the distance between them. This results in having overlapped anchor zones. Determining all the anchor zones a vehicle resides in, may be quite a challenge.

Our approach is very simple and complexity optimised. Based on the position of the vehicle, the box in which the vehicle resides from the grid can be determined. Each of the corners represents the center of an anchor zone. If the radius of an anchor zone is equal with the distance between the anchor zones, the vehicle can reside in some or all the anchor zones placed in the corners of its box as shown in Figure 3.4. If the radius is bigger, than we process all the anchor zones that are farther and contain the vehicle. Opposed to iterating over all anchor zones and perform operations, this method reduces the computational complexity.

When an encounter takes place i.e. a module receives a message and $onData()$ method is called we must check if both nodes reside in the same anchor zone as the calculation of *criticality condition* requires. Thus, we keep the above algorithm for anchor zones calculation, but we verify if both vehicles, the sender or the receiver reside in the same anchor zone.

The following values are recorded and calculated in our simulation :

- *maxTransitNodes* represents the maximum number of nodes that exists at a given time in the anchor zone.
- *avgContactsInSJNTime* represents the average number of contacts in their sojourn time.
- *avgTimeInAnchor* denotes the average sojourn time of vehicles in an anchor zone.
- *criticality* which is equal to $maxTransitNodes$ * $avgContactsInSJNTime$ * $avgTimeInAnchor$.

Chapter 4

Simulation Environment and Evaluation

In this section we will discuss about the simulation environment, traffic generation and the evaluation of resulted data.

4.1 Environment

The environment for our simulation required by Omnet++ was generated using OpenStreetMap (7) and several SUMO tools. We decided to use three different city scenarios to show that floating content in urban environments can be feasible regardless of particularity of road architecture. The cities we used as simulation environment are: SanFrancisco, Beijing and Erlangen.

The road network file is generated from the exported map (from OpenStreetMap) file using *netconvert*. It has many paramaters that steer how the network is imported and how the resulting SUMO-network is generated. We removed isolated edges because they have no influence and benefit for the simulation scenario. We also removed the unnecessary nodes that split edges without being at a junction. We clustered junctions with traffic lights and joined the traffic light logic.

OpenStreetMap contains not only the road network, but also a wide range of additional polygons such as buildings and rivers. They represent the obstacles existing on the map which generate signal attenuation and influence the communication between vehicles.

As suggested in (12), we also chose two different anchor zone radii: $a = r \epsilon \{200\text{m}, 500\text{m}\}$. To evaluate the feasibility of floating content in the urban area we placed anchor zones every 200m horizontally and vertically across the entire simulation area. Figure 4.1 shows how anchor zones are distributed on the simulation map.

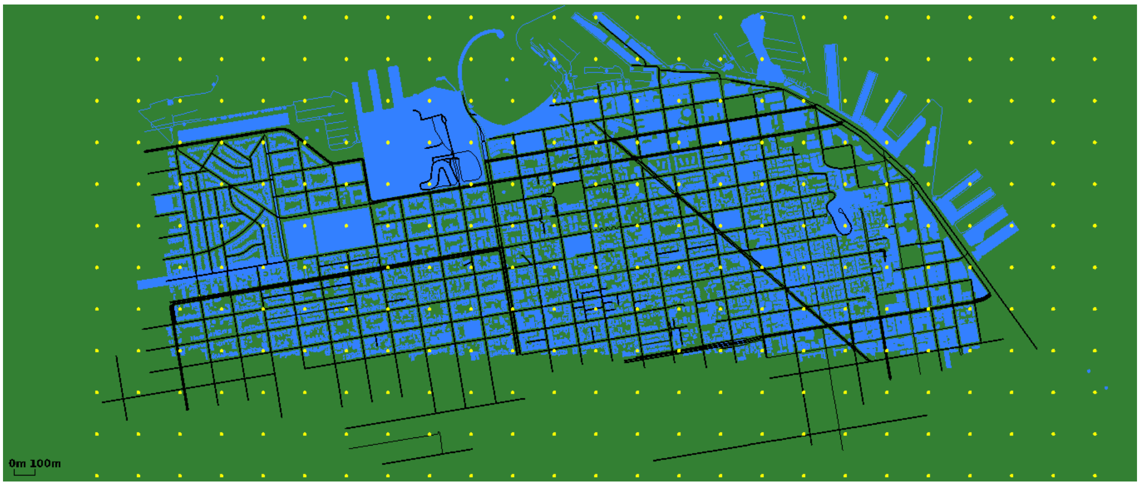


Figure 4.1: Anchor zones every 200m over the simulation area

4.2 Mobility Simulation

There are many methods used to obtain suitable mobility pattern for simulations in urban vehicular scenarios. In this section we briefly describe several relevant mobility patterns.

One of the most popular mobility models to evaluate mobile ad hoc network (MANET) routing protocols is called "Random Waypoint Model". It became popular due to its simplicity and wide availability. Each node following this mobility pattern pauses for a fixed number of seconds and selects a random destination in the simulation area and a random speed between 0 and some maximum speed. When it arrives at the destination it repeats the previous step. Despite of its simplicity, the analysis from (21) revealed that it can be harmful when simulating vehicular ad hoc networks.

Real mobility traces is another method to obtain mobility models and presents a clear improvement over the random mobility. Traces are obtained from real vehicles such as taxis and buses which simply move in an ordinary day in a city. The mobility of nodes is tracked using on board equipment or road side equipment. Although they generate the most realistic mobility patterns, it is clear that it is influenced by other non tracked vehicles whose movement does not appear in the recorded traces. The major problem is that it lacks flexibility and does not allow performing exhaustive evaluation of ad hoc network protocols. For example we cannot change vehicle density without modifying speed (2).

As we stated in the previous chapter we use the SUMO road traffic simulator which overcome the restrictions of real traces with no loss of realism. SUMO defines two concepts used to generate traffic for a map generated network: trips which denotes the origin and the destination for a single vehicle, and flows which defines the same trip for multiple vehicles. We used a couple of SUMO tools to generate our traffic demand:

- *randomTrips.py* is a random trip generator. It inserts a new vehicle every second with a trip having a random origin and destination. It doesn't check if the two locations on the map are connected.
- *duarouter* generates the traffic demand using a file with trips and flows. It is made up of vehicles with assigned routes, calculated using the Dijkstra algorithm. Unconnected trips are deleted.

We ensured that every car travels a minimum distance of 1500 meters from origin to destination. *randomTrips.py* distribute the vehicles randomly on their starting edges taking into account their speed and the edge maximum speed. We decided to place a bigger proportion of high speed vehicles on the map which will be placed on lanes with higher maximum speeds. This way, there will be heavy traffic on highways and anchor zones with higher *criticality conditions* should appear in the areas where the traffic jams may occur. It is a method to emphasize the anchor zones with high floating probability without loss of generality. The speed ranges from 5m/s (18km/h) to (79km/h).

4.3 Evaluation

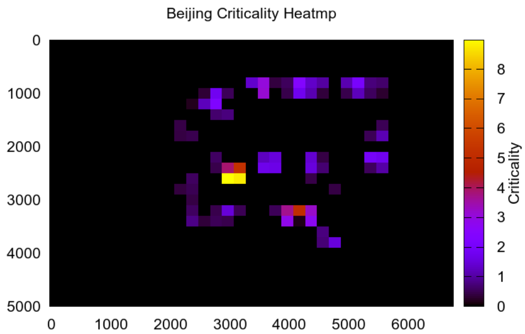
We decided to collect data from 300s of simulation. Although the time interval appear to be short, it is sufficient for gathering consistent data. Since the application relies on data broadcasting, it forces the simulator to schedule thousands of messages which results in usage of a great amount of time and memory e.g. 10 minutes of simulation for San Francisco city is processed in 1 hour and 45 minutes. We analyzed the probability of information floating taking into consideration the particularity of the road network as well as the spatial distribution of vehicles, and also the evolution during the simulation using the SIR model.

4.3.1 Beijing

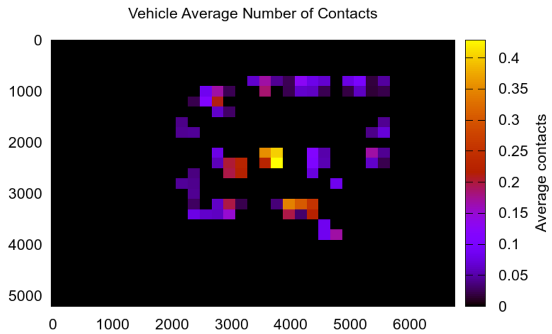
Figure 4.2a displays an area from Beijing city on which we conducted our simulations. More specifically the figure is a snapshot that displays the road network from an area of 6.7×5.2 km^2. The map is coloured with respect to the amount of time a car has to wait due to the traffic lights placed in intersections. Road segments coloured in red have a waiting time of about 40s, more than the ones coloured in black which have a continuous flow of vehicles and thus a waiting time close to 0s.

The figures below display the *criticality* heatmap along with the average number of contacts a vehicle encounters during its sojourn time. The similarity between them is clearly visible.

(a) Beijing traffic waiting time heatmap



(b) Beijing Criticality Map $r = 200m$

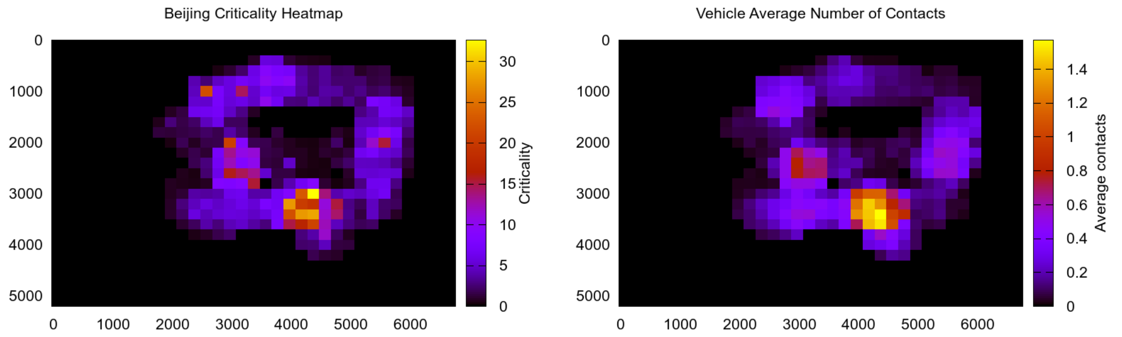


(c) Average Number of Contacts Map $r = 200m$

Figure 4.2: Beijing 300s simulation with 200m anchor size

The black areas state that the *criticality* factor is nearly 0, thus the probability of information floating is very low. Indeed, the black areas occur in places between streets or outside the traffic road network, where neither vehicles, nor their radio range can reach them. In the 200m scenario, the anchor zones with a higher probability of floating scatter the map. In the blue coloured areas, appears an average *criticality condition*, slightly above 1, the criticality threshold. According to the figure 4.2c, 9 out of 10 vehicles do not have any interaction with other vehicles, during their sojourn time. The reason could be the small amount of time spent in that location or lack of neighbours at the time a beacon is sent out.

The bright yellow area, occurs naturally in the road intersection. The expectations for an information to live rise with respect to the average waiting time on a road segment. The *criticality factor* rise well over 8 as well as the average number



(a) Beijing Criticality Map $r = 500m$ **(b)** Average Number of Contacts Map $r = 500m$

Figure 4.3: Beijing 300s simulation with 500m anchor size

of contacts a vehicles has during its sojourn time.

Increasing the anchor range yields better floating results. Figure 4.3 shows that information has a high probability of living along the road network. Also, the small light yellow areas in figure 4.3b reflect that every node has at least one contact during its sojourn time. The *criticality condition* increases 4 times the one computed for a range of 200m.

Figure 4.4 shows the information evolution within an anchor zone during a simulation time of 1000s: infected and healed. We decided to set the *ttl* parameter (time-to-live) to 600s, a realistic number that could be used for short lasting events and long enough to collect consistent data. We chose an anchor zone with a high number of information exchanges to provide a detailed picture of its evolution and avoided the ones which disappeared before the expiry time because of stochastic fluctuations.

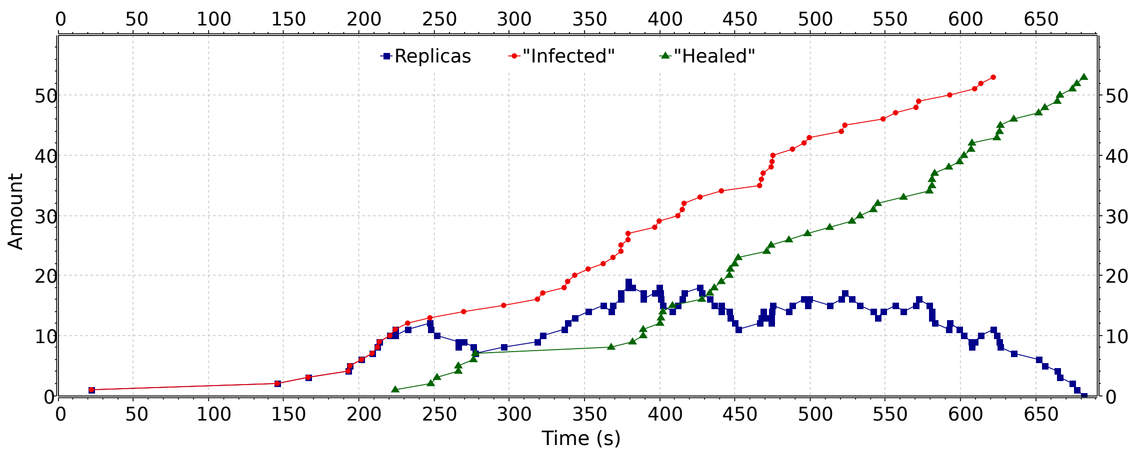


Figure 4.4: Information life evolution with 500m range. The number of copies is equal with the total number of “infections” minus the total number of “healings”.

We recorded the number of replicas an information has during the simulation denoted by *Replicas* label. The other two labels are 2 classes which belong to the epidemic SIR model: *infected* denote the information carrying nodes, and *healed* the ones which previously had a copy of the information and deleted it. The deletion could occur if the distance between the vehicle and the geo-location origin was bigger than the anchor radius or the information availability expired.

It is important to determine whether the information will spread or not. And if it does, for how long, and how it will develop in time. As we presented in subsection 2.4.2, if $S_0 > S_c = a/r$ the content will spread. The plot in 4.5 confirms the condition for a 500m anchor radius: the threshold μ is around 0 during most of the information lifetime and the amount of susceptibles is greater than the threshold. This means that they could possibly receive a copy in the near future and become "infected". The end time period states that the number of susceptibles become less than μ and thus the replication process reaches the end. The reason the threshold suddenly becomes so big is because the application deletes the item once the lifetime expires. The "infected" will become susceptible again, fact confirmed by the figure which shows a slight increase.

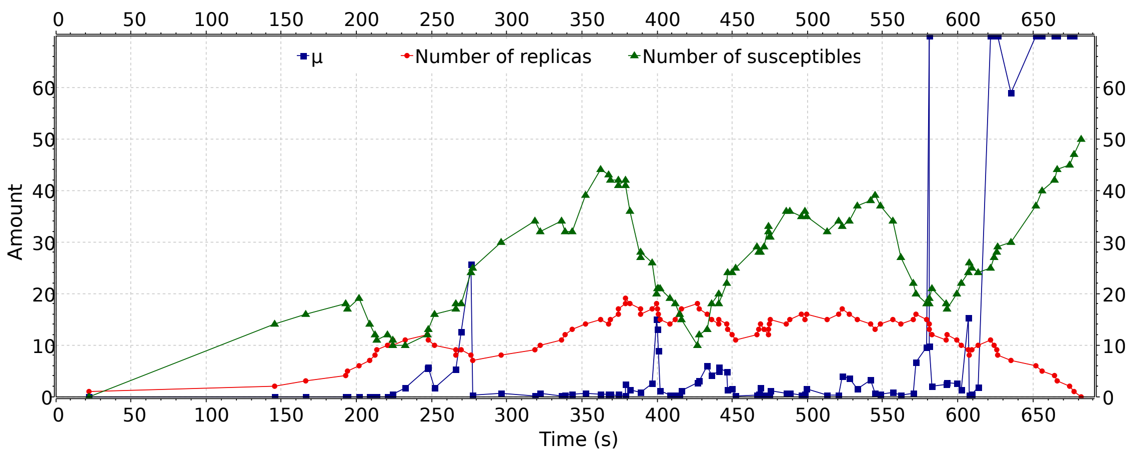


Figure 4.5: When $S > \mu$ the number of replicas the number of replicas can increase, on the other hand, when $S < \mu$, the number of replicas decrease towards 0; μ represents the epidemy threshold; the anchor zone radius is 500m

Figure 4.6 depicts how the information develops with respect to radius size. Establishing the optimum range enables the equipment and application to save battery energy and have a better resources management.

Anchor zones with $r \in \{100m, 200m\}$ yield a short life expectancy. In our scenario the content lasts for about 100s, just 1/6 of the due time. Such distances are probably more suitable if used indoor public places like shopping malls, metro stations, usually overcrowded places were the message exchanging is possible.

Ranges within $\{300m,\ 400m\}$ increase the life time up to 500s, but the information disappears ahead of time. The range of 500m keeps the item alive until its termination time. As long as the range is big enough to hold a considerable amount of vehicles inside, the information will last.

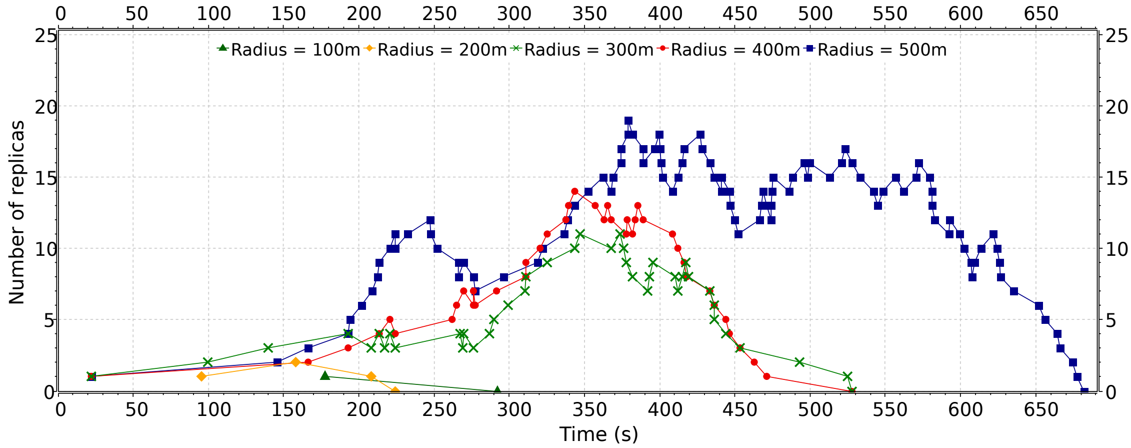


Figure 4.6: Information life evolution with respect to radius size

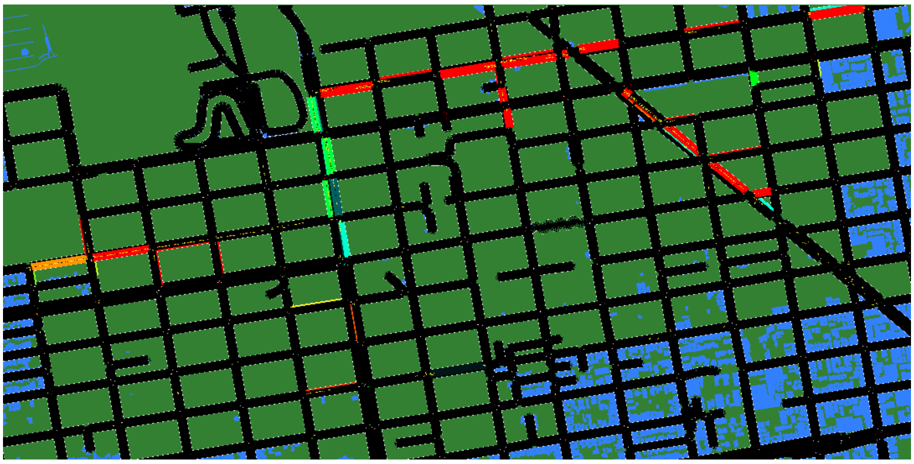
4.3.2 San Francisco

The area selected from San Francisco used in our simulations has 4.5×2.5 km^2 and contains a road network made up from road streets with different numbers of lanes on each way. We removed the bicycle, pedestrian, cityrail and public transport segment roads and left only the segments for regular road vehicles. Figure 4.7a displays the road average waiting time vehicles must be aware of, because of the traffic lights logic. Traffic jams occur on the red road segments which determine the vehicles to spend a big amount of time in place.

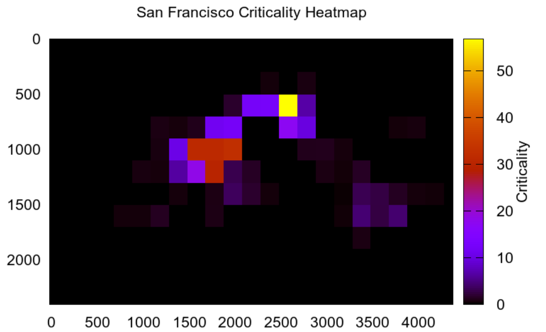
The traffic generator created random trips for the vehicles used in simulation, having random starting points and destinations. However, the vehicles majority choose the high speed roads in order to finish their trips, and thus traffic jams occur. This explain the red coloured segments in the figure.

The *criticality* heatmap and the average number of contacts map created with 200m anchor radius, match the coloured figure in 4.7a. Black areas appear around the two-lane streets which scatter the map, between the San Francisco blocks. The $criticality factor$ is close to 0 which is supported by the low density of vehicles and the fact that they don't remain around to much.

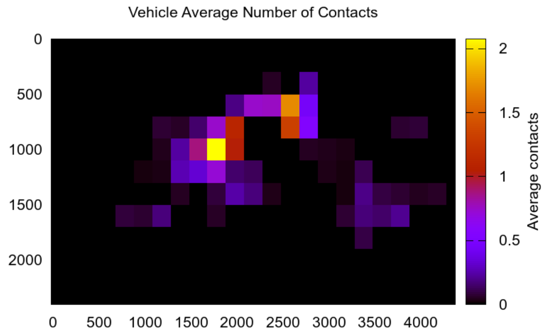
The areas which are expected to keep the information floating are coloured in blue and yellow. The diagonally placed street is clearly an information carrying medium, the average number of contacts being close to one. The intersection with the top horizontal street clusters the vehicle, raising the number of contacts up to 2



(a) San Francisco traffic waiting time heatmap



(b) San Francisco Criticality Map $r = 200m$

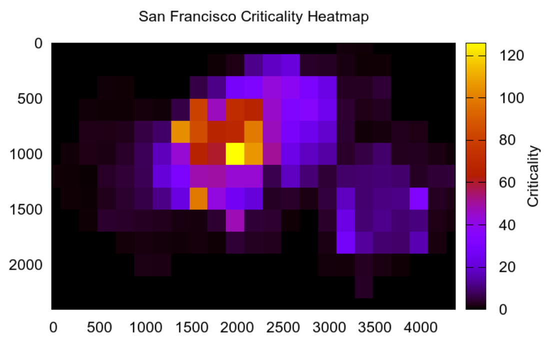


(c) Average Number of Contacts Map $r = 200m$

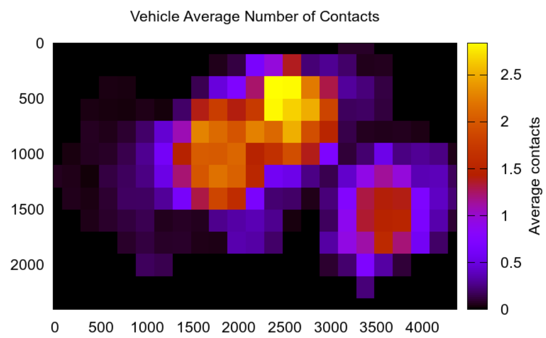
Figure 4.7: SanFrancisco 300s simulation with 200m anchor size

for each car which travels the location. Thus, the *criticality condition* reaches over 50 in the same area.

Figure 4.8 depicts the results using an anchor range of 500m. While the average number of contacts slightly increases from 2 to 2.5 the *criticality* reaches 120 in the top spots, nearly 2.5 times than the former approach. This is possible because the anchor zones now cover larger areas, thus involving more cars in the exchanging process. The bigger radius make the anchor zones more diffusive inside the heatmap confirming that a user may receive much more items spread on a larger area. In the 200m scenario, the areas are more concentrated, with the need of being close to the high traffic zones in order for the replication to be possible. The big values that appear over a big portion of the map are an indicator that even in the probabilistic system used the information will float.



(a) Criticality Map $r = 500m$



(b) Average Number of Contacts Map $r = 500m$

Figure 4.8: San Francisco 300s simulation with 500m anchor size

4.3.3 Erlangen

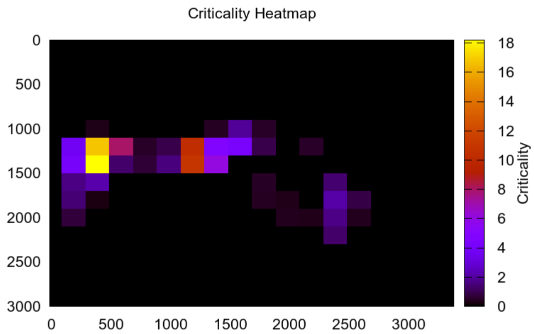
For our third and final scenario, we chose an area of Erlangen, which is 3.5×3.1 km^2 and has a less dense road network than San Francisco, although is about the same size. Figure 4.9 displays the road segments with a high probability of holding often traffic jams, and thus, forcing the vehicles to stay in place a considerable amount of time.

It is clearly visible that islands which occur in the plots take the shape of the main street in Erlangen; black areas generated a criticality below 1 and thus, are not a suitable place for sustaining information floating.

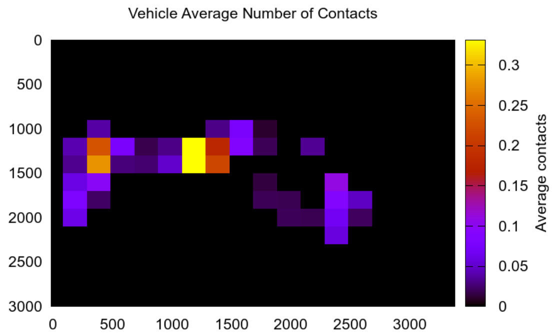
Naturally, the main street, which crosses the entire area, has a consistent flow of vehicles with an average number of contacts of 0.1. After the calculation, the average criticality on the main street becomes 6.

Figure 4.10 depicts the results for a scenario with an anchor zone radius of 500m, keeping the same effect seen in other 2 cities. The criticality increases nearly 4 times which is also supported by the fact that each node had at least one encounter during the period it crossed the anchor location.

(a) Erlangen traffic waiting time heatmap

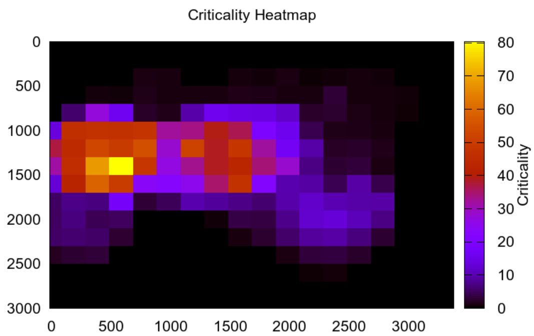


(b) Criticality Map $r = 200m$

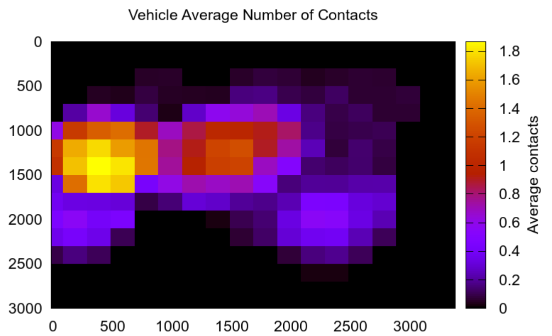


(c) Average Number of Contacts Map $r = 200m$

Figure 4.9: Erlangen 300s simulation with 200m anchor size



(a) Criticality Map $r = 500m$



(b) Average Number of Contacts Map $r = 500m$

Figure 4.10: Erlangen 300s simulation with 500m anchor size

Bibliography

[1] Worldwide mobile phone users: H1 2014 forecast and comparative estimates, 2014.

[2] Miguel Báguena, Sergio M Tornell, Alvaro Torres, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni. Vacamobil: Vanet car mobility manager for omnet++. In *Communications Workshops (ICC), 2013 IEEE International Conference on*, pages 1057–1061. IEEE, 2013.

[3] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for p2p and mobile ad-hoc networks. *Analysis*, 2004.

[4] Michael Behrisch Daniel Krajzewicz, Jakob Erdmann and Laura Bieker. Recent development and applications of sumo - simulation of urban mobility. *Analysis*, pages 128–138, 2012.

[5] David Eckhoff and Christoph Sommer. A multi-channel ieee 1609.4 and 802.11 p edca model for the veins framework. In *Proceedings of 5th ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems: 5th ACM/ICST International Workshop on OMNeT++.(Desenzano, Italy, 19-23 March, 2012). OMNeT+*, 2012.

[6] Pasi Lassila Jussi Kangasharju Jrg Ott Esa Hyyti, Jorma Virtamo. When does content float? characterizing availability of anchored information in opportunistic content sharing. *Analysis*, 2011.

[7] Mordechai (Muki) Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, oct 2008.

[8] Herbert W Hethcote. The mathematics of infectious diseases. *SIAM review*, 42(4):599–653, 2000.

[9] Jon Einar Kolberg. Bmw assist. *Online*, 2014.

[10] A. Kopke, M. Swigulski, K. Wessel, D. Willkomm, P. T. Klein Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating

wireless and mobile networks in omnet++ the mixim vision. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 71:1–71:8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[11] Francesca Martelli, Maria Elena Renda, and Paolo Santi. Measuring ieee 802.11 p performance for active safety applications in cooperative vehicular systems. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1–5. IEEE, 2011.

[12] Jorg Ott, Esa Hyytia, Pasi Lassila, Tobias Vaegs, and Jussi Kangasharju. Floating content: Information sharing in urban areas. *Simulation*, pages 136–146, 2011.

[13] Jorg Ott, Esa Hyytia, Pasi Lassila, Tobias Vaegs, and Jussi Kangasharju. Floating content: Information sharing in urban areas. *Simulation*, pages 136–146, 2011.

[14] Harshad Phule. *Public Safety Application for Approaching Emergency Vehicle Alert and Accident Reporting in VANETs Using WAVE*. PhD thesis, Rochester Institute of Technology, 2012.

[15] Christoph Sommer, David Eckhoff, Reinhard German, and Falko Dressler. A computationally inexpensive empirical model of ieee 802.11 p radio shadowing in urban environments. In *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, pages 84–90. IEEE, 2011.

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

[17] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[18] Volvo. Volvo on call. *Online*, 2014.

[19] Axel Wegener, MichałPiórkowski, Maxim Raya, Horst Hellbruck, Stefan Fischer, and Jean-Pierre Hubaux. Traci: An interface for coupling road traffic

and network simulators. In *Proceedings of the 11th Communications and Networking Simulation Symposium*, CNS '08, pages 155–163, New York, NY, USA, 2008. ACM.

[20] wikipedia. Delay-tolerant networking @MISC, 2011.

[21] Jungkeun Yoon, Mingyan Liu, and Brian Noble. Random waypoint considered harmful. In *INFOCOM 2003. twenty-second annual joint conference of the IEEE computer and communications. IEEE societies*, volume 2, pages 1312–1321. IEEE, 2003.