Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Emergency Message Dissemination and broadcasting protocols

Emergency Message Dissemination (EMD) is a fundamental application in VANET to prevent traffic accidents, thereby reducing death and injury rates. Such task can be executed by the VANET itself by turning it into an infrastructure-less self-organizing network, where the dissemination is carried out by specific protocols.

Since traffic information, especially emergency data, has a broadcast-oriented nature (i.e. it is of public interest), it is more appropriate to disseminate it using broadcasting routing scheme rather than unicast or multicast ones. **5989903**

This choice leads to some advantages, such as:

* the fact that vehicles do not need to know the destination address and how to calculate a route towards it;
* a greater coverage of vehicles interested in the information, useful also in lossy scenarios, especially when paired with controlled redundancy schemes;
* a greater efficiency in bandwidth usage.

The idea behind existing algorithms consists in designating the next forwarder in the multi-hop chain from the source of the alert to the target region where the sensitive data has to be delivered. Ideally, the farthest vehicle from a previous forwarder in the dissemination direction should be given priority when designating the next forwarder. However, due to unreliable wireless channel, the designation of farthest vehicle can fail and interrupt the message dissemination. Due to this, next forwarder designation keeps into consideration also other vehicles (called potential forwarder candidates, PFCs) which have received an Alert Message. The PFCs participate in contention to elect the farthest forwarder candidate (FFC) who will continue disseminating the message.

In order to carry out the forwarder designation process, the main idea consists in differentiating waiting times (WT) of PFCs. Each PFC should select a waiting time ranging from 0 to a predefined upper bound (PUB). To guarantee the correct designation of the farthest vehicle as forwarder, PFCs choose their waiting time inversely proportional to the distance between the PFC and the previous forwarder. This way other candidates can detect the transmission from the FFC and suppress their transmission.

Advancements in research on Emergency Message Dissemination has lead to the development of a number of broadcasting protocols. However, as identified by Panichpapiboon et al.**5989903**, most of them belong to one of two main categories:

* Single-hop Broadcasting Protocols, in which no flooding is employed. Instead, vehicles periodically select and broadcast only a subset of the packets it has received;
* Multi-hop Broadcasting Protocols, in which packets are transmitted through the network via flooding by some of the neighbors of the source. It is of utmost importance to reduce the number of redundant transmission in order not to waste bandwidth and saturate the channel.

1.1.1 Single-hop Broadcasting Protocols

Vehicles employing Single-Hop Broadcasting protocols will not flood received packets immediately through the network. Instead, vehicles use information from packets to update their database and periodically rebroadcast only a fraction of that information. The two variables these kind of protocols can work on to aim for network efficiency are:

* *Broadcast Interval*, i.e. the amount of time between retransmissions, which should keep into consideration both freshness of information and potential redundancy in transmissions;
* *Relevancy of information* to broadcast: as stated before, only relevant information (i.e. a subset of all the information) should be broadcast.

Single-Hop protocols can be further subdivided into two categories:

1. Fixed Broadcast Interval, which keep the Broadcast Interval fixed. Some examples are:
   - *TrafficInfo***4621303**, a protocol in which vehicles record, among other information, travel times on road segments (identified by an ID) and keep them on their on-board database. Vehicles periodically exchange information about the learned travel times based on the relevance of such information. The relevance is calculated using a ranking algorithm which uses the current position of the vehicle and the current time (i.e. relevance decreases with distance and time), broadcasting only the k most important information.
   - *TrafficView***1263039**, in which vehicles exchange information about speed and position and record it in their database. Data about different vehicles is then aggregated into a single record using one of two aggregation algorithms:
     - the *ratio-based* algorithm, which assigns an aggregation ratio to each portion of a road: the more important the road is, the higher the aggregation ratio will be, increasing the accuracy of the information of that area;
     - the *cost-based* algorithm, an algorithm which keeps into consideration the cost of aggregating different records. The aggregation cost is defined as the loss of accuracy the aggregation will bring about.
2. Adaptive Broadcast Interval, which adapt the Broadcast Interval based on dynamic information. Some examples are:

- *Collision Ratio Control Protocol (CRCP)***4357748**, a scheme according to which vehicles exchange information about location, speed and road ID. The Broadcast Interval is dynamically controlled based on the amount of detected collisions and bandwidth efficiency: the protocol tries to maintain the number of collisions under a certain threshold by doubling the Broadcast Interval every time the threshold is exceeded. Otherwise, the Broadcast Interval is decreased by one second when the bandwidth efficiency decreases too much.

  Moreover, the authors propose three different methods for selecting the data to be transmitted:

  - *Random Selection*: a vehicle selects a random information in its database and broadcasts it;
  - *Vicinity Priority Selection*: vehicles give priority to information of nearby areas;
  - *Vicinity Priority Selection with Queries*: similar to Vicinity Priority Selection, with the possibility of querying information for a certain area.
- *Abiding Geocast:***4531929**, which aims to deliver an Alert Message to a specific area where the warning is still relevant. Only vehicles that are travelling towards the effective area can participate in contention to broadcast the message. Moreover, broadcast is dynamically adjusted based on transmission range, speed, and distance between the potential forwarder and the destination area, increasing when such distance increases or the potential forwarder's speed decreases.
- *Segment-oriented Data Abstraction and Dissemination (SODAD)***1402433**, a protocol according to which roads are divided into segments and each vehicle can both discover information itself and collect it from neighbor's transmissions. Whenever a vehicle receives a transmission from another vehicle, the information received will be classified as either one of two events:
  - a *provocation* event that will decrease the Broadcast Interval;
  - a *mollification* event that will increase the Broadcast Interval.

  The classification is done via comparison of the newly received data with the information stored in the vehicle's on-board database. The vehicle assigns a higher weight if the difference between information coming from these two sources is high. The weight will be then compared against a threshold to establish whether a provocation of mollification event has taken place.

1.1.2 Multi-hop Broadcasting Protocols

Multi-hop Broadcasting Protocols can be further subdivided into two categories:

(a) Delay based protocols, which assign a different waiting time before re-broadcasting the message to each vehicle. This delay is usually inversely proportional to the distance between the source and the potential sender. Some examples are:

- *Urban Multi-hop Broadcast (UMB)* **Korkmaz:2004:UMB:1023875.1023887**, designed to solve the broadcast redundancy, hidden node and reliability problems in multi-hop broadcasting using *Request-to-Broadcast (RTB)* and *Clear-To-Broadcast (CTB)* packets;

- *Smart Broadcast (SB)* **4025102** and *Efficient Directional Broadcast (EDB)* **4340158**, which try to reduce the delay introduced by *UMB* and remove the *RTB* and *CTB* packets, respectively;
- *Vehicle-density-based Emergency Broadcasting (VDEB)* **5663803**, a slotted broadcasting protocol which keeps vehicle density into consideration when computing waiting time slots;
- *Reliable Method for Disseminating Safety Information (RMDSI)* **4591259**, which aims to offer better performances when the network becomes fragmented by making a forwarder keep a copy of the packet it has broadcasted until it hears a retransmission (or until the packet lifetime expires). If no retransmission is heard within a certain time limit, the forwarder tries to find the next node which can relay the message using a small control packet;
- *Multi-hop Vehicular Broadcast (MHVB)* **4068699**, a protocol that keeps traffic congestion into consideration by checking whether the number of neighbors of a vehicle is greather than a certain threshold and its speed is smaller than another threshold. When a node detects congestion, it increases its broadcast interval in order to try to reduce the network load;
- *Reliable Broadcasting of Life Safety Messages (RBLSM)* **4458046**, whose main objective is reliability, and a higher priority is given to the vehicle nearest to the sender instead to the one farthest from it, due to the assumption that the closer the vehicle is, the more reliable it is considered since its received signal strength is higher.

(b) Probabilistic-based Multi-hop Broadcasting Protocols. The idea behind these kind of protocols is similar to the one behind Delay based protocols, but instead of assigning a different rebroadcast delay to each vehicle, a different rebroadcast probability is assigned. Each protocol differs in the function that assigns probabilities. Some examples of probabilistic-based protocols are:

- *Weighted p-Persistence* **4407231**, in which every PFC computes its own rebroadcast probability based on distance between itself and the transmitter. The formula used is the following:

$$p_{ij} = \frac{D_{ij}}{R} \tag{1.1}$$

  where D_{ij} is the distance between transmitter i and PFC j and R is the transmission range. Based on this function, the probability to rebroadcast is proportional to the distance between the PFC and the transmitter. The abovementioned formula does not keep into account vehicle density and also assumes that the transmission range is fixed and known to all vehicles.
- *Optimized Adaptive Probabilistic Broadcast (OAPB)***1543865** *and AutoCast (AC)* **4350058**, which both keep the vehicle density into consideration when computing the forwarding probability by making vehicle periodically exchange Hello Messages. Thanks to those messages, each vehicle can compute the number of neighbors and then use this information accordingly.

– *Irresponsible Forwarding (IF)* **47402775426212**, a protocol that considers vehicle density like *OAPB* and *AC*, but the formula used is not a simple linear function. In fact, the rebroadcast probability assignment function is the following:

$$p = e^{-\frac{\rho_s(z-d)}{c}} \tag{1.2}$$

where ρ_s is the vehicle density, z is the transmission range, d is the distance between the PFC and the transmitter and $c \geq 1$ is a shaping parameter which influences the rebroadcast probability. *Irresponsible Forwarding* aims to offer a solution that can scale with network density.

The previous work**ROM2017** focused on the implementation and testing of Fast-Broadcast, a Multi-hop delay based protocol firstly presented in **4199282**. The main focus of the algorithm is to try to overcome the assumption of a fixed and known transmission range, which other protocols often tacitly assume. The protocol will be further analyzed in Chapter 3.

The authors of ROFF **6906275**, another Multi-Hop delay based protocol, state that existing protocols are affected by two problems:

* the perfect suppression of redundant transmissions, by which potential forwarders which have lost the contention detect the transmission from the farthest vehicle and suppress their transmission. However this suppression can not always be guaranteed due to short difference between waiting times. In fact, if the timer of a potential forwarder expires before it has heard the transmission from the FFC, a redundant transmission will occur;
* the disuniformity and the costant change in spatial vehicle distribution in VANETs. Existing protocols which keep into consideration the distance between PFC and previous forwarder do not keep into consideration large empty spaces in the waiting time computation, leading to unnecessary wait.

ROFF's solutions to these problems and the implementation of the protocol will be analyzed in Chapter 4.

1.2 Radio Propagation Models

Since field testing in VANETs, especially in large scenarios, is usually expensive and difficult to execute in real settings, researchers usually carry out their tests in a simulated environment, such as ns-3 (Section 5.1). In order to model the transmission of signal throughout various media, several radio propagation models have been developed. A Radio Propagation Model (RPM) is an empirical mathematical formulation used to model the propagation of radio waves as a function of frequency, distance, transmission power and other variables. Over the years various RPMs have been developed, some aiming at modelling a general situation, and others more useful in specific scenarios. For example, implementations range from the more general free space model, where only distance and power are considered, to more complex models which account for shadowing, reflection, scattering, and other multipath losses. Moreover, it is important to keep into consideration the computational complexity and scalability of the model: some have poor accuracy but are scalable, while others have very

good accuracy but can only work for small sets of nodes. As always, it is very important to find the right trade-off between complexity and accuracy.

The authors of **6298165** classify the propagation models offered by the network simulator ns-3 in three different categories:

* **Abstract** propagation loss models, for example the Maximal Range model (also known as Unit Disk), which establishes that all transmissions within a certain range are received without any loss;
* **Deterministic** path loss models, such as the Friis propagation model, which models quadratic path loss as it occurs in free space, and Two Ray Ground, which assume propagation via two rays: a direct (LOS) one, and the one reflected by the ground;
* **Stochastic** fading models such as the Nakagami model, which uses stochastic distributions to model path loss.

These traditional models, especially the stochastic ones, work quite well to describe the wireless channel characteristics from a macroscopic point of view. However, given the probabilistic nature of the model, single transmissions are not affected by the mesoscopic and microscopic effects of the sorrounding environment. To keep these effects into consideration, researchers have utilized Ray-Tracing, a geometrical optics technique used to determine all possible signal paths between the transmitter and the receiver, considering reflection, diffraction and scattering of radio waves, suitable both for 2D and 3D scenarios **245274 765022**.

However, a Ray-Tracing based approach, while producing a fairly accurate model, is not very scalable due to its high computational complexity, especially in a real-time scenario. To overcome this problem, the authors of **STEPANOV200861** have resorted to a fairly computationally expensive pre-processing, but this leads to the need of pre-processing every scenario (and also every change in the scenario).

The RPM utilized in this work will be presented in Section 2.1.

Chapter 2

Models

In this Chapter the models implemented and utilized in this work will be presented, namely the Obstacle Shadowing propagation loss model and the Junction model.

2.1 Obstacle Shadowing propagation loss model

The original thesis **ROM2017**, after having analyzed various works concerning shadowing in urban scenarios **Giordano:2010:CST:1860058.1860065 4020783** used a deterministic RPM called Obstacle Shadowing propagation loss model presented in **5720204** and implemented by the authors of **Carpenter:2015:OMI:2756509.2756512**. This propagation model calculates the loss in signal strength due to the shadowing effect of obstacles such as buildings.

The authors of **5720204** designed the model as an extension of well-established fading models, which can be expressed by Equation 2.1, where:

* P are the transmit or receive powers of the radios;
* G are the antenna gains;
* L indicate the terms capturing loss effects during transmission.

$$P_r[dBm] = P_t[dBm] + G_t[dB] + G_r[dB] - \sum L_x[dB] \tag{2.1}$$

Common RPMs can be written as components L of 2.1 and chained to obtain the compound attenuation. For example, Equation 2.2 and 2.3 represent respectively the Two-Ray Ground and Log-Normal models.

$$L_{TwoRayGround} = 10 \lg \left(\frac{d^4 L}{h_t^2 h_t^2} \right) \qquad [dB] \tag{2.2}$$

$$L_{LogNorm} = 10 \lg \left(X_\sigma \right) \qquad [dB] \tag{2.3}$$

The authors extended the general model shown in 2.1 adding a L_{obs} term for each obstacle in the line of sight between sender and receiver. The term is described by Equation 2.4, where:

* n is the number of times that the line of sight intersects the borders of the obstacle;
* d_m is the length of the obstacle's intersections;
* β represents the attenuation due to the exterior wall of a building, in dB per wall;
* γ represents an approximation of the internal structure of a building, in dB per meter.

Parameters β and γ can be fitted to represent different types of buildings. $\beta \approx 9.6$ dB per wall and $\gamma \approx 0.4$ dB/m are the values proposed by the authors for buildings in suburban areas.

$$L_{obs} = \beta n + \gamma d_m \tag{2.4}$$

Figure 2.1 shows an example of transmission where the signal encounters $n = 4$ walls.

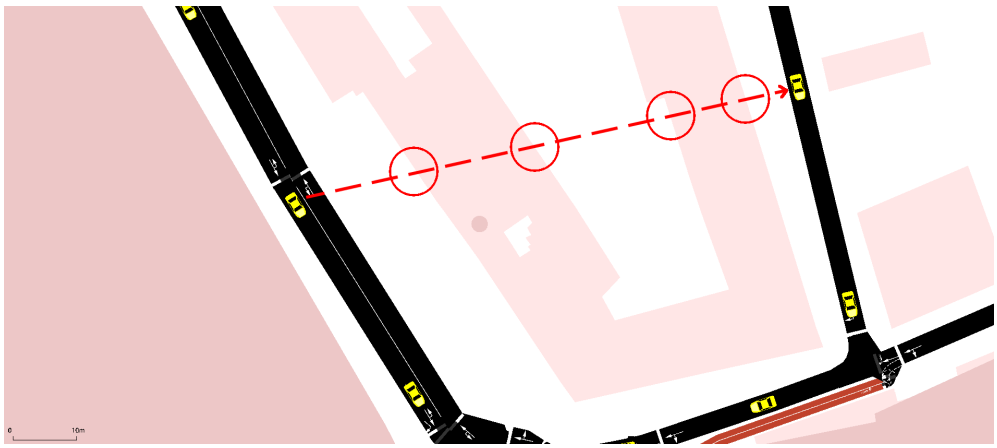

Figure 2.1: Example of obstacle shadowing in vehicle-to-vehicle communication. Walls encountered by the signal are surrounded in red circles

2.2 Junction modeling

Part of this work consisted in implementing extensions for the algorithms which will be presented in Chapter 3 and 4 to utilize junctions in a smart way. These extensions aim to reach a higher number of vehicles during the Alert Message propagation. In order for those extension to work, a model to identify and represent junctions was necessary. Using data retrieved from OpenStreetMap elaborated through SUMO (see Section 5.2 for the full process), it has been possible to retrieve information about junctions present inside a urban scenario. Figure 2.2 shows the JSON representation of a junction.

```
<junction id="122666592" type="priority" x="1204.30" y="2735.10" incLanes="13446476#0_0
-13433275#0_0 -13446476#1_0" intLanes=":122666592_0_0 :122666592_9_0 :122666592_10_0
:122666592_3_0 :122666592_4_0 :122666592_5_0 :122666592_6_0 :122666592_7_0 :122666592_11_0"
shape="1205.98,2741.13 1210.40,2736.50 1208.80,2733.84 1208.24,2731.90 1207.84,2729.55 1207.61,
2726.79 1207.54,2723.62 1201.14,2723.60 1200.81,2725.51 1200.41,2725.91 1199.85,2725.92 1199.13,
2725.56 1198.25,2724.82 1193.81,2729.43">
```

Figure 2.2: Example of data about a junction retrieved from OpenStreetMap and elaborated through SUMO

Among other data, the most useful information about a junction are:

* the x and y coordinates, which identify its center;
* the *shape* attribute, containing coordinates which represent its shape.

Using from this data it was possible to establish the junction modeling process, represented in Figure 2.3. Since the *shape* attribute resulted in a polygon too small for the purpose of this work, a bounding box which extends the polygon by 20 meters in each direction has been implemented.

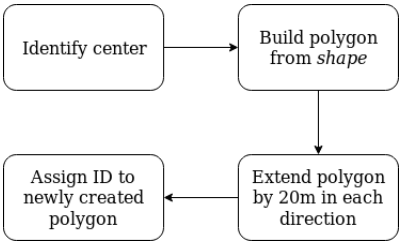

Figure 2.3: Junction modeling process

Figure 2.4 shows the results of the above-mentioned process executed on six different junctions. The red polygon is the polygon defined by the *shape* attribute in the JSON definition of a junction, while the yellow rectangle is the bounding box. Each vehicle knows whether it is inside a junction based on its coordinates. A vehicle is inside a junction only if its coordinates are inside of one of the bounding boxes.

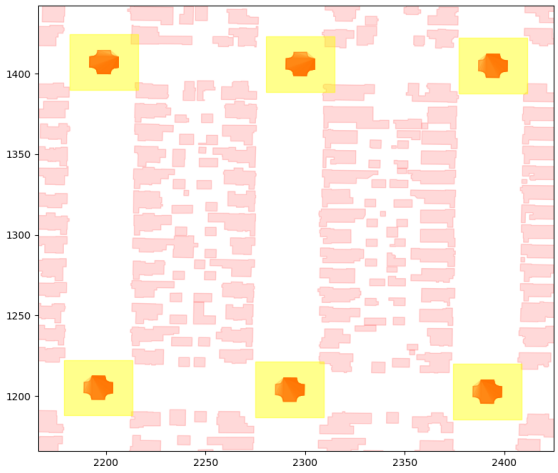

Figure 2.4: Example of 6 junctions in an urban scenario

Chapter 3

Fast-Broadcast

Fast-Broadcast **4199282** is a multi-hop routing protocol for vehicular communication. Its main feature consists in breaking the assumption that all vehicles should know, *a priori*, their fixed and constant transmission range. This assumption is often unreasonable, especially in VANETs and urban environments, where electromagnetic interferences and obstacles such as buildings heavily influence the transmission range.

Fast-Broadcast employs two different phases:

1. the **Estimation Phase**, during which vehicles estimate their frontward and backward transmission range;
2. the **Broadcasts Phase**, during which a vehicle sends an Alert Message and the other cars need to forward it in order to propagate the information.

3.1 Estimation Phase

During this phase, vehicles try to estimate their frontward and backward transmission range by the means of Hello Messages. These beacons are sent periodically via broadcast to all the neighbors of a vehicle.

Time is divided into turns and, in order to keep estimations fresh, data collected during a certain turn is kept for the duration of the next turn, then discarded. The parameter *turnSize* specifies the duration of a turn: the authors suggest a duration of one second. A bigger *turnSize* could guarantee less collisions to the detriment of freshness of information. On the other hand, the effects of a smaller *turnSize* are specular to those just presented.

Using this approach, vehicles can estimate two different values:

* *Current-turn Maximum Front Range (CMFR)*, which estimates the maximum frontward distance from which another car can be heard by the considered one;
* *Current-turn Maximum Back Range* (CMBR), which estimates the maximum backward distance at which the considered car can be heard.

When the turn expires, the value of these variables is stored in the *LMFR* and *LMBR* variables (*Latest-turn Maximum Front Range* and *Latest-turn Maximum Back Range*, respectively). The algorithm uses both last turn and current turn data because the former guarantees values calculated with a larger pool of Hello Messages, while the latter considers fresher information.

When sending a Hello Message (Algorithm 1), the vehicle initially waits for a random time between 0 and *turnSize*. After this, if it has not heard another Hello Message or a collision, it proceeds to transmit a Hello Message containing the estimation of its frontward transmission range.

When receiving a Hello Message (Algorithm 2), the vehicle retrieves its position and the sender's position, calculates the distance between these two positions and then updates the CMFR field if the message comes from ahead, otherwise CMBR is updated. The new value is obtained as the maximum between the old CMFR or CMBR value, the distance between the vehicle and the sender, and the sender's transmission range estimation included in the Hello Message.

Algorithm 1 Hello message sending procedure for 1D

```
1: for each turn do
2:     sendingTime ← random(turnSize)
3:     wait(sendingTime)
4:     if ¬ (heardHelloMsg() ∨ heardCollision()) then
5:         helloMsg.declaredMaxRange ← max(LMFR, CMFR)
6:         helloMsg.senderPosition ← retrievePosition()
7:         transmit(helloMsg)
8:     end if
9: end for
```

Algorithm 2 Hello message receiving procedure for 1D

```
1: myPosition ← retrievePosition()
2: senderPosition ← helloMsg.senderPosition
3: declaredMaxRange ← helloMsg.declaredMaxRange
4: d ← distance(myPosition, senderPosition)
5: if receivedFromFront(helloMsg) then
6:     CMFR ← max(CMFR, d, declaredMaxRange)
7: else
8:     CMBR ← max(CMBR, d, declaredMaxRange)
9: end if
```

3.2 Broadcast Phase

This phase is activated once a vehicle sends an Alert Message. The other cars can exploit the estimation of transmission ranges to reduce redundancy in message broadcast. Each vehicle can exploit this information to assign itself a forwarding priority inversely proportional to the relative distance: the higher the relative distance, the higher the priority.

When the Broadcast Phase is activated, a vehicle sends an Alert Message with application specific data. Broadcast specific data is also piggybacked on the Alert Message, such as:

* *MaxRange:* the maximum range a transmission is expected to travel backward before the signal becomes too weak to be received. This value is utilized by following vehicles to rank their forwarding priority;
* *SenderPosition*: the coordinates of the sender.

Upon reception, each vehicle waits for a random time called *Contention Window* (*CW*). This window ranges from a minimum value (*CWMin*) and a maximum one (*CWMax*) depending on sending/forwarding car distance (*Distance*) and on the estimated transmission range (*MaxRange*), according to formula 3.1. It is quite easy to see that the higher the sender/forwarder distance is, the lower the contention window is.

$$\left\lfloor \left(\frac{\text{MaxRange} - \text{Distance}}{\text{MaxRange}} \times (\text{CWMax} - \text{CWMin}) \right) + \text{CWmin} \right\rfloor \tag{3.1}$$

If another forwarding of the same message coming from behind is heard during waiting time, the vehicle suppresses its transmission because the message has already been forwarded by another vehicle farther back in the column. On the contrary, if the same message is heard coming from the front, the procedure is restarted using the new parameters. The vehicle can forward the message only if the waiting time expires without having received the same message.

Algorithm 3 and 4 describe the logic behind the Broadcast Phase.

Algorithm 3 Alert Message generation procedure for 1D

```
1: alertMsg.maxRange ← max(LMBR, CMBR)
2: alertMsg.position ← retrievePosition()
3: transmit(alertMsg)
```

Algorithm 4 Alert Message forwarding procedure for 1D

```
1: cwnd ← computeCwnd()
2: waitTime ← random(cwnd)
3: wait(waitTime)
4: if sameBroadcastHeardFromBack() then
5:     exit()
6: else if sameBroadcastHeardFromFront() then
7:     restartBroadcastProcedure()
8: else
9:     alertMsg.maxRange ← max(LMBR, CMBR)
10:    alertMsg.senderPosition ← retrievePosition()
11:    transmit(alertMsg)
12: end if
```

3.3 Multiple dimensions extension

The original work **4199282** considered only a strip-shaped road, where it was easy to define directions and establish when a message came from the front or the back. In **BAR2017** an extension considering two dimensions was proposed. This work proposes a modification to that version, since it caused some delivery problems in particular scenarios. This section will firstly present the original extension of **BAR2017** and then discuss the proposed modification.

The modifications to the Fast-Broadcast algorithm are the following:

1. Utilizing only one parameter between CMBR and CMFR (thus considering only CMR):

2. Including the position of the vehicle which originally generated the Alert Message in addition to the position of the sender of the message.

When a vehicle receives an Alert Message, the origin-vehicle distance is confronted with the origin-sender distance: the vehicle can forward the message only if the former is greater than or equal to the latter, otherwise it simply discards the message.

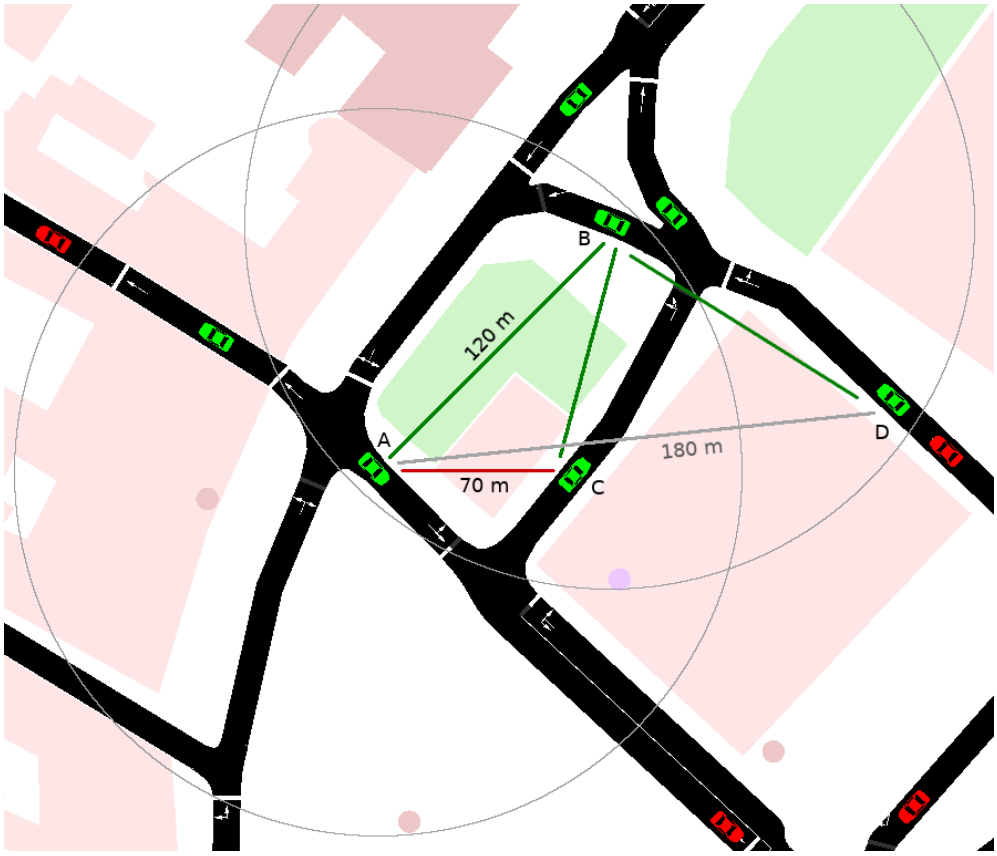

Figure 3.1: Example of Fast-Broadcast in 2D scenario

For example, suppose that vehicle A is the origin of the Alert Message and B receives it, but C doesn't due to an obstacle in the line of sight. B computes origin-vehicle distance, $d(A, B)$, and origin-sender distance, $d(A, A)$, which in this case are respectively 120 and 0m. Since origin-vehicle is greater than origin-sender, B can forward the Alert Message.

Now suppose that C receives the message from B. C computes origin-vehicle distance, $d(A, C)$, and origin-sender distance, $d(A, B)$, which amount to 70 and 120m respectively. Since the former is not greater than or equal to the latter, C is not a candidate for forwarding and suppresses the transmission.

D receives the message from B as well. D is a good candidate for transmission since the origin-vehicle distance, which amounts to 180m, is greater than origin-sender distance, equal to 120m.

However, the condition by which vehicles suppressed their transmission when the origin-sender distance was smaller than the origin-sender distance causes delivery

problems in particular scenarios, reported in Appendix TODO

. Hence, that condition has been dropped in the algorithm implemented in this work. Referring to the example in Figure 3.1, this change makes vehicle C a forwarder candidate as well.

Algorithm 5, 6, 7 and 8 show the implementation for Fast-Broadcast for multiple dimensions (mainly 2D, but this version actually works also for 3D scenario, for example scenarios with only drones or with drones and vehicles).

Algorithm 5 Hello message sending procedure for 2D

```
1: for each turn do
2:     sendingTime ← random(turnSize)
3:     wait(sendingTime)
4:     if ¬ (heardHelloMsg() ∨ heardCollision()) then
5:         helloMsg.declaredMaxRange ← max(LMR, CMR)
6:         helloMsg.senderPosition ← retrievePosition()
7:         transmit(helloMsg)
8:     end if
9: end for
```

Algorithm 6 Hello message receiving procedure for 2D

```
1: myPosition ← retrievePosition()
2: senderPosition ← helloMsg.senderPosition
3: declaredMaxRange ← helloMsg.declaredMaxRange
4: d ← distance(myPosition, senderPosition)
5: CMR ← max(CMR, d, declaredMaxRange)
```

Algorithm 7 Alert Message generation procedure for 2D

```
1: alertMsg.maxRange ← max(LMR, CMR)
2: alertMsg.senderPosition ← retrievePosition()
3: alertMsg.originPosition ← retrievePosition()
4: transmit(alertMsg)
```

Algorithm 8 Alert Message forwarding procedure for 2D

```
1: cwnd ← computeCwnd()
2: waitTime ← random(cwnd)
3: wait(waitTime)
4: if sameBroadcastHeardFromBack() then
5:     exit()
6: else if sameBroadcastHeardFromFront() then
7:     restartBroadcastProcedure()
8: else
9:     alertMsg.maxRange ← max(LMR, CMR)
10:    alertMsg.senderPosition ← retrievePosition()
11:    transmit(alertMsg)
12: end if
```

3.3.1 Hello Message Header Structure

The Hello Message header structure is represented in Figure 3.2.

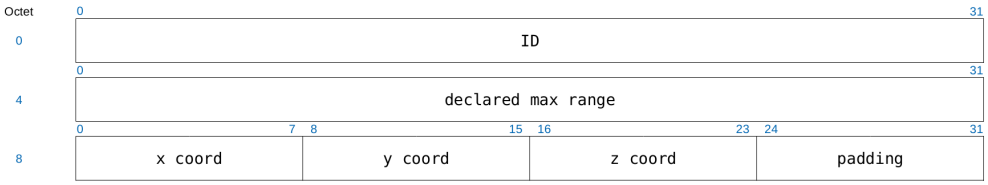

Figure 3.2: Fast-Broadcast Hello Message header structure

The fields contained in the Hello Message are the following:

* *ID*: unique ID of the Hello Message's sender;
* *declared max range*: maximum range detected by the vehicle during Estimation Phase;
* *x coord*, *y coord* and *z coord*: coordinates of the Hello Message's sender.

3.3.2 Alert Message Header Structure

The Alert Message header structure is represented in Figure 3.3.

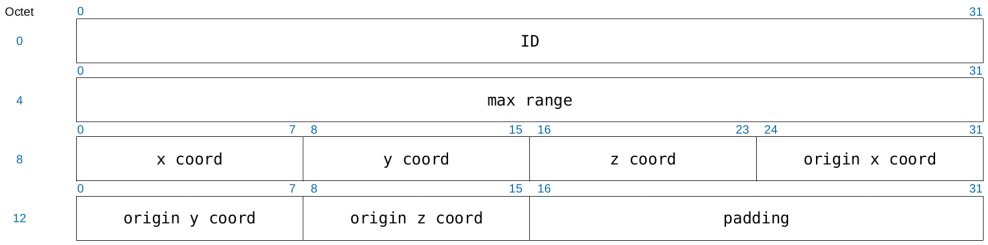

Figure 3.3: Fast-Broadcast Alert Message header structure

The fields contained in the Alert Message are the following:

* *ID*: unique ID of the Alert Message's sender;
* *max range*: maximum range detected by the vehicle during Estimation Phase;
* *x coord*, *y coord* and *z coord*: coordinates of the Hello Message's sender;
* *origin x*, *origin y* and *origin z coord*: coordinates where the Alert initially originated from.

3.4 Smart Junction Fast-Broadcast (SJ-Fast-Broadcast)

This work proposes an extension for Fast-Broadcast which keeps junctions into consideration in order to achieve a greater delivery ratio of Alert Messages. The idea comes from the following observation: whenever buildings block signal propagation

across different roads, the message is relayed only through the road segments inside the forwarder's field of view. This causes the Alert Message propagation to spread:

* in one direction, if the forwarder is not inside a junction and the road segment is surrounded by buildings which block the signal;
* in three directions, whenever the forwarder lies inside a junction.

Actually the signal travels respectively in two and four directions, but we are not considering the direction where the previous Alert Message is coming from (i.e. we are only considering forwarding towards previously uncovered areas in this analysis). Referring to Figure 3.4, if the message is coming from the top and A and B forwards it, then only vehicles in the north-south road will be reached. Vehicles on the west-east road will be reached by the Alert Message at a later point, if the propagation circles back to them through another road, otherwise they will never be informed about the alert. If C is also forced to forward, then we have coverage across all road segments and the propagation can continue towards all directions. Figure 3.5 shows an example where SJ-Fast-Broadcast is employed and a greater coverage is achieved.

Figure 3.4: Usual Fast-Broadcast Alert Message propagation

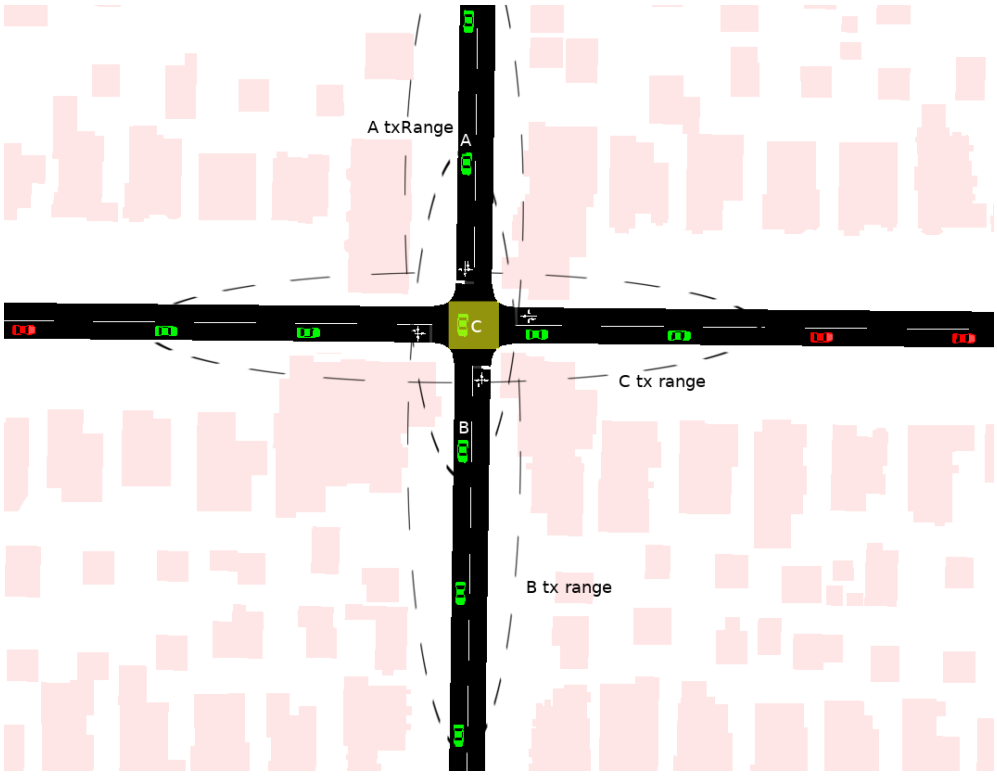

Figure 3.5: SJ-Fast-Broadcast Alert Message propagation

The proposed extension works by having nodes inside a junction participate in a second contention with all other vehicles inside the same junction. When the original timer calculated by Fast-Broadcast expires for one of the vehicles inside the junction, it forwards the message. All other vehicles inside the same junction suppress their transmission. This way the propagation can proceed in all directions, while the propagation in the normal direction can proceed without additional delays. Algorithm 9 and 10 show the code for SJ-Fast-Broadcast's Broadcast Phase. Estimation Phase's algorithms remain the same as Algorithm 5 and 6. Line 4 of Algorithm 10 shows that a vehicle suppresses its transmission in one of the following two cases:

* if the vehicle has heard the same broadcast coming from the back and is not inside a junction;
* if the vehicle has heard the same broadcast coming from the back, is inside a junction with ID j and the sender is also inside the same junction.

Algorithm 9 SJ-Fast-Broadcast Alert Message generation procedure

```
1: alertMsg.maxRange ← max(LMR, CMR)
2: alertMsg.senderPosition ← retrievePosition()
3: alertMsg.originPosition ← retrievePosition()
4: alertMsg.senderInJunction ← isSenderInJunction()
5: alertMsg.junctionId ← getJunctionId()
6: transmit(alertMsg)
```

Algorithm 10 SJ-Fast-Broadcast Alert Message forwarding procedure

```
1: cwnd ← computeCwnd()
2: waitTime ← random(cwnd)
3: wait(waitTime)
4: if (sameBroadcastHeardFromBack() ∧ vehicleNotInJunction()) ∨ (sameBroad-
   castHeardFromBack() ∧ vehicleInJunction(j) tel∧ alertMsg.senderInJunction ∧
   alertMsg.junctionId == j) then
5:     exit()
6: else if sameBroadcastHeardFromFront() then
7:     restartBroadcastProcedure()
8: else
9:     alertMsg.maxRange ← max(LMBR, CMBR)
10:    alertMsg.senderPosition ← retrievePosition()
11:    transmit(alertMsg)
12: end if
```

3.4.1 Alert Message Header Structure

The Hello Message header structure is the same as Figure 3.2, hence it is not reported here. The Alert Message header structure is represented in Figure 3.6.

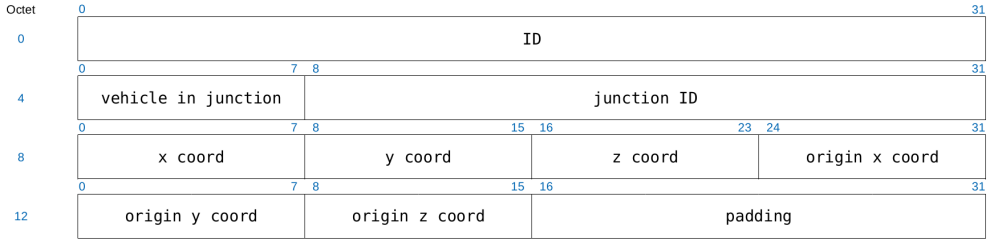

Figure 3.6: SJ-Fast-Broadcast Alert Message header structure

The additional fields are the following:

* *vehicle in junction*: whether the vehicle which is send this Alert Message is inside a junction;
* *junction ID*: the ID of the junction the sender is inside.

Chapter 4

ROFF

RObust and Fast Forwarding scheme (ROFF) is a protocol proposed by Hongseok Yoo and Dongkyun Kim in **6906275**. This chapter will present the two main problems tackled by ROFF already introduced in Section 1.1, namely the perfect suppression of redundant transmissions, which will be explained in 4.1.1 , and the disuniformity and the costant change in spatial vehicle distribution in VANETs, addressed in section 4.1.2.

4.1 Forwarder Selection Problem

4.1.1 Collision Analysis

The first problem tackled by ROFF concerns collisions caused by nodes who start to transmit at the same time. This results in a collision in the area resulting from the intersection of the nodes' transmission ranges.

Suppose that $S_f = \{f_i | 0 < i \leq N, i \in \mathbb{N}\}$ is the set of PFCs ordered in ascending order by the distance between the previous forwarder and the PFC, where N is the number of PFCs and f_n is the FFC. We define f_0 as the previous forwarder. Based on the most common idea in existing protocols, ideally a PFC f_i suppresses its scheduled transmission whenever it receives the transmission from f_N. In order to achieve suppression, vehicles from f_i to $f_i - 1$ should wait until they receive the transmission from f_N before forwarding. If a vehicle forwards the message before having received the transmission from f_N, then a collision will occur. As stated in Section 1.1, existing protocols employ a strategy for waiting time assignment by which each PFC calculates its waiting time based on the distance between itself and the previous forwarder (that is $distance = d_{f_i,f_0}$). As a consequence, successful suppression of all PFCs (f_1 to f_{N-1}, f_{N-1} included) can be achieved only if the timer of f_{N-1} is long enough to detect the transmission from f_N. The authors define $minDiff$ as the minimum time difference between f_N and f_{N-1} to prevent f_{N-1} from forwarding.

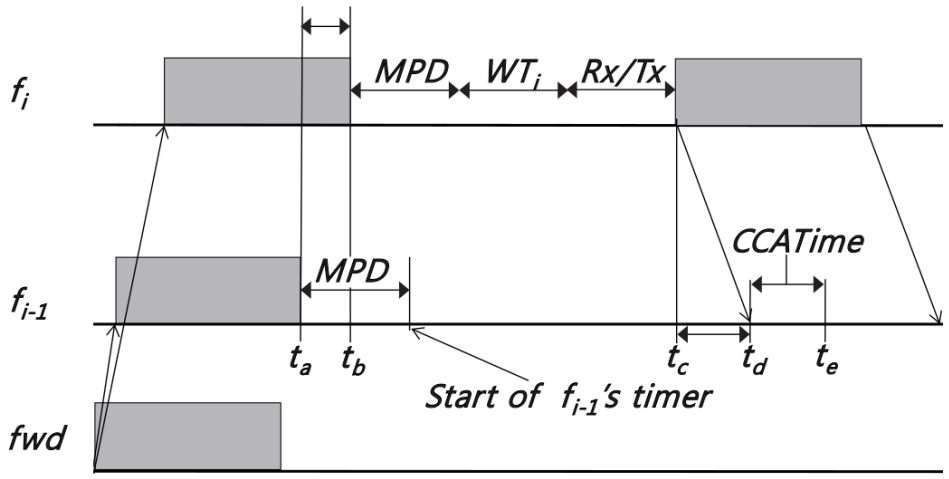

Figure 4.1: Definition of $minDiff$ between f_N and f_{N-1} (**6906275**)

Figure 4.1 depicts a situation where fwd is the forwarder and f_i (farther from fwd than f_{i-1}) relays the message before f_{i-1}. The two vehicles f_i and f_{i-1} complete receiving the message at different times (t_a and t_b) due to propagation delay (calculated as $pd = d/s$, where d is the distance between two points in space and s is the wave propagation speed of the medium, i.e. $s = c$, the speed of light, in wireless communication). After reception we have additional amount of time in order to process and retransmit the message:

* each PFC spends MAC Processing Delay (MPD) to process the message and then waits for WT_i calculated according to whichever multi-hop algoritm is being used. As explained previously, this waiting time is inversely proportional to the distance between the PFC and fwd;
* after timer expiration, each PFC wait for Rx/Tx turnaround time (Rx/Tx), in order to switch their interface from reception to transmission mode.

After f_i has forwarded the message, f_{i-1} starts receiving it at t_d, $t_d - t_c$ being equal to pdf_{i-1}, f_i. The time between the PHY module of f_{i-1} starts reception and the MAC module of the same vehicle is aware of reception is called $CCATime$. Hence, if f_{i-1}'s timer expires between t_a and t_e, f_{i-1}'s transmission will collide with f_i's. In order to accomplish succesful suppression, f_{i-1}'s timer should not expire before t_e.

Given the fact that propagation delay is not controllable and MPD, Rx/Tx and $CCATime$ are usually standard-defined parameters, an algorithm can only manage the difference between waiting times of f_i and f_{i-1} (represented by WT_i and WT_{i-1} respectively in the following formula). To achieve succesful suppression, f_{i-1} should wait until the forwarding from f_i is detected by f_{i-1}'s MAC layer, so $MPD + WT_{i-1}$ should be greater than $t_e - t_a$. Hence, the formula for $minDiff$ is the following:

$$minDiff = (pd_{fwd,f_i} - pd_{fwd,f_{i-1}}) + pd_{f_i,f_{i-1}} + Rx/Tx + CCATime \tag{4.1}$$

4.1.2 Latency Analysis

The second problem ROFF tries to overcome is the effect of empty space in vehicle distribution on forwarding latency.

The region where PFCs are placed can be defined as *naive forwarding area* (NFA) and is defined as the intersection between:

* the transmission range of a forwarder fwd;
* the area in the opposite movement direction of the same forwarder fwd.

The distribution of vehicle inside NFA can vary in time; moreover, vehicles are not usually placed at the same distance, so empty spaces of various sizes exist inside the area.

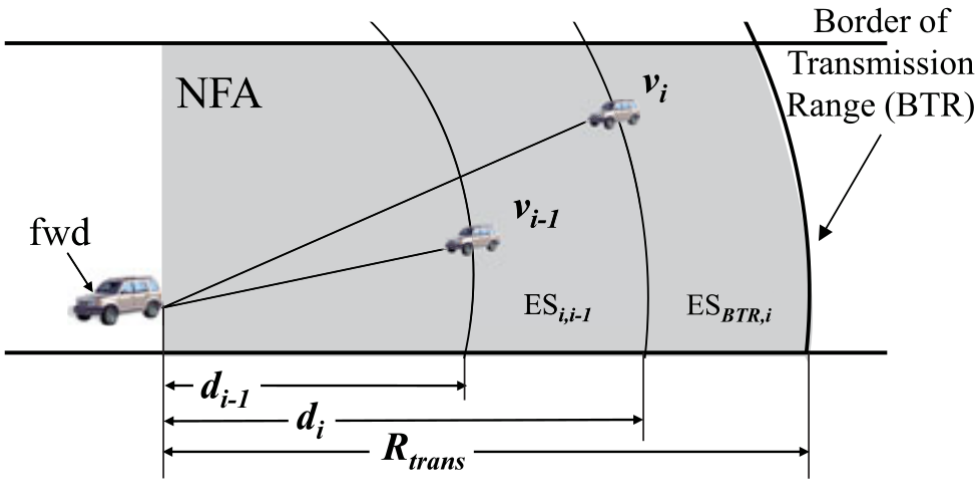

Figure 4.2: Definition of empty space (**6906275**)

Suppose that $S_v = \{v_i | 0 < i \leq M\}$ is a set of M vehicles inside NFA with vehicles ordered by distance between the vehicle and the previous forwarder in ascending order. Referring to Figure 4.2 , the empty space $ES_i, i-1$ between v_i and v_{i-1} is the segment within the two circles centered in fwd with radius d_{i-1} and d_i respectively. Hence, the size of $ES_i, i-1$ is equal to $d_i - di - 1$.

Large empty spaces have a negative effect on forwarding latency. There is no guarantee that the farthest vehicle from the previous forwarder become the FFC: lossy channel environments, shadowing and other phenomena can make any vehicle within NFA become the forwarder. Suppose that fwd is the previous forwarder and there are two vehicles, A and B, inside NFA where A is farther from fwd than B, A has not received the transmission from fwd while B has. The empty space ES_A, B between A and B influences the forwarding latency:

* if ES_A, B is small, B relays the message after a short waiting time;
* if ES_A, B is large, B waits needlessly (since there is no other vehicle farther from fwd which has received the message) a large amount of time.

ROFF aims at resolving the effect of empty spaces by allowing vehicles to choose waiting time inversely proportional to their *unique forwarding priority* proportional to the distance between the vehicle and the previous forwarder, instead of using directly such distance in the waiting time computation.

4.2 ROFF Algorithm

The ROFF algorithm works under the following two assumptions:

1. each vehicle has access to a GPS system and a digital map;
2. vehicles periodically (e.g. every 100 milliseconds) broadcast a Hello message containing various information, such as its position, velocity, etc. The period between each Hello message broadcast is called *Beacon Interval*.

The algorithm is composed of three components, as depicted in Figure 4.3.

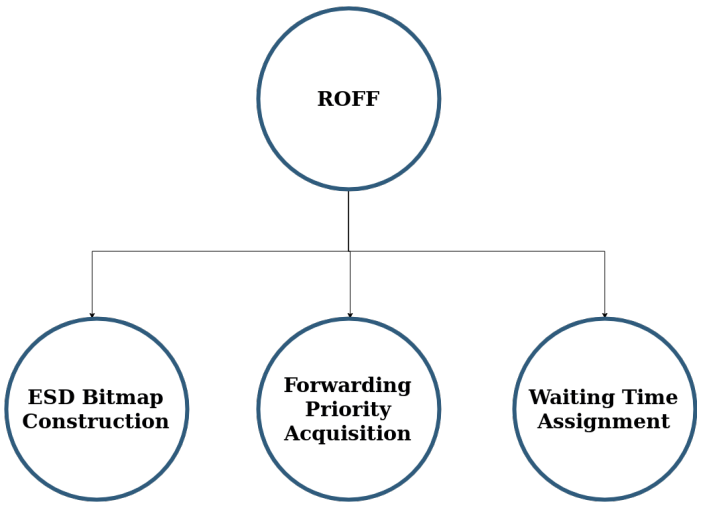

Figure 4.3: Components of ROFF algorithm

The algorithm in short works as follows (additional information will be given in the following sections):

* when a forwarder relays an Alert Message, it also broadcasts a special structure called ESD Bitmap which describe the empty space distribution within the forwarders' NFA;
* PFCs which receive the Alert Message and the ESD Bitmap decide whether they are eligible for contention based on the ESD Bitmap;
* Eligible PFCs contend by choosing different waiting times based on a unique forwarding priority.

4.2.1 ESD Bitmap Construction

Upon Hello Message reception, each vehicle uses the data within the message in order to update and maintain a structure called Neighbor Table (NBT) which monitors its local view (i.e. all the vehicles in the neighborhood of said vehicle). Each entry of the NBT contains:

* the ID of the neighbor;
* the position of the neighbor;

* the reception time of the Hello Message to verify the freshness of information and remove outdates entries.

An example of Neighbor Table of node with ID 0 is represented in Picture 4.4.

ID	Position	Reception time
1	(150, 120, 0)	13:25:31
2	(130, 145, 0)	13:25:32
3	(180, 200, 0)	13:25:31

Figure 4.4: Example of Neighbor Table of node with ID 0

Thanks to the NBT, the forwarder can advertise its neighborhood to other vehicles. We can define the set of neighbors detected by the forwarder as its *local view*. If a vehicle who receives an Alert Message is not present inside the local view of the forwarder, then that vehicle cannot participate in contention.

The NBT could be piggybacked as-is on the Alert Message, but the authors of ROFF identified some problems with this solution, such as the great overhead caused by the size of IDs. Hence, the proposed solution consists in compressing the NBT in a bitmap-like structure called ESD Bitmap.

In order to build the ESD Bitmap, first of all the forwarder measures the distance between itself and each of its neighbors listed in the NBT. Due to GPS granularity, distances are expressed at meter-level and can be represented with non-negative integers.

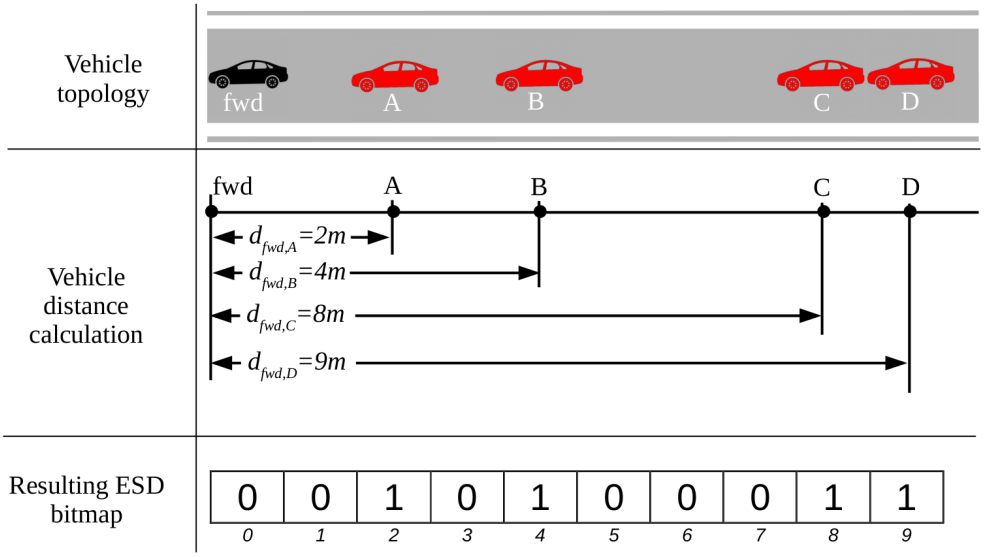

Figure 4.5: ESD Bitmap construction with $k = 1$

After distance measurement, *fwd* builds a bitmap to represent the measured distances by setting the *i-th* bit to either:

* 1 if there is a PFC distant d from *fwd* such that $k * i \leq d \leq k * (i + 1) - 1$;

∗ 0 otherwise.

The parameter k, called *distance range*, identifies how many distances can be identified by the same bit in the ESD Bitmap. This parameter can be controlled to manage the compression level and the accuracy of the Bitmap. Increasing k decreases the number of bits of the Bitmap, but every bit will identify k different distances, hence losing precision. Picture 4.5 shows the construction of an ESD Bitmap using $k = 1$.

An ESD Bitmap using $k = 2$ is show in Figure 4.6. It is possible to see that the bit in position 4 represents distances d such that $8 \leq d \leq 9$.

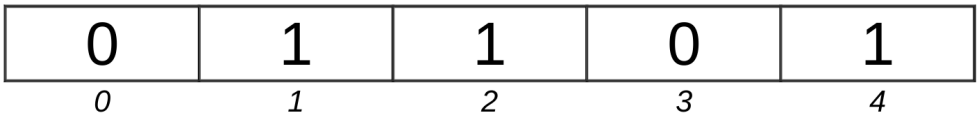

Figure 4.6: Resulting ESD Bitmap with $k = 2$

4.2.2 Forwarding Priority Acquisition

Whenever a PFC f_i receives the Alert Message with included the ESD Bitmap from fwd, it checks whether the distance between itself and the previous forwarder is listed inside the bitmap (i.e. f_i checks whether the bit in position $d(fwd, f_i)$ in the bitmap is equal to 1, assuming $k = 1$). If so, f_i can participate in contention since it is inside the *local view* of fwd. Otherwise, is cannot participate in contention to forward the Alert Message.

After f_i has entered contention, it can calculate its own forwarding priority based on the distances express by the ESD Bitmap. These distances are the distances of every other PFC which can participate in contention. Assuming that L_{dist} is the list of distances ordered in ascending order and $distance = d(fwd, f_i)$, f_i's forwarding priority is set as the rank of $distance$ inside L_{dist}. This way PFCs that are farther away from fwd are assigned a higher forwarding priority, while PFCs nearer fwd are assigned a lower forwarding priority.

ROFF avoids assigning the same priority to two different PFCs by a technique called *ID-based contention*. Using this technique, whenever a PFC f_i receives an Alert Message it checks whether its NBT contains another vehicle f_j whose distance from fwd is the same as f_i's (hence the distance is identified by the same bit in the bitmap). If so, f_i participates in contention only if its ID is higher than the ID of f_j. In other words, if inside f_i's NBT there exists an entry for f_j such that $d(fwd, f_i) = d(fwd, f_j)$ and $ID_{f_j} > ID_{f_i}$, then f_i defers its transmission. Figure 4.7 depicts a scenario where node A does not participate in contention due to ID-based contention.

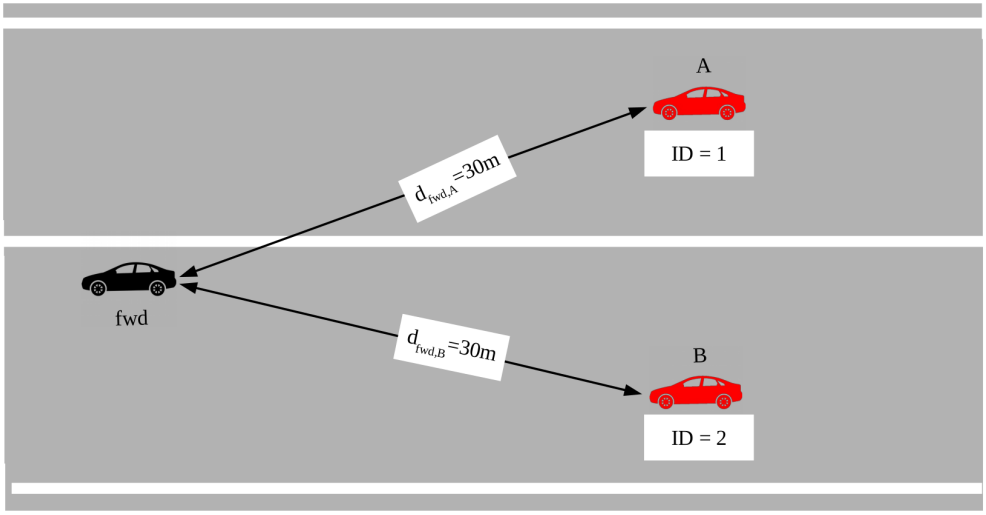

Figure 4.7: ID-based contention: node A defers its transmission

4.2.3 Waiting Time Assignment

Once forwarding priorities are determined, ROFF assigns a waiting time to each vehicle that is inversely proportional to the priority. Letting $PFC(i)$ be the PFC with forwarding priority i and WT_p the waiting time of PFC p, in order to achieve successful suppression of transmission by $PFC(i-1)$, the waiting time of $PFC(i-1)$ should be at least longer by $minDiff_{PFC(i),PFC(i-1)}$ than waiting time of $PFC(i)$ (i.e. $WT_{PFC(i-1)} \geq WT_{PFC(i)} + minDiff_{PFC(i),PFC(i-1)}$). Hence, the waiting time of $PFC(k)$ is the sum of $minDiff$s (calculated using Equation 4.1) between PFCs with forwarding priorities lower than k, as shown in Equation 4.2. PFC with top priority (being equal to 1) broadcasts immediately and does not have to wait any time.

$$WT_{PFC(k)} = \sum_{i=2}^{k} minDiff_{PFC(i),PFC(i-1)} (k \geq 2) \tag{4.2}$$

Figure 4.8 shows the application of 4.2 in the scenario represented in Figure 4.5.

PFC	Priority	Waiting time
D	1	0
C	2	minDiff(C, D)
B	3	minDiff(B, C) + minDiff(C, D)
A	4	minDiff(A, B) + minDiff(B, C) + minDiff(C, D)

Figure 4.8: Waiting time assignment based on vehicle distribution depicted in Figure 4.5

When calculating $minDiff$ as explained in Section 4.1.1, , MPD, Rx/Tx and $CCATime$ are fixed system parameters; the only variables are propagation delays between vehicles, which are dependent on distance. Since coordinates are not included in the ESD Bitmap (only distances are), each PFC needs to identify the coordinates of the previous forwarder's neighbors. In other words, whenever a PFC f_i receives the ESD Bitmap from fwd and calculates its own waiting time, it checks for each distance $dist$ in L_{dist} whether there is a vehicle f_j distant $dist$ from fwd (i.e. $d(fwd, f_j) = dist$. If such vehicle exists, then f_i uses f_j coordinates in the propagation delay (hence $minDiff$) calculation, otherwise it disregards $dist$ in the computation since possibly there is no vehicle distant $dist$ from fwd in f_i's local view.

4.3 Multiple dimensions extension

In the original work **6906275**, ROFF has been implemented in the network simulator ns-2. Since retrieval of the original code has proven to be impossible, part of this thesis concerned the implementation of ROFF in ns-3. Moreover, the original article focuses only on 1D message propagation along a strip-shaped area. This work proposes a 2D and 3D extension of ROFF based on the idea of Fast-Broadcast presented in Section 3.3. The main addition to the original algorithms consists in the suppression of scheduled transmissions only when an Alert Message is heard coming from a vehicle farther from the initial sender than the current vehicle. More in detail, if fwd is the origin of the Alert Message, A is the previous forwarder and B is a PFC with a scheduled transmission. B will suppress its transmission only if it hears an Alert Message coming from D such that $d_{fwd,D} > d_{fwd,B}$. Algorithm 11, 12, 13 and 14 show ROFF's phases in detail.

Algorithm 11 Hello message sending procedure for 2D

```
1: for each beaconInterval do
2:     helloMsg.ID ← retrieveID()
3:     helloMsg.senderPosition ← retrievePosition()
4:     transmit(helloMsg)
5: end for
```

Algorithm 12 Hello message receiving procedure for 2D

```
1: id ← helloMsg.ID
2: sp ← helloMsg.senderPosition
3: time ← currentTime()
4: addNBTEntry(id, sp, time)
```

Algorithm 13 Alert Message generation procedure for 2D

```
1: esdBitmap ← createEsdBitmap(NBT)
2: alertMsg.esdBitmapSize ← esdBitmap.size
3: alertMsg.esdBitmap ← esdBitmap
4: alertMsg.senderPosition ← retrievePosition()
5: alertMsg.originPosition ← retrievePosition()
6: transmit(alertMsg)
```

Algorithm 14 Alert Message forwarding procedure for 2D

```
1: esdBitmap ← alertMsg.esdBitmap
2: if vehicleNotPresentInEsd() ∨ lostIDContention() then
3:     exit()
4: else
5:     listOfDistances ← computeListOfDistances(esdBitmap)
6:     myPosition ← retrievePosition()
7:     senderPosition ← alertMsg.senderPosition
8:     dist ← distance(myPosition, senderPosition)
9:     myRank ← getRank(dist, listOfDistances)
10:    wait ← computeWaitTime(myRank)
11:    if sameBroadcastHeardFromBack() then
12:        exit()
13:    else if sameBroadcastHeardFromFront() then
14:        restartBroadcastProcedure()
15:    else
16:        esdBitmap ← createEsdBitmap(NBT)
17:        alertMsg.esdBitmapSize ← esdBitmap.size
18:        alertMsg.esdBitmap ← esdBitmap
19:        alertMsg.senderPosition ← retrievePosition()
20:        transmit(alertMsg)
21:    end if
22: end if
```

4.3.1 Hello Message Header Structure

The Hello Message header structure is represented in Figure 4.9.

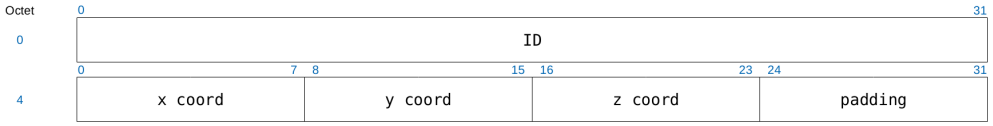

Figure 4.9: ROFF Hello Message header structure

The fields contained in the Hello Message are the following:

* *ID*: unique ID of the Hello Message's sender;
* *x coord*, *y coord* and *z coord*: coordinates of the Hello Message's sender.

4.3.2 Alert Message Header Structure

The Alert Message header structure is represented in Figure 4.10.

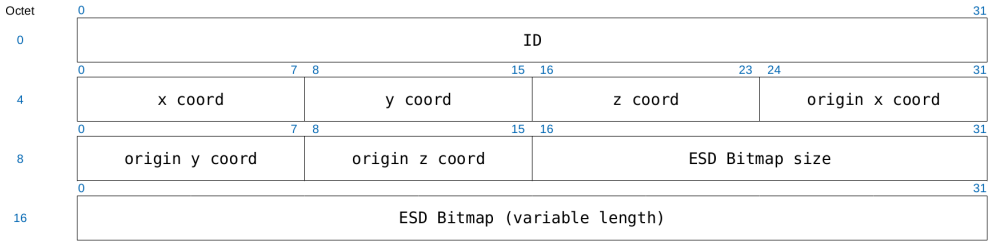

Figure 4.10: ROFF Alert Message header structure

The fields contained in the Alert Message are the following:

* *ID*: unique ID of the Alert Message's sender (or forwarder);
* *x coord*, *y coord* and *z coord*: coordinates of the Alert Message's forwarder;
* *origin x*, *origin y* and *origin z coord*: coordinates where the Alert initially originated from;
* *ESD Bitmap size*: field which indicates how many bytes the ESD Bitmap occupies;
* *ESD Bitmap*: the ESD Bitmap itself, with variable length.

4.4 Smart Junction ROFF (SJ-ROFF)

Since ROFF does not consider junctions in the default algorithm, the proposal explained in Section 3.4 could also bring benefits to it. Hence, we have implemented a Smart Junction variant of ROFF (called SJ-ROFF) following the logic of SJ-Fast-Broadcast. The modifications to the base algorithm are fairly simple: two fields have been added to the Alert Message Header to indicate whether the sender is inside a junction (and the ID of the junction, if it is). Algorithm **??** and 16 show the updated functionalities of Alert Message creation and forwarding. Hello Message algorithms are unchanged.

Algorithm 15 Alert Message generation procedure for 2D

```
1: esdBitmap ← createEsdBitmap(NBT)
2: alertMsg.esdBitmapSize ← esdBitmap.size
3: alertMsg.esdBitmap ← esdBitmap
4: alertMsg.senderPosition ← retrievePosition()
5: alertMsg.originPosition ← retrievePosition()
6: alertMsg.senderInJunction ← isSenderInJunction()
7: alertMsg.junctionId ← getJunctionId()
8: transmit(alertMsg)
```

Algorithm 16 Alert Message forwarding procedure for 2D

```
1: esdBitmap ← alertMsg.esdBitmap
2: if vehicleNotPresentInEsd() ∨ lostIDContention() then
3:     exit()
4: else
5:     listOfDistances ← computeListOfDistances(esdBitmap)
6:     myPosition ← retrievePosition()
7:     senderPosition ← alertMsg.senderPosition
8:     dist ← distance(myPosition, senderPosition)
9:     myRank ← getRank(dist, listOfDistances)
10:    wait ← computeWaitTime(myRank)
11:    if (sameBroadcastHeardFromBack() ∧ vehicleNotInJunction()) ∨ (sameBroad-
   castHeardFromBack() ∧ vehicleInJunction(j) tel∧ alertMsg.senderInJunction ∧
   alertMsg.junctionId == j) then
12:        exit()
13:    else if sameBroadcastHeardFromFront() then
14:        restartBroadcastProcedure()
15:    else
16:        esdBitmap ← createEsdBitmap(NBT)
17:        alertMsg.esdBitmapSize ← esdBitmap.size
18:        alertMsg.esdBitmap ← esdBitmap
19:        alertMsg.senderPosition ← retrievePosition()
20:        transmit(alertMsg)
21:    end if
22: end if
```

4.4.1 Alert Message Header Structure

The Hello Message header structure is the same as Figure 4.9, hence it is not reported here. The Alert Message header structure is represented in Figure 4.11.

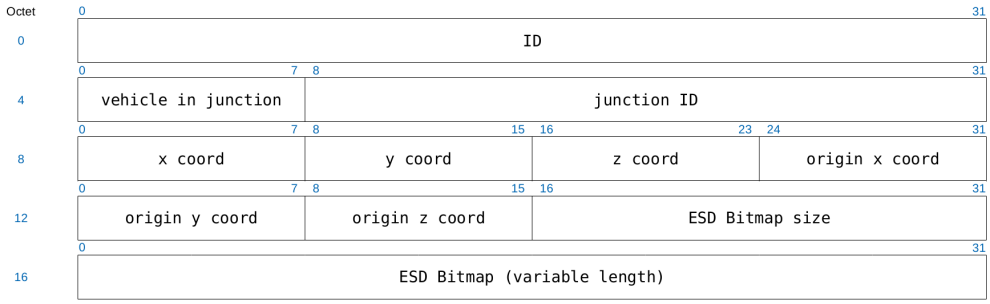

Figure 4.11: SJ-ROFF Alert Message header structure

The additional fields are the following:

* *vehicle in junction*: whether the vehicle which has forwarded this Alert Message is inside a junction;
* *junction ID*: the ID of the junction the sender is inside.

Chapter 5

Tools and applications

5.1 Network Simulator 3

Network Simulator 3 (ns-3) is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.

ns-3 development began in 2006 by a team lead by Tom Henderson, George Riley, Sally Floyd and Sumit Roy. Its first version was released on June 30, 2008.

ns-3 is the successor of ns-2, released in 1989. The fact that the former was built from scratch makes it impossible to have backward compatibility. In fact, ns-2 used oTCL scripting language to describe network topologies and C++ to write the core of the simulation. This choice was due to avoid the very time consuming C++ code recompilation, exploiting the interpreted language oTCL. ns-2 mixed the "fast to run, slow to change" C++ with the "slow to run, fast to change" oTCL language. Since compilation time was not an issue with modern computing capabilities, ns-3 developers chose to utilize exclusively C++ code (and optional Python bindings) to develop simulations.

5.1.1 Modules and module structure

ns-3 is composed of various modules, which are groups of classes, examples and tests each related to a certain feature. The components of a module work in a cohesive way in order to offer APIs to other modules and users. Some examples of built-in modules are:

* WiFi;
* AODV;
* CSMA.

The obstacle shadowing propagation loss model and the Fast Broadcast algorithm have been implemented as modules too.

Modules follow a prototypical structure in order to promote clarity and offer built-in documentation. Figure 5.1 shows the typical module structure.

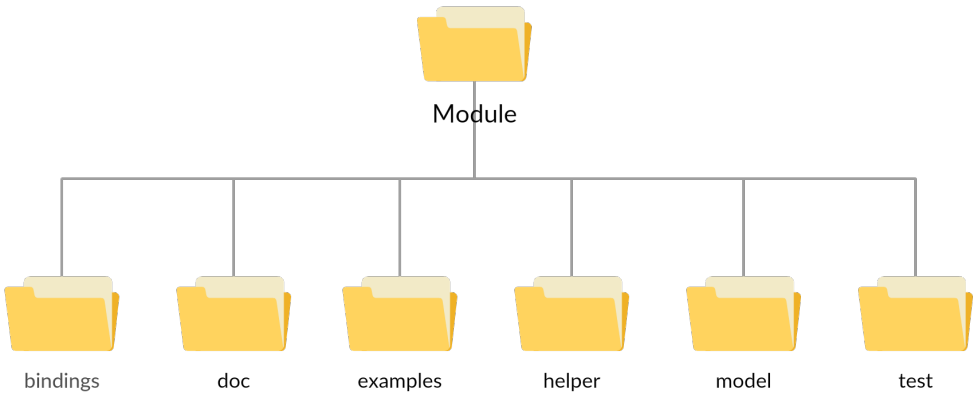

Figure 5.1: ns-3 module structure

The following directories can be found inside a module's root directory:

* **bindings:** Python bindings used to make the module's API compatible with Python;
* **doc:** documentation of the module;
* **examples:** examples and proof of concepts of what can be done using the module;
* **helper:** higher level APIs to make the module easier to use;
* **model:** headers and source files which implement the module's logic;
* **test:** test suite and test cases to test the module.

5.1.2 Key elements

The element at the base of ns-3 is called *node*, instance of `ns3::Node`. A node can be thought of as a shell of a computer. Various other elements can be added to nodes, such as:

* NetDevices (e.g. NICs, which enable nodes to communicate over *channels*);
* protocols;
* applications.

The applications implement the logic of a simulation. For example, the `UdoEchoClientApplication` and `UdpEchoServerApplication` can be used to implement a client/server application which exchange and print the packets' content over the network. The Fast Broadcast protocol has been implemented as an application as well.

The *channels* model various type of transmission media, such as the wired and the wireless ones.

5.1.3 Structure of a simulation

A simulation can be implemented in many ways, but in most cases the following steps are executed:

* manage command line arguments (e.g. number of nodes to consider in the simulation, transmission range, etc.);
* initialize all the necessary fields in classes;
* create nodes;
* set up physical and MAC layers;
* set up link layer, routing protocols and addresses;
* configure and install applications on nodes;
* position nodes and (optionally) give them a mobility model;
* schedule user defined events, such as transmissions of packets;
* start the simulation;
* collect and manage output data.

5.1.4 NetAnim

Netanim is an offline animator tool based on the Qt toolkit. It collects an XML tracefile during the execution of a simulation and can be used to animate the simulation, analyzing packet transmissions and contents.

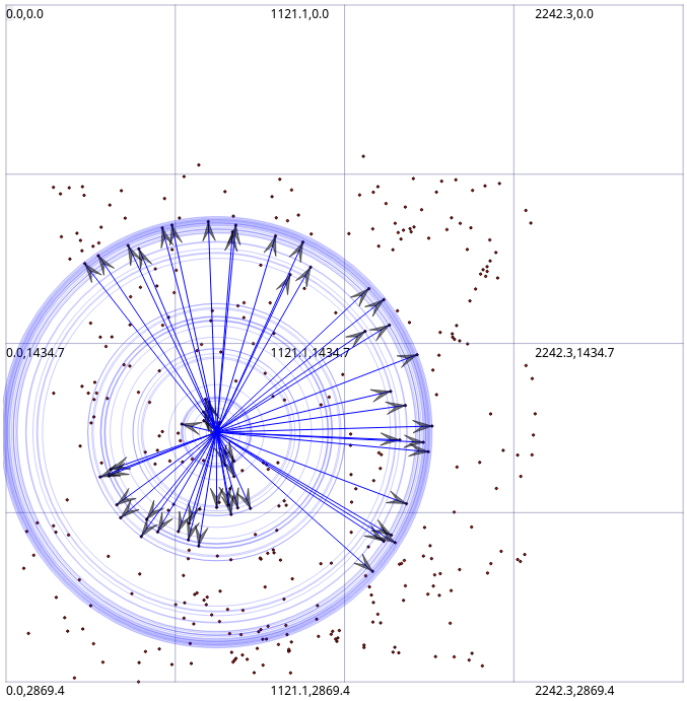

Figure 5.2: Packet transmission in NetAnim

5.2 Simulation of Urban MObility

Simulation of Urban MObility is an open-source road traffic simulation package. It is written in C++ and licensed under GPLv3.

It offers different tools to analyze and manage real maps from the urban mobility point of view, including pedestrian movement and various types of vehicles.

The original work, starting from real maps obtained from OpenStreetMap (OSM), **ROM2017** utilized SUMO to produce two files:

1. a `.poly` file using the SUMO tool *Polyconvert*. This file contains information about all the obstacles, such as buildings, useful for the Obstacle Shadowing module;

2. a `.ns2mobility` file using the SUMO tool *TraceExporter*. This file contains information about the vehicles and their positioning.

The process of generating these two files necessary for ns-3 simulations requires some intermediate steps. The full process is represented in Figure 5.3.

The original work considered a only a distance of 25 meters between vehicles; this work considers various distances, ranging from 15 to 45 meters. Figure 5.4 shows the same scenario (Padua) with different distances between vehicles.

One of the intermediate results, the `.net` file which describes the road infrastructure using a graph resulting from *Netconvert*, has been utilized for junction modeling, as explained in Section 2.2.

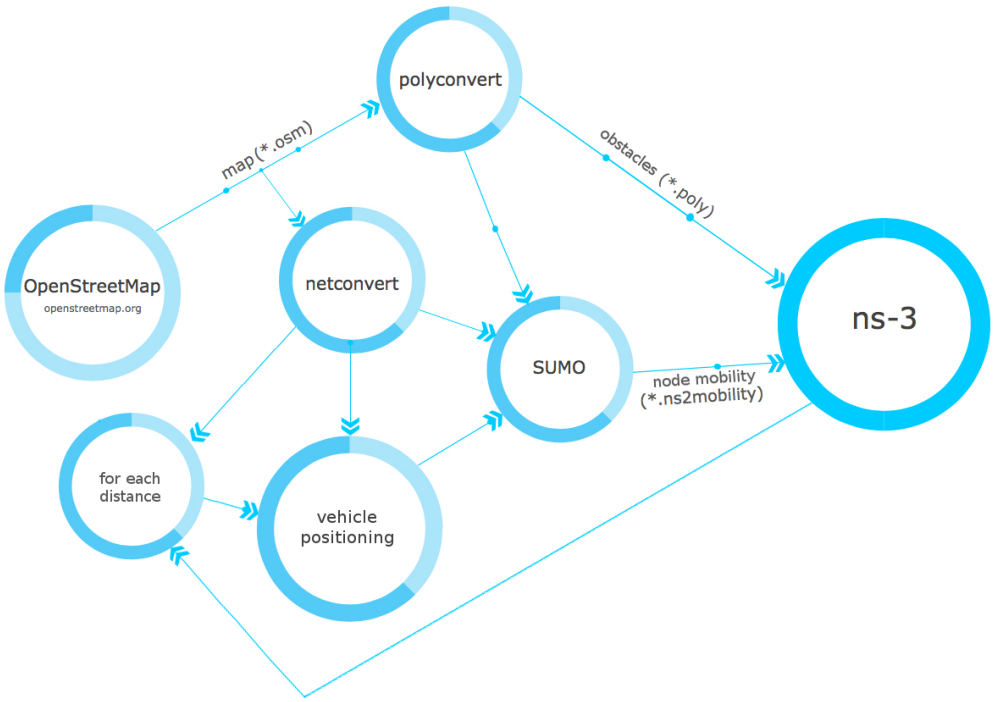

Figure 5.3: Steps to generate necessary files using SUMO (**ROM2017**)

Figure 5.4: Padua scenario with vehicle distance equals to 15 meters (top) and 45 meters (bottom)

Chapter 6

Simulations

After the protocols under consideration have been presented in Chapters 3 and 4, this Chapter will contain the results of the simulation carried out through various scenario of increasing complexity.

6.1 Metrics

Before proceeding with the presentation of simulation results, metrics used to evaluate the protocol's performances will be presented in this section.

6.1.1 Total Delivery Ratio (TDR)

This metric is used to detect how many vehicles have received the Alert Message in total. Ideally, broadcasting protocols should be able to reach all the reachable nodes in the scenario in order to warn them of the danger. The metric is calculated as follows:

$$TDR = \frac{no.\ of\ vehicles\ successfully\ receiving\ the\ Alert\ Message}{no.\ of\ vehicles\ in\ the\ scenario} \tag{6.1}$$

All scenarios are created in a way such that every node can be reached by multi-hop propagation regardless of the source of the Alert Message. In other words, from graph theory point of view, the graph resulting from vehicle distribution where two nodes are connected only if they are within transmission range of each other is a connected graph. This way, the term at the denominator of Equation 6.1 is equivalent to the number of vehicles reachable by pure flooding.

6.1.2 Total Delivery Ratio On Circumference (TDROC)

This metric is used to detect how many vehicles on the circumference have received the Alert Message. The circumference is built starting from the source of the Alert Message. The way it is built depends on scenario topology (e.g. 1D, 2D, etc) and will be explained more thoroughly in the following Sections. The main idea behind the circumference consists in considering only vehicles far from the source of the AM. The metric is calculated as follows:

$$TDROC = \frac{no.\ of\ vehicles\ on\ circ.\ successfully\ receiving\ the\ Alert\ Message}{no.\ of\ vehicles\ on\ circ.} \quad (6.2)$$

The same consideration about vehicles and reachability by pure flooding presented in Section 6.1.1 is also valid for Equation 6.2.

6.1.3 Number Of Hops (NOH)

This metric is used to measure the mean number of hops required in order to propagate the Alert Message from the source to all vehicles reached on the circumference. This value is obviously dependent on the paths taken by forwarder Alert Messages from the source to all destinations. We have that *ONOH* is always greater than or equal to the *Optimal Number of Hops BNOH* (i.e. $ONOH \leq NOH$). *BNOH* is calculated using the following formula:

$$BNOH = \frac{Circumference\ Radius}{Transmission\ Range}$$

The Number Of Hops metric can be calculated as:

$$NOH = \frac{\sum_{p \in RC} no.\ of\ hops\ from\ source\ to\ p}{no.\ of\ vehicles\ on\ circ} \quad (6.3)$$

where *RC* is the set of vehicles which have successfully received the message on the circumference.

The *NOH* metric (whose value should be as close as possible to *BNOH*) is an important indicator of the effectiveness of the multi-hop protocol in choosing the farthest forwarder during contention.

Figure 6.1 can be used as an example to calculate *NOH*. Using Equation 6.3, the mean number of hops is equal to 3.

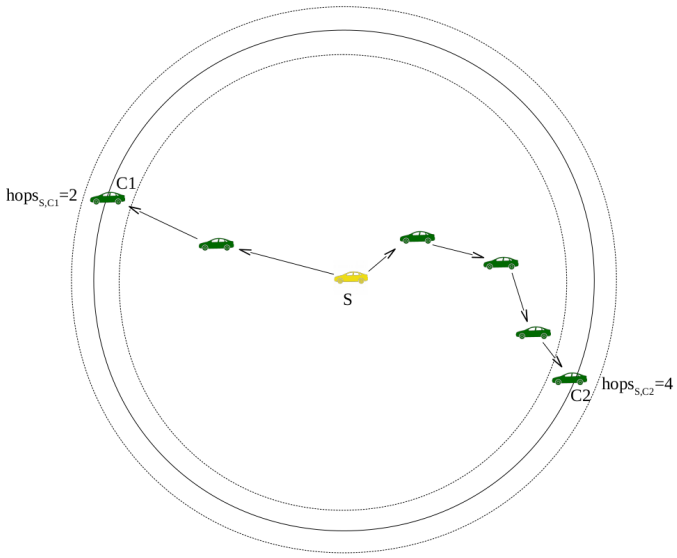

Figure 6.1: Example of *NOH* calculation with Alert Message starting from node S

6.1.4 Number Of Slots (NOS)

This metric is used to measure the number of slots required in order to propagate the Alert Message from the source to all vehicles reached on the circumference. High values of this metric means that the multi-hop protocol introduces a lot of waiting time before each forwarding, hence increasing end-to-end delay and hurting the timeliness of the emergency message propagation.

$$NOH = \frac{\sum_{p \in RC} no.\ of\ slots\ waited\ along\ path\ from\ source\ to\ p}{no.\ of\ vehicles\ on\ circ} \tag{6.4}$$

where RC is the set of vehicles which have successfully received the message on the circumference The number of slots along the path from node a to b at the numerator of Equation 6.4 is the sum of the slots waited by each forwarder before relaying the Alert Message.

Based on the scenario represented in Figure 6.2 and Equation 6.4, the mean number of slots from source S to nodes on the circumference is 100.

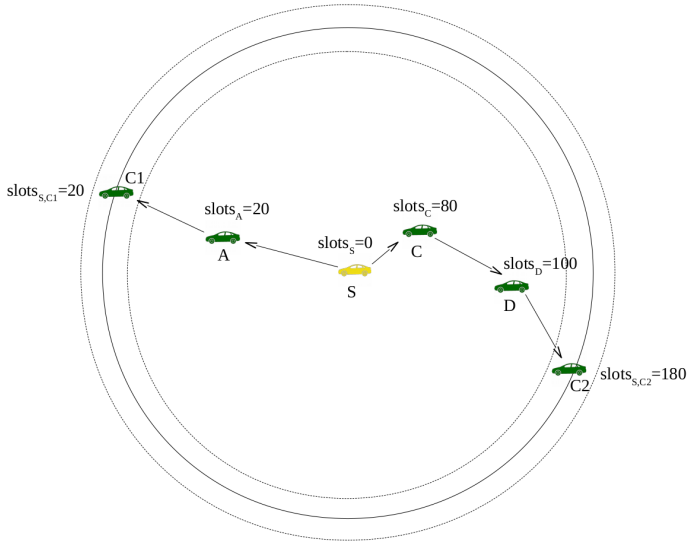

Figure 6.2: Example of *NOS* calculation with Alert Message starting from node S

6.1.5 Forwarding Node Number (FNN)

This metric is used to measure the number of vehicles which forward the Alert Message. The value of this metric is an indicator of the effectiveness of the multi-hop protocol in successfully suppressing scheduled transmissions from PFCs after the FFC has relayed the message. The metric is calculated as follows:

$$FNN = no.\ of\ vehicles\ forwarding\ the\ Alert\ Message \tag{6.5}$$

6.2 Platoon scenario

The first scenario taken into consideration is a simple platoon scenario, where vehicles are placed in a strip-like area 15 kilometers long. Vehicles are 25 meters distant from each other. Transmission ranges of 100, 300 and 500 has been employed during simulations. Parameters for this scenario are included in Table ??.

Parameter	Value	
Scenario configuration		
Road length	15000	m
Distance between vehicles	25	m
Circumference radius	14000	m
Number of vehicles	600	
Source of alert message position	Left of platoon	
Mobility model	ns3::ConstantPosition	
Network configuration		
Packet size	100	byte
Transmission standard	802.11b	
Frequency	2.4	GHz
Channel bandwidth	22	MHz
Transmission speed	11	Mbps
Transmission powers	-7.0, 4.6, 13.4	dBm
Transmission range	100, 300, 500	m
Modulation	DSSS	
Propagation loss model	ns3::TwoRayGround	
Shadowing model	No	
Propagation delay model	ns3::ConstantSpeed	
Junction modeling	No	
Protocols configuration		
FB contention window	[32, 1024]	slot
ROFF distance range (k parameter)	1	
Number of simulations per configuration	1000	

Table 6.2: Platoon scenario configuration

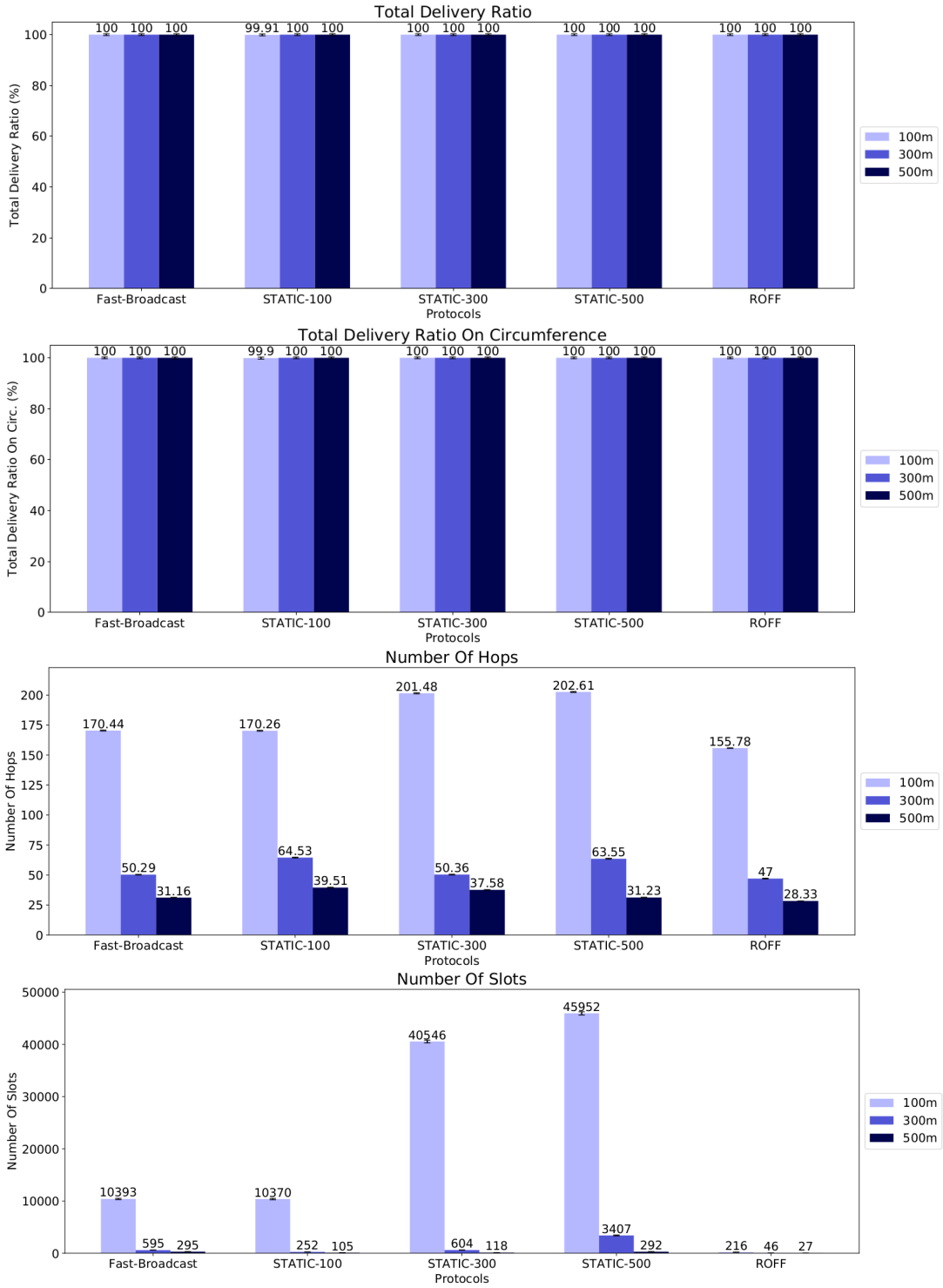

Figure 6.3: *TDR*, *TDROC*, *NOH* and *NOS* metrics for Platoon scenario

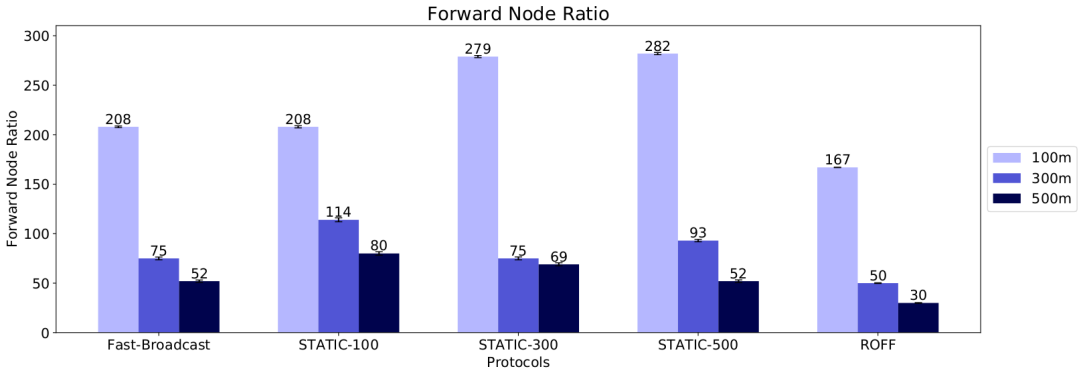

Figure 6.4: *FNN* metric for Platoon scenario

Since this scenario is one-dimensional, the circumference simply consists in vehicles distant 14.000 ± 12 meters from the source of Alert Message. Both metrics about Delivery Ratio (global and on circumference) are close to 100%, so the algorithms successfully propagate the AM until the end of the platoon.

Considering the Number Of Hops, ROFF's results are 11,76%, 6,54% and 9,08% lower than Fast-Broadcast's results respectively for 100, 300 and 500 meters transmission range. ROFF's results are close to the optimal number of hops, respectively 140, 46,6 and 28 for the same transmission ranges as above. This means that the forwarder selection algorithms based on ESD Bitmap works better in choosing the farthest vehicle from the previous forwarder compared to the contention window approach for 1D scenarios. Considering STATIC variants of Fast-Broadcast, it is possible to observe that the STATIC-tx variant produces comparable results with Fast-Broadcast with *tx* transmission range (e.g. STATIC-100 value is comparable to Fast-Broadcast with 100 meters transmission range), as expected. The STATIC protocol performs worse (hence the Number of Hops increases) whenever the transmission range is underestimated (e.g. STATIC-100 with 300 meters transmission range, compared to Fast-Broadcast with the same transmission range) or overestimated (e.g. STATIC-500 with 100 meters transmission range, compared to Fast-Broadcast with the same transmission range). This behaviour is expected, as reported in **BAR2017**.

Regarding the Number of Slots, ROFF's performs much better than Fast-Broadcast. The metric's value is decreased respectively by 97,92%, 92,27% and 90,85% using ROFF. The waiting time calculation, based on unique forwarding priority instead of distance, guarantees a much lower wait compared to Fast-Broadcast contention window approach. As before, STATIC approaches produce results comparable with Fast-Broadcast when the transmission range estimation is correct. Instead, the Number of Slots greatly increases when the transmission range is underestimated and decreases when the transmission range is overestimated. In this last case, the decrease in NOS comes with an increase in Number Of Hops reported in the previous paragraph, so an overestimation of transmission range is not desirable.

Lastly, it is possible to see that ROFF's achieves a better suppression of redundant transmissions, guaranteeing a decrease of 19,71%, 33,33% and 42,31$ respectively for each one of the transmission ranges considered.

6.3 Grid scenario

After tests on the 1D Platoon scenario have been carried out, the next step consisted in testing the algorithms' performances in a 2D Grid scenario, employing also the shadowing model introduced in 2.1. Parameters for this scenario are included in Table 6.4.

Parameter	Value	
Scenario configuration		
Road length	4800	m
Distance between roads	300	m
Road width	10	m
Number of roads (vertical)	17	
Number of roads (horizontal)	17	
Distance between vehicles	25	m
Circumference radius	2000	m
Number of vehicles	6528	
Source of alert message position	Center	
Mobility model	ns3::ConstantPosition	
Edge of buildings	290	m
Number of buildings	255	m
Network configuration		
Shadowing model	Obstacle Shadowing	
Junction modeling	No	
Number of simulations per configuration	1000	

Table 6.4: Grid scenario configuration

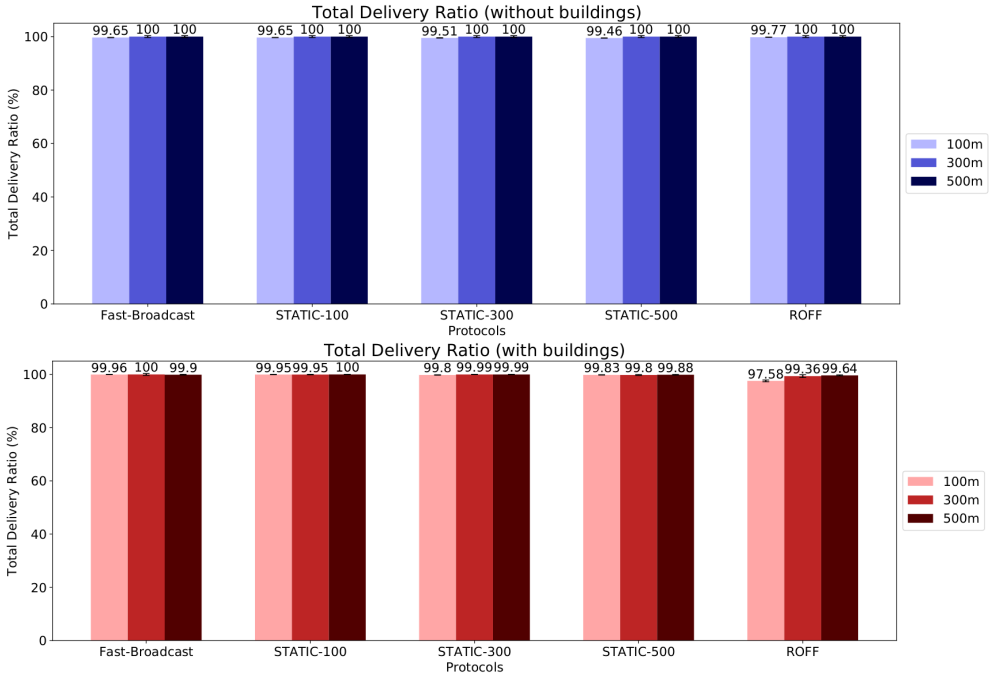

Figure 6.5: *TDR* without buildings (top) and with buildings (bottom) for Grid scenario

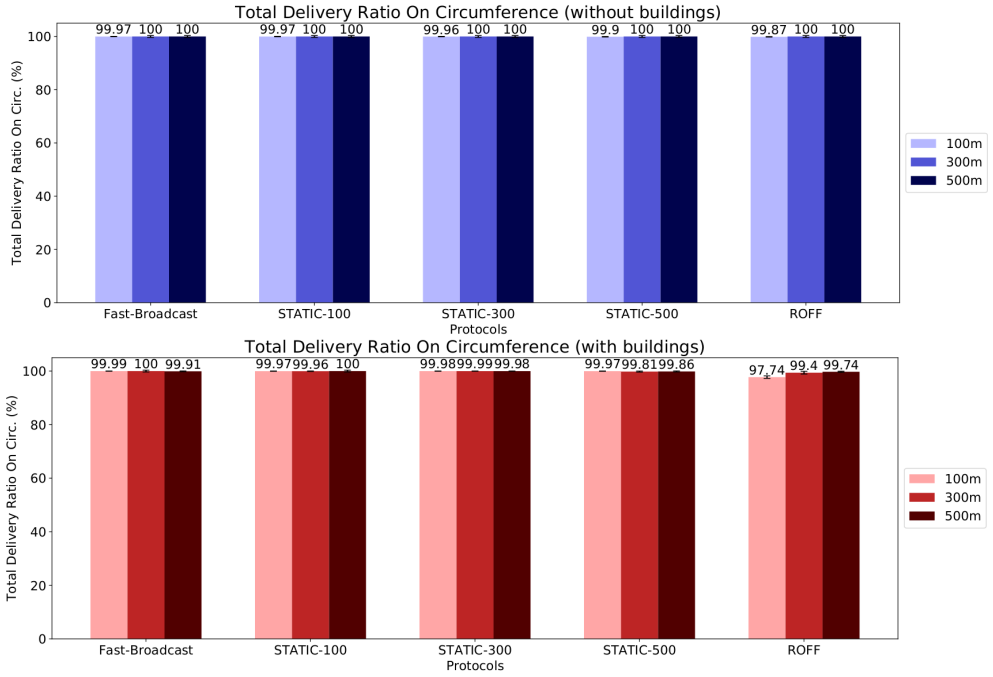

Figure 6.6: *TDROC* without buildings (top) and with buildings (bottom) for Grid scenario

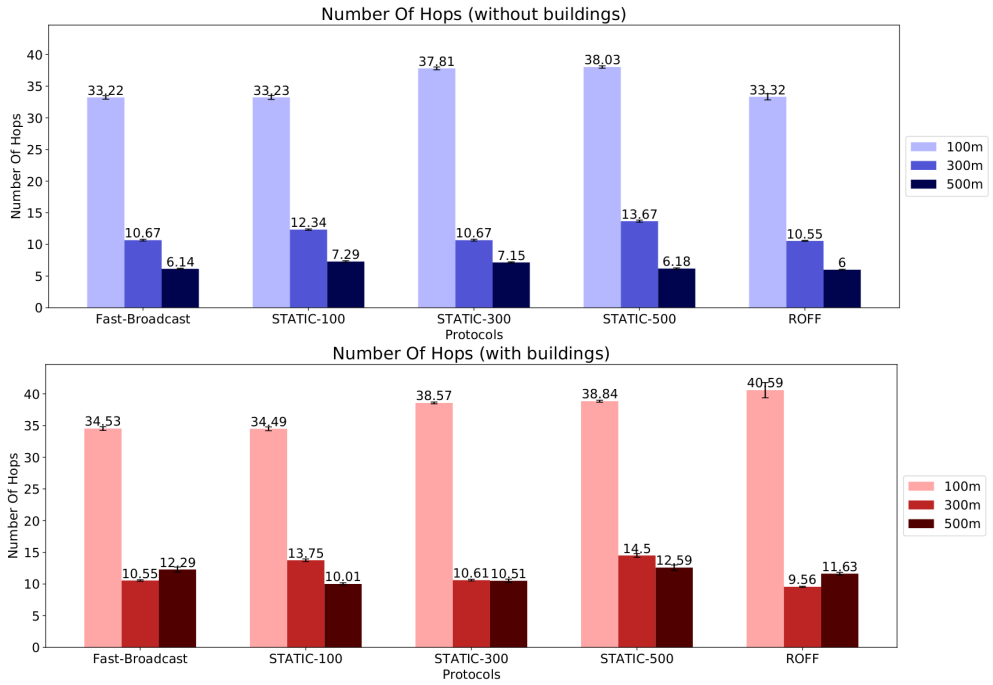

Figure 6.7: *NOH* without buildings (top) and with buildings (bottom) for Grid scenario

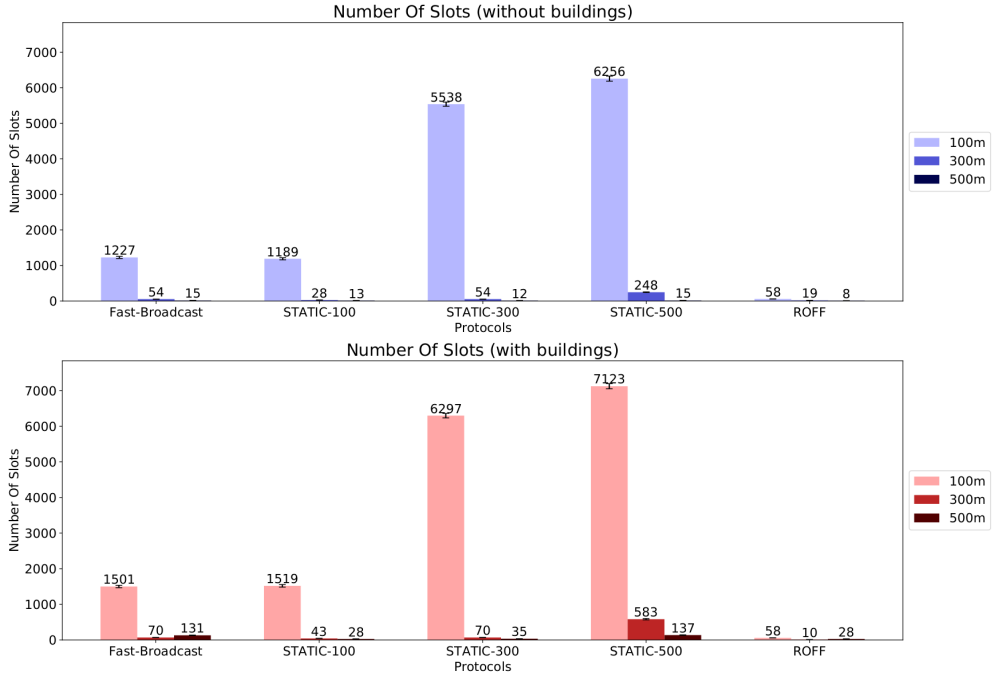

Figure 6.8: *NOS* without buildings (top) and with buildings (bottom) for Grid scenario

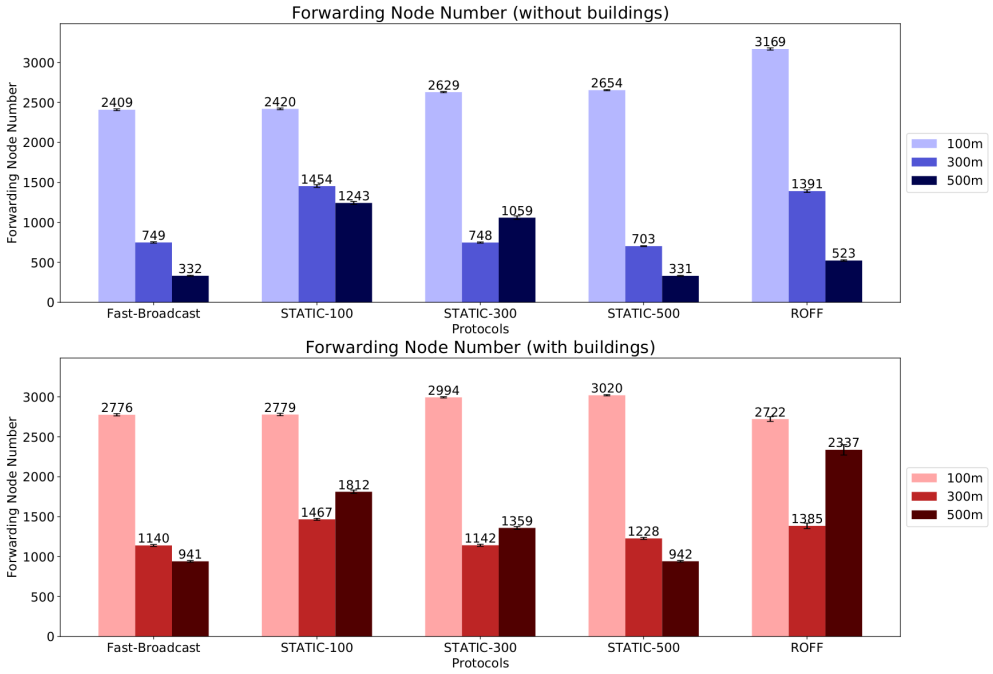

Figure 6.9: *FNN* without buildings (top) and with buildings (bottom) for Grid scenario

All the algorithms perform well as far as delivery ratios are concerned (Figure 6.5 6.6), both for scenarios with and without buildings. This means that the shadowing caused by the Obstacle Model does not create no-reach zones, and the signal manages to find its way to almost all of the vehicles. The regular pattern by which vehicles are placed may help with that: further testing where this hypothesis is removed will be presented in the next sections. Even though delivery ratios are unchanged, the effect of the Obstacle model can be observed qualitatively comparing Figure 6.10 and 6.11 for Fast-Broadcast, and Figure 6.12 and 6.13 for ROFF. The effects of obstacles on propagation cause the signal to travel only through roads segments where line of sight is possible. Instead, without obstacles the signal can be propagated freely in all directions.

Moreover, Fast-Broadcast and ROFF can be compared with each other by looking at Figure 6.10 and 6.12 for the scenario without buildings, and Figure 6.11 and 6.13 for the scenario with buildings. In both cases, ROFF designates the furthest PFC as the next forwarder much more reliably than Fast-Broadcast thanks to the Neighbor Table mechanism. Instead, Fast-Broadcast relies on a random choice of waiting time, so the next designated forwarder is sometimes not one of the farthest vehicles from the previous forwarder. Based on this observation, we can expect a much lower NOH value for ROFF. Quantitative analysis of this and other metrics will be reported in the next paragraphs of this section.

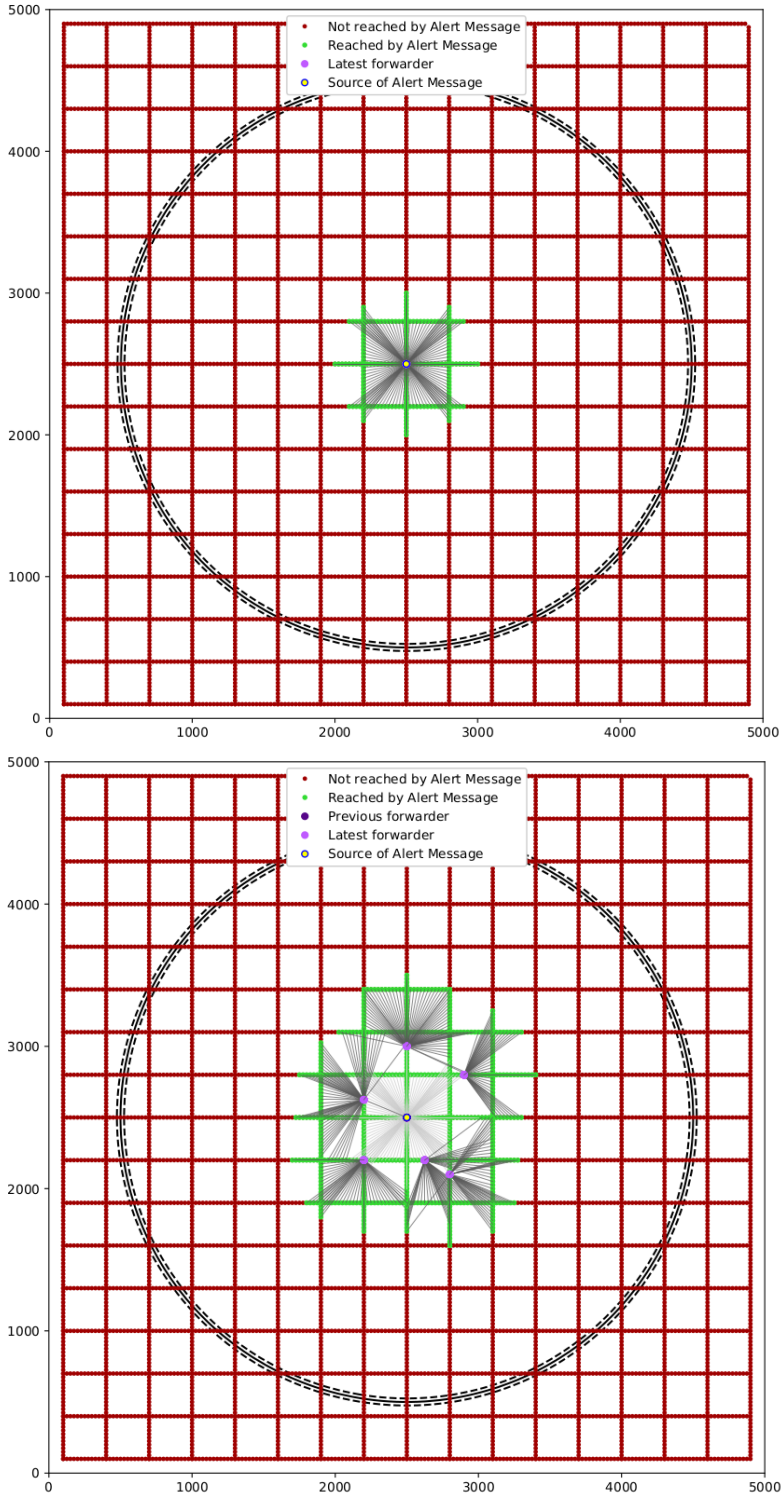

Figure 6.10: Fast-Broadcast after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500 meters transmission range and without the Obstacle model

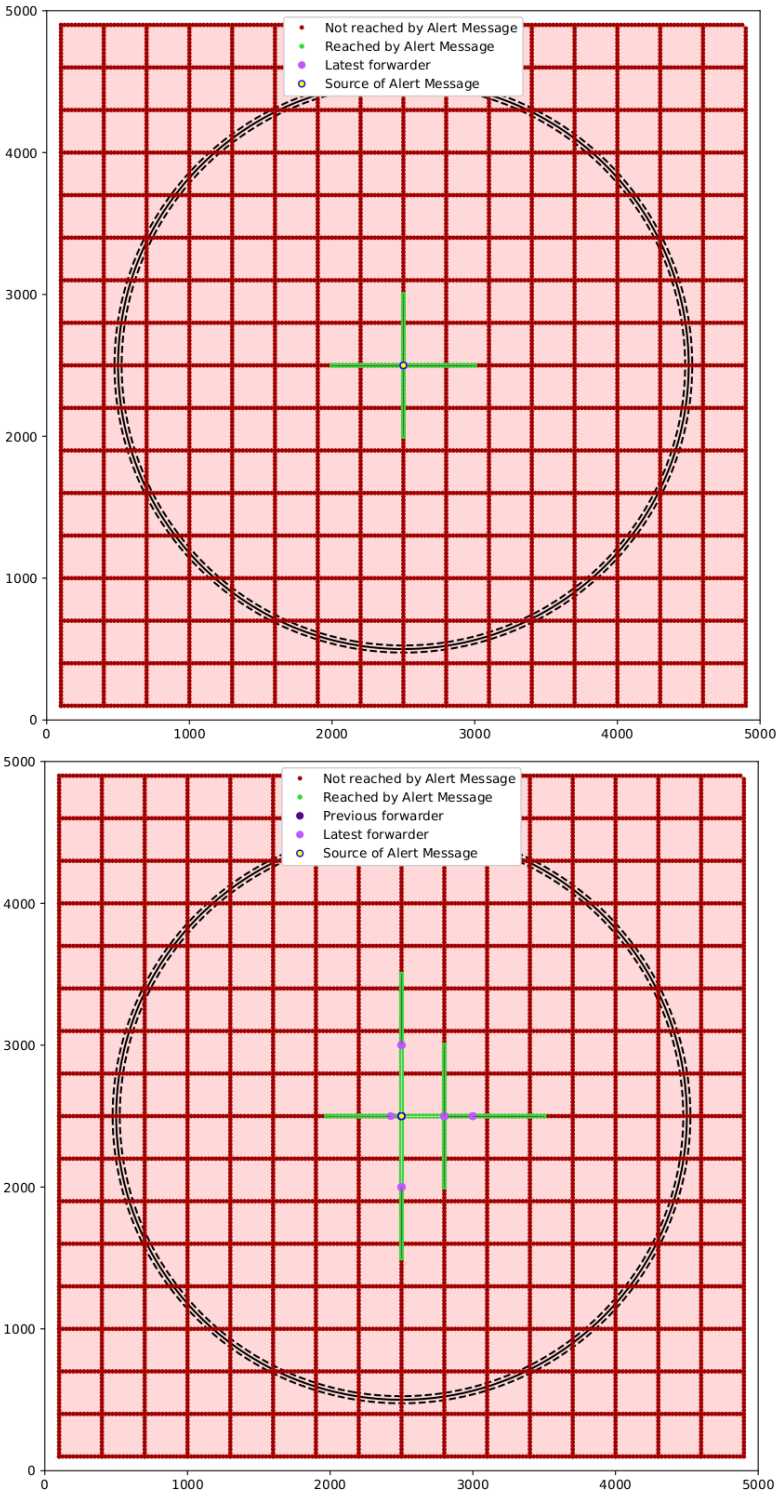

Figure 6.11: Fast-Broadcast after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500 meters transmission range and with the Obstacle model

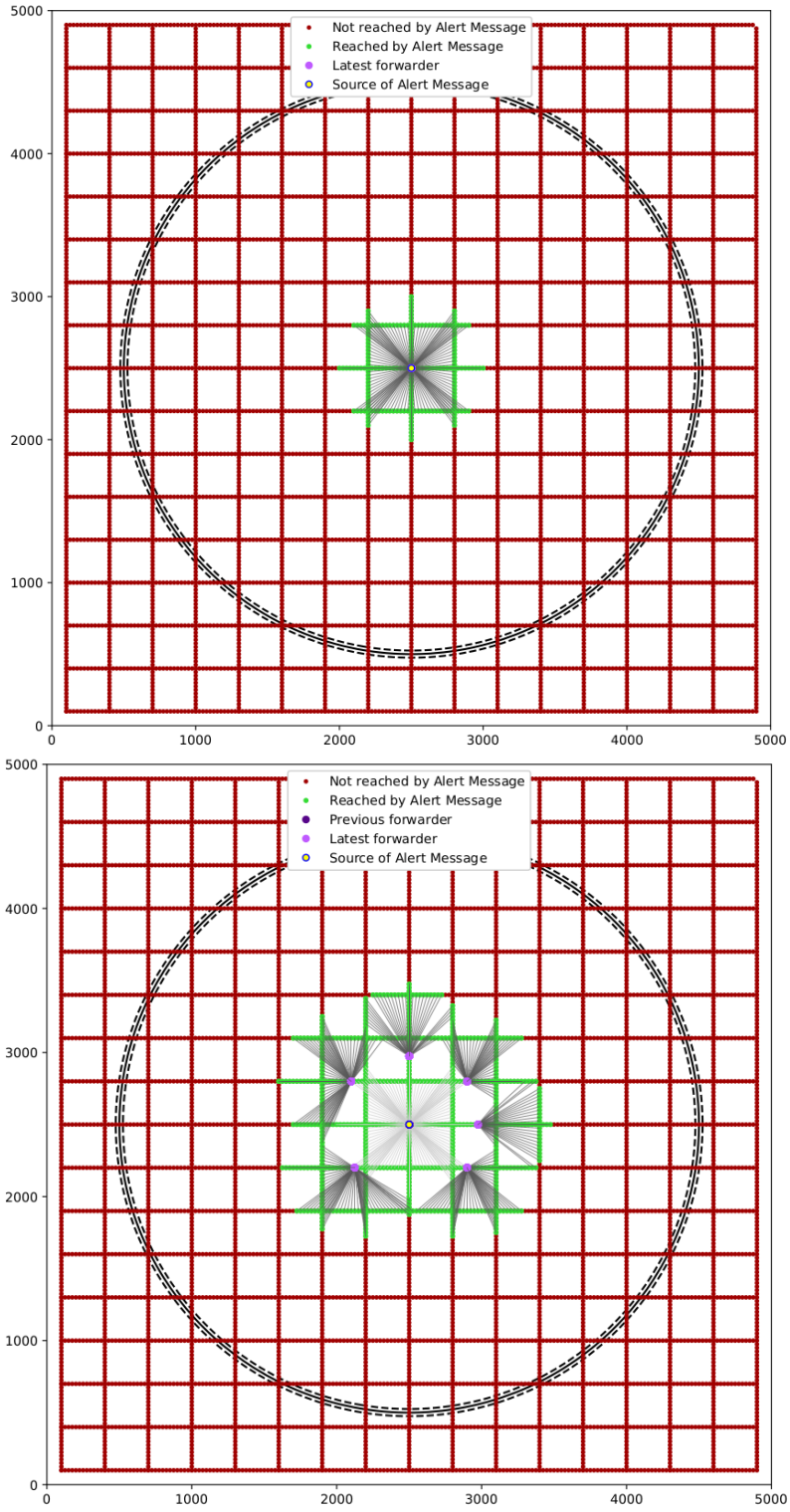

Figure 6.12: ROFF after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500 meters transmission range and without the Obstacle model

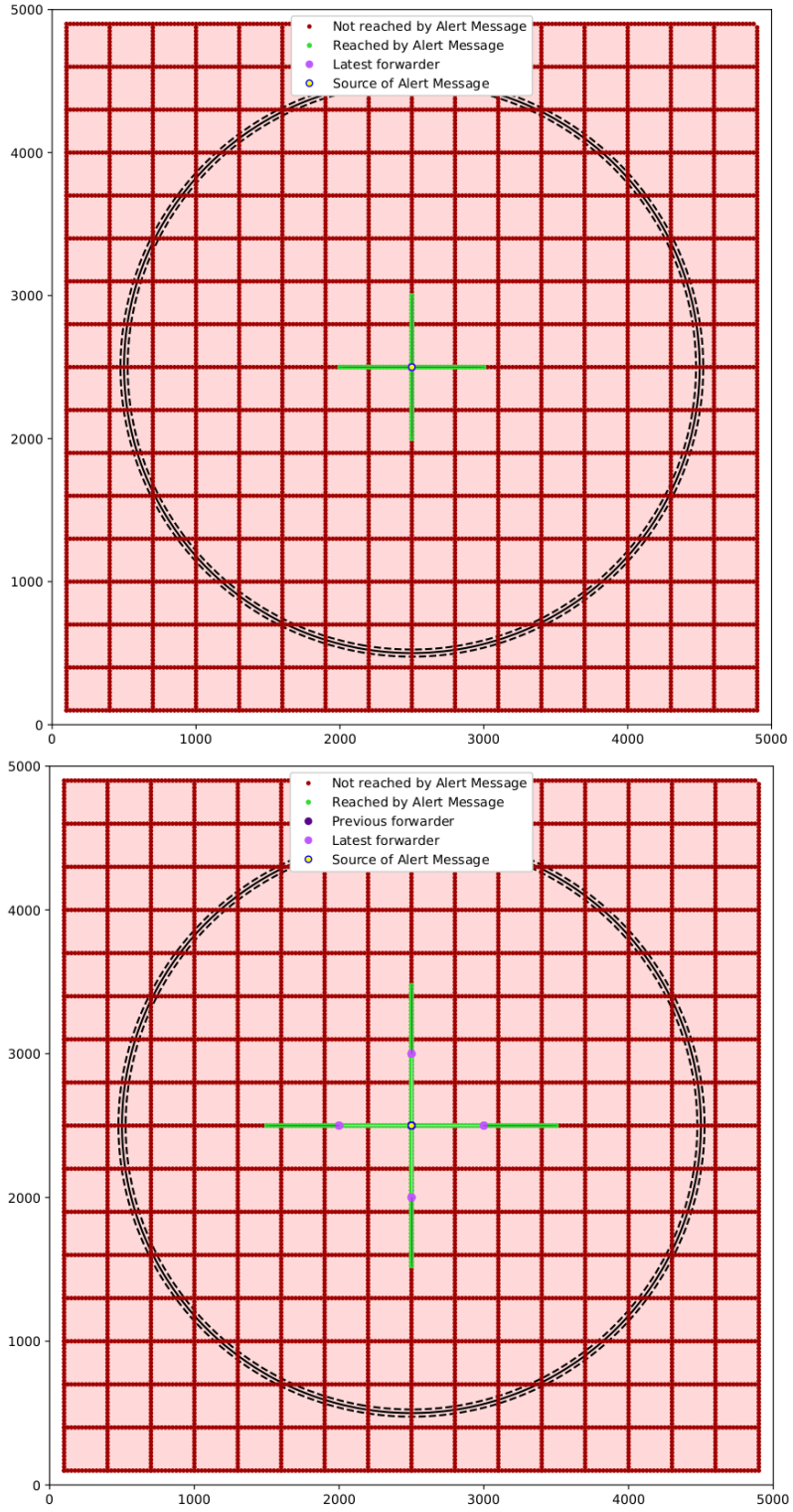

Figure 6.13: ROFF after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500 meters transmission range and with the Obstacle model

If we focus on the Number Of Hops (Figure 6.7), we notice that the introduction of buildings causes the number of hops to increases for transmission ranges of 100 and 500 meters, while decreases for 300 meters tranmission range. This is probably due to the fact that the distance between roads is also 300 meters and the starting vehicle is inside a junction. This means that the effect of buildings causes the signal to travel along roads, hopping from junction to junction, and reaching the circumference much faster. Another unusual observervation consists in ROFF with 100 meters transmission range, where the value increases by a lot. This case needs further examination in order to be explained. Apart from this case, ROFF achieves better results than Fast-Broadcast, with results 9% and 3.47% better for 300 and 500 meters transmission range respectively.

The analysis of the Number Of Slots (Figure 6.8) shows similar results: the shadowing model causes the metric's value to rise in all configurations except for ROFF with 300 meters transmission range. ROFF continues to outperform Fast-Broadcast with results similar to those observed in the Platoon scenario.

Considering the Forwarding Node Number (Figure 6.9), the situation is much different than the previous scenario. It is possible to see in both graphs that ROFF's values are higher than Fast-Broadcast's across all configurations. Despite the almost-perfect suppression of redundant transmission works in a 1D scenario, as observer in the previous section, the mechanism does not seem to have much of an effect in a 2D scenario, regardless of the presence of buildings. A possible explanation of this phenomenon might be the one reported in Figure 6.14, which indicates a possible Alert Message propagation process. Suppose that S is the previous forwarder: the other nodes on the circumference (distant txRange from S) are the elected FFCs which win the contention and relay the message. Due to how ROFF works, those nodes send the message at the same time, hence if we focus on A and B we have a collision area (the red area in Figure 6.14). The nodes inside that collision area receive the forwarding from A and B at the same time and a collision occurs. As a consequence, they are not aware that the message has already been forwarded and some of them will fire their transmission. This process is repeated for every forward (and also for other FFCs other than A and B in the example) and so this could cause the increase in the FNN value. In other words, ROFF is guaranteed to cause the collision area shown in the example due to the exact waiting time calculation for nodes at the same distance from S. Fast-Broadcast is less affected by this problem since the waiting time calculation is not deterministic (the waiting time is chosen randomly from the interval [1...CW] where CW is calculated using Formula 3.1).

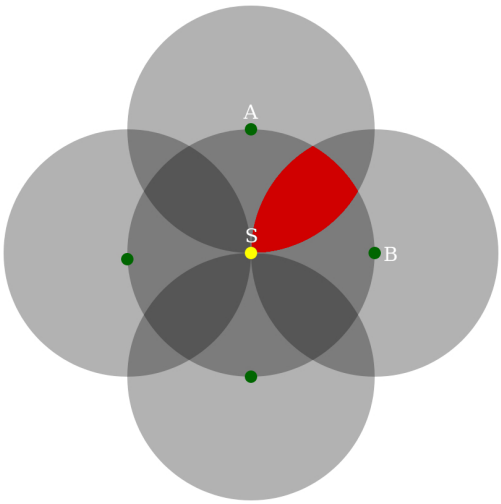

Figure 6.14: Collision area in 2D ROFF

6.4 Los Angeles urban scenario

The next step consisted in expanding the 2D scenario with realistic urban data taken from OpenStreetMap and processed by SUMO as explained in Section 5.2. In addition to the shadowing model, in this scenario (and also in the urban scenario regarding Padua, which will be presented in the next section), a *smart junction* variant of the algorithms has been introduced in order to try to increase the delivery ratios.

Figure 6.15 depicts the scenario with vehicles (green dots), buildings (purple polygons) and junctions (yellow polygons).

All configurations are reported in Figure 6.16. In order to facilitate the understanding of the various configurations, the following graphs will utilize shades of color similar to those in Figure 6.16 (e.g. graphs with shades of blue will represent a scenario without buildings and without junctions, while shades of orange will represent a scenario with buildings and with junctions). Parameters for this scenario are included in Table 6.6.

Figure 6.15: Los Angeles urban scenario depiction

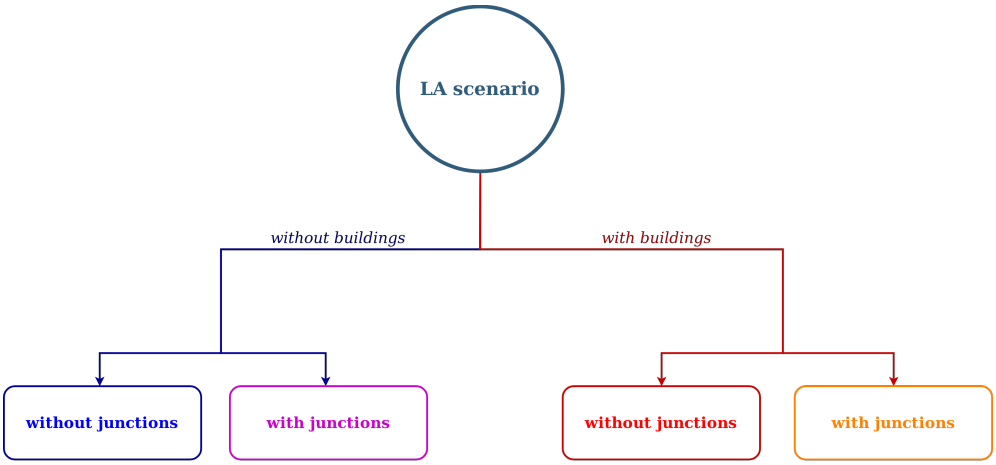

Figure 6.16: Los Angeles urban scenario test overview

Parameter	Value	
Scenario configuration		
Latitude N	33.9654	°
Latitude S	33.9478	°
Longitude W	-118.3260	°
Longitude E	-118.3055	°
Road length	1200	m
Distance between vehicles	25	m
Circumference radius	1000	m
Number of vehicles	1465	
Source of alert message position	Center	
Number of buildings	8241	
Mobility model	ns3::ConstantPosition	
Number of junctions	1288	
Network configuration		
Shadowing model	Obstacle Shadowing	
Junction modeling	Yes	
Number of simulations per configuration	5000	

Table 6.6: Los Angeles urban scenario configuration

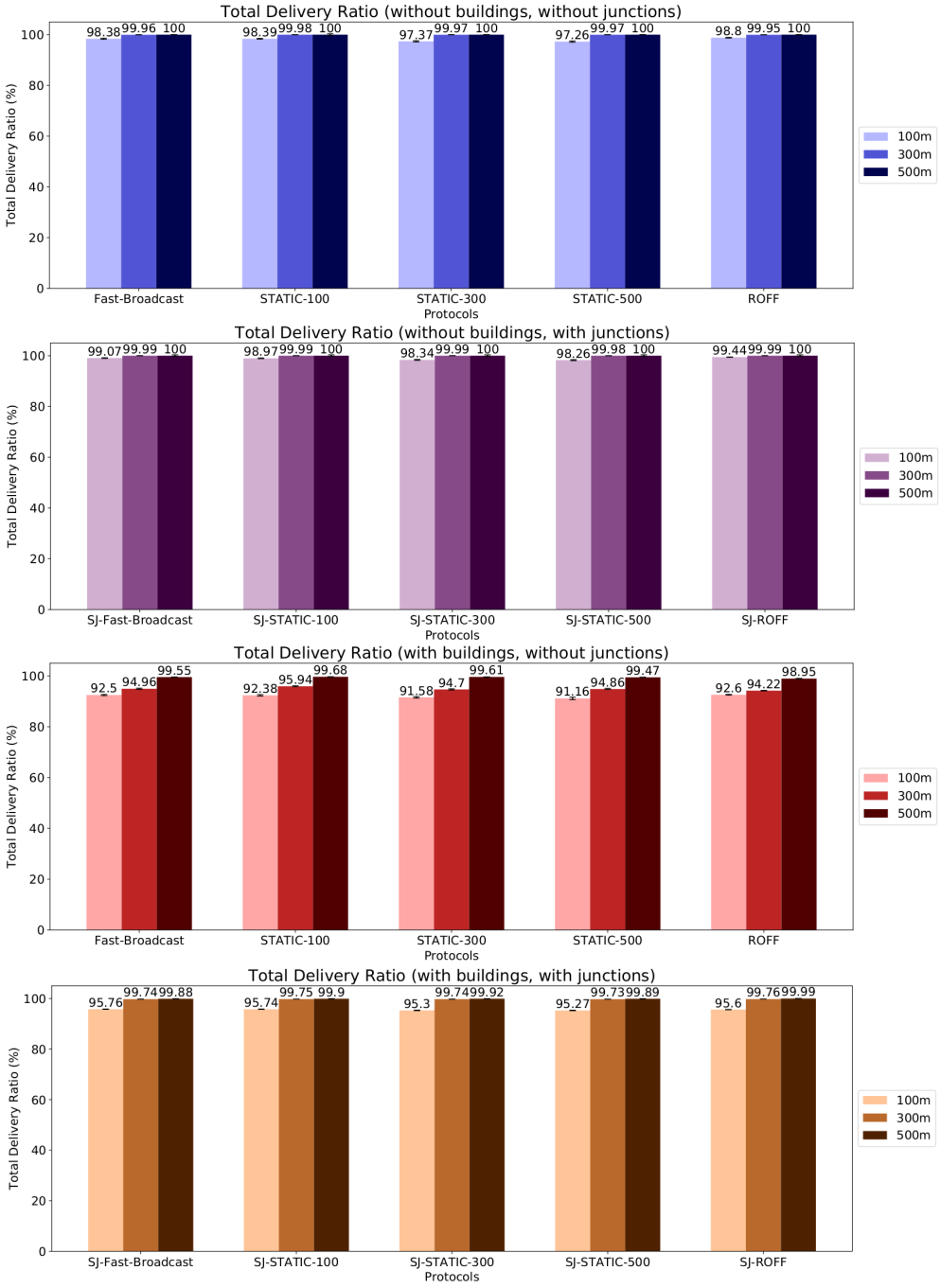

Figure 6.17: *TDR* for Los Angeles urban scenario

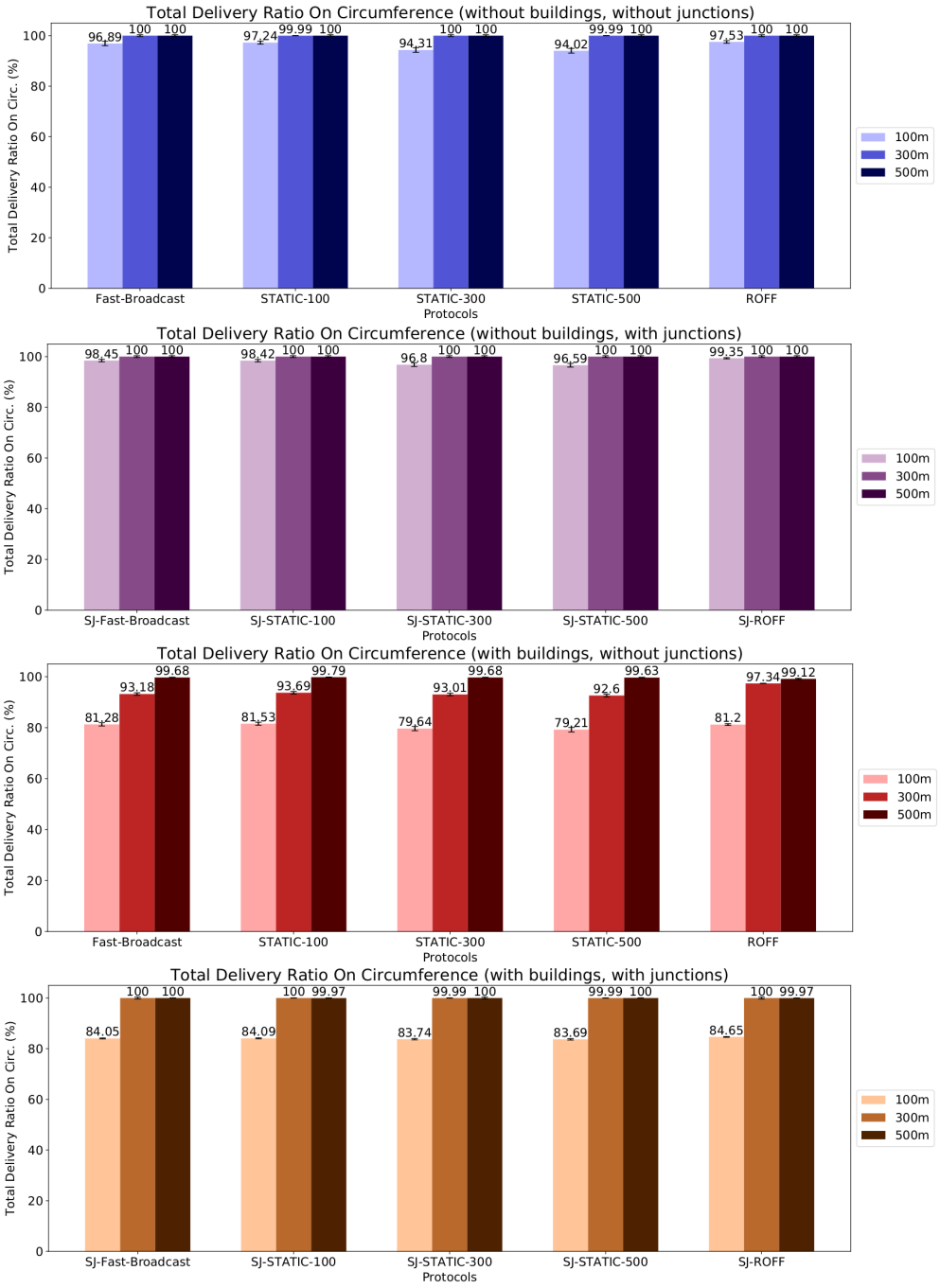

Figure 6.18: *TDROC* for Los Angeles urban scenario

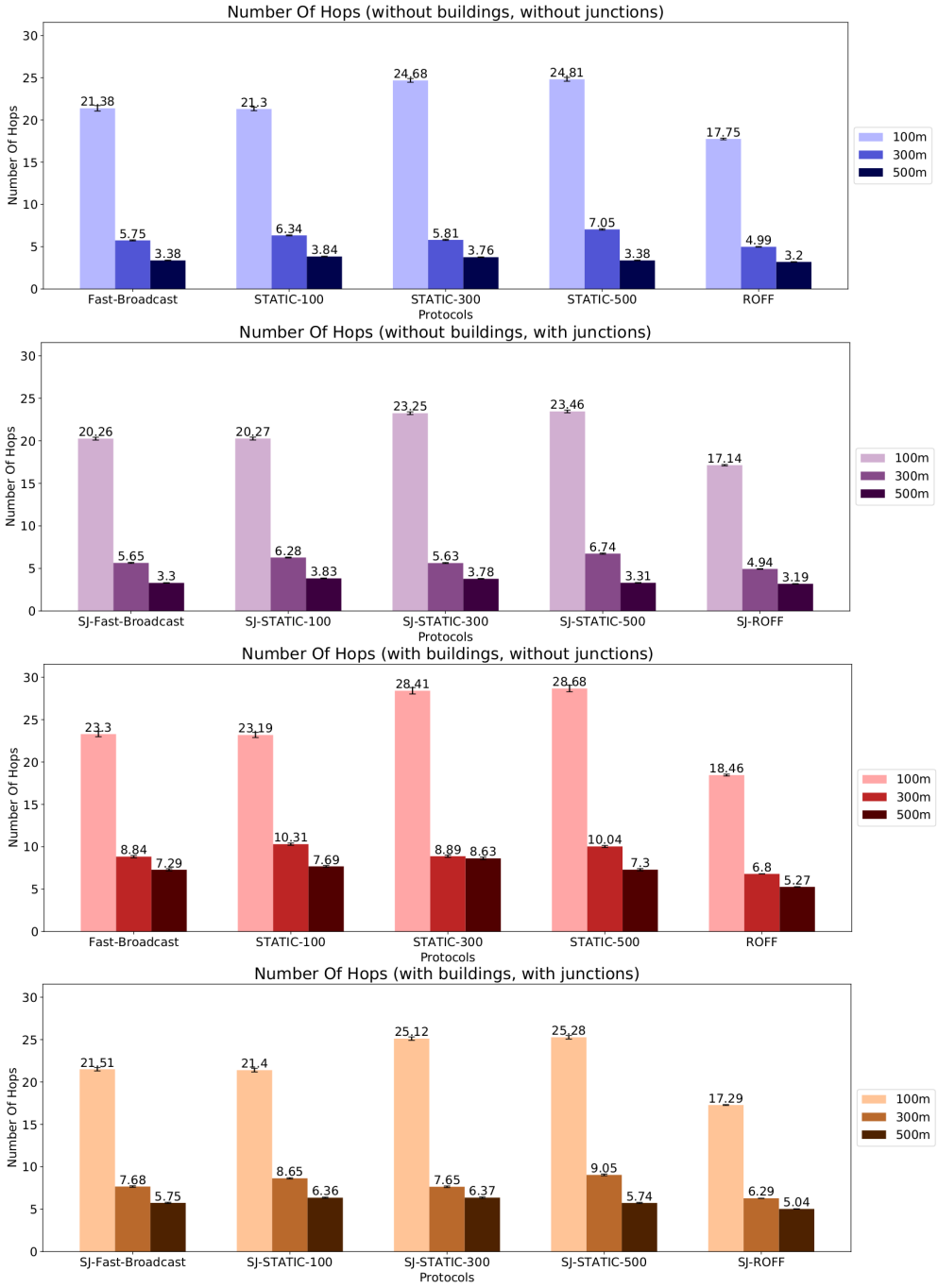

Figure 6.19: *NOH* for Los Angeles urban scenario

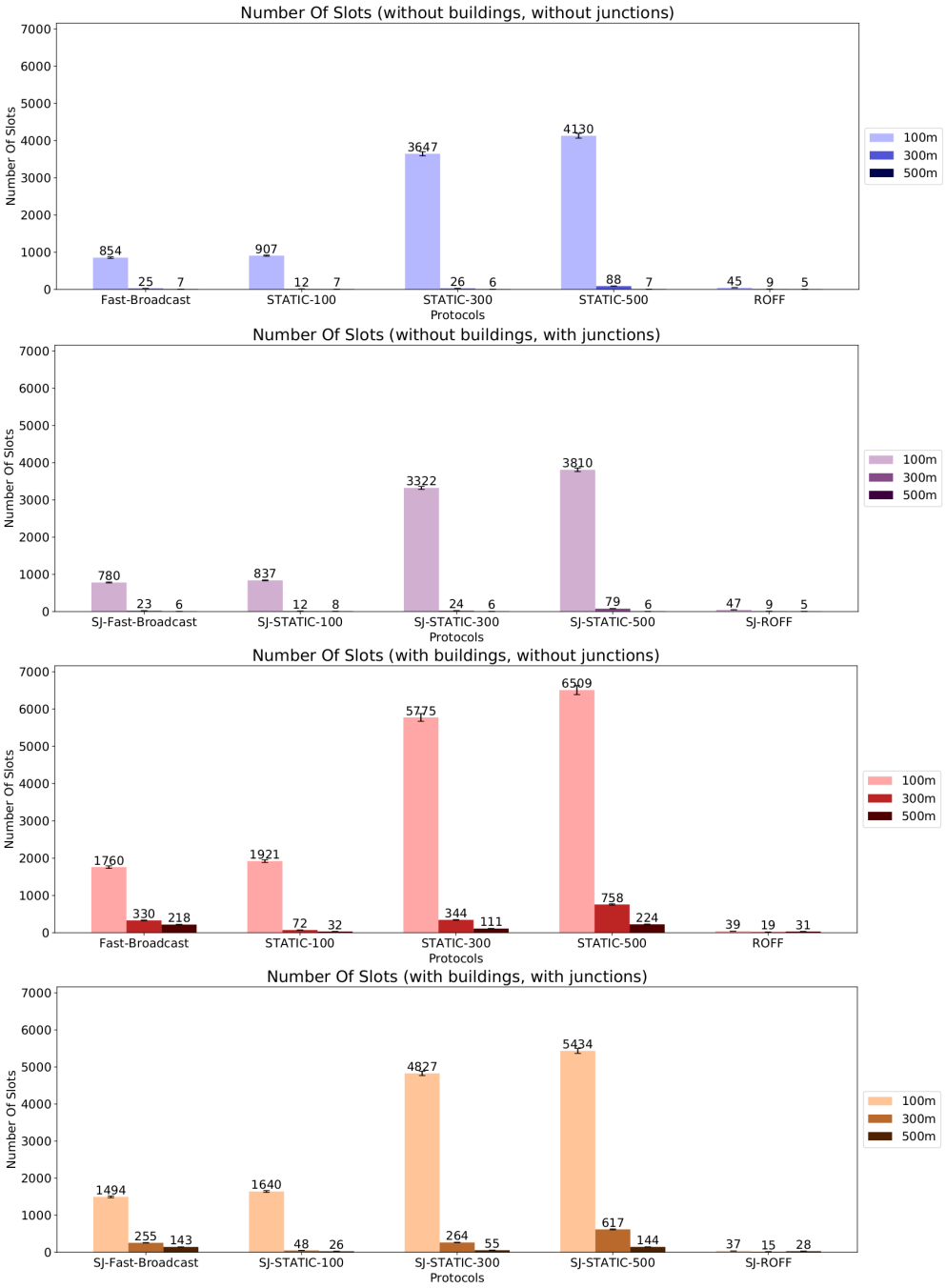

Figure 6.20: *NOS* for Los Angeles urban scenario

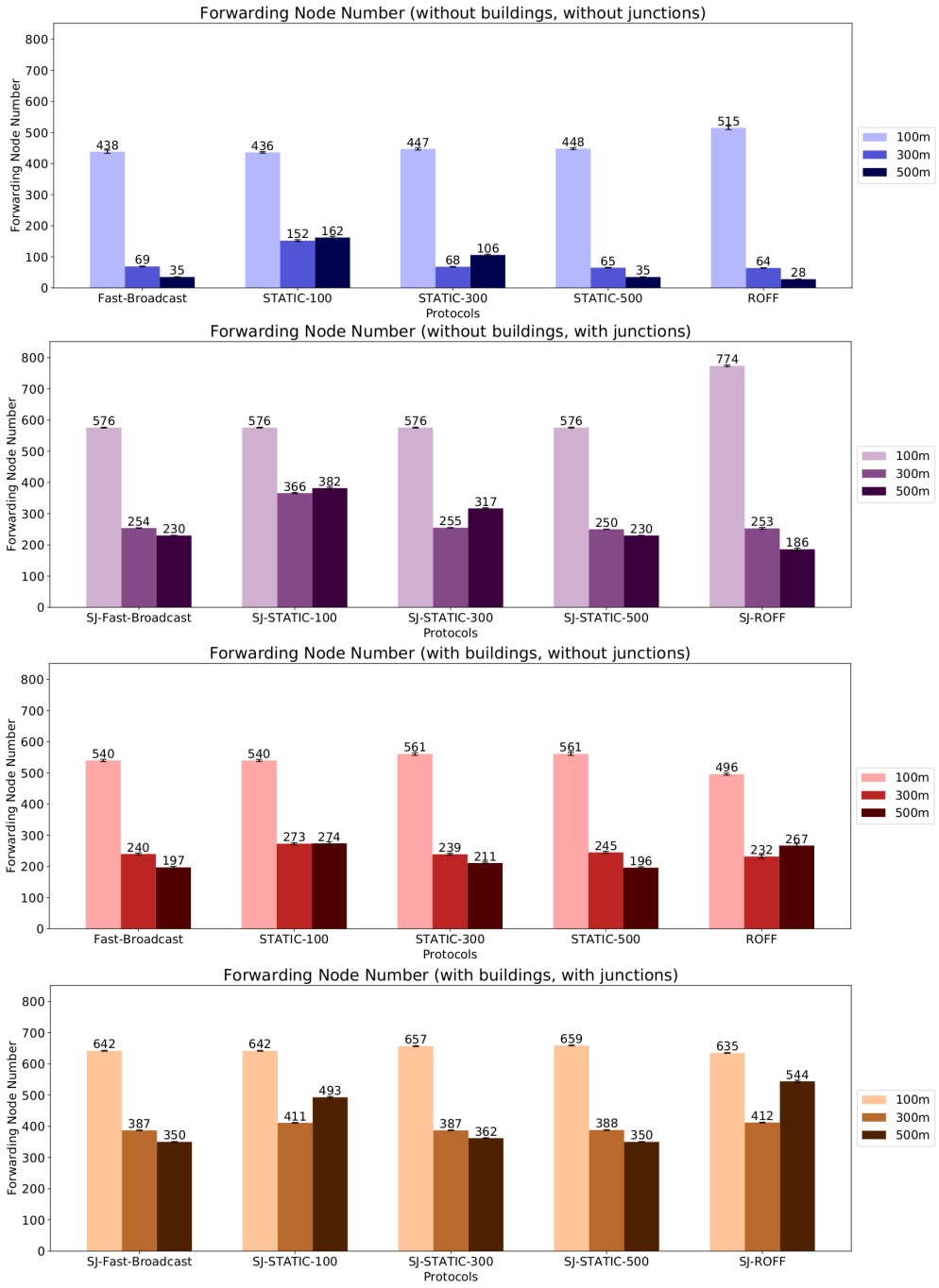

Figure 6.21: *FNN* for Los Angeles urban scenario

It is possible to observe the effects of the Obstacle Shadowing model on *Total Delivery Ratio* and *Total Delivery Ratio On Circumference*. The effects are pretty mild across all configurations, but are more noticeable with lower transmission ranges (100 and 300 meters). For example, introducing the shadowing model without considering junctions leads to a decrease of 5.97% (98.38 to 92.5) in *TDR* (Figure 6.17) with a transmission

range of 100 meters and a decrease of 5% (99.96 to 94.96) with transmission range of 300 meters for Fast-Broadcast. Results for ROFF are comparable, with decreases of 6.28% and 5.73% respectively.

The decrease in *TDROC* (Figure 6.17) is much more noticeable, especially for the 100 meters transmission range. The value decrease by 16.11% for Fast-Broadcast and 16.74% for ROFF. This means that introducing building shadowing leads to more problems when the Alert Message has to reach further distances. This phenomenon can be seen in Figure 6.22. Vehicles not reached by the Alert Message are more concentrated towards the circumference instead of the center.

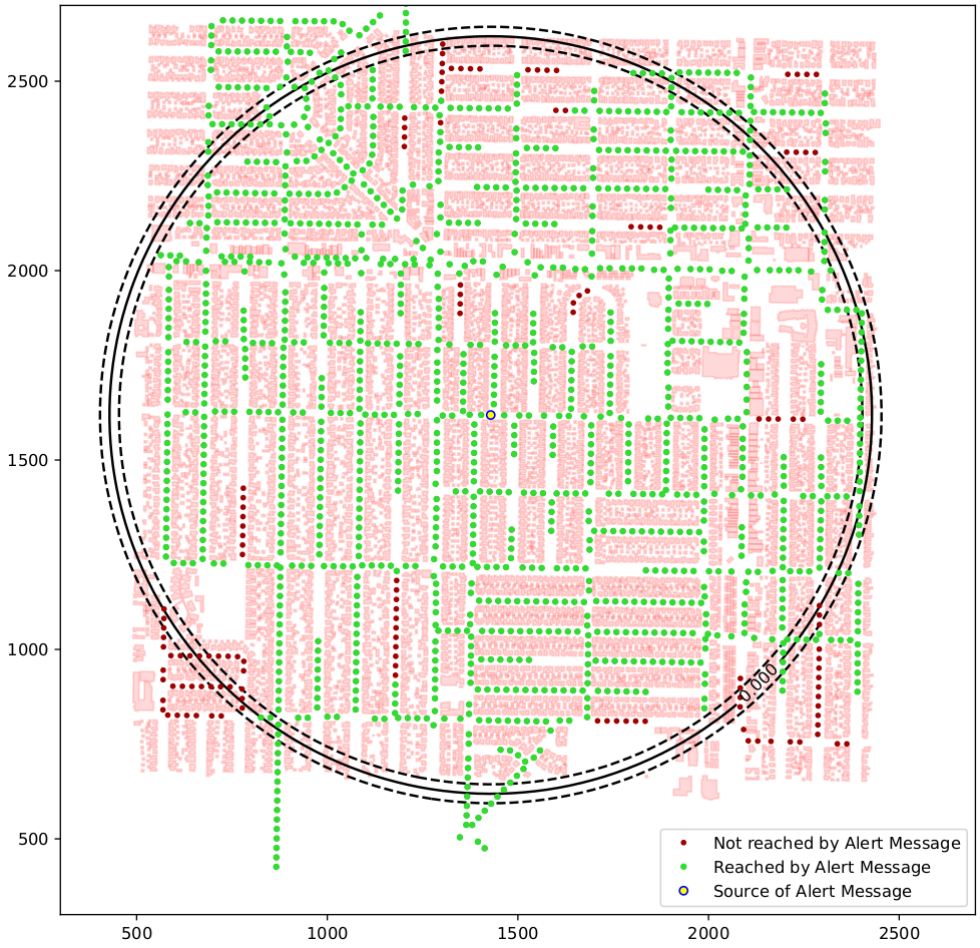

Figure 6.22: Alert Message delivery for Los Angeles scenario with buildings and without junctions (Fast-Broadcast with 100 meters transmission range)

The introduction of the smart junction variants of the algorithms proves to be fairly effective, increasing *TDR* by 3,52% and 5,03% when employing SJ-Fast-Broadcast instead of Fast-Broadcast for the scenario with buildings and transmission range of 100 and 300 meters respectively. Increases of the same magnitude can be noticed when utilizing SJ-ROFF instead of ROFF. With regards to *TDROC*, both SJ-Fast-Broadcast and SJ-ROFF reach 100% for 300 and 500 meters transmission ranges, while having a

mild increase of around 3,40% for 100 meters transmission range.

Considering the Number Of Hops to reach the circumference (6.19), we can observe that the introduction of the shadowing model leads to an increase in the metric's value across all algorithms. Comparing the first and third configuration, *NOH* rises by 8.98, 53,74 and 115,68% for Fast-Broadcast, and by 3.85, 36.27 and 64.69% for ROFF, for the three increasing transmission ranges. We notice that:

1. the rise increases with the increase in transmission range;
2. Fast-Broadcast is affected in a greater way by the shadowing model compared to ROFF.

The first point is expected, since the one-hop progress is greater in a scenario with a higher transmission range. This leads to a greater probability for the signal to run into a building. In a scenario with a lower transmission range, transmissions have a lower probability to run into a building and follow more closely the road segment even without the effects of buildings. Hence, they are less likely to be influenced by the model. The second point may be due to the fact that the Neighbor Table continues to work well to identify the FFC along the road segment, leading to a smaller increase in the number of hops.

SJ-Fast-Broadcast and SJ-ROFF need less hops in order to reach the circumference than their counterparts which do not take junctions into consideration. This holds true for both scenarios with and without buildings. This is probably due to the additional transmissions inside junctions, which help to propagate the signal along road segments in a linear way, hence covering more space with each hop. SJ-ROFF performs better than SJ-Fast-Broadcast with regards to *NOH*. The difference between the two algorithms decreases as the transmission range increases.

Regarding *NOS* (Figure 6.20), Fast-Broadcast results increase by 106.09, 1220 and 3014% while ROFF's ones increase by 111.11 and 520% for 300 and 500 meters transmission range, and decrease by 4.44% for 100 meters transmission range, when going from a building-less scenario to a scenario with buildings. Moreover, Fast-Broadcast's values are 4412.82, 1636.84 and 603.23% higher than ROFF's values in the scenario with buildings. This goes to show the effect of empty space distribution on waiting times (here represented by the number of slots waited): when the distance between the FFC and the previous forwarder is close to the estimated transmission range, then Fast-Broadcast's waiting slots are pretty close to the minimum. But when obstacles are introduced, there might not be any PFC close to the estimated transmission range. Hence, the FFC will wait a very long time. ROFF is not affected by this phenomenon thanks to the forwarding priority acquisition, as introduced in Chapter 4. Comparing the third and fourth image in Figure 6.20, we can see that the introduction of junctions is beneficial for both algorithms, leading to a decrease in *NOS*. This is a consequence of the abovementioned decrease in Number Of Hops.

Concerning the Forwarding Node Number (Figure 6.21), the value increases when the shadowing model is introduced. ROFF with 100 meters transmission range is the only configuration where the values decreases. This configuration needs further testing in order to be explained. The introduction of junctions brings about an increase in the number of forwardings for all algorithms, as expected. Focusing on the fourth image of Figure 6.21, SJ-ROFF is affected greatly by this increase, almost equalling SJ-Fast-Broadcast with 100 meters transmission range, and with *FNN* much greater values in 300 and 500 meters transmission range configurations. This might be caused by the introduction of transmissions inside junctions, which exacerbate the collision

problem already reported in the previous section (Figure 6.14), leading to an imperfect suppression of scheduled transmissions. The problem is not as severe as in the Grid scenario since the node distribution is less regular, but overall Fast-Broadcast performs slightly better in terms of FNN.

6.5 Padua urban scenario

As reported in the previous section the Los Angeles scenario, despite being realistic, still had a certain degree of regularity for what concerns vehicle distribution and scenario topology. In fact, roads and sidewalks are pretty large and the overall road positioning resembles a Grid scenario. The next step in testing consisted in employing the algorithms in a more difficult scenario, with narrower roads and intersections, smaller (if any) sidewalks and pedestrian zones where no traffic is allowed. The chosen scenario is located in Padua and, as the previous one, data about the scenario have been retrieved by OSM and processed through SUMO. The scenario is depicted in Figure 6.23, while its configuration is reported in Table 6.8.

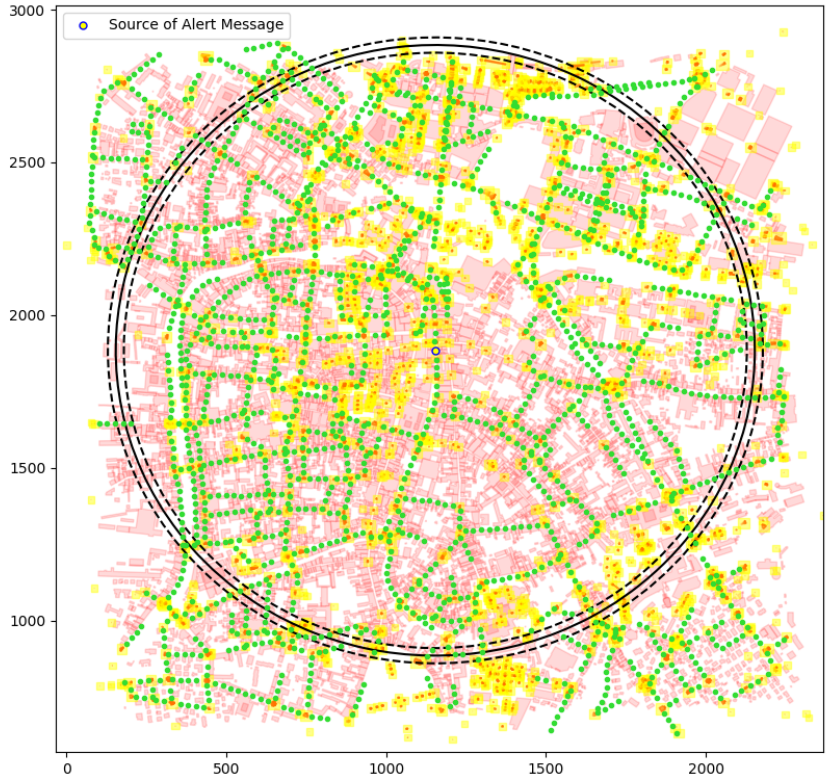

Figure 6.23: Padua urban scenario depiction

Parameter	Value	
Scenario configuration		
Latitude N	45.4171	°
Latitude S	45.3981	°
Longitude W	11.8654	°
Longitude E	11.8923	°
Road length	1200	m
Distance between vehicles	25	m
Circumference radius	1000	m
Number of vehicles	1775	
Source of alert message position	Center	
Number of buildings	6322	
Mobility model	ns3::ConstantPosition	
Number of junctions	3231	
Network configuration		
Shadowing model	Obstacle Shadowing	
Junction modeling	Yes	
Number of simulations per configuration	5000	

Table 6.8: Los Angeles urban scenario configuration

6.6 Los Angeles smart city scenario

After the comparison of the algorithms in 2D scenarios, we wanted to test them in a mixed 2D-3D scenario, where drones were also employed. Drones are utilized in many military and civil applications, such as agriculture, environmental protection and traffic flow control. It is foreseeable that they will also be a part of the development of the so called "smart cities", defined as "the use of discrete new technology applications such as RFID and Internet Of Things through more holistic conception of intelligent, integrated working that is closely linked to the concept of living and user generated services"**smartCity**. One of the possible applications of drones in a urban scenario could make them help vehicles in Emergency Message Dissemination in order to exploit their greater field of view and bypass the effects of ground level shadowing and obstacles.

This scenario is built upon the Los Angeles urban scenario presented in Section 6.4. Two layers of drones were added to the ground level of vehicles:

* the first layer is situated at a height of 30 meters from ground level and employs 732 drones;
* the second layer, employing the same number of drones, is situated at a height of 60 meters.

Drones are more spaced from one another than vehicles, with an average distance of 50 meters. We also wanted to test the effects of the Obstacle Model on drones in this scenario. Since the maximum height of a building in the data retrieved from

OpenStreetMap is 21.9 meters, drones would not have their line of sight affected by the obstacles. Hence, as an additional configuration of this scenario, all buildings have been heightened to be taller than the second layer of drones (e.g. every building is high 100 meters).

The TDR metric considers delivery to all entities in the scenario (both vehicles and drones), while the circumference relative to the area of interest continues to be defined in the same way as the 2D Los Angeles urban scenario, hence TDROC concerns exclusively vehicles.

All configurations are reported in Figure 6.24, whose colors will guide the following graph results. Table 6.10 reports all the scenario settings.

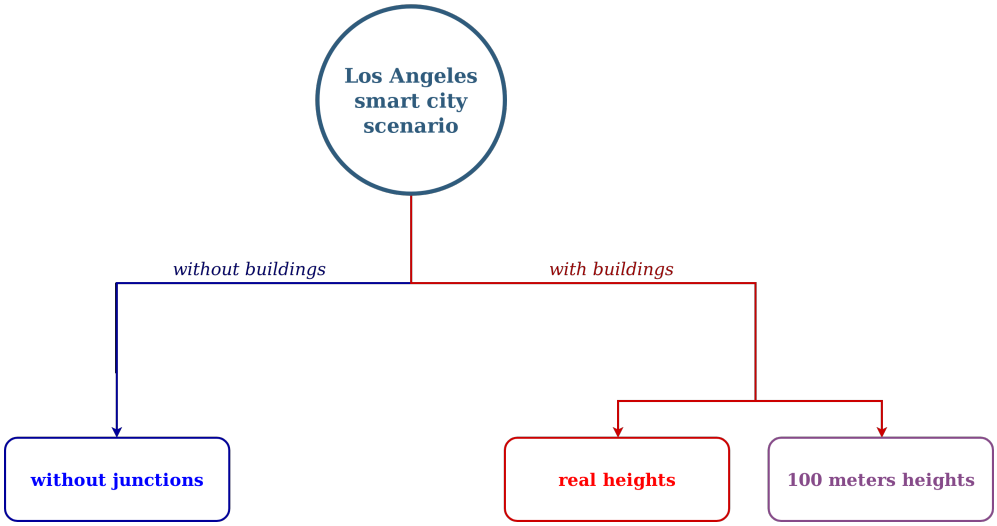

Figure 6.24: Test overview of Los Angeles smart city scenario

Parameter	Value	
Scenario configuration		
Circumference radius	1000	m
Number of vehicles	1465	
Distance between vehicles	25	m
Source of alert message position	Center	
Number of drone layers	2	
Height of first drone layer	30	m
Number of drones in first layer	732	
Height of second drone layer	60	m
Number of drones in second layer	732	
Distance between drones (average)	50	m
Number of buildings	8241	
Building heights	Real, 100	m
Mobility model	ns3::ConstantPosition	
Network configuration		
Shadowing model	Obstacle Shadowing	
Junction modeling	No	
Number of simulations per configuration	5000	

Table 6.10: Los Angeles urban scenario configuration

6.7 Hello Message forging scenario

During our analysis we wanted to test how vulnerable the algorithms were to attacks where some malicious attacker tries to increase the end to end delay (the NOS metric in our case). The attack consists in sending fake Hello Messages to vehicles during the Estimation Phase in order to mess with their estimations. We tested two different levels of severity on the attack: a low severity one, reported in Section 6.7.1, and a high severity one, reported in Section 6.7.2. Several percentages of affected vehicles (i.e. the vehicles which receive the forged hello message bursts) ranging from 0 to 50% have been tested. Both tests were carried out using the Los Angeles urban scenario without the effect of the Obstacle model. Only Fast-Broadcast and ROFF algorithms (and their Smart Junction variants) were tested.

6.7.1 Low severity attack

In the low severity attack the effect of 100 different forged Hello Messages was tested. Each forged message reports a fake position such that the detected distances by the affected node (the receiver) ranges from 301 ($txRange + 1$) to 40 ($txRange + 100$). Parameters for this scenario are reported in Table 6.12.

Parameter	Value	
Scenario configuration		
Vehicles affected by forging	0, 10, 20, 30, 40, 50	%
Number of forged messages	100	
Forged distances	301, 302,...,400	
Network configuration		
Transmission power	4.6	dBm
Transmission range	300	
Shadowing model	No	
Junction modeling	No	
Number of simulations per configuration	1000	

Table 6.12: High severity forging attack scenario configuration based on Los Angeles urban scenario

6.7.2 High severity attack

In the high severity attack the number of forged Hello Messages was increased to 1000. Moreover, even the fake positions included in the messages have been increased in a way such that the receivers detected distances ranging from 10000 to 11000 meters. Parameters for this scenario are reported in Table 6.14.

Parameter	Value	
Scenario configuration		
Vehicles affected by forging	0, 10, 20, 30, 40, 50	%
Number of forged messages	1000	
Forged distances	301, 302,...,400	
Network configuration		
Transmission power	4.6	dBm
Transmission range	300	
Shadowing model	No	
Junction modeling	No	
Number of simulations per configuration	1000	

Table 6.14: High severity forging attack scenario configuration based on Los Angeles urban scenario

6.8 Varying vehicle density scenario

6.9 Contention window

Parameter	Value
Network configuration	
Shadowing model	No
Junction modeling	No
Protocols configuration	
FB contention window	[16, 128], [32, 1024] slot
ROFF distance range (k parameter)	1
Number of simulations per configuration	1000

Table 6.16: Smaller contention window scenario configuraton based on Los Angeles urban scenario

Acronyms

LOS Line of sight. 6

NIC Network Interface Controller. 34

RPM Radio Propagation Model. 5, 7

VANET Vehicular Ad-Hoc Network. 1, 11