

Università degli Studi di Padova  
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"  
CORSO DI LAUREA MAGISTRALE IN INFORMATICA



**Fast Message Propagation  
Over IoV Scenarios**

*Master's Degree Thesis*

*Supervisor*

Prof. Claudio E. Palazzi

*Co-supervisor*

Dott. Armir Bujari

*Author*

Jordan Gottardo

---

ACADEMIC YEAR 2018-2019



# Abstract

The increasingly pervasive use of technology in the automotive industry and urban environment requires the development of broadcasting algorithms to deliver messages across vehicular ad-hoc networks (VANETs). These kinds of networks are created spontaneously and rely on vehicle-to-vehicle communication, without the need of any infrastructure or prior network topology knowledge by nodes. It is foreseeable that, in the near future, VANETs could be exploited in order to run heterogeneous applications, ranging from leisure-oriented functionalities such as video streaming and gaming, to more serious data exchanging services to monitor traffic congestion. One important application consists in emergency message distribution, where message delivery, timeliness and other life-safety related metrics are paramount. Whereas most of related work in this field is focused on one-dimensional topologies (i.e., car platooning in highways), this thesis consists in the reimplementations and redesign for two and three-dimensional scenarios of the RObust and Fast Forwarding algorithm (ROFF) and the Fast-Broadcast algorithm to compare them in various urban scenarios of increasing complexity. Considering urban scenarios, the Obstacle Model will be employed to take into account the shadowing effects of buildings on signal propagation. Moreover, this thesis proposes a Smart Junction extension for both algorithms, SJ-Fast-Broadcast and SJ-ROFF, which increase message delivery ratios by exploiting the presence of vehicles within road junctions in scenarios where the shadowing effects of obstacles is significant.



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>   | <b>1</b>  |
| 1.1      | Single-hop Broadcasting Protocols . . . . .                 | 2         |
| 1.2      | Multi-hop Broadcasting Protocols . . . . .                  | 3         |
| 1.3      | Radio Propagation Models . . . . .                          | 5         |
| <b>2</b> | <b>Obstacle and Junction Models</b>                         | <b>7</b>  |
| 2.1      | Obstacle Shadowing propagation loss model . . . . .         | 7         |
| 2.2      | Junction model . . . . .                                    | 8         |
| <b>3</b> | <b>Fast-Broadcast</b>                                       | <b>11</b> |
| 3.1      | Estimation Phase . . . . .                                  | 11        |
| 3.2      | Broadcast Phase . . . . .                                   | 12        |
| 3.3      | Multiple dimensions extension . . . . .                     | 13        |
| 3.3.1    | Hello Message Header Structure . . . . .                    | 16        |
| 3.3.2    | Alert Message Header Structure . . . . .                    | 16        |
| 3.4      | Smart Junction Fast-Broadcast (SJ-Fast-Broadcast) . . . . . | 16        |
| 3.4.1    | Alert Message Header Structure . . . . .                    | 19        |
| <b>4</b> | <b>ROFF</b>   | <b>21</b> |
| 4.1      | Forwarder Selection Problem . . . . .                       | 21        |
| 4.1.1    | Collision Analysis . . . . .                                | 21        |
| 4.1.2    | Latency Analysis . . . . .                                  | 22        |
| 4.2      | ROFF Algorithm . . . . .                                    | 24        |
| 4.2.1    | ESD Bitmap Construction . . . . .                           | 24        |
| 4.2.2    | Forwarding Priority Acquisition . . . . .                   | 26        |
| 4.2.3    | Waiting Time Assignment . . . . .                           | 27        |
| 4.3      | Multiple dimensions extension . . . . .                     | 28        |
| 4.3.1    | Hello Message Header Structure . . . . .                    | 29        |
| 4.3.2    | Alert Message Header Structure . . . . .                    | 29        |
| 4.4      | Smart Junction ROFF (SJ-ROFF) . . . . .                     | 30        |
| 4.4.1    | Alert Message Header Structure . . . . .                    | 31        |
| <b>5</b> | <b>Tools and applications</b>                               | <b>33</b> |
| 5.1      | Network Simulator 3 . . . . .                               | 33        |
| 5.1.1    | Modules and module structure . . . . .                      | 33        |
| 5.1.2    | Key elements . . . . .                                      | 34        |
| 5.1.3    | Structure of a simulation . . . . .                         | 34        |
| 5.1.4    | NetAnim . . . . .   | 35        |
| 5.2      | Simulation of Urban MObility . . . . .                      | 36        |

|   |            |
|---|------------|
| <b>6 Simulations</b>  | <b>39</b>  |
| 6.1 Metrics . . . . .   | 39         |
| 6.1.1 Total Delivery Ratio (TDR) . . . . .                    | 39         |
| 6.1.2 Total Delivery Ratio On Circumference (TDROC) . . . . . | 40         |
| 6.1.3 Number Of Hops (NOH) . . . . .                          | 40         |
| 6.1.4 Number Of Slots (NOS) . . . . .                         | 41         |
| 6.1.5 Forwarding Node Number (FNN) . . . . .                  | 42         |
| 6.2 Platoon scenario . . . . .                                | 42         |
| 6.3 Grid scenario . . . . .                                   | 45         |
| 6.4 Los Angeles urban scenario . . . . .                      | 55         |
| 6.5 Padua urban scenario . . . . .                            | 65         |
| 6.6 Los Angeles smart city scenario . . . . .                 | 72         |
| 6.7 Hello Message forging scenario . . . . .                  | 80         |
| 6.7.1 Low severity attack . . . . .                           | 81         |
| 6.7.2 High severity attack . . . . .                          | 82         |
| 6.8 Different vehicle density scenario . . . . .              | 83         |
| 6.9 Smaller contention window . . . . .                       | 90         |
| 6.10 Number of Hello Messages computation . . . . .           | 94         |
| <b>7 Conclusions</b>  | <b>97</b>  |
| 7.1 Future works . . . . .                                    | 98         |
| <b>A Fast-Broadcast modifications</b>                         | <b>99</b>  |
| A.1 Fast-Broadcast's improvement on delivery ratios . . . . . | 100        |
| A.2 Fast-Broadcast's non-pejorative effects . . . . .         | 103        |
| <b>Acronyms</b>   | <b>107</b> |
| <b>Bibliography</b>   | <b>109</b> |

# List of Figures

|   |    |
|---|----|
| 2.1 Example of obstacle shadowing in vehicle-to-vehicle communication.                                  | 8  |
| Walls encountered by the signal are surrounded in red circles . . . . .                                 |    |
| 2.2 Example of data about a junction retrieved from OpenStreetMap and elaborated through SUMO . . . . . | 9  |
| 2.3 Junction modeling process . . . . .   | 9  |
| 2.4 Example of 6 junctions in an urban scenario . . . . .   | 9  |
| <br>  |    |
| 3.1 Example of Fast-Broadcast in 2D scenario . . . . .  | 14 |
| 3.2 Fast-Broadcast Hello Message header structure . . . . .   | 16 |
| 3.3 Fast-Broadcast Alert Message header structure . . . . .   | 16 |
| 3.4 Regular Fast-Broadcast Alert Message propagation . . . . .  | 17 |
| 3.5 SJ-Fast-Broadcast Alert Message propagation . . . . .   | 18 |
| 3.6 SJ-Fast-Broadcast Alert Message header structure . . . . .  | 19 |
| <br>  |    |
| 4.1 Definition of <i>minDiff</i> between $f_N$ and $f_{N-1}$ ([21]) . . . . .                           | 22 |
| 4.2 Definition of empty space ([21]) . . . . .  | 23 |
| 4.3 Components of ROFF algorithm . . . . .  | 24 |
| 4.4 Example of Neighbor Table of node with ID 0 . . . . .   | 25 |
| 4.5 ESD Bitmap construction with $k = 1$ . . . . .  | 25 |
| 4.6 Resulting ESD Bitmap with $k = 2$ . . . . .   | 26 |
| 4.7 ID-based contention: node A defers its transmission . . . . .                                       | 27 |
| 4.8 Waiting time assignment based on vehicle distribution depicted in Figure 4.5 . . . . .              | 27 |
| 4.9 ROFF Hello Message header structure . . . . .   | 29 |
| 4.10 ROFF Alert Message header structure . . . . .  | 30 |
| 4.11 SJ-ROFF Alert Message header structure . . . . .   | 31 |
| <br>  |    |
| 5.1 ns-3 module structure . . . . .   | 34 |
| 5.2 Packet transmission in NetAnim . . . . .  | 35 |
| 5.3 Steps to generate necessary files using SUMO ([19]) . . . . .                                       | 36 |
| 5.4 Padua scenario with vehicle distance equals to 15m (top) and 45m (bottom) . . . . .                 | 37 |
| <br>  |    |
| 6.1 Example of <i>NOH</i> calculation with Alert Message starting from node S                           | 41 |
| 6.2 Example of <i>NOS</i> calculation with Alert Message starting from node S                           | 42 |
| 6.3 <i>TDR</i> metric for Platoon scenario . . . . .  | 43 |
| 6.4 <i>TDROC</i> , <i>NOH</i> , <i>NOS</i> and <i>FNN</i> metrics for Platoon scenario . . . . .        | 44 |
| 6.5 <i>TDR</i> without buildings (top) and with buildings (bottom) for Grid scenario . . . . .          | 46 |

|      |  |    |
|------|--|----|
| 6.6  | <i>TDROC</i> without buildings (top) and with buildings (bottom) for Grid scenario . . . . .   | 47 |
| 6.7  | <i>NOH</i> without buildings (top) and with buildings (bottom) for Grid scenario . . . . .   | 47 |
| 6.8  | <i>NOS</i> without buildings (top) and with buildings (bottom) for Grid scenario . . . . .   | 48 |
| 6.9  | <i>FNN</i> without buildings (top) and with buildings (bottom) for Grid scenario . . . . .   | 48 |
| 6.10 | Fast-Broadcast after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500m transmission range and without the Obstacle model . . . . .  | 50 |
| 6.11 | Fast-Broadcast after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500m transmission range and with the Obstacle model . . . . .     | 51 |
| 6.12 | ROFF after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500m transmission range and without the Obstacle model . . . . .            | 52 |
| 6.13 | ROFF after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500m transmission range and with the Obstacle model . . . . .               | 53 |
| 6.14 | Collision area in 2D scenarios for ROFF . . . . .  | 55 |
| 6.15 | Los Angeles urban scenario depiction . . . . .   | 56 |
| 6.16 | Los Angeles urban scenario test overview . . . . .   | 56 |
| 6.17 | <i>TDR</i> for Los Angeles urban scenario . . . . .  | 58 |
| 6.18 | <i>TDROC</i> for Los Angeles urban scenario . . . . .  | 59 |
| 6.19 | <i>NOH</i> for Los Angeles urban scenario . . . . .  | 60 |
| 6.20 | <i>NOS</i> for Los Angeles urban scenario . . . . .  | 61 |
| 6.21 | <i>FNN</i> for Los Angeles urban scenario . . . . .  | 62 |
| 6.22 | Alert Message delivery for Los Angeles scenario with buildings and without junctions (Fast-Broadcast with 100m transmission range) . . . . . | 63 |
| 6.23 | Padua urban scenario depiction . . . . .   | 65 |
| 6.24 | <i>TDR</i> for Padua urban scenario . . . . .  | 67 |
| 6.25 | <i>TDROC</i> for Padua urban scenario . . . . .  | 68 |
| 6.26 | <i>NOH</i> for Padua urban scenario . . . . .  | 69 |
| 6.27 | <i>NOS</i> for Padua urban scenario . . . . .  | 70 |
| 6.28 | <i>FNN</i> for Padua urban scenario . . . . .  | 71 |
| 6.29 | Los Angeles smart city scenario with real height buildings (top) and very high buildings (bottom) . . . . .                                  | 73 |
| 6.30 | Test overview of Los Angeles smart city scenario . . . . .   | 74 |
| 6.31 | <i>TDR</i> for Los angeles smart city scenario . . . . .   | 75 |
| 6.32 | <i>TDROC</i> for Los angeles smart city scenario . . . . .   | 76 |
| 6.33 | <i>NOH</i> for Los angeles smart city scenario . . . . .   | 77 |
| 6.34 | <i>NOS</i> for Los angeles smart city scenario . . . . .   | 78 |
| 6.35 | <i>FNN</i> for Los angeles smart city scenario . . . . .   | 79 |
| 6.36 | <i>NOS</i> for low severity forging scenario . . . . .   | 81 |
| 6.37 | <i>NOS</i> for high severity forging scenario . . . . .  | 83 |
| 6.38 | <i>TDR</i> for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges . . . . .             | 84 |
| 6.39 | <i>TDROC</i> for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges . . . . .           | 85 |
| 6.40 | <i>NOH</i> for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges . . . . .             | 86 |
| 6.41 | <i>NOS</i> for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges . . . . .             | 87 |

|      |  |     |
|------|--|-----|
| 6.42 | <i>FNN</i> for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges . . . . .                   | 88  |
| 6.43 | <i>FNN</i> for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges (different angle) . . . . . | 89  |
| 6.44 | <i>TDR</i> for [16, 128] contention window (top) and [32, 1024] contention window (bottom) . . . . .   | 91  |
| 6.45 | <i>TDROC</i> for [16, 128] contention window (top) and [32, 1024] contention window (bottom) . . . . .   | 91  |
| 6.46 | <i>NOH</i> for [16, 128] contention window (top) and [32, 1024] contention window (bottom) . . . . .   | 92  |
| 6.47 | <i>NOS</i> for [16, 128] contention window (top) and [32, 1024] contention window (bottom) . . . . .   | 92  |
| 6.48 | <i>FNN</i> for [16, 128] contention window (top) and [32, 1024] contention window (bottom) . . . . .   | 93  |
| 6.49 | Number of Hello Messages sent for Fast-Broadcast with 100m (top), 300m (middle) and 500m (bottom) transmission range . . . . .                     | 95  |
| A.1  | <i>TDR</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario . . . . .  | 100 |
| A.2  | <i>TDROC</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario . . . . .  | 101 |
| A.3  | <i>NOH</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario . . . . .  | 101 |
| A.4  | <i>NOS</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario . . . . .  | 102 |
| A.5  | <i>FNN</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario . . . . .  | 102 |
| A.6  | <i>TDR</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario . . . . .  | 103 |
| A.7  | <i>TDROC</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario . . . . .                                      | 104 |
| A.8  | <i>NOH</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario . . . . .  | 104 |
| A.9  | <i>NOS</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario . . . . .  | 105 |

|  |     |
|--|-----|
| A.10 <i>FNN</i> for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario . . . . . | 105 |
|--|-----|

## List of Tables

|  |    |
|--|----|
| 6.2 Platoon scenario configuration . . . . .   | 43 |
| 6.4 Grid scenario configuration . . . . .  | 46 |
| 6.6 Los Angeles urban scenario configuration . . . . .   | 57 |
| 6.8 Los Angeles urban scenario configuration . . . . .   | 66 |
| 6.10 Los Angeles urban scenario configuration . . . . .  | 74 |
| 6.12 Low severity forging attack scenario configuration based on Los Angeles urban scenario . . . . .  | 81 |
| 6.14 High severity forging attack scenario configuration based on Los Angeles urban scenario . . . . . | 82 |
| 6.16 Different vehicle densities scenario configuration based on Padua urban scenario . . . . .        | 84 |
| 6.18 Contention window scenario configuraton based on Los Angeles urban scenario . . . . .             | 90 |
| 6.19 Number of Hello Messages sent for every turn . . . . .  | 95 |
| 7.2 Fast-Broadcast and ROFF's pros and cons . . . . .  | 98 |

# Chapter 1

## Introduction

Emergency Message Dissemination (EMD) is a fundamental application in [VANETs](#) 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 schemes rather than unicast or multicast ones.[\[1\]](#)

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 the farthest vehicle can fail and interrupt the message dissemination. Therefore, the next forwarder designation keeps into consideration also other vehicles (called potential forwarder candidates, PFCs) which have received an Alert Message. The PFCs participate in a contention to elect the farthest forwarder candidate (FFC) which 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 EMD has lead to the development of a number of broadcasting protocols. However, as identified by Panichpapiboon et al.[\[1\]](#), 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 transmissions in order not to waste bandwidth and saturate the channel.

## 1.1 Single-hop Broadcasting Protocols

Vehicles employing Single-Hop Broadcasting protocols will not flood received packets 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 kinds of protocol can work on to aim for greater network efficiency are:

- \* *Broadcast Interval*, which is the amount of time between retransmissions. When modulating this variable, it is important to 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** protocols, which keep the Broadcast Interval fixed. Some examples are:
  - *TrafficInfo*[2], 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*[3], in which vehicles exchange information about speed and position and record it in their database. Data about different vehicles are 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** protocols, which adapt the Broadcast Interval based on dynamic information. Some examples are:
  - *Collision Ratio Control Protocol (CRCP)*[4], 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*[5], 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)*[6], a protocol according to which roads are divided into segments and each vehicle can both discover information itself and collect it from neighbors' 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 or mollification event has taken place.

## 1.2 Multi-hop Broadcasting Protocols

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

1. **Delay based protocols**, which assign a different waiting time before rebroadcasting 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)*[7], 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)*[8] and *Efficient Directional Broadcast (EDB)* [9], which try to reduce the delay introduced by *UMB* and remove the *RTB* and *CTB* packets, respectively.
  - *Vehicle-Density-based Emergency Broadcasting (VDEB)*[10], a slotted broadcasting protocol which keeps vehicle density into consideration when computing waiting time slots.

- *Reliable Method for Disseminating Safety Information (RMDSI)*[11], 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)*[12], a protocol that keeps traffic congestion into consideration by checking whether the number of neighbors of a vehicle is greater 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)*[13], 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.

**2. Probabilistic-based Multi-hop Broadcasting Protocols.** The idea behind these kinds of protocol 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*[14], in which every PFC computes its own rebroadcast probability based on the distance between itself and the transmitter. The formula used is the following:

$$p_{ij} = \frac{D_{ij}}{R} \quad (1.1)$$

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 vehicle density into account and also assumes that the transmission range is fixed and known to all vehicles.

- *Optimized Adaptive Probabilistic Broadcast (OAPB)*[15] and *AutoCast (AC)*[16], 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)*[17][18], 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:

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

where  $\rho_s$  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[19] focused on the implementation and testing of Fast-Broadcast, a Multi-hop delay based protocol firstly presented in [20]. 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[21], another Multi-Hop delay based protocol, state that existing protocols are affected by two main problems:

- \* the unperfect 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 constant 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.

### 1.3 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 case, 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 the computational complexity and scalability of the model into consideration: 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 [22] 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 surrounding 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 [23] [24].

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 [25] 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

# Obstacle and Junction Models

In this Chapter the models implemented and utilized in this work will be presented, namely the Obstacle Shadowing propagation loss model (Section 2.1) and the Junction model (Section 2.2).

### 2.1 Obstacle Shadowing propagation loss model

The original thesis [19], after having analyzed various works concerning shadowing in urban scenarios [26] [27], used a deterministic RPM called Obstacle Shadowing propagation loss model presented in [28] and implemented by the authors of [29]. This propagation model calculates the loss in signal strength due to the shadowing effect of obstacles such as buildings.

The authors of [28] 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.

$$P_r[dBm] = P_t[dBm] + G_t[dB] + G_r[dB] - \sum L_x[dB] \quad (2.1)$$

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.

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

$$L_{LogNorm} = 10 \lg (X_\sigma) \quad [dB] \quad (2.3)$$

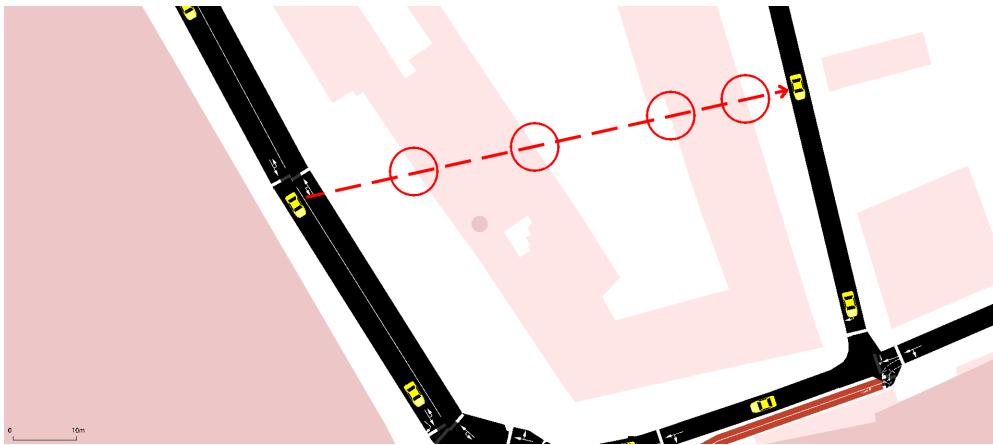
The authors extended the general model shown in 2.1 adding a  $L_{obs}$  term for each obstacle in the line of sight between sender and receiver. The term is described by Equation 2.4, where:

- \*  $n$  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;
- \*  $\beta$  represents the attenuation due to the exterior wall of a building, in dB per wall;
- \*  $\gamma$  represents an approximation of the internal structure of a building, in dB per meter.

Parameters  $\beta$  and  $\gamma$  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.

$$L_{obs} = \beta n + \gamma d_m \quad (2.4)$$

[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 model

Part of this work consisted in implementing extensions for the algorithms which will be presented in Chapter [3](#) and [4](#) to utilize road 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.

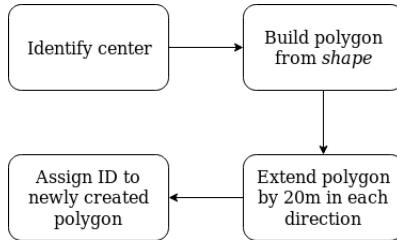
```
<junction id="122666592" type="priority" x="1204.30" y="2735.10" inLanes="13446476#0_0
-13433275#0_0 -13446476#1_0" intLanes=":122666592_0_0 :122666592_9_0 :122666592_10_0
:122666592_3_0 :122666592_4_0 :122666592_5_0 :122666592_6_0 :122666592_7_0 :122666592_11_0"
shape="1205.98,2741.13 1210.40,2736.50 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 20m 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 [20] 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 **Broadcast 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  $\leftarrow$  random(turnSize)
3:   wait(sendingTime)
4:   if  $\neg$  (heardHelloMsg()  $\vee$  heardCollision()) then
5:     helloMsg.declaredMaxRange  $\leftarrow$  max(LMFR, CMFR)
6:     helloMsg.senderPosition  $\leftarrow$  retrievePosition()
7:     transmit(helloMsg)
8:   end if
9: end for
```

---



---

**Algorithm 2** Hello message receiving procedure for 1D

---

```

1: myPosition  $\leftarrow$  retrievePosition()
2: senderPosition  $\leftarrow$  helloMsg.senderPosition
3: declaredMaxRange  $\leftarrow$  helloMsg.declaredMaxRange
4: d  $\leftarrow$  distance(myPosition, senderPosition)
5: if receivedFromFront(helloMsg) then
6:   CMFR  $\leftarrow$  max(CMFR, d, declaredMaxRange)
7: else
8:   CMBR  $\leftarrow$  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.

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

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 [20] 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 [30] 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 [30] and then discuss the proposed modification.

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

- 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.

As explained in the original extension, 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 becoming the next forwarder 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 A. 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 scenarios, 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  $\leftarrow$  random(turnSize)
3:   wait(sendingTime)
4:   if  $\neg$  (heardHelloMsg()  $\vee$  heardCollision()) then
5:     helloMsg.declaredMaxRange  $\leftarrow$  max(LMR, CMR)
6:     helloMsg.senderPosition  $\leftarrow$  retrievePosition()
7:     transmit(helloMsg)
8:   end if
9: end for
```

---



---

**Algorithm 6** Hello message receiving procedure for 2D

---

```

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

---



---

**Algorithm 7** Alert Message generation procedure for 2D

---

```

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

---



---

**Algorithm 8** Alert Message forwarding procedure for 2D

---

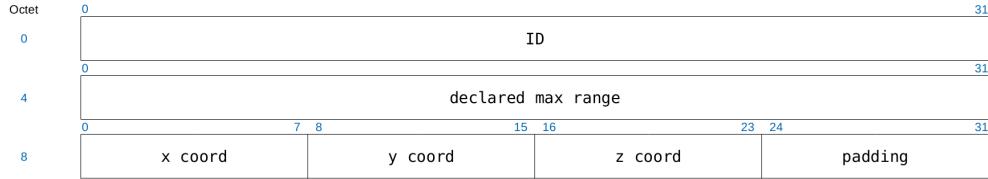
```

1: cwnd  $\leftarrow$  computeCwnd()
2: waitTime  $\leftarrow$  random(cwnd)
3: wait(waitTime)
4: if sameBroadcastHeardFromBack() then
5:   exit()
6: else if sameBroadcastHeardFromFront() then
7:   restartBroadcastProcedure()
8: else
9:   alertMsg.maxRange  $\leftarrow$  max(LMR, CMR)
10:  alertMsg.senderPosition  $\leftarrow$  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.



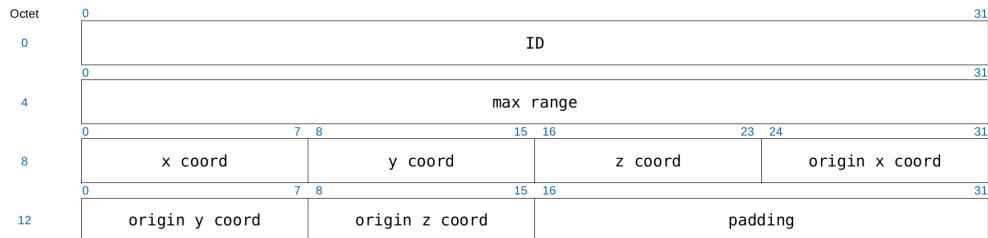
**Figure 3.2:** Fast-Broadcast Hello Message header structure

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

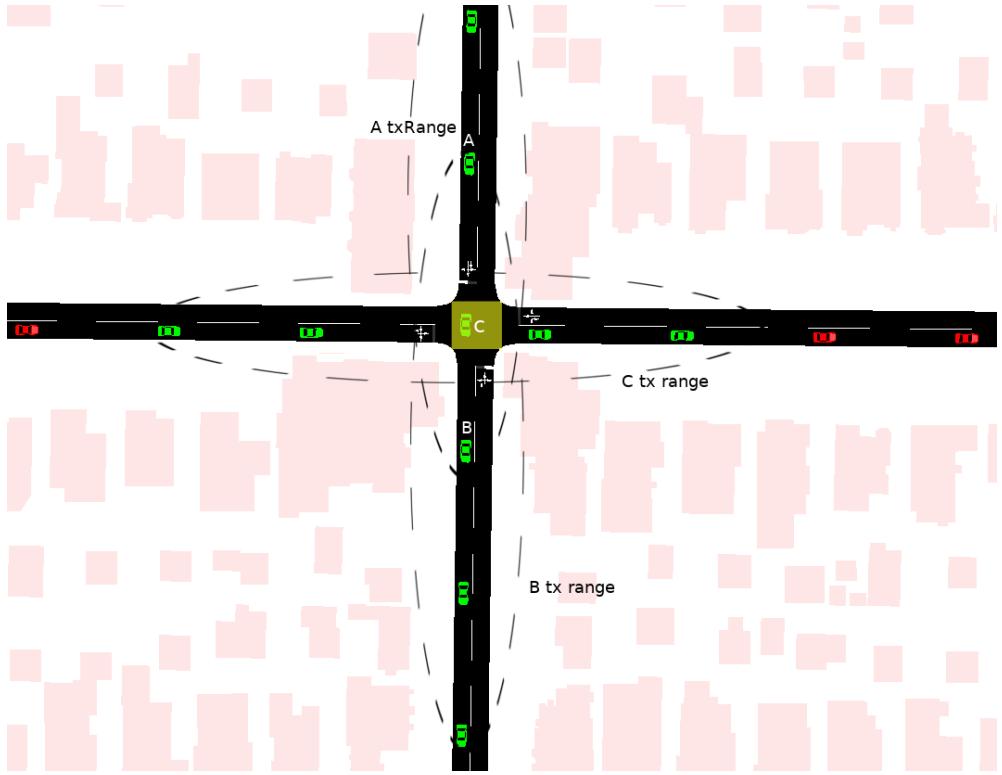
across different roads, the message is relayed only through the road segments inside the forwarder's field of view. This causes the Alert Message propagation to spread:

- \* 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:** Regular 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

---

```

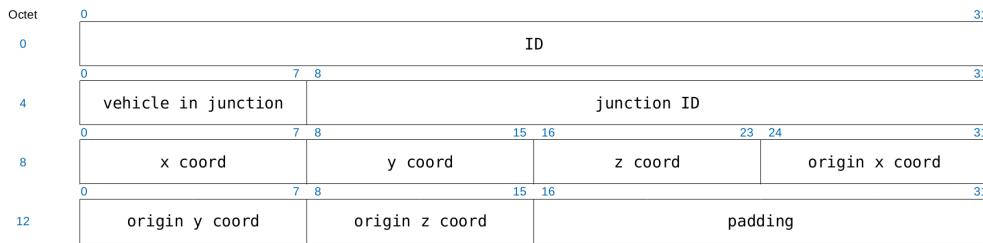
1: cwnd ← computeCwnd()
2: waitTime ← random(cwnd)
3: wait(waitTime)
4: if (sameBroadcastHeardFromBack() ∧ vehicleNotInJunction()) ∨ (sameBroadcastHeardFromBack() ∧ vehicleInJunction(j) ∧ alertMsg.senderInJunction ∧ alertMsg.junctionId == j) then
5:   exit()
6: else if sameBroadcastHeardFromFront() then
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

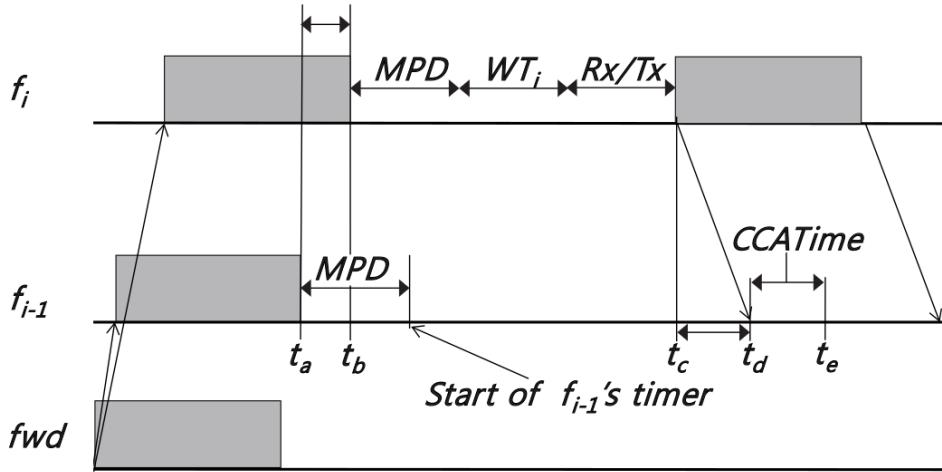
RObust and Fast Forwarding scheme (ROFF) is a protocol proposed by Hongseok Yoo and Dongkyun Kim in [21]. This chapter will present the two main problems tackled by ROFF already introduced in Section 1, namely the perfect suppression of redundant transmissions, which will be explained in 4.1.1 , and 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, 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  $f_{N-1}$  ([21])

Figure 4.1 depicts a scenario 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 algorithm 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 waits 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  $pd_{f_{i-1}, f_i}$ . 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 successful 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 successful 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:

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

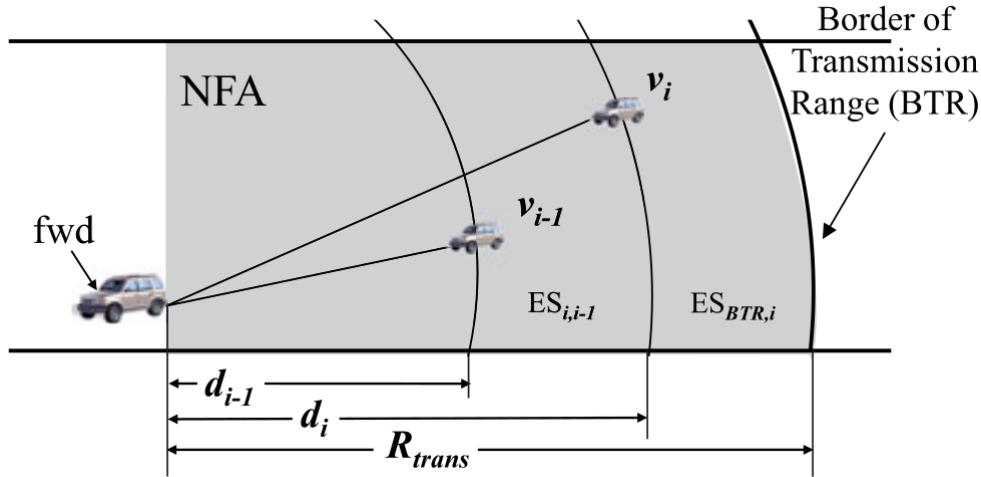
#### 4.1.2 Latency Analysis

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 ([21])

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  $d_i - d_{i-1}$ .

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.

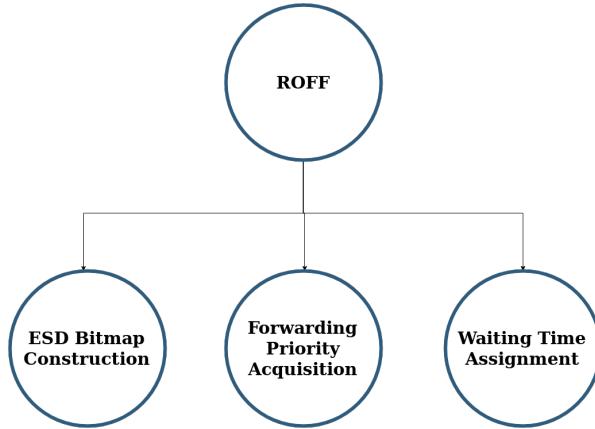
ROFF aims at resolving the effect of empty spaces by allowing vehicles to choose waiting time directly proportional to their *unique forwarding priority* proportional to the distance between the vehicle and the previous forwarder, instead of using directly such distance in the waiting time computation.

## 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 describes the empty space distribution within the forwarder's 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 outdated entries.

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

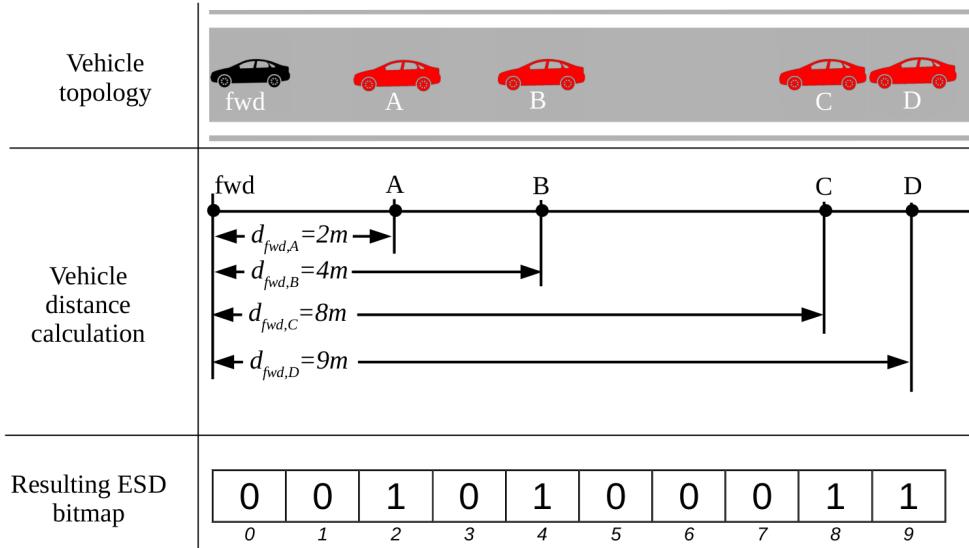
| ID | Position      | Reception time |
|----|---------------|----------------|
| 1  | (150, 120, 0) | 13:25:31       |
| 2  | (130, 145, 0) | 13:25:32       |
| 3  | (180, 200, 0) | 13:25:31       |

**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. Figure 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  $8 \leq d \leq 9$ .

|          |          |          |          |          |
|----------|----------|----------|----------|----------|
| <b>0</b> | <b>1</b> | <b>1</b> | <b>0</b> | <b>1</b> |
| 0        | 1        | 2        | 3        | 4        |

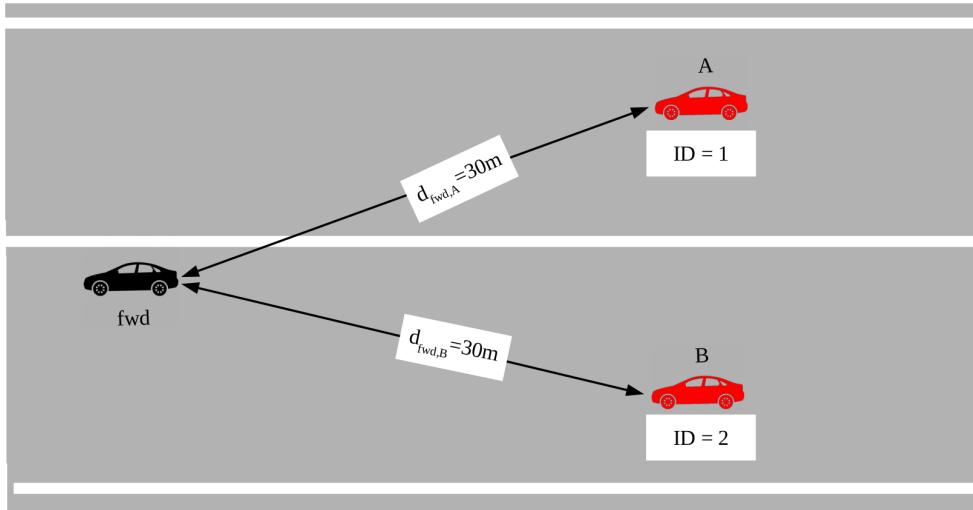
**Figure 4.6:** Resulting ESD Bitmap with  $k = 2$

#### 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  $\text{minDiff}_{PFC(i),PFC(i-1)}$  than waiting time of  $PFC(i)$  (i.e.,  $WT_{PFC(i-1)} \geq WT_{PFC(i)} + \text{minDiff}_{PFC(i),PFC(i-1)}$ ). Hence, the waiting time of  $PFC(k)$  is the sum of  $\text{minDiffs}$  (calculated using Equation 4.1) between PFCs with forwarding priorities lower than  $k$ , as shown in Equation 4.2. The PFC with top priority (being equal to 1) broadcasts immediately and does not have to wait any time.

$$WT_{PFC(k)} = \sum_{i=2}^k \text{minDiff}_{PFC(i),PFC(i-1)} \quad (k \geq 2) \quad (4.2)$$

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        | $\text{minDiff}(C, D)$   |
| B   | 3        | $\text{minDiff}(B, C) + \text{minDiff}(C, D)$                        |
| A   | 4        | $\text{minDiff}(A, B) + \text{minDiff}(B, C) + \text{minDiff}(C, D)$ |

**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.,  $d(fwd, f_j) = dist$ ) inside its own NBT. If such vehicle exists, then  $f_i$  uses  $f_j$ 's 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 [21], 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

---

```

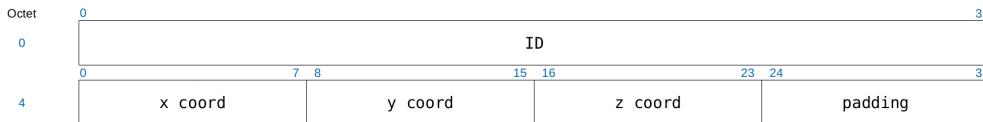
1: esdBitmap ← alertMsg.esdBitmap
2: if vehicleNotPresentInEsd() ∨ lostIDContention() then
3:   exit()
4: else
5:   listOfDistances ← computeListOfDistances(esdBitmap)
6:   myPosition ← retrievePosition()
7:   senderPosition ← alertMsg.senderPosition
8:   dist ← distance(myPosition, senderPosition)
9:   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.



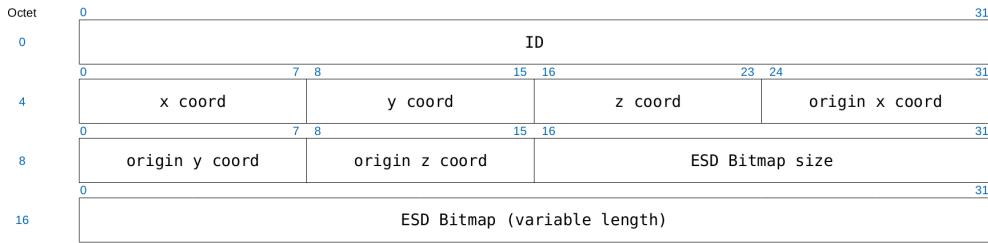
**Figure 4.9:** ROFF Hello Message header structure

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 located within a junction (and the ID of the junction, if it is). Algorithm 15 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

---

```

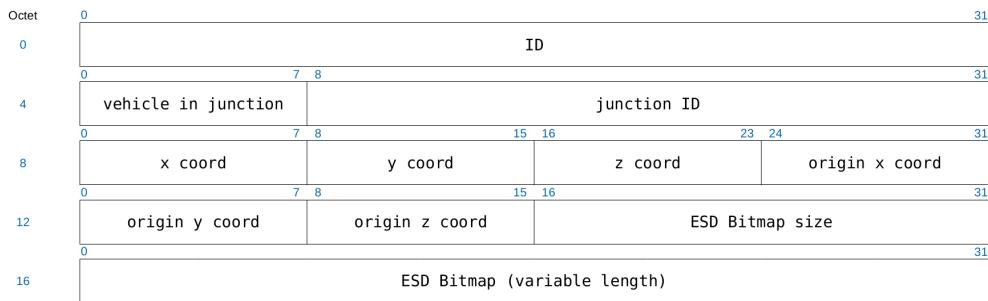
1: esdBitmap ← alertMsg.esdBitmap
2: if vehicleNotPresentInEsd() ∨ lostIDContention() then
3:   exit()
4: else
5:   listOfDistances ← computeListOfDistances(esdBitmap)
6:   myPosition ← retrievePosition()
7:   senderPosition ← alertMsg.senderPosition
8:   dist ← distance(myPosition, senderPosition)
9:   myRank ← getRank(dist, listOfDistances)
10:  wait ← computeWaitTime(myRank)
11:  if (sameBroadcastHeardFromBack() ∧ vehicleNotInJunction()) ∨ (sameBroadcastHeardFromBack() ∧ vehicleInJunction(j) ∧ 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.

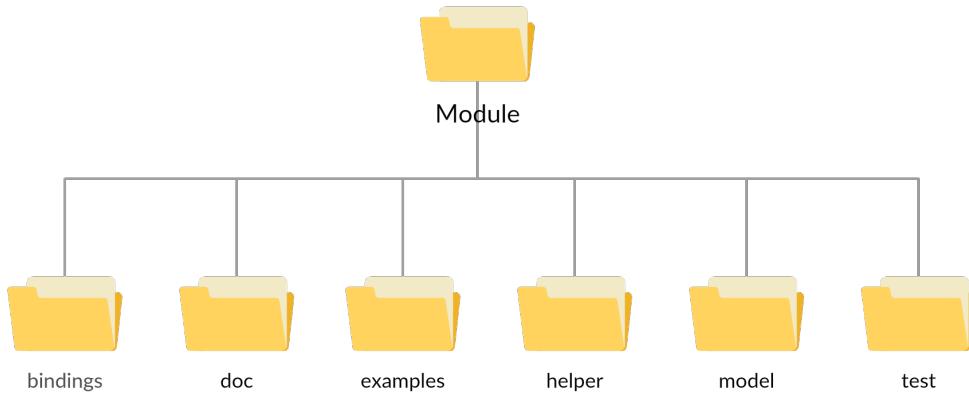
#### 5.1.1 Modules and module structure

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

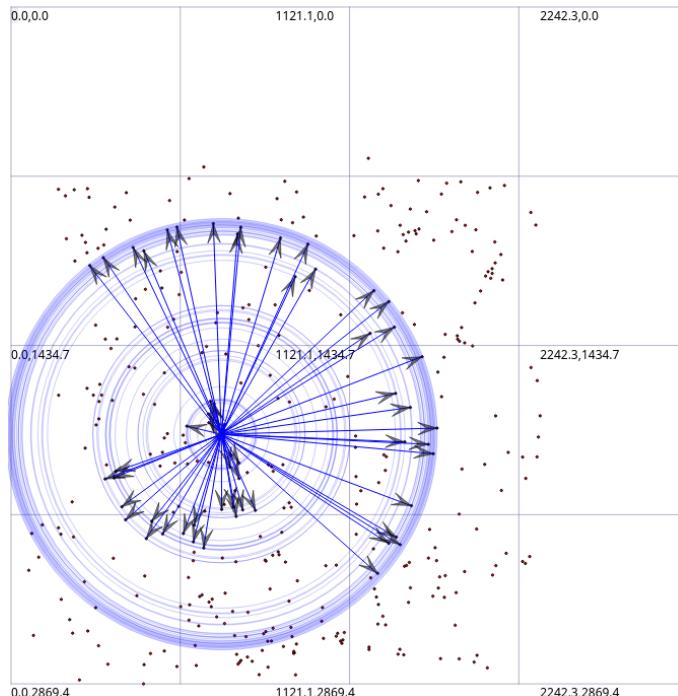
### 5.1.3 Structure of a simulation

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.

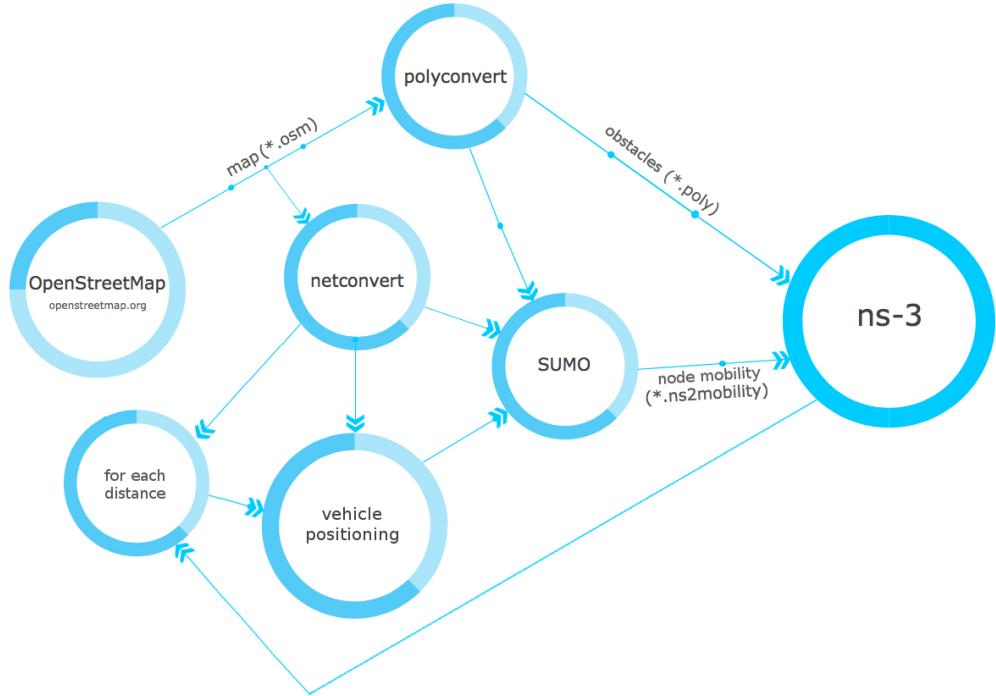
The original work [19] , starting from real maps obtained from OpenStreetMap (OSM), utilized SUMO to produce two files:

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 25m between vehicles; this work considers various distances, ranging from 5m to 45m. 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 ([19])



**Figure 5.4:** Padua scenario with vehicle distance equals to 15m (top) and 45m (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 scenarios of increasing complexity. Several algorithms will be tested, namely:

- \* Fast-Broadcast and SJ-Fast-Broadcast;
- \* ROFF and SJ-ROFF;
- \* STATIC variants of Fast-Broadcast (e.g., STATIC-100), which behave like Fast-Broadcast but do not employ a transmission range estimation. They only use a static parameter (for example 100m for STATIC-100) and use it in the Broadcast Phase;
- \* SJ-STATIC variants of Fast-Broadcast, which also considers junctions during message propagation, but do not employ a transmission range estimation.

Numeric results in every chart will be coupled with their 95% confidence interval. All following simulation have been run on the High Performance Computing platform of the University of Padua, Department of Mathematics [31].

### 6.1 Metrics

Before proceeding with the presentation of simulation results, the metrics used to evaluate the protocols' 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:

$$TDR = \frac{\text{no. of vehicles successfully receiving the Alert Message}}{\text{no. of vehicles in the scenario}} \quad (6.1)$$

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 (i.e., the area of interest where the alert data has to be delivered) 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. Basically, this metric considers only vehicles far from the source of the AM, instead of all vehicles as in the previous metric. Hence, this is used to measure how effective the algorithm is at delivering the message far from the origin. The metric is calculated as follows:

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

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 forwarded Alert Messages from the source to all destinations. We have that *NOH* is always greater than or equal to the *Optimal Number of Hops* *ONOH* (i.e.,  $ONOH \leq NOH$ ). *ONOH* is calculated using the following formula:

$$ONOH = \frac{\text{Circumference Radius}}{\text{Transmission Range}}$$

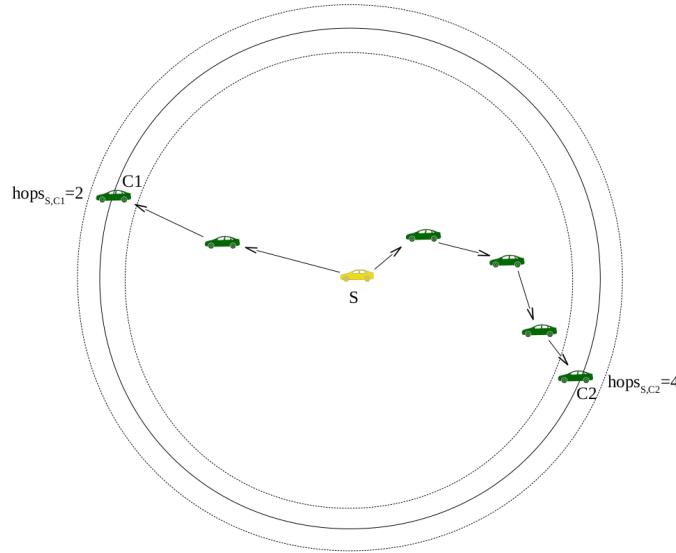
The Number Of Hops metric can be calculated as follows:

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

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 *ONOH*) 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

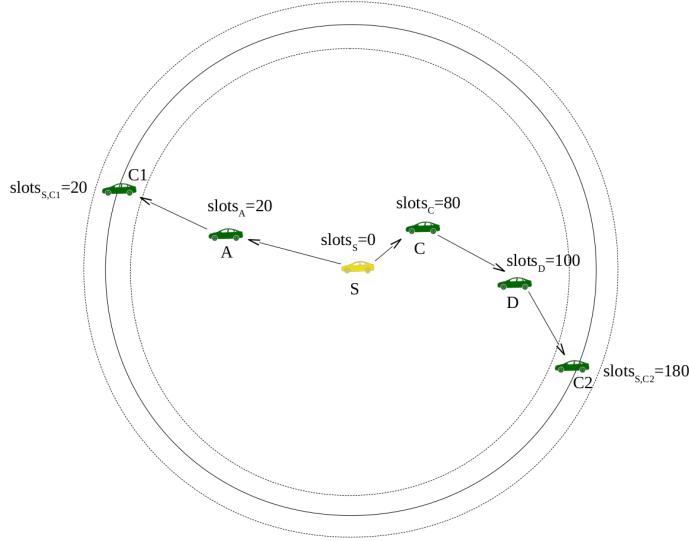
#### 6.1.4 Number Of Slots (NOS)

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.

$$NOS = \frac{\sum_{p \in RC} \text{no. of slots waited along path from source to } p}{|RC|} \quad (6.4)$$

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)

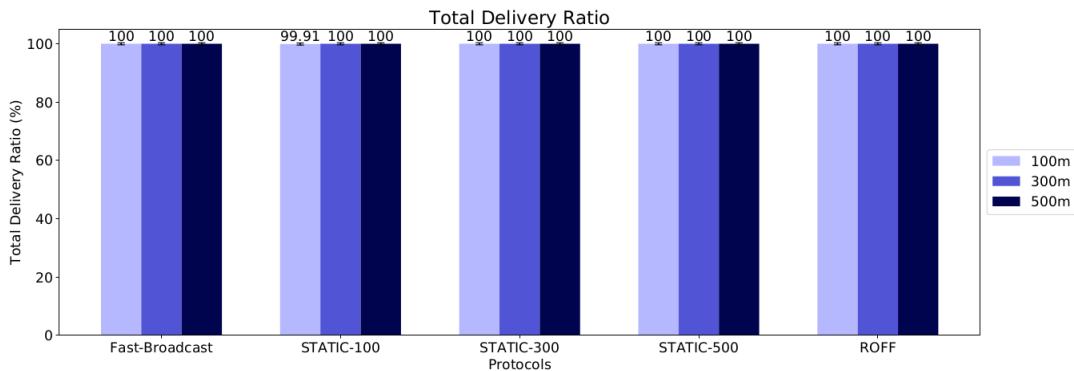
This metric is used to measure the number of vehicles which forward the Alert Message. The value of this metric is an indicator of the effectiveness of the multi-hop protocol in successfully suppressing scheduled transmissions from [PFCs](#) after the [FFC](#) has relayed the message. The metric is calculated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

$$FNN = \text{no. of vehicles forwarding the Alert Message} \quad (6.5)$$

## 6.2 Platoon scenario

The first scenario taken into consideration is a simple one-dimensional platoon scenario, where vehicles are placed in a strip-like area 15km long. Vehicles are 25m distant from each other. Transmission ranges of 100m, 300m and 500m has been employed during simulations. Parameters for this scenario are included in Table 6.2. These parameters will also be valid for all following scenarios, unless specified otherwise.

| 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       |      |
| Network configuration                   |                       |      |
| Packet payload 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                  |      |
| Mobility model                          | ns3::ConstantPosition |      |
| Propagation loss model                  | ns3::TwoRayGround     |      |
| Propagation delay model                 | ns3::ConstantSpeed    |      |
| Shadowing model                         | No                    |      |
| 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 metric for Platoon scenario

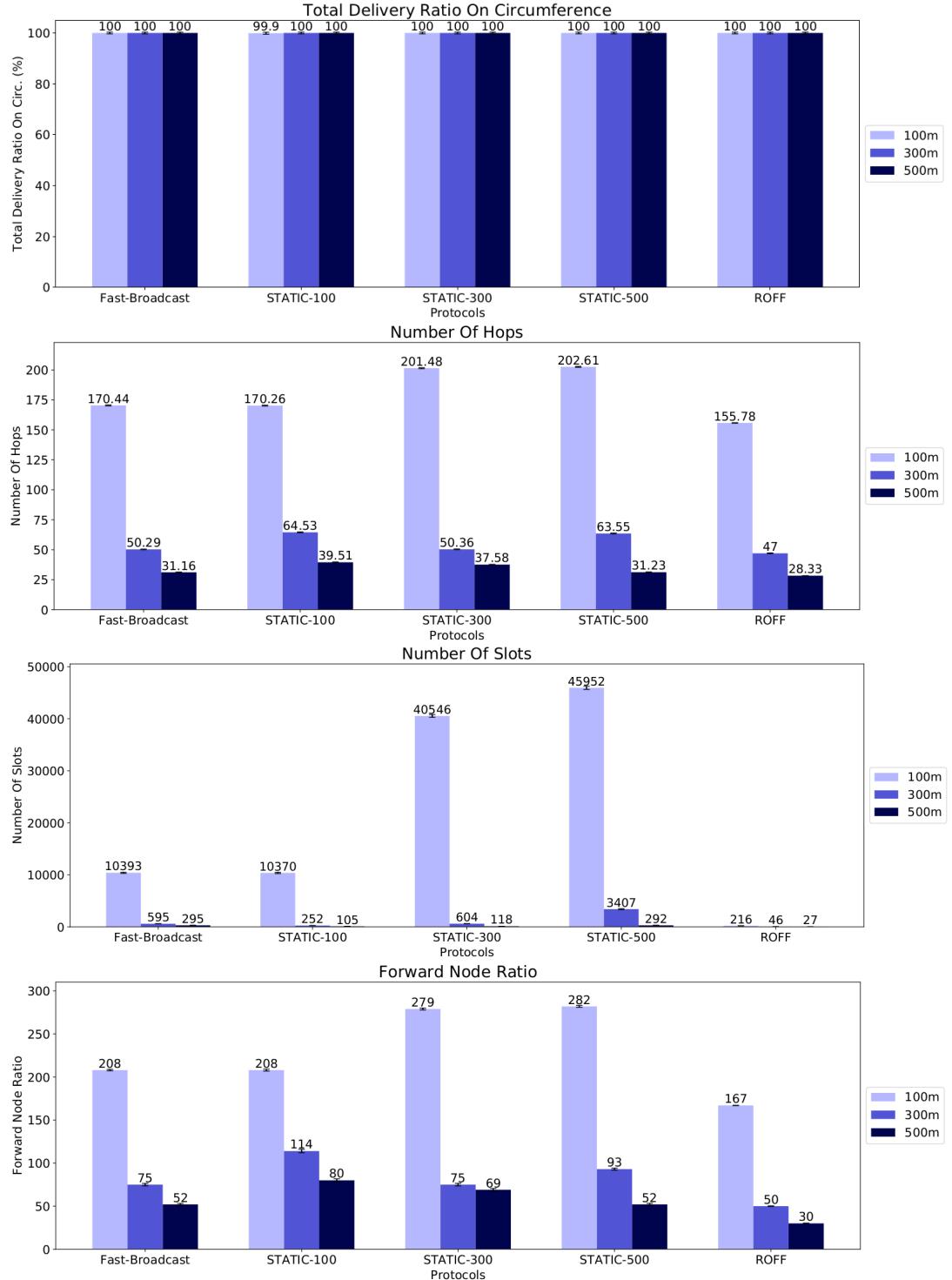


Figure 6.4: TDROC, NOH, NOS and FNN metrics for Platoon scenario

Since this scenario is one-dimensional, the circumference simply consists in vehicles distant  $14.000 \pm 25\text{m}$  from the source of Alert Message. Both metrics about delivery ratios (global and on circumference) are exactly 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 100m, 300m and 500m 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 at 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 100m 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 300m transmission range, compared to Fast-Broadcast with the same transmission range) or overestimated (e.g., STATIC-500 with 100m transmission range, compared to Fast-Broadcast with the same transmission range). This behaviour is expected, as reported in [30].

Regarding the Number of Slots, ROFF 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's 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.

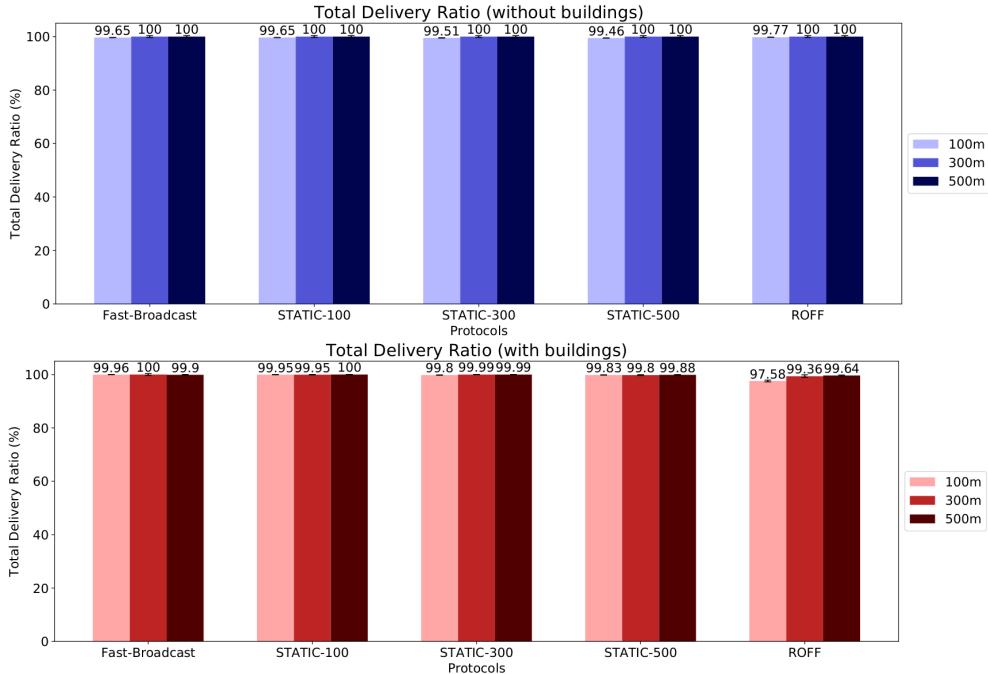
Lastly, it is possible to see that ROFF achieves a better suppression of redundant transmissions, guaranteeing a decrease of 19.71%, 33.33% and 42.31% respectively for each one of the transmission ranges considered.

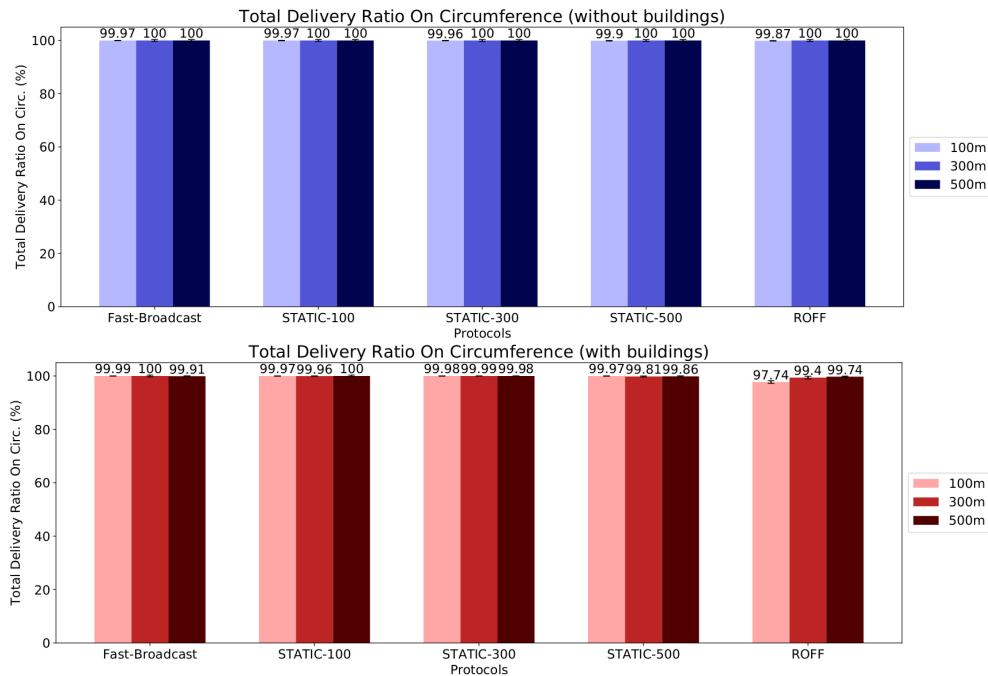
### 6.3 Grid scenario

After tests on the 1D Platoon scenario have been carried out, the next step consisted in testing the algorithms' performances in a two-dimensional Grid scenario, employing also the shadowing model introduced in 2.1. The Grid scenario consisted in a grid built by 17 north-south and 17 west-east roads, distant 300m from each other. Each road is 10m wide. Vehicles are placed only inside roads and are 25m distant from each other. The source of the Alert Message is in the middle of the grid, and the circumference has a radius of 2000m. For metrics calculation, vehicles are considered inside the circumference if they are  $2000 \pm 25\text{m}$  far from the source. Buildings are placed inside the blocks generated by road intersections, and are square shaped with an edge of 290m.

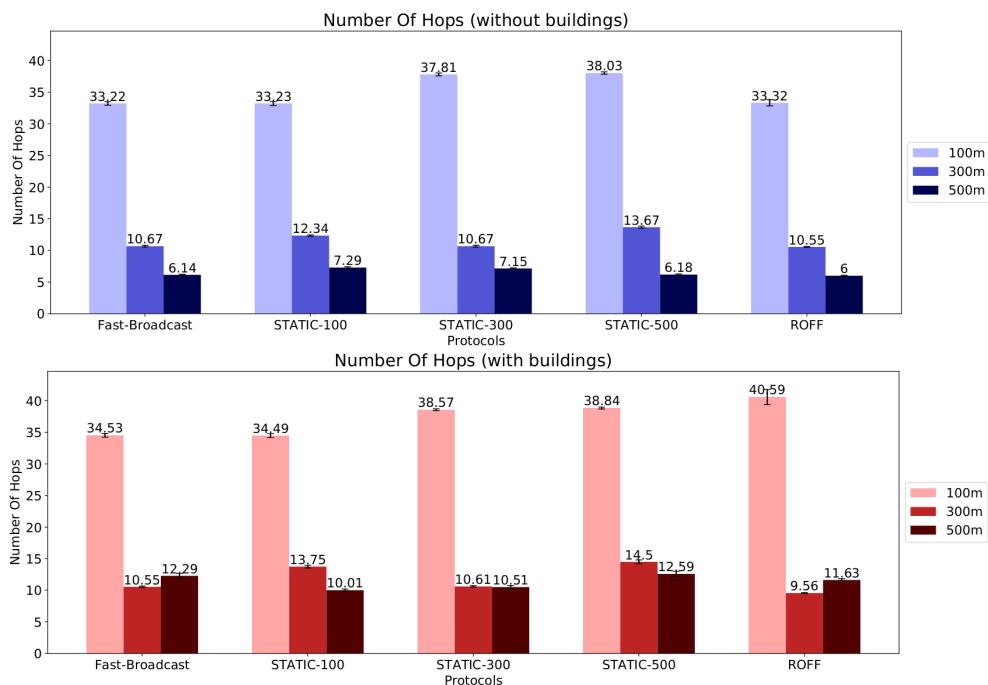
Parameters for this scenario are included in Table 6.4.

| 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             |   |
| Edge of buildings                       | 290                | m |
| Number of buildings                     | 255                | m |
| Simulator configuration                 |                    |   |
| Shadowing model                         | Obstacle Shadowing |   |
| Junction modeling                       | No                 |   |
| Number of simulations per configuration | 1000               |   |

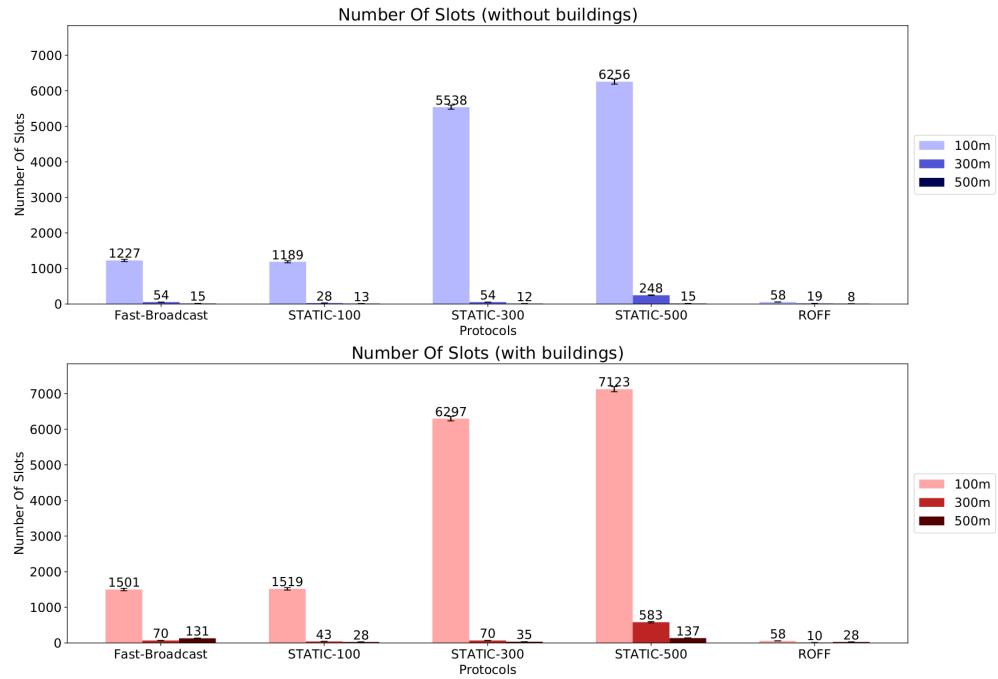
**Table 6.4:** Grid scenario configuration**Figure 6.5:** TDR without buildings (top) and with buildings (bottom) for Grid scenario



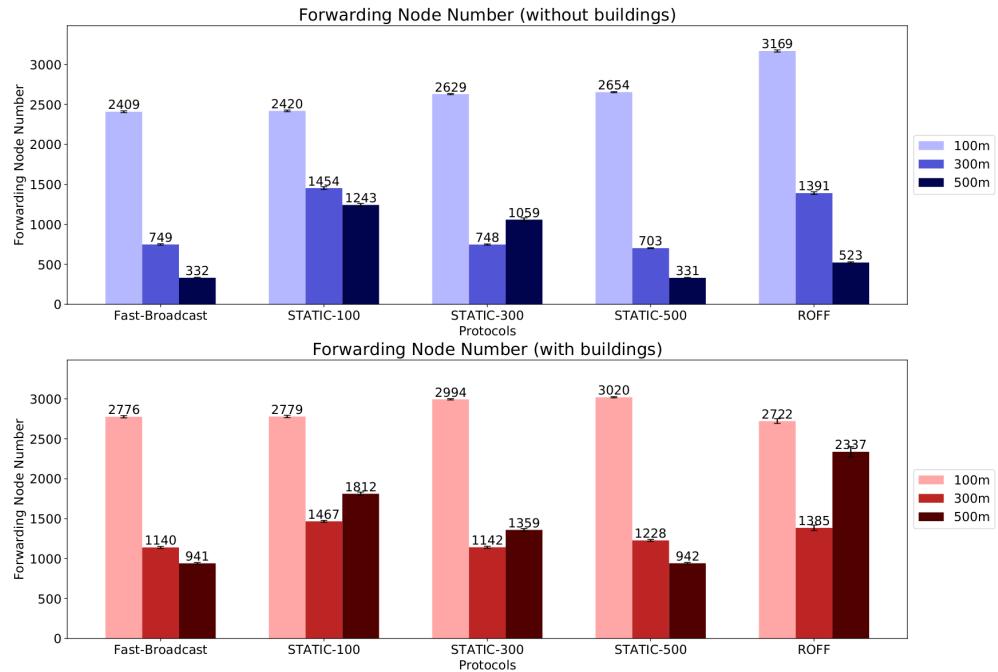
**Figure 6.6:** TDROC without buildings (top) and with buildings (bottom) for Grid scenario



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



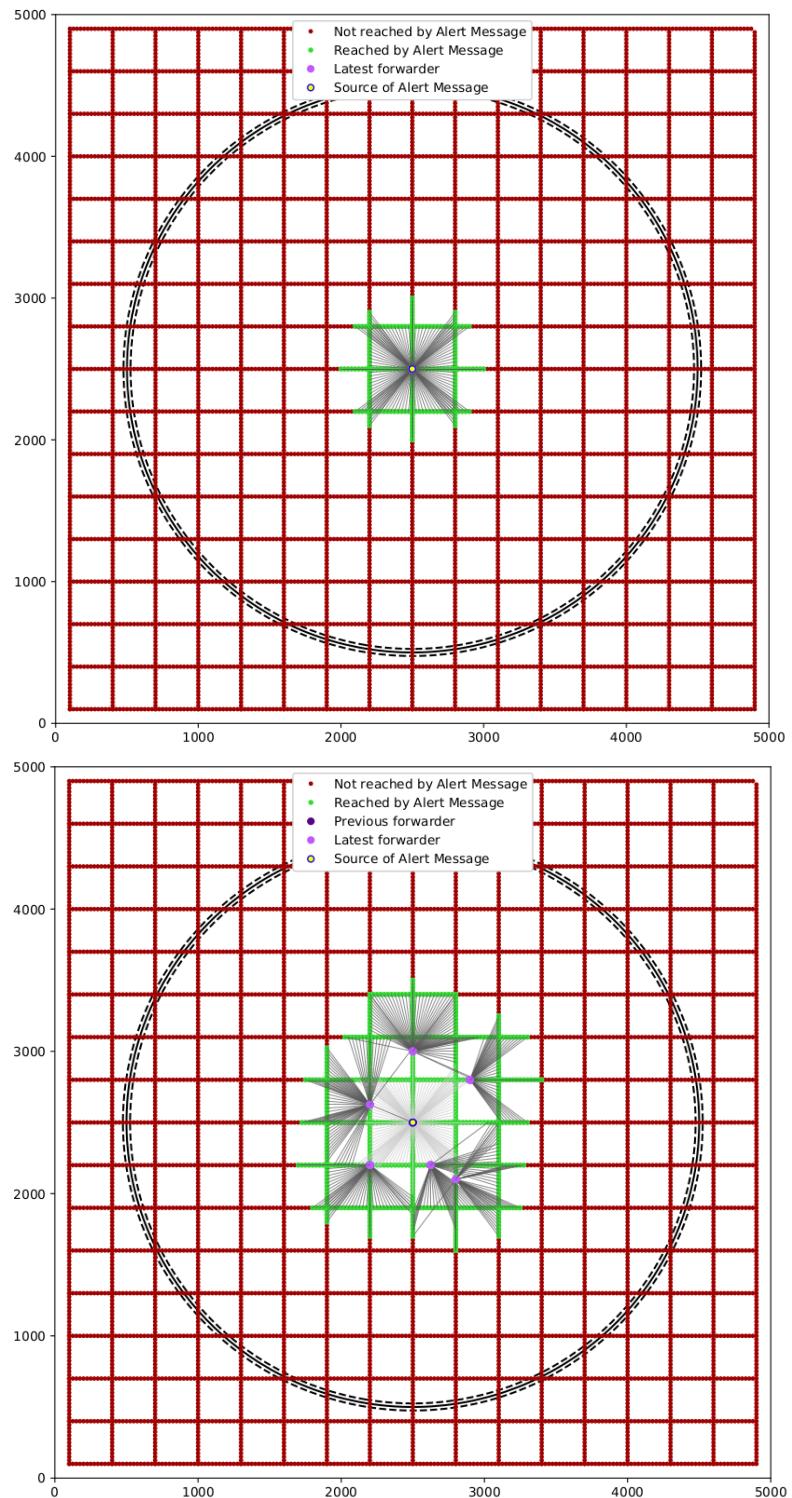
**Figure 6.8:** NOS without buildings (top) and with buildings (bottom) for Grid scenario



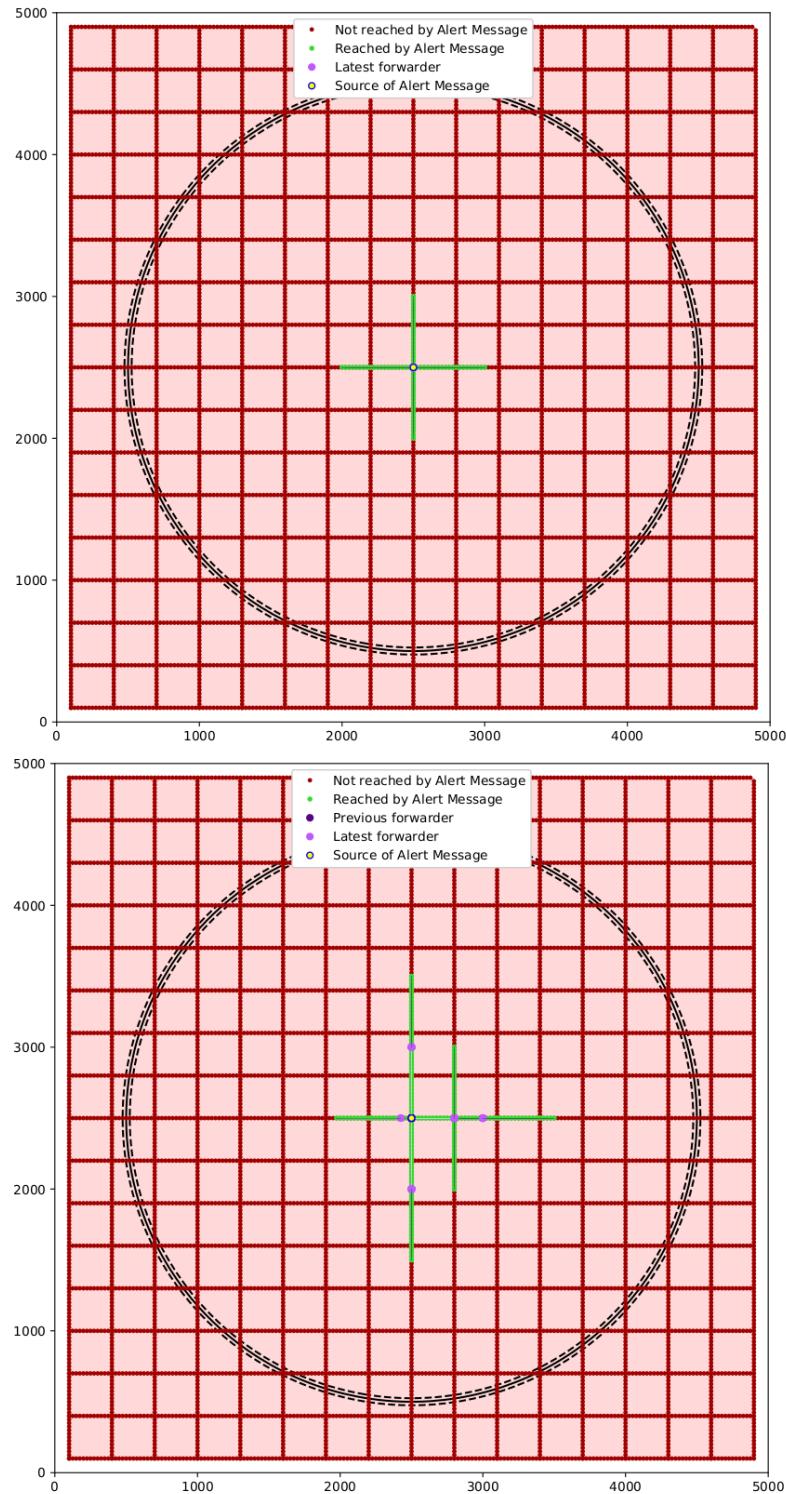
**Figure 6.9:** FNN without buildings (top) and with buildings (bottom) for Grid scenario

All the algorithms perform well as far as delivery ratios are concerned (Figure 6.5 and 6.6), for both configurations 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.

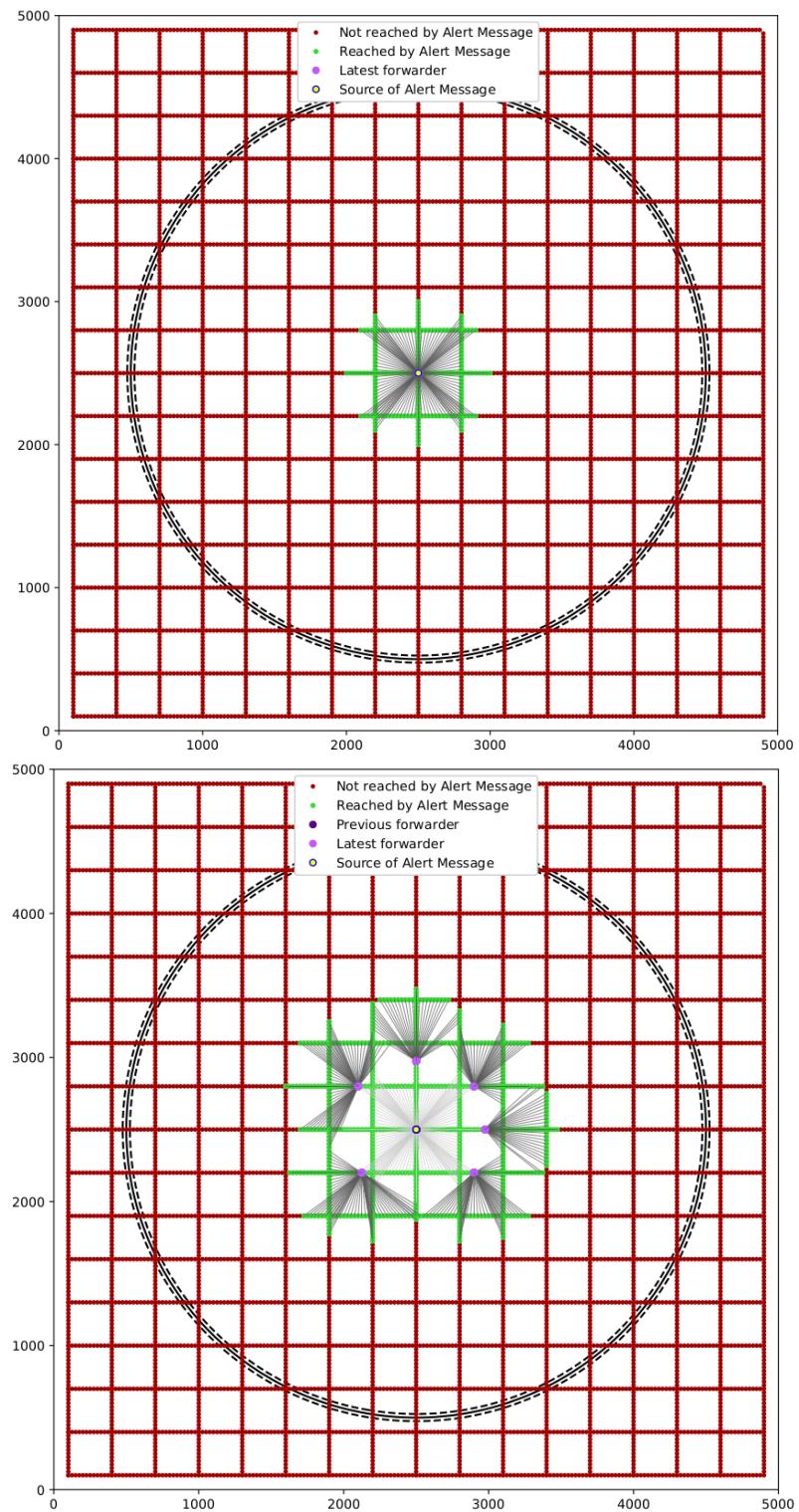
Moreover, Fast-Broadcast and ROFF can be compared with each other by looking at Figure 6.10 and 6.12 for the scenario without buildings, and Figure 6.11 and 6.13 for the scenario with buildings. In both cases, ROFF designates the farthest PFC as the next forwarder much more reliably than Fast-Broadcast 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.



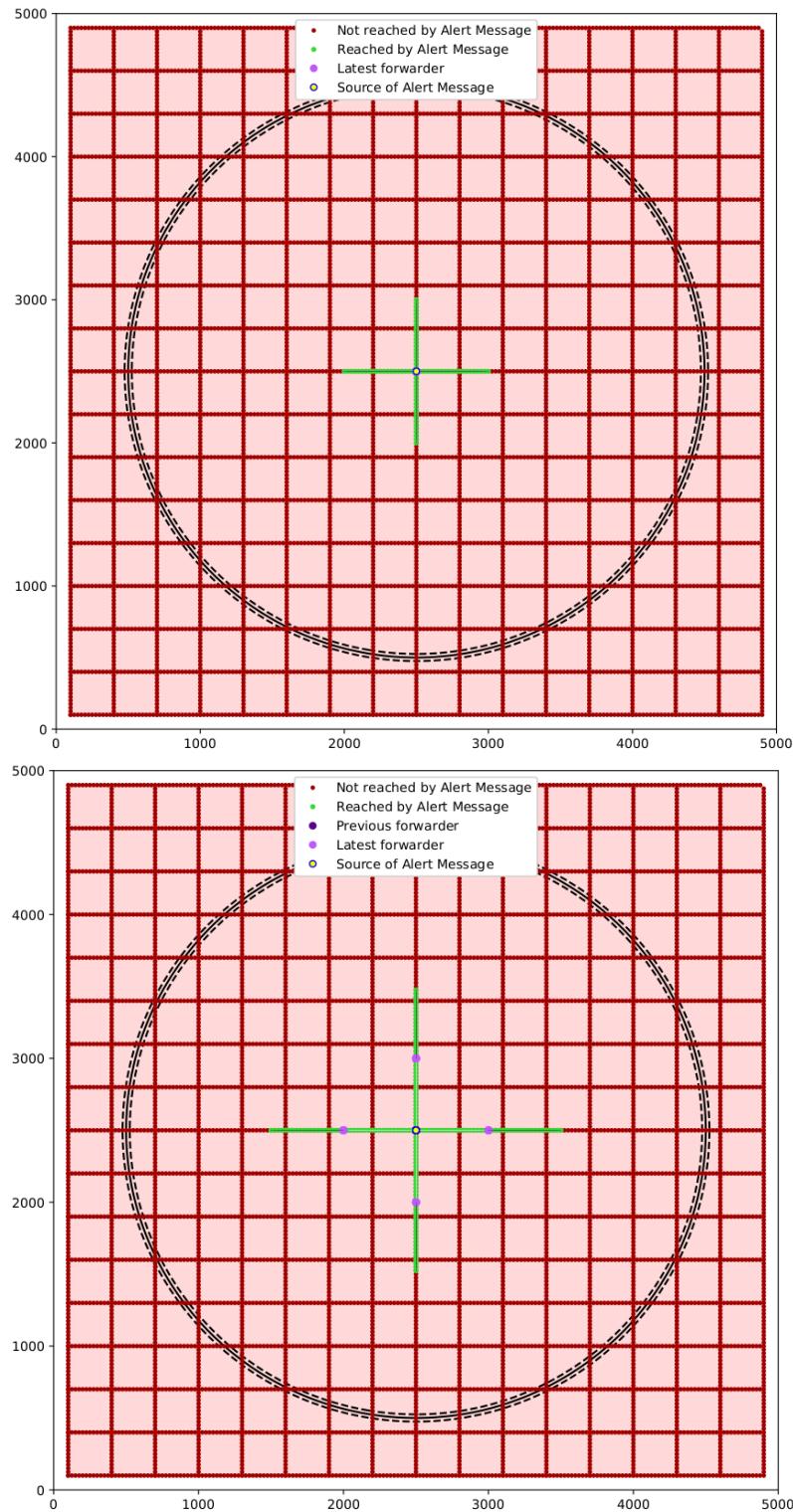
**Figure 6.10:** Fast-Broadcast after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500m transmission range and without the Obstacle model



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



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

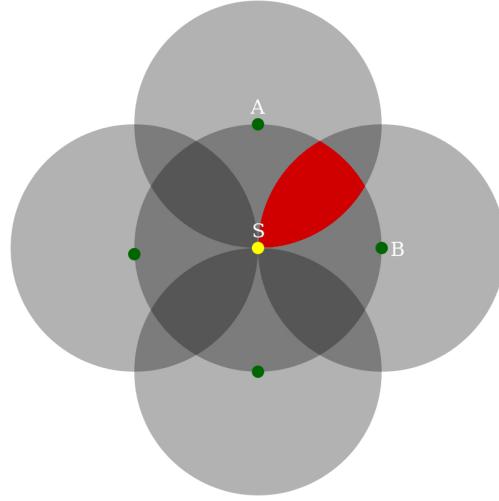


**Figure 6.13:** ROFF after 1 hop (top) and 2 hops (bottom) in Grid scenario with 500m transmission range and with the Obstacle model

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 100m and 500m, while decreases for 300m transmission range. This is probably due to the fact that the distance between roads is also 300m 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 observation consists in ROFF with 100m 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 500m 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 300m 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 observed 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 exactly the same time, hence if we focus on A and B we have a guaranteed 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).



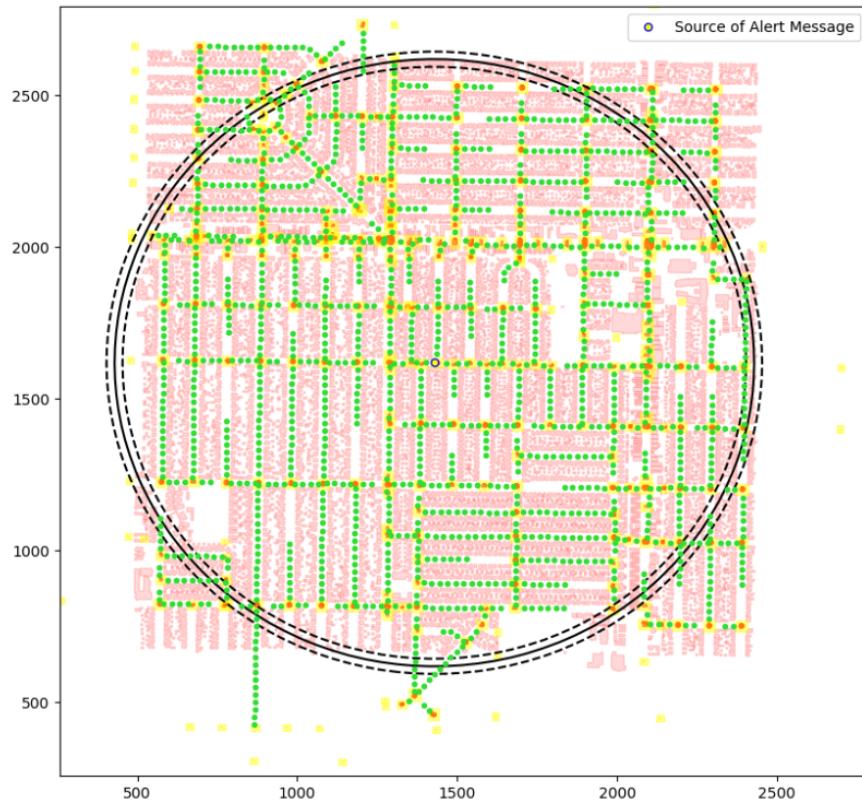
**Figure 6.14:** Collision area in 2D scenarios for ROFF

## 6.4 Los Angeles urban scenario

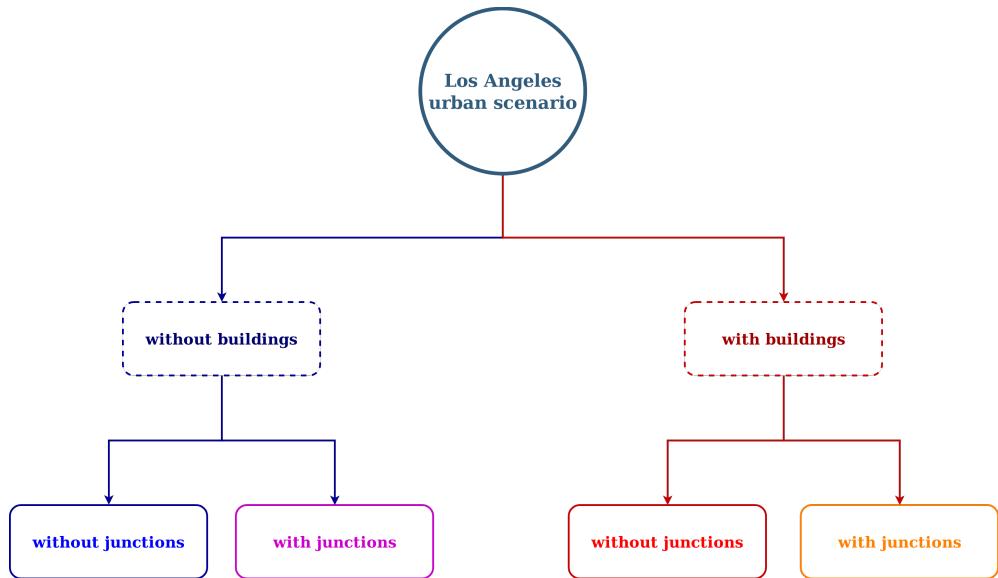
The next step consisted in expanding the 2D scenario with realistic urban data taken from OpenStreetMap and processed by SUMO as explained in Section 5.2. In addition to the shadowing model, in this scenario (and also in the urban scenario regarding Padua, which will be presented in the next section),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

Figure 6.15 depicts the scenario with vehicles (green dots), buildings (pink 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 reported 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.



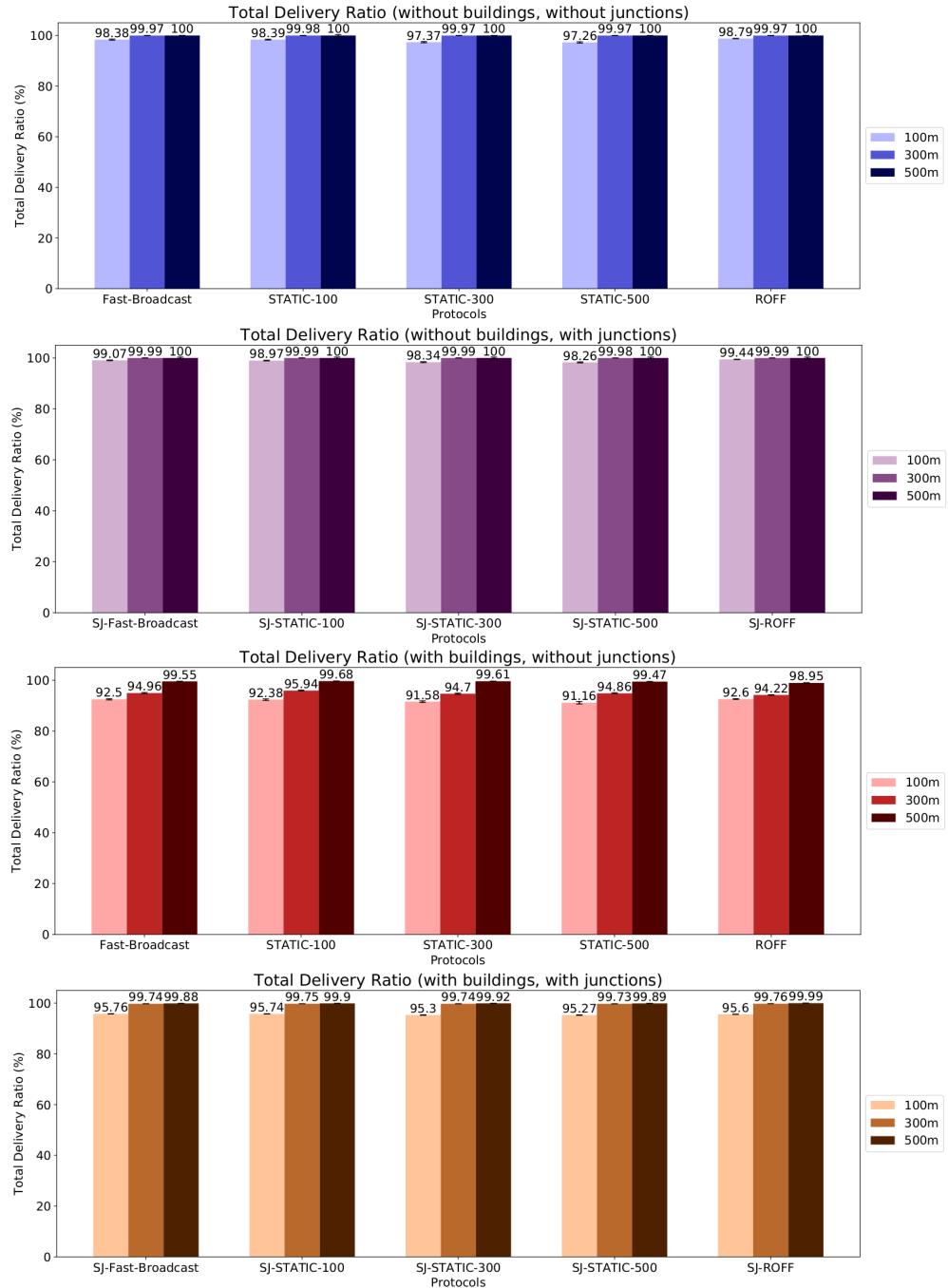
**Figure 6.15:** Los Angeles urban scenario depiction

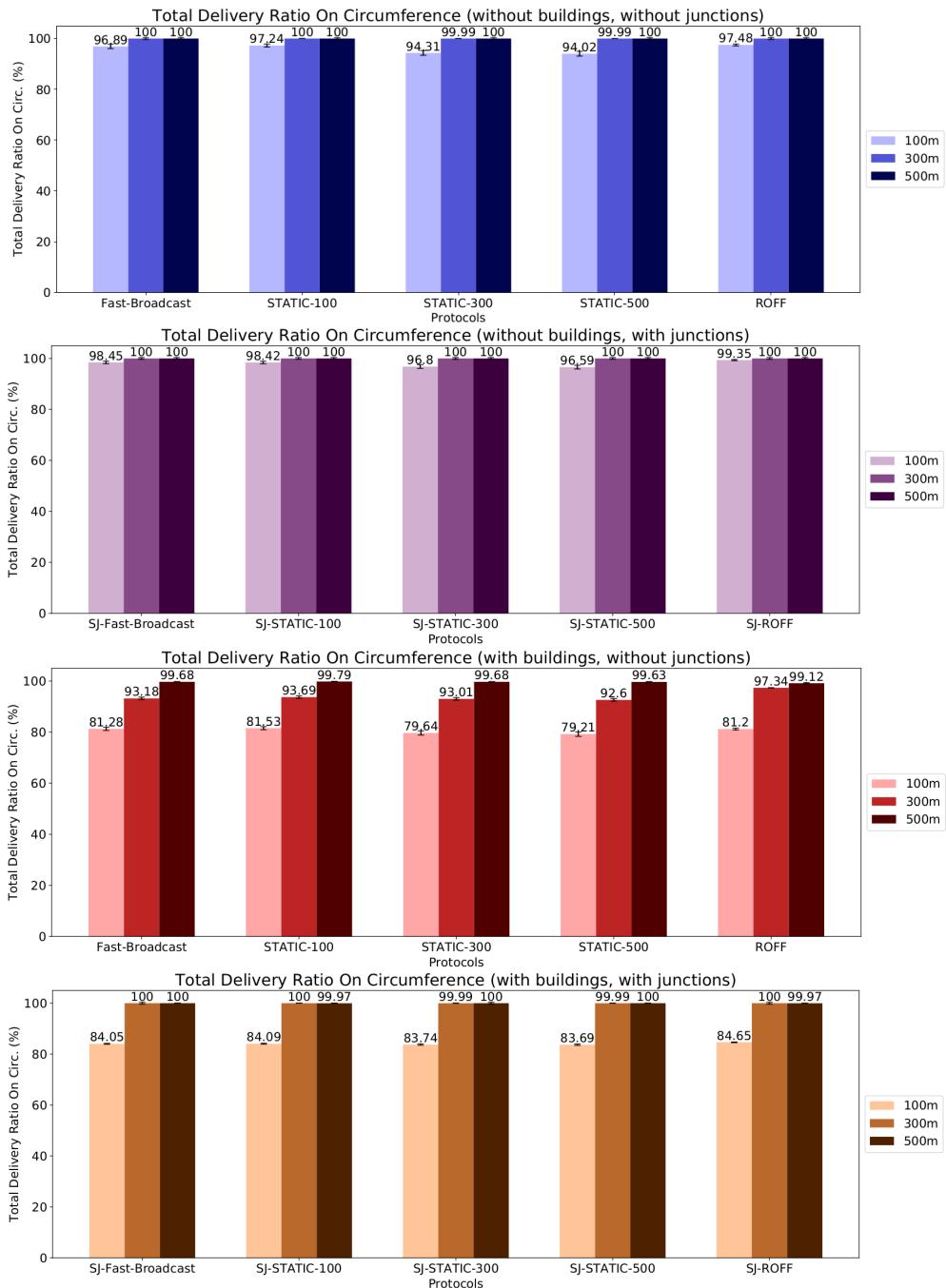


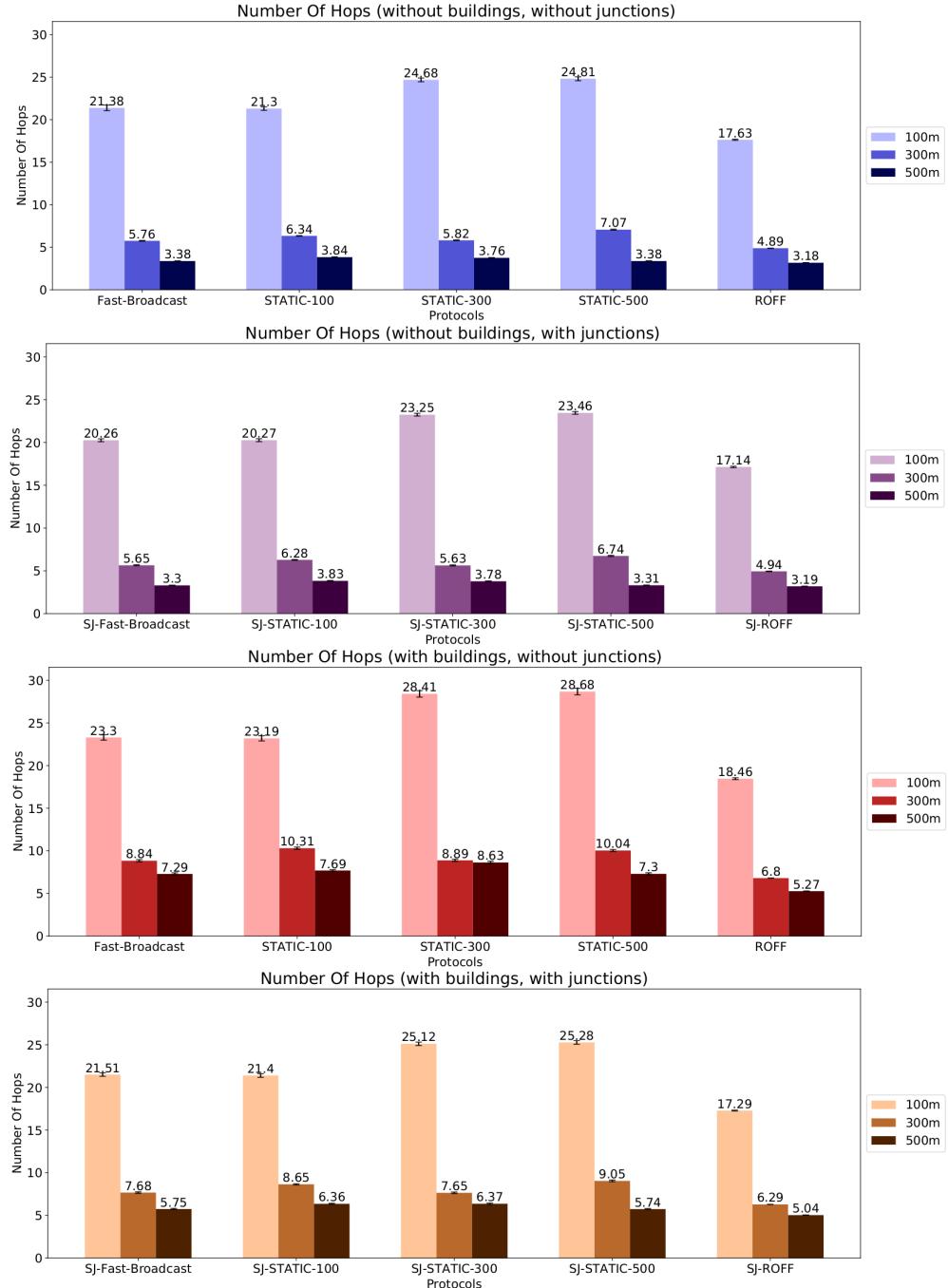
**Figure 6.16:** Los Angeles urban scenario test overview

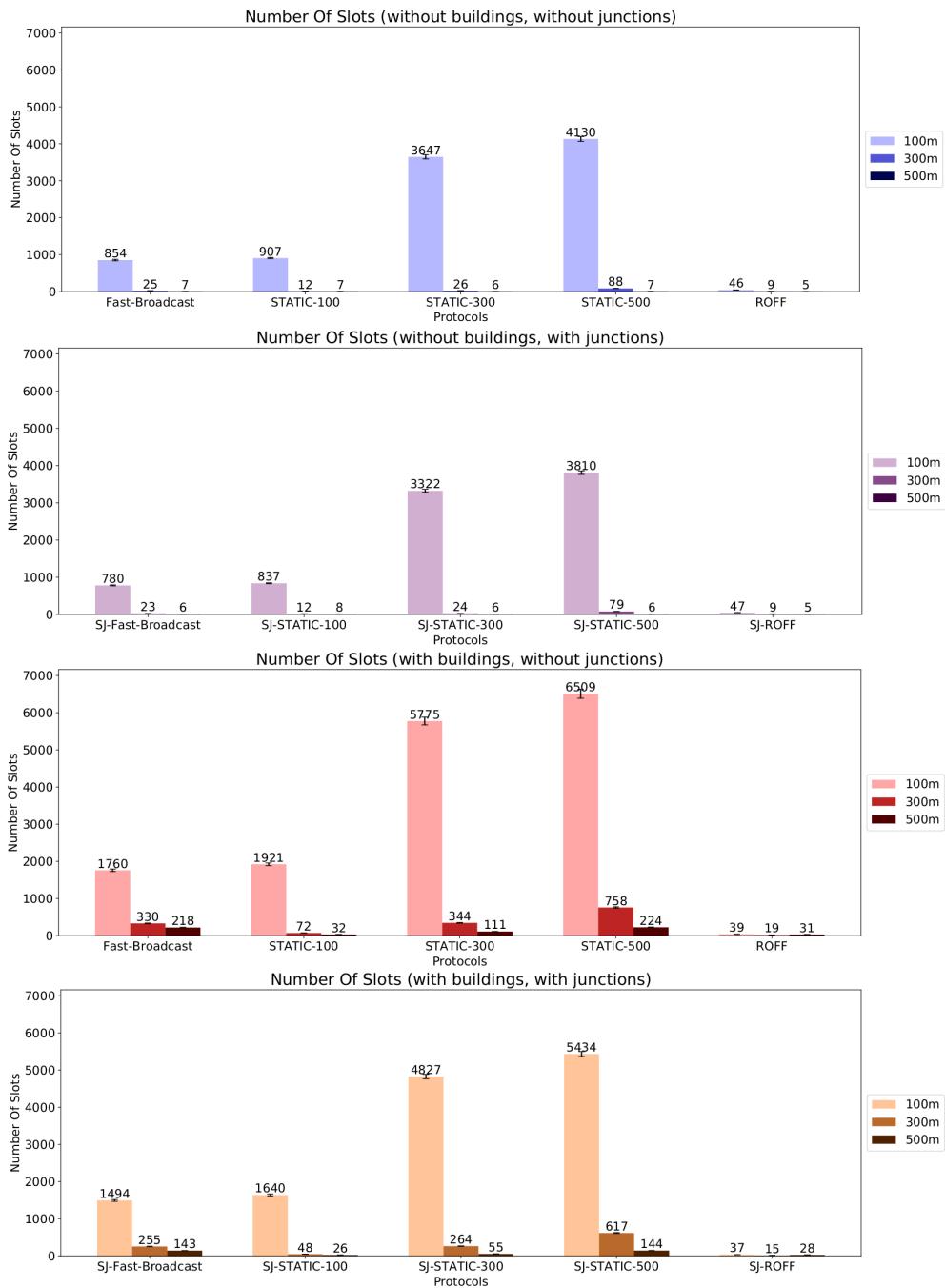
| Parameter                               | Value              |   |
|---|--------------------|---|
| Scenario configuration                  |                    |   |
| Latitude N                              | 33.9654            | ° |
| Latitude S                              | 33.9478            | ° |
| 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               |   |
| Number of junctions                     | 1288               |   |
| Simulator configuration                 |                    |   |
| Shadowing model                         | Obstacle Shadowing |   |
| Junction modeling                       | Yes                |   |
| Number of simulations per configuration | 4500               |   |

**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

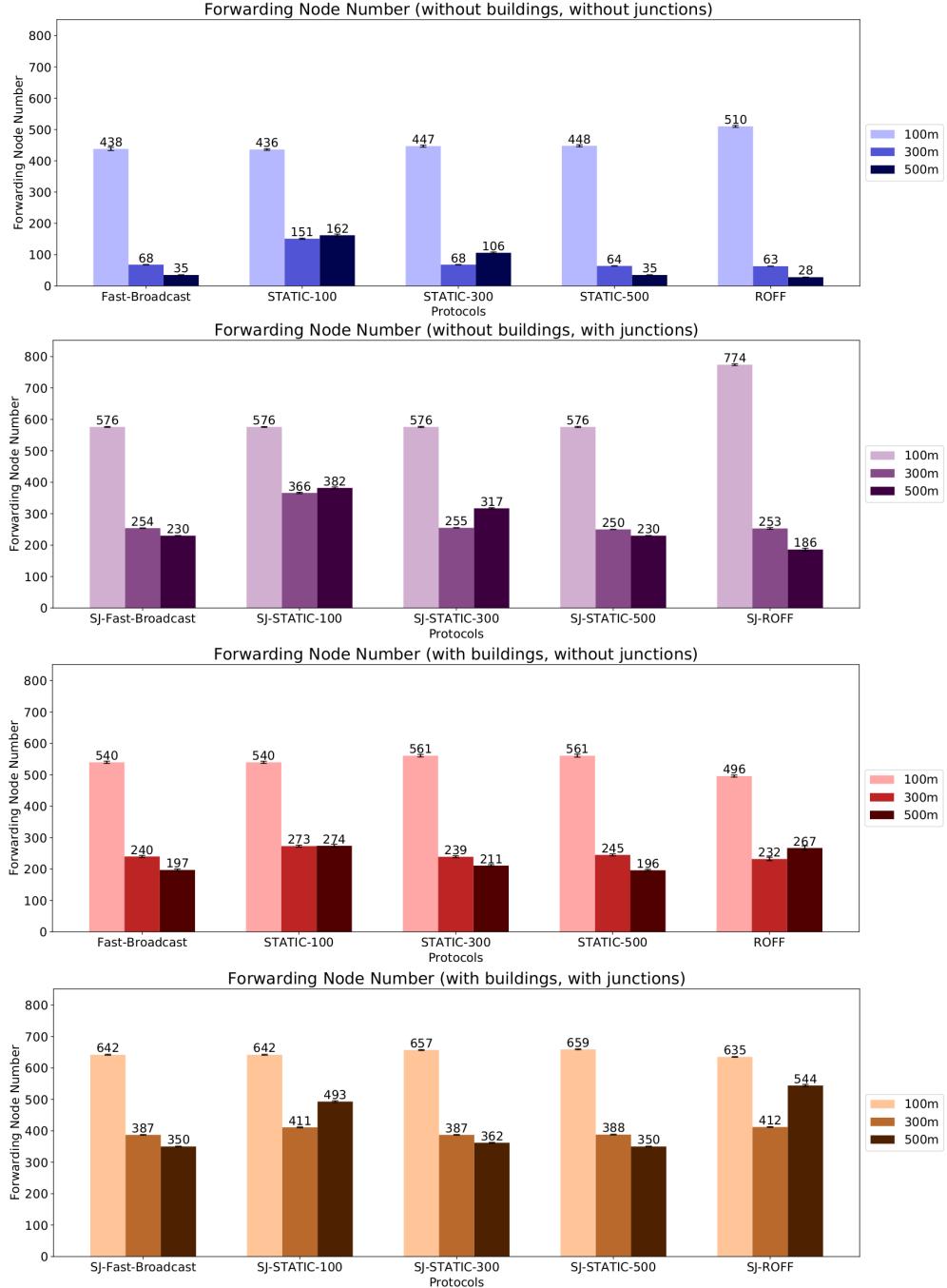
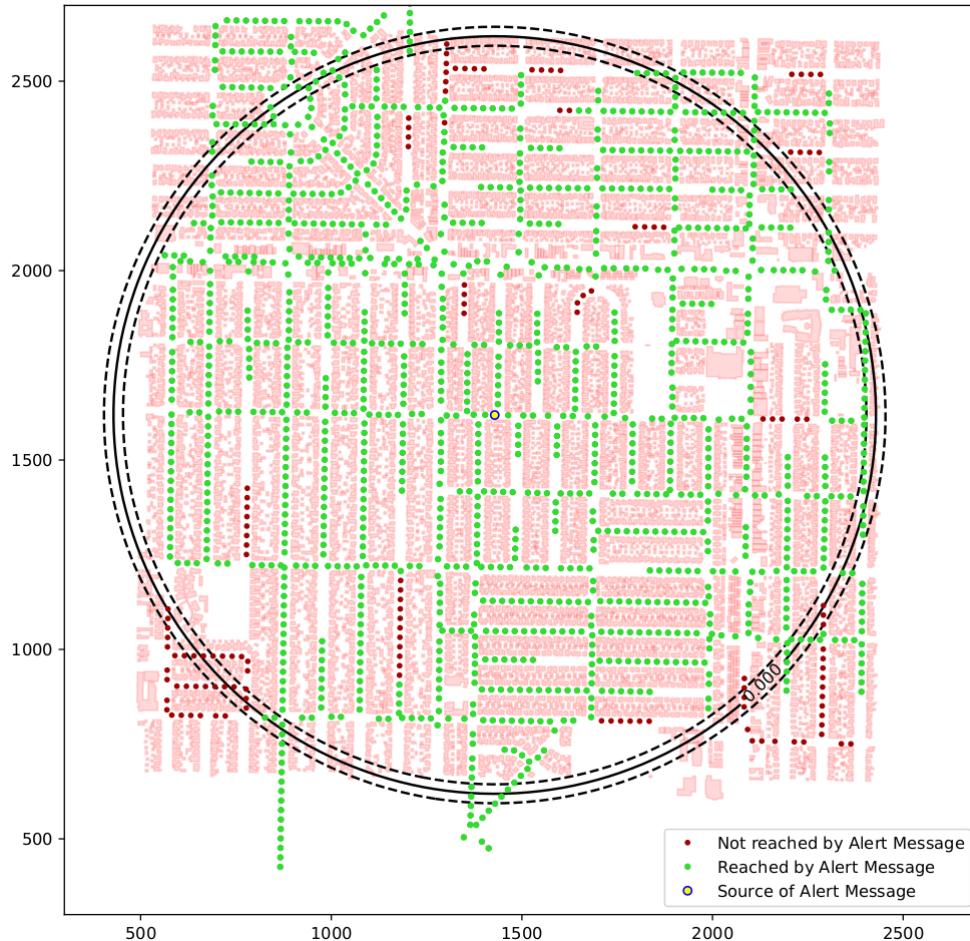


Figure 6.21: FNN for Los Angeles urban scenario

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 (100m and 300m). 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 100m and a decrease of 5% (99.96 to 94.96) with transmission range of 300m 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 100m 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 farther 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 100m 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 100m and 300m 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 300m and 500m transmission ranges, while having a

mild increase of around 3.40% for 100m transmission range.

Considering the Number Of Hops to reach the circumference (Figure 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 ROFF's Neighbor Table mechanism 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.

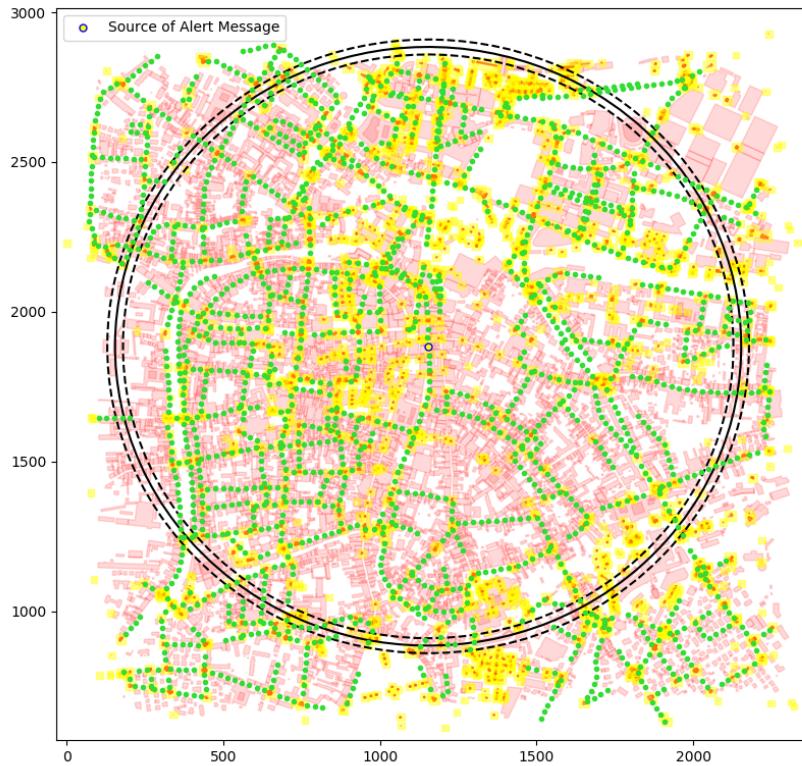
Regarding *NOS* (Figure 6.20), Fast-Broadcast results increase by 106.09%, 1220% and 3014% while ROFF's ones increase by 111.11% and 520% for 300 and 500m transmission range, and decrease by 4.44% for 100m transmission range, when going from a building-less scenario to a scenario with buildings. Moreover, 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 whose distance is 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 100m 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 100m transmission range, and with much greater *FNN* values in 300 and 500m 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 better than ROFF 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               |
| Number of junctions                     | 3231               |
| Simulator configuration                 |                    |
| Shadowing model                         | Obstacle Shadowing |
| Junction modeling                       | Yes                |
| Number of simulations per configuration | 4500               |

**Table 6.8:** Los Angeles urban scenario configuration

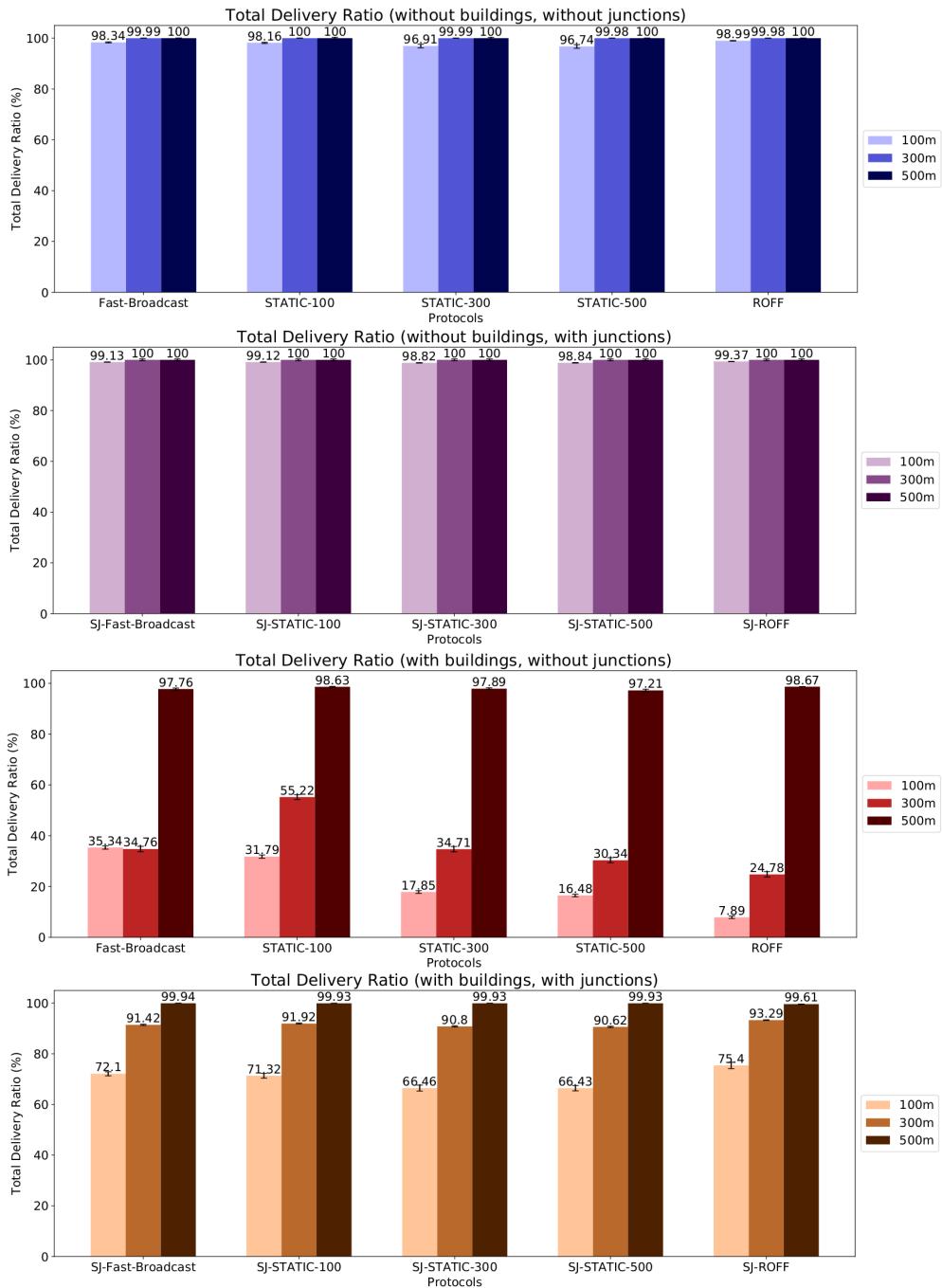
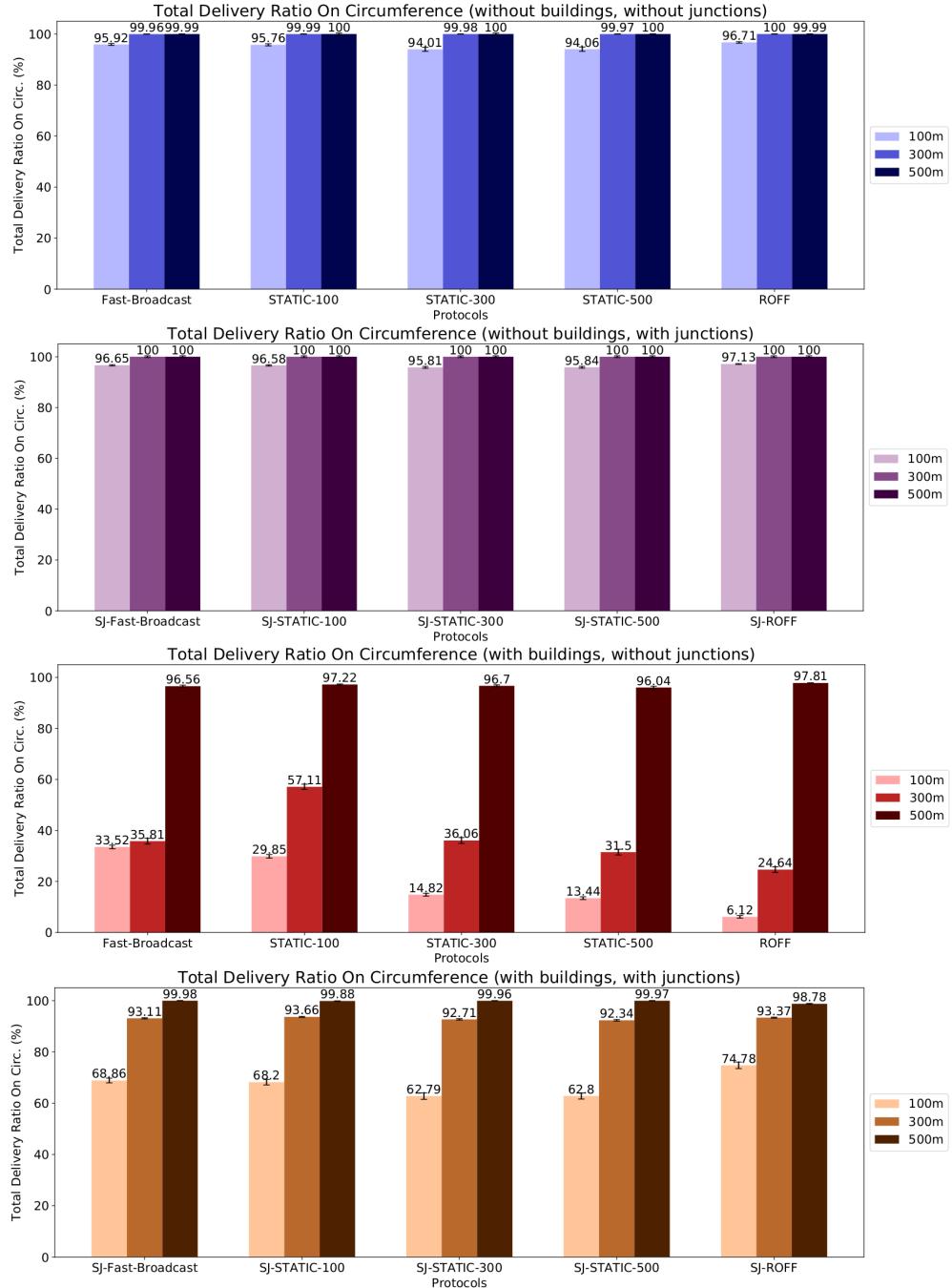
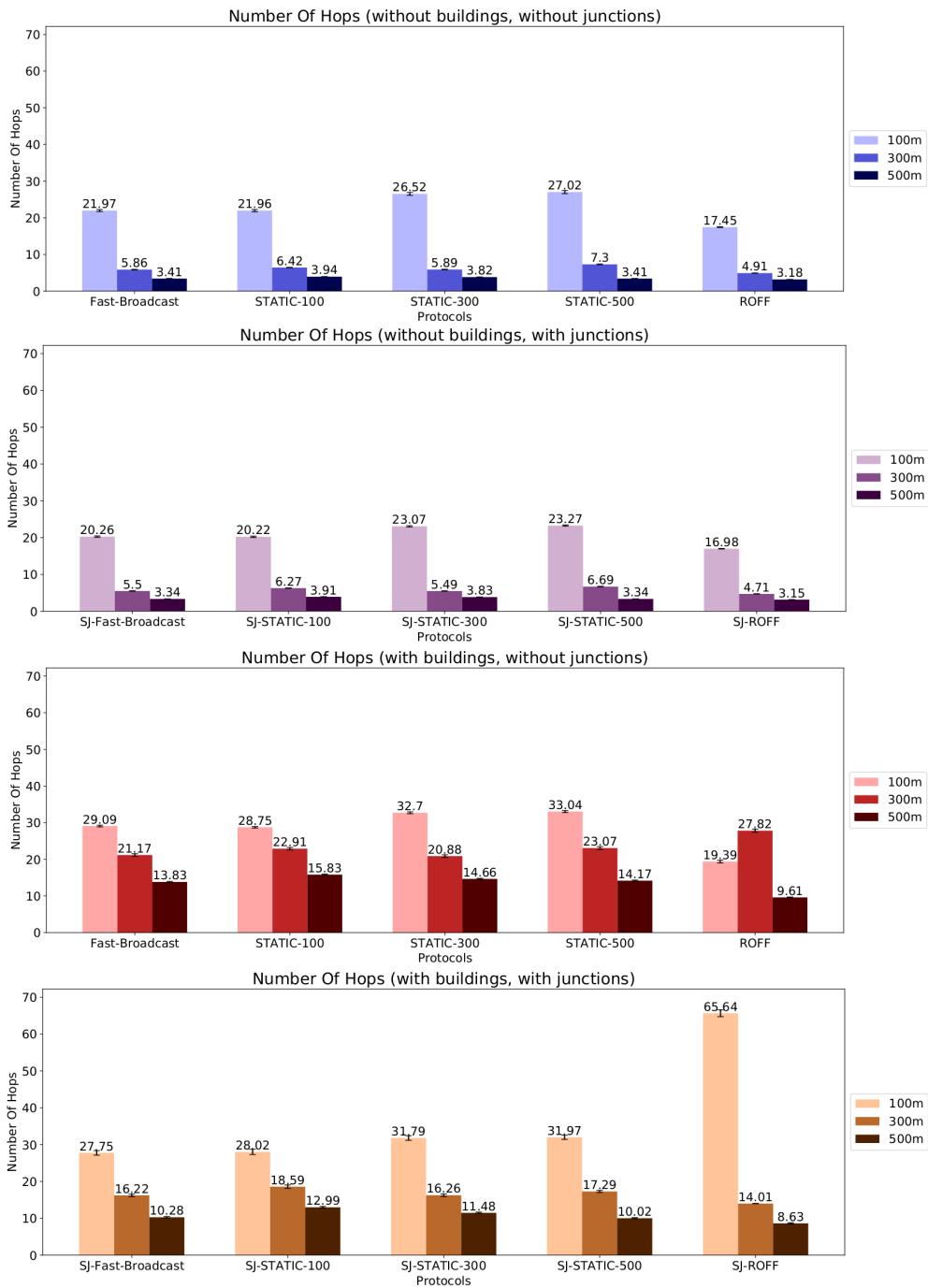
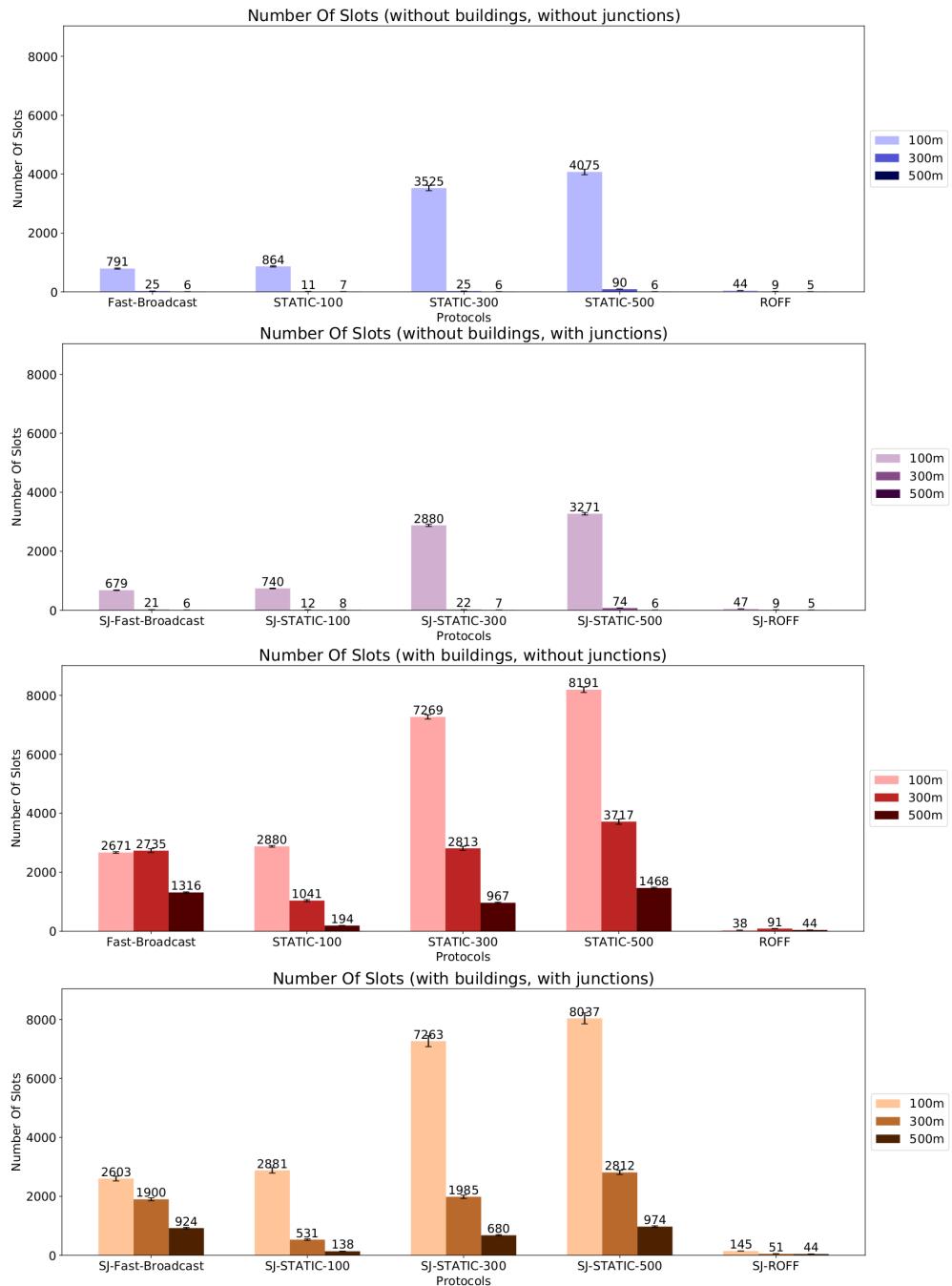


Figure 6.24: TDR for Padua urban scenario

**Figure 6.25:** TDROC for Padua urban scenario

**Figure 6.26:** NOH for Padua urban scenario



**Figure 6.27:** NOS for Padua urban scenario

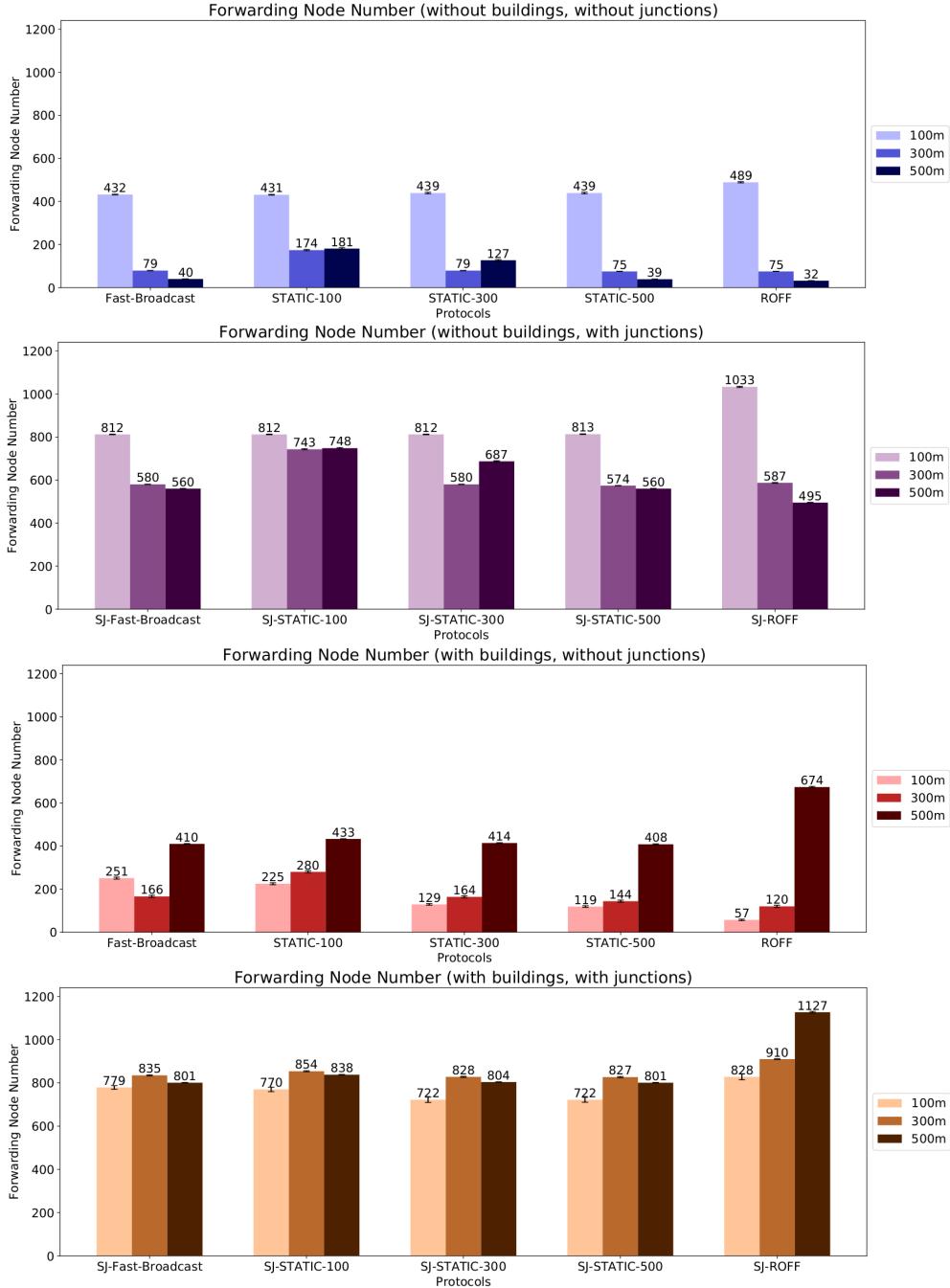


Figure 6.28: FNN for Padua urban scenario

In this scenario the effects of obstacle shadowing on delivery ratios is quite large, especially for the 300m and 500m cases, for all algorithms. We have a decrease in TDR (Figure 6.24) of around 65% for Fast-Broadcast in both transmission ranges when introducing buildings (first and third images), which results in quite poor performances. The Smart Junction variants of Fast-Broadcast manages to reach pretty good results in

Total Delivery Ratio (fourth image), with 72.1% and 91.42% for the two transmission ranges. ROFF's results are even worse before the introduction of junctions, with *TDR* below 10% for the lowest transmission range. Even in this case, SJ-ROFF manages to improve the values, slightly overcoming SJ-Fast-Broadcast's results.

The same observations hold true even for *TDROC* (Figure 6.24). Hence, the benefits of junctions are twofold even for this scenario regarding delivery ratios.

Concerning *NOS*, the increase due to the obstacle model is even more pronounced than the previous scenario for both algorithms. Due to the lower coverage, it is possible that the few messages which reach the circumference go through a very irregular path and one-hop progress is quite low. The introduction of junctions, as in the Los Angeles scenario, causes a decrease in *NOH*. One anomalous result is ROFF with 100m transmission range, whose values shoots up by more than 232% when comparing it with SJ-ROFF, and is much higher than SJ-Fast-Broadcast under the same conditions. This phenomenon might be due to the harsher vehicle distribution, obstacle and junction positioning which cause the signal to follow very long paths from the source to the circumference. With the other two transmission ranges, SJ-ROFF outperforms SJ-Fast-Broadcast.

Results for *NOH* and *FNN* are comparable with the ones observed in previous scenarios.

## 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”[32]. 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.

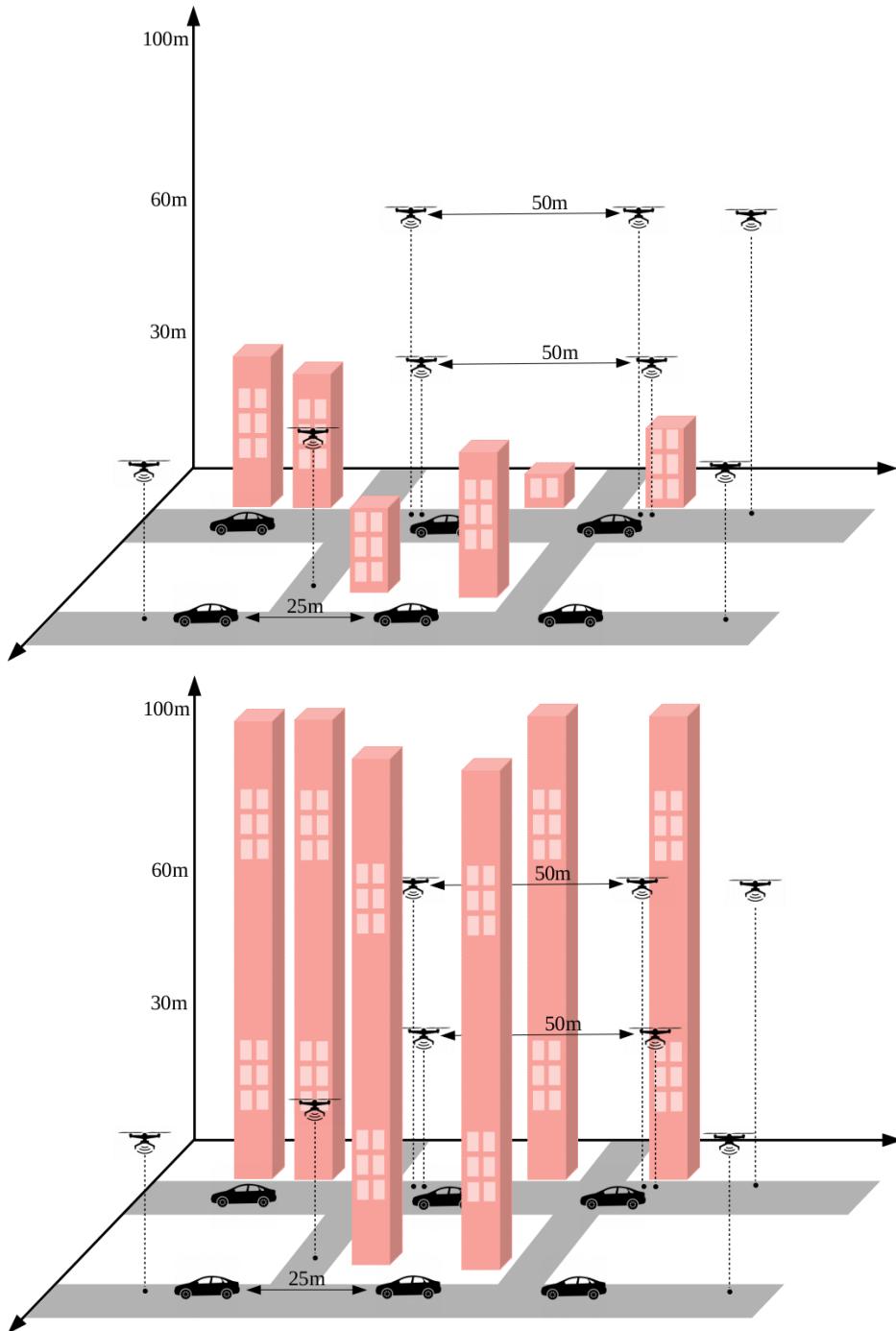
This scenario is built upon the Los Angeles urban scenario presented in Section 6.4. Two layers of drones were added to the ground level of vehicles:

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

Drones are more spaced from one another than vehicles, with an average distance of 50m. 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 21.9m, 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 100m).

One important fact to notice is that, even if we have called the object placed inside the two layers “drones”, this scenario applies also if those objects were other kinds of network connected entities, such as customers' access points or IoT devices.

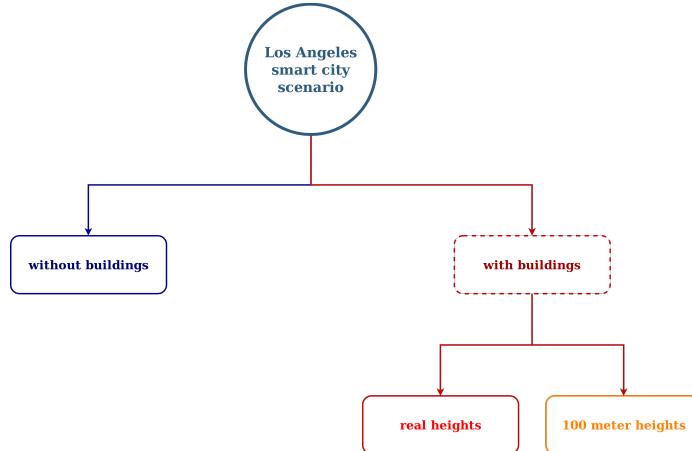
Figure 6.29 shows a representation of the two scenarios with buildings.



**Figure 6.29:** Los Angeles smart city scenario with real height buildings (top) and very high buildings (bottom)

In this scenario, 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.

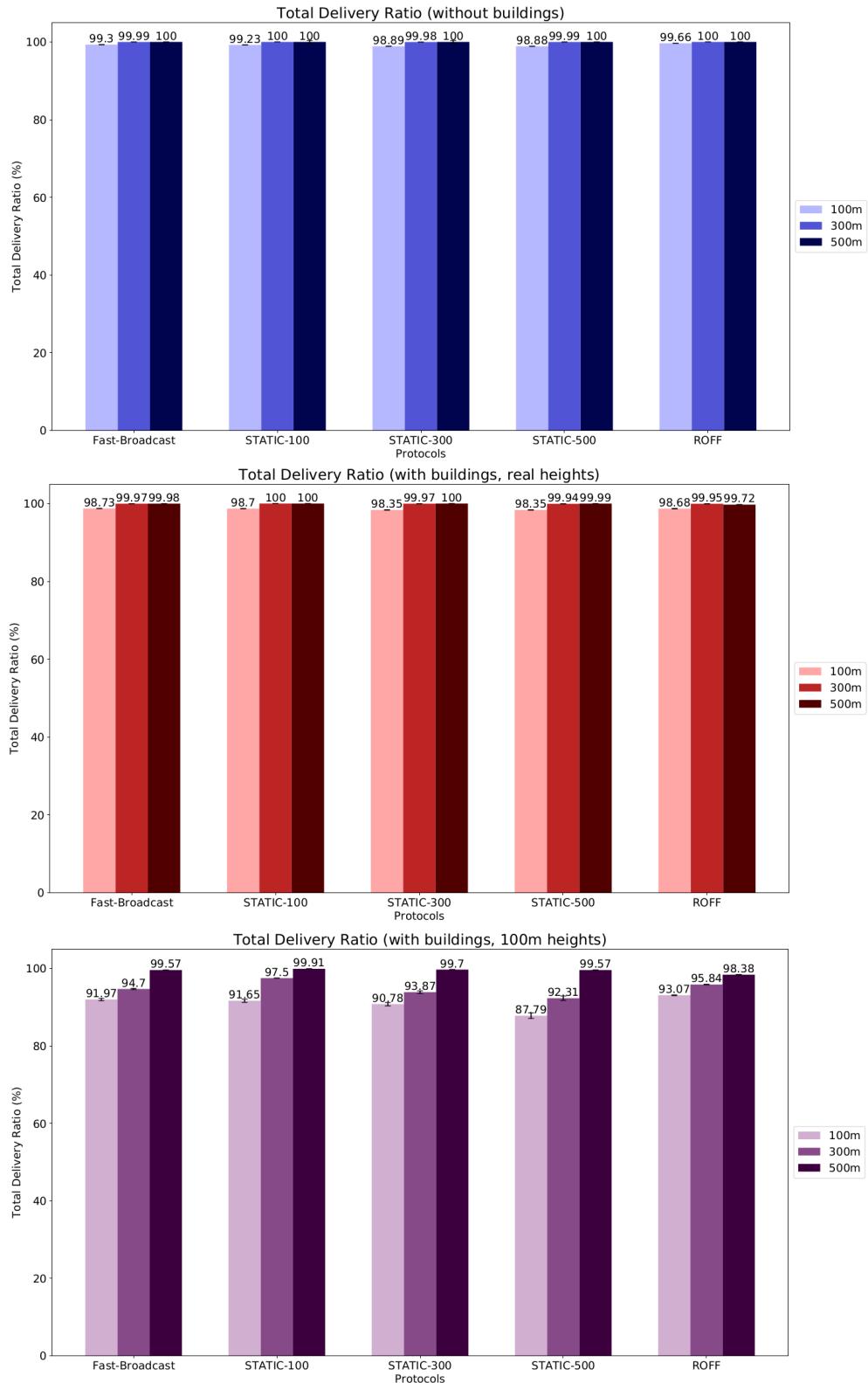
All configurations are reported in Figure 6.30, whose colors will guide the following graph results. Table 6.10 reports all the scenario settings.

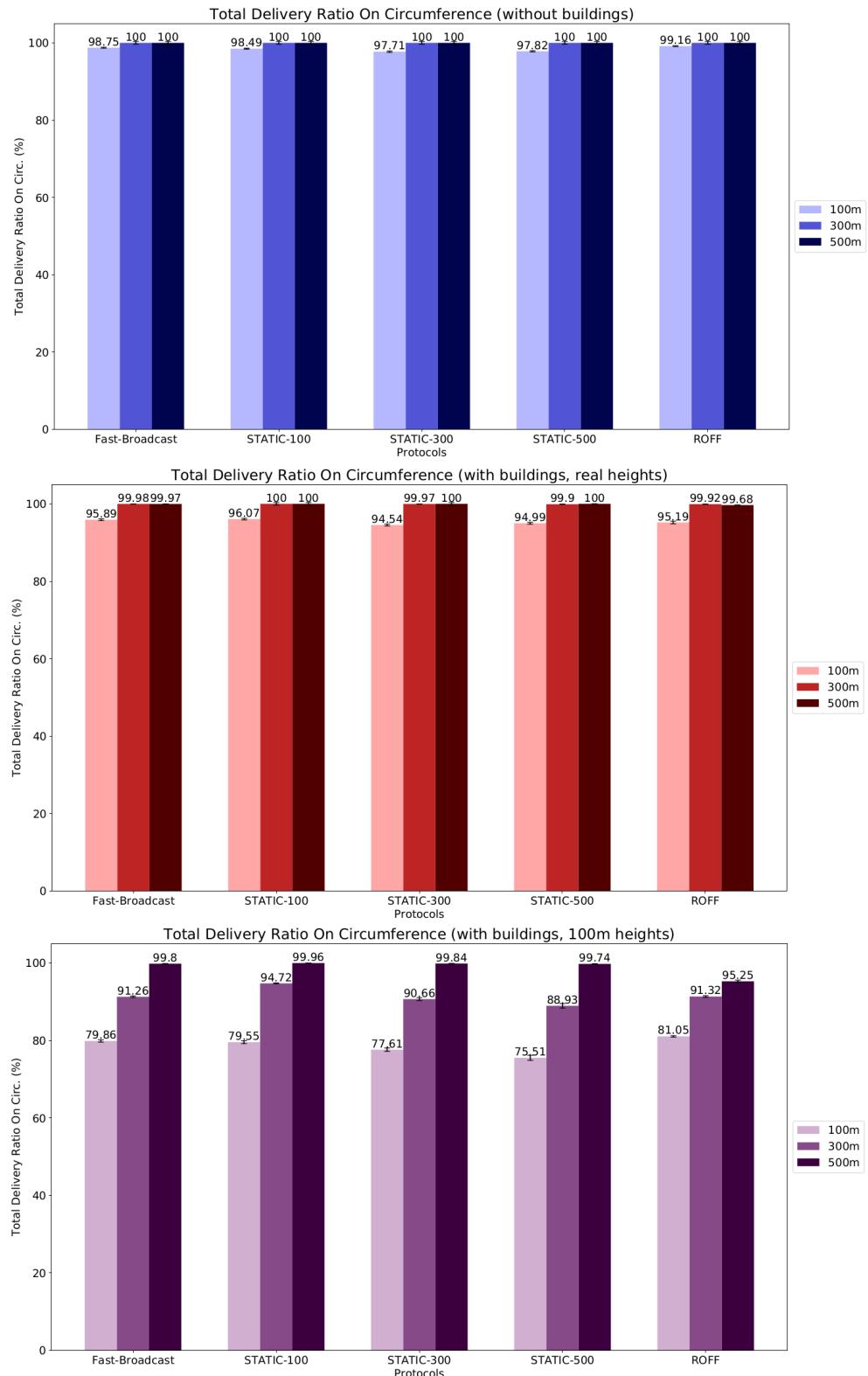


**Figure 6.30:** Test overview of Los Angeles smart city scenario

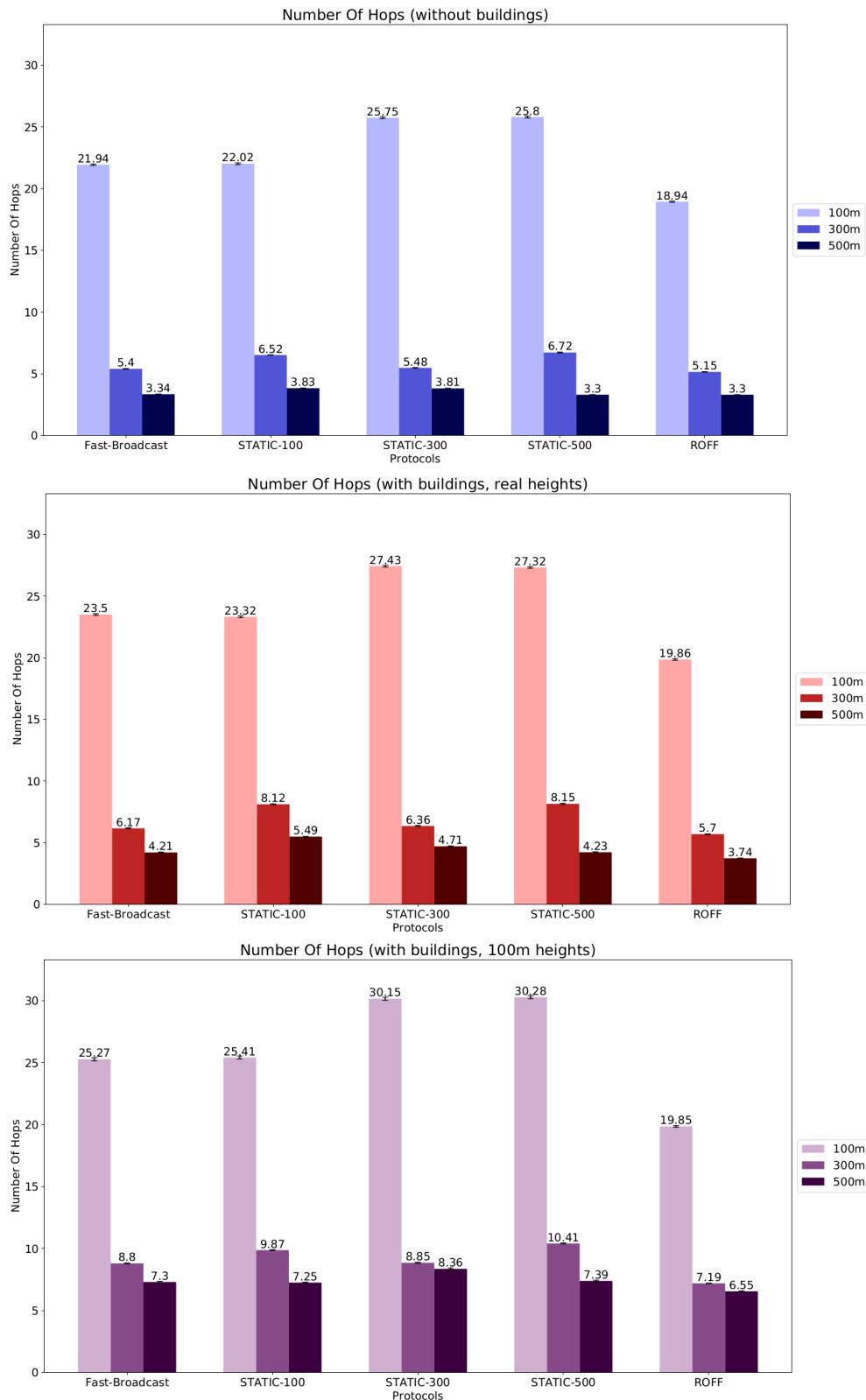
| Parameter                               | Value              |   |
|---|--------------------|---|
| Scenario configuration                  |                    |   |
| Circumference radius                    | 1000               | m |
| Number of vehicles                      | 1465               |   |
| 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 |
| Simulator configuration                 |                    |   |
| Shadowing model                         | Obstacle Shadowing |   |
| Junction modeling                       | No                 |   |
| Number of simulations per configuration | 1000               |   |

**Table 6.10:** Los Angeles urban scenario configuration

**Figure 6.31:** TDR for Los angeles smart city scenario



**Figure 6.32:** TDROC for Los angeles smart city scenario

**Figure 6.33:** NOH for Los angeles smart city scenario

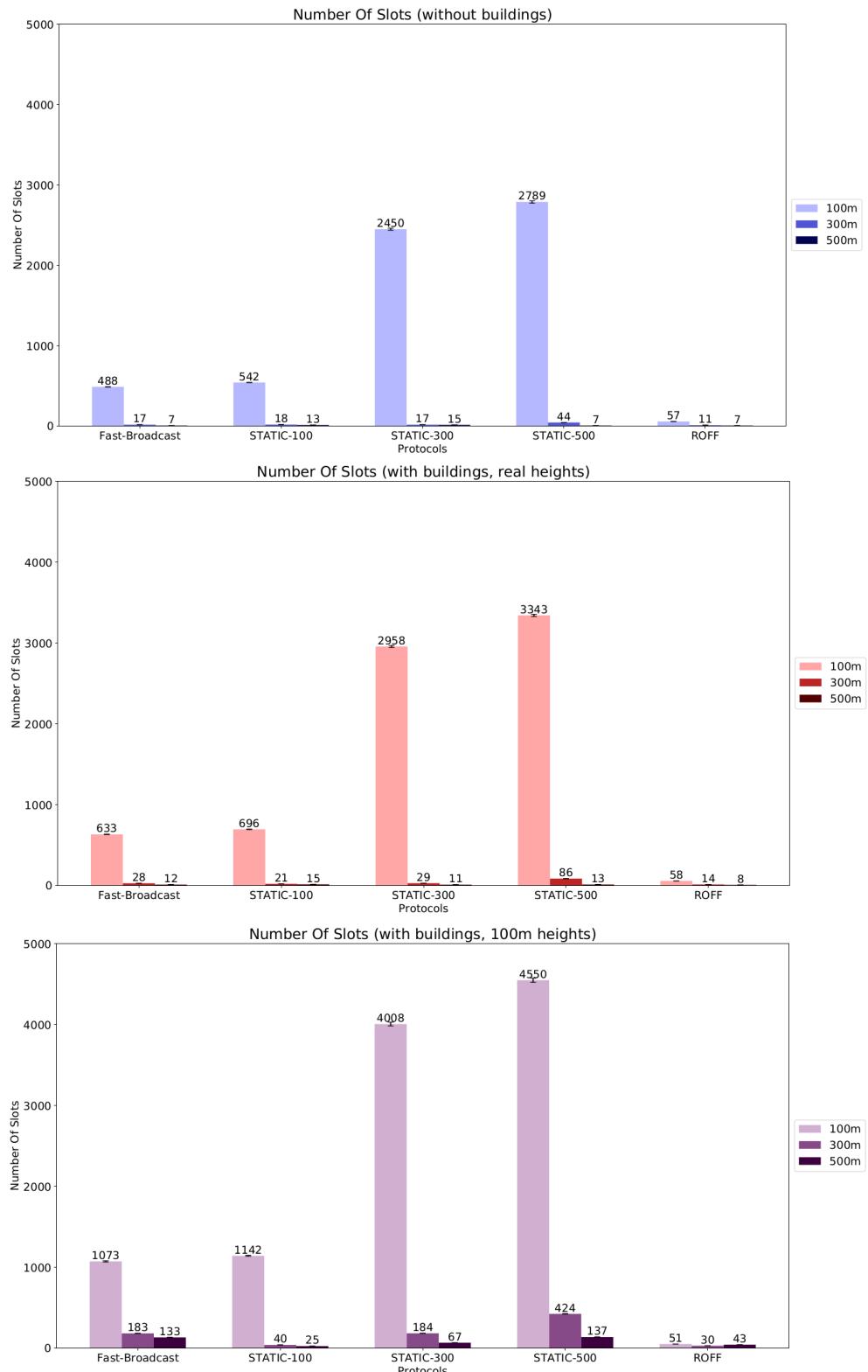
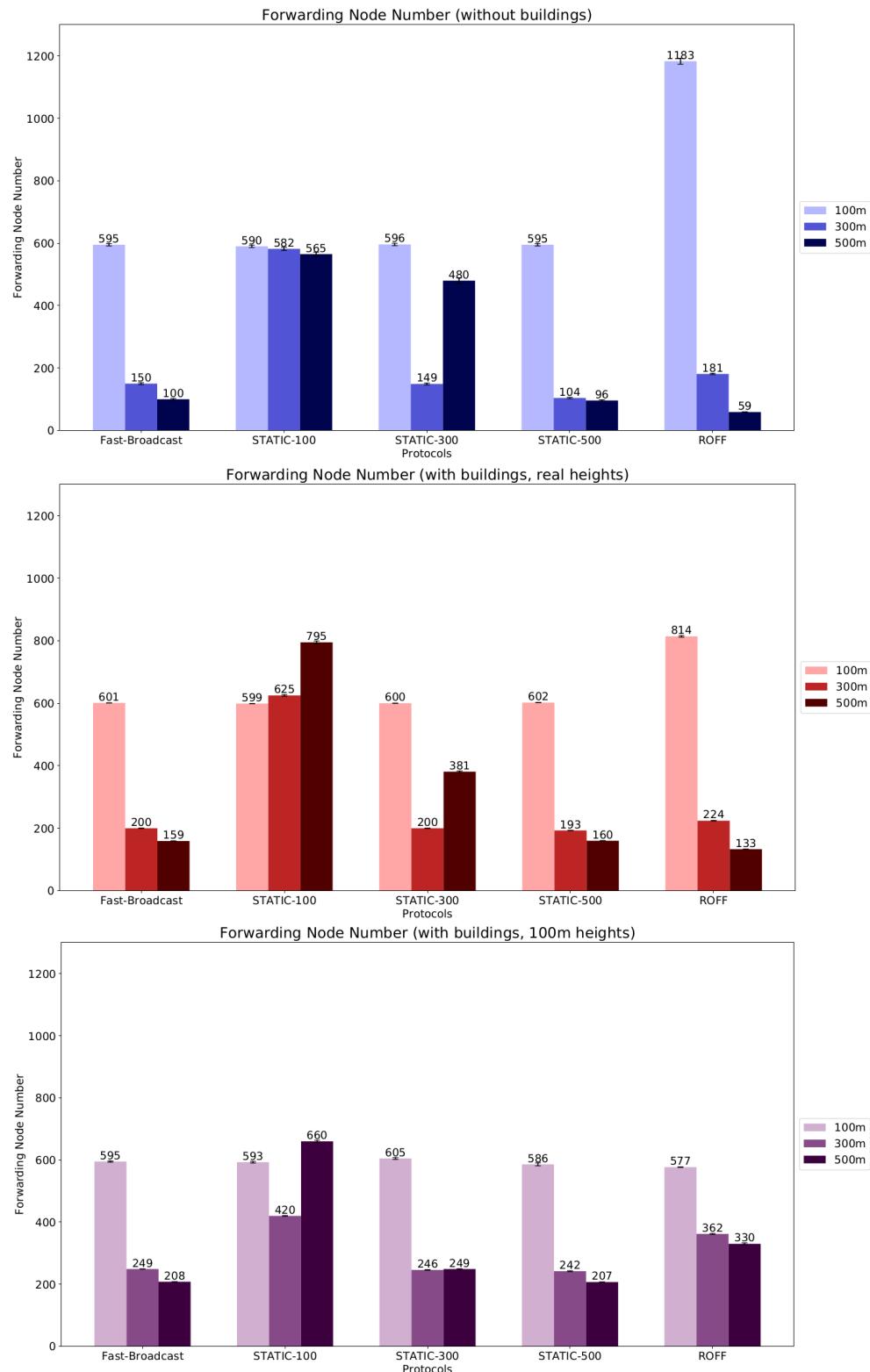


Figure 6.34: NOS for Los angeles smart city scenario

**Figure 6.35:** FNN for Los angeles smart city scenario

Judging by the results shown in Figure 6.31 and 6.32, drones seem to help a lot in signal propagation thanks to their altitude. In fact, the scenario without buildings and with real-height buildings yield comparable results. In the third scenario, when the obstacles's heights exceed the second drone layer's height, the shadowing effect is more pronounced for lower transmission range configurations. Overall the total delivery ratio is still acceptable, with neither Fast-Broadcast or ROFF falling under 90%. *TDROC* in configurations with 100m transmission range is affected in a greater way. In fact, as specified above, in this scenario the *TDROC* metric is only about vehicles on the ground, so the greater shadowing due to high buildings yields the expected results.

The effects of obstacles on *NOH* (Figure 6.33) are consistent with those observed in previous scenarios. The metric's value further increases with high buildings, since the signal probably propagates more often on the ground instead of covering larger distances thanks to inter-street drone to vehicle communications. With high buildings the propagation is restrained inside the road, hence the effects of drones are much more limited.

Concerning *NOS* (Figure 6.34), we can notice that the increase is much higher when going from obstacles with real heights to 100m high obstacles. For example, focusing on the 300m transmission range configuration, for Fast-Broadcast we have an increase of 64.71% from the first to the second image and of 553.57% from the second to the third. For ROFF those values are 27.27% and 114.29%. Once again, this phenomenon could be due to the same effect explained in the previous paragraph: with inter-street communication heavily impeded and the increase in *NOH*, *NOS* increases as a consequence.

Focusing on the 300m and 500m transmission range configurations, *FNN* (Figure 6.35) increases for Fast-Broadcast and ROFF, both when going from the first image to the second and from the second to the third. This is expected, since in both cases the shadowing effects are increased and a greater number of forwarders is necessary to cover the same area. For the 100m case, we can notice some anomalous results: Fast-Broadcast's value stays around the same number, while ROFF's decreases considerably in the last two scenarios. The same observation about collision areas in ROFF introduced in Section 6.3 might explain this behaviour, but it would affect in a similar way also the 300m and 500m configurations. Hence, this phenomenon needs more thorough testing in order to be explained.

Overall, the two main algorithms seem to work well even in this mixed 2D and 3D scenario, with similar advantages and disadvantages reported in previous urban scenarios.

## 6.7 Hello Message forging scenario

In all previous tests all entities participating in the message propagation told the truth about their position and did not try to manipulate the algorithms in order to hinder the delivery process. 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 having a malicious node send fake Hello Messages bursts to vehicles during the Estimation Phase in order to mess with their estimations. These forged messages included fake information, mainly a wrong (overestimated) transmission range and fake IDs. 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.,

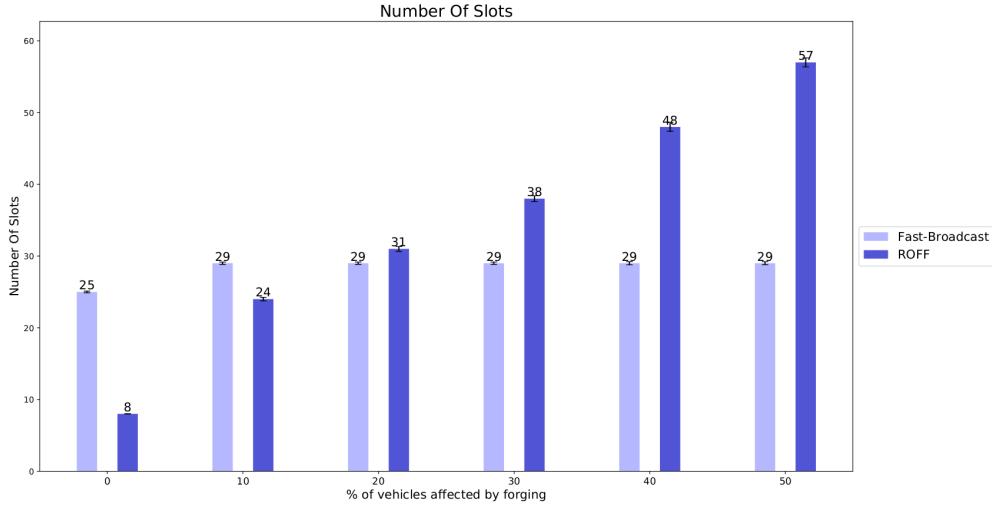
the vehicles which receive the forged Hello Message bursts) ranging from 0 to 50% have been tested. Both tests were carried out using the Los Angeles urban scenario without the effect of the Obstacle Model. Only Fast-Broadcast and ROFF algorithms were tested.

### 6.7.1 Low severity attack

In the low severity attack the effect of 150 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 450 ( $txRange + 150$ ). 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               | 150                   |     |
| Forged distances                        | 301, 302,...,450      |     |
| Simulator configuration                 |                       |     |
| Transmission power                      | 4.6                   | dBm |
| Transmission range                      | 300                   | m   |
| Shadowing model                         | No                    |     |
| Junction modeling                       | No                    |     |
| Number of simulations per configuration | 1000                  |     |

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



**Figure 6.36:** NOS for low severity forging scenario

The hello forging attack fills the Neighbor Table of affected ROFF nodes with fake information about neighbors. The priority acquisition process is hence tainted by the fake PFCs, which get top priority due to forged distances being greater than the real transmission range. It is possible to see an increasing trend in ROFF results when the percentage of affected vehicles is increased. In fact, when more vehicles are affected by this attack, the probability of having affected nodes participating in the source-to-circumference message propagation increases, enlarging *NOS* as a consequence.

Fast-Broadcast is only affected in a small way when increasing the percentage of affected nodes from 0% to 10%. In fact, Fast-Broadcast utilizes only the estimated value of the transmission range for its waiting time computation, and does not take into account other information about the neighbors of a node. Hence, the only variable which affects Fast-Broadcast's *NOS* is the maximum estimated transmission range, which is 450m in this scenario. This value gets propagated throughout the network over time, despite the number of vehicles initially affected by the forging. As a consequence, the metric's value stabilizes after the percentage of affected vehicles is greater than 0% and does not increase further when that percentage is increased.

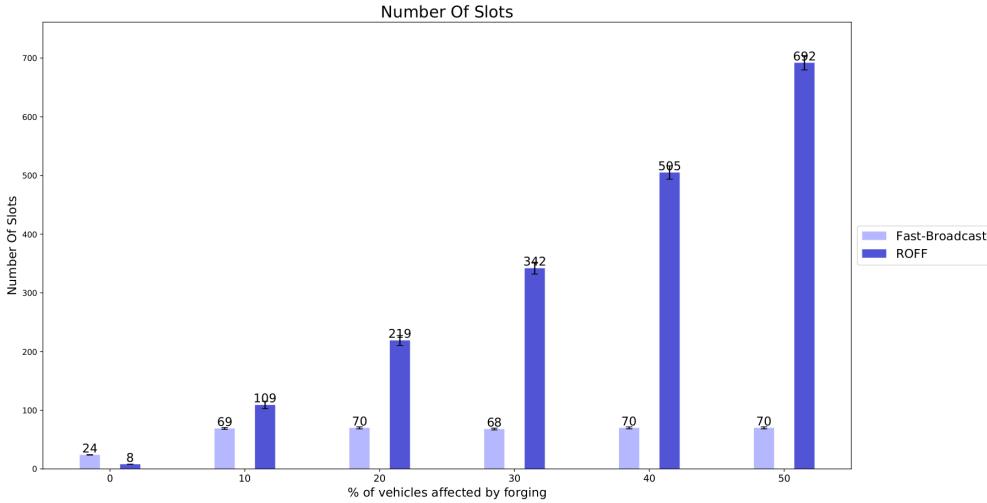
We can notice that ROFF's performances are similar to Fast-Broadcast's with 20% of affected vehicles. After this threshold, ROFF's value increase further.

### 6.7.2 High severity attack

In the high severity attack the number of forged Hello Messages was increased to 1000. As a consequence, the fake positions included in the messages increased in a way such that the receivers detected distances ranging from 301m to 1300m. 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,...,1300       |
| Simulator configuration                 |                         |
| Transmission power                      | 4.6 dBm                 |
| Transmission range                      | 300 m                   |
| 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



**Figure 6.37:** NOS for high severity forging scenario

In this case it is possible to observe that ROFF is affected a lot more than Fast-Broadcast by the attack. Even when jumping from 0% to only 10% of affected nodes, ROFF's *NOS* increases more than tenfold, with a value 57.97% higher than Fast-Broadcast under the same conditions. Fast-Broadcast is affected as well when going from 0% to 10%, but the increase is much lower. The reason is the same noticed in the previous Section: Fast-Broadcast is affected only by the maximum estimated transmission range (1300m in this scenario), while ROFF is also affected by the great number of information about fake neighbors. Higher percentages of affected vehicles increase the Number Of Slots even more for ROFF, while Fast-Broadcast caps out after 10% and its metric's value does not increase.

One can notice that making the attack stronger requires a smaller percentage of affected vehicles in order to make ROFF's the loser against Fast-Broadcast in this kind of confrontation.

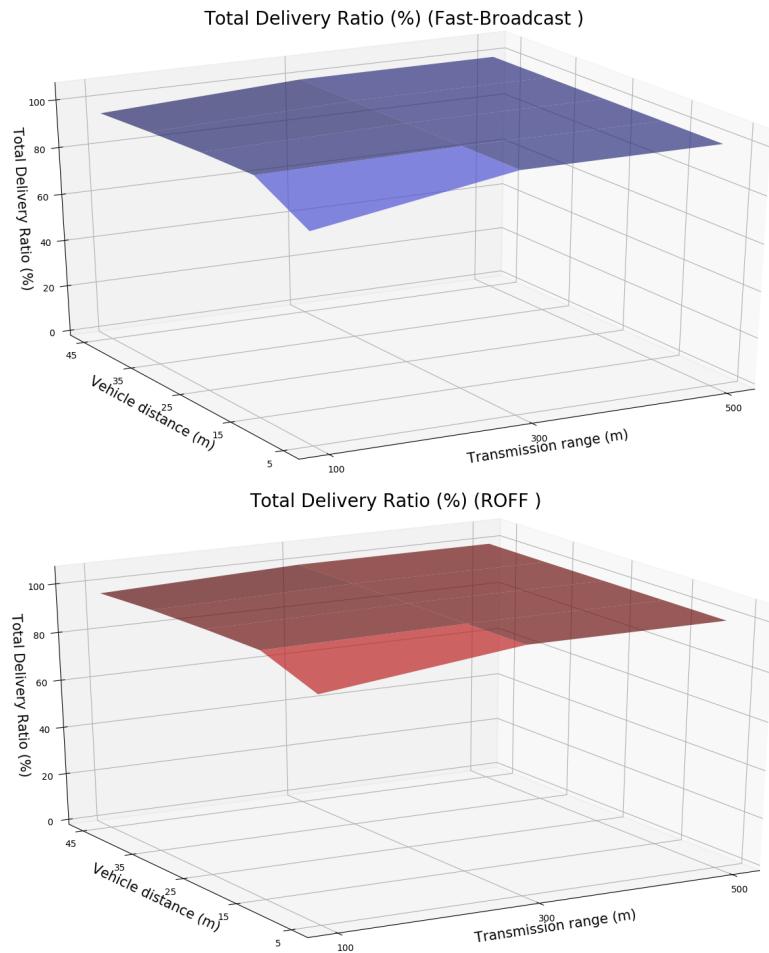
As a consequence of these attacks, a possible solution could be the implementation of identification techniques between vehicles in order to discard forged messages.

## 6.8 Different vehicle density scenario

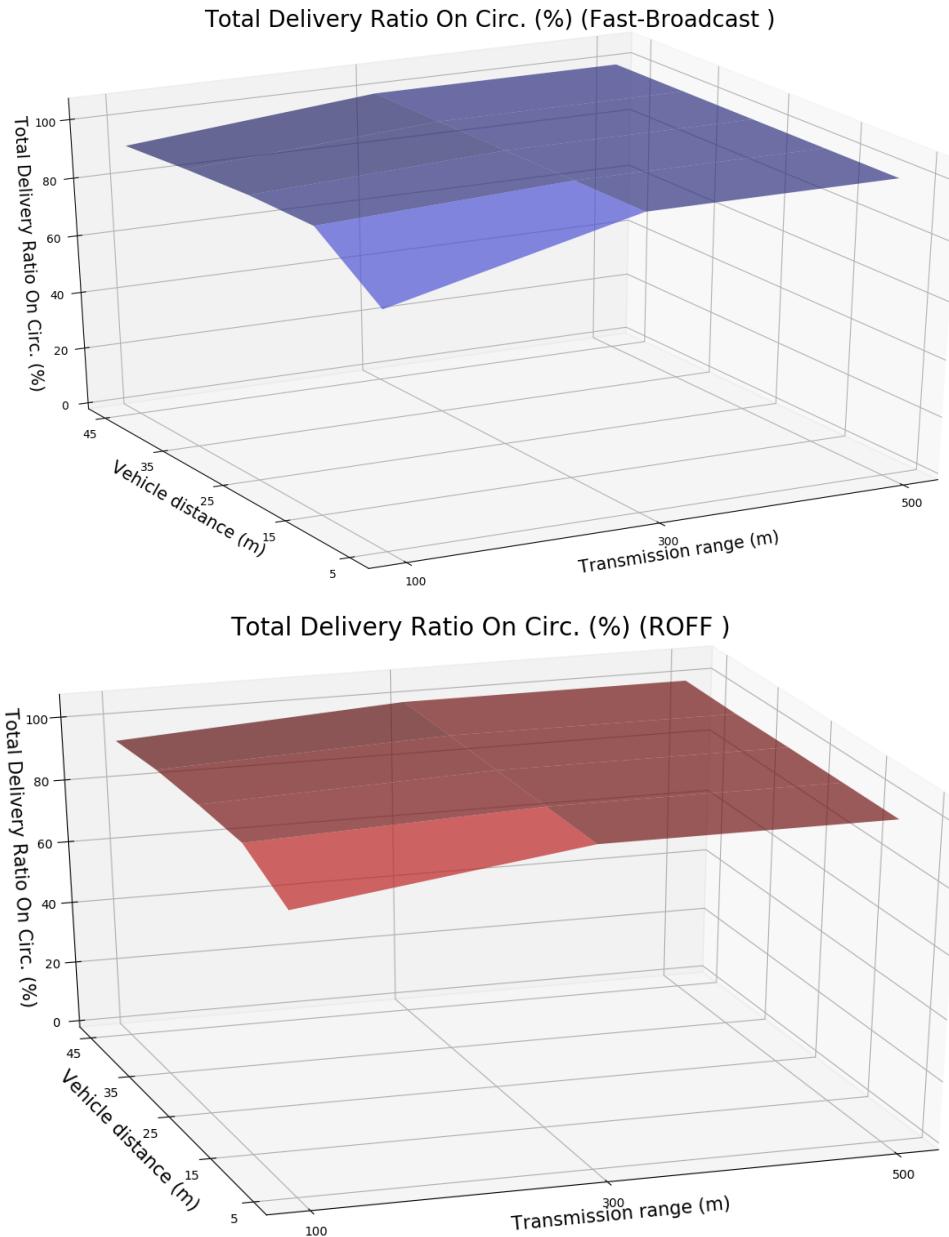
The previous scenarios all employed a distance between vehicles equal to 25m. Real vehicle distributions are obviously different than this idealistic scenario. Moreover, good broadcasting algorithms should work well regardless of vehicle distribution and density. Hence, in this section different vehicle distributions (generated by varying the distances between vehicles) have been tested. The scenario is based on the Padua urban scenario (Section 6.5), with vehicle distances ranges from 5m to 45m, and without the effects of buildings. Obviously, a decrease in distance between vehicles implies an increase in vehicle density (and also on the total number of vehicles in the scenario). Parameters are reported in Table 6.16.

| Parameter                 | Value                        |
|---------------------------|------------------------------|
| Scenario configuration    |                              |
| Distance between vehicles | 5, 15, 25, 35, 45            |
| Number of vehicles        | 4975, 2856, 1776, 1318, 1072 |
| Simulator configuration   |                              |
| Transmission powers       | -7.0, 4.6, 13.4 dBm          |
| Transmission range        | 100, 300, 500 m              |
| Shadowing model           | No                           |
| Junction modeling         | No                           |

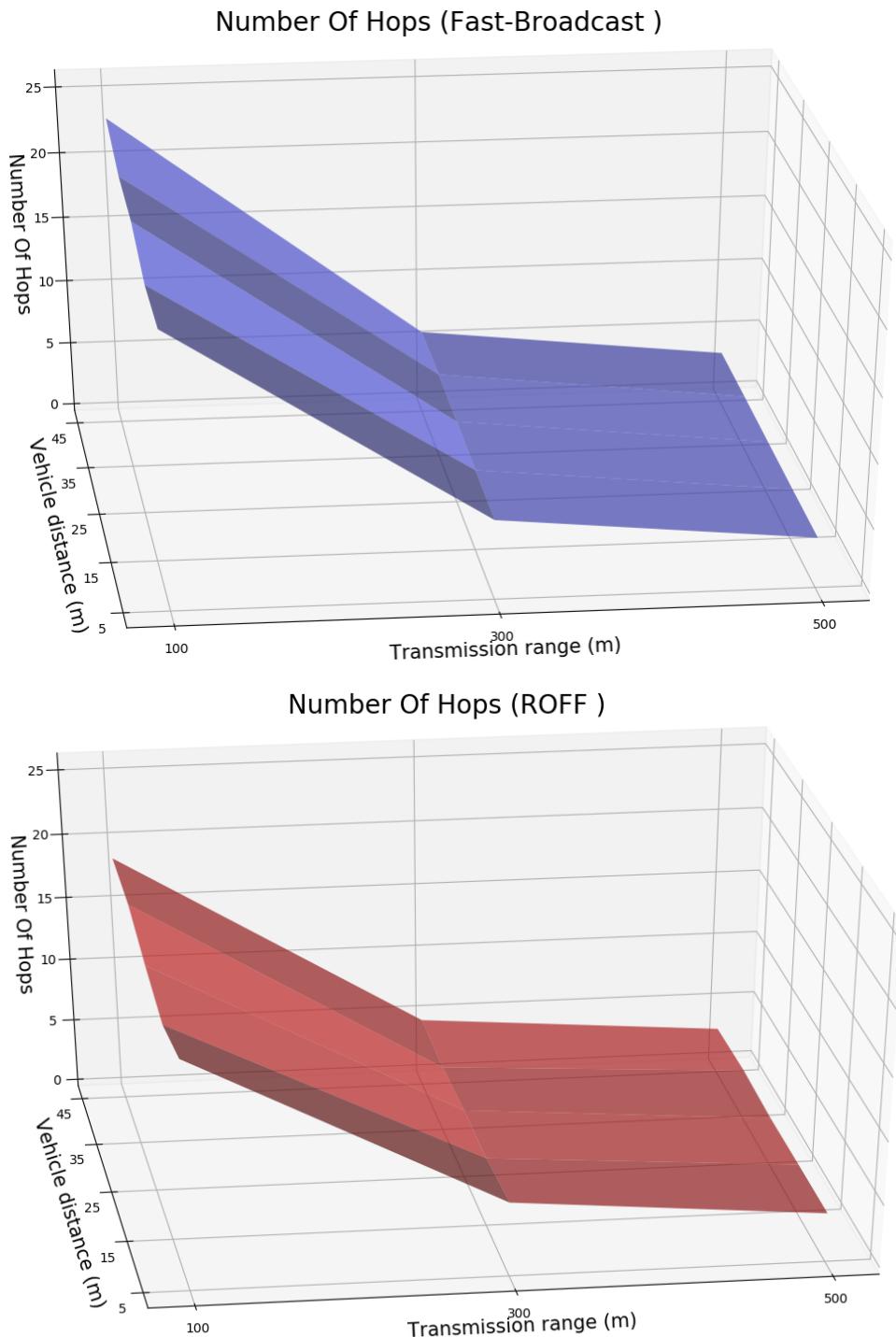
**Table 6.16:** Different vehicle densities scenario configuration based on Padua urban scenario



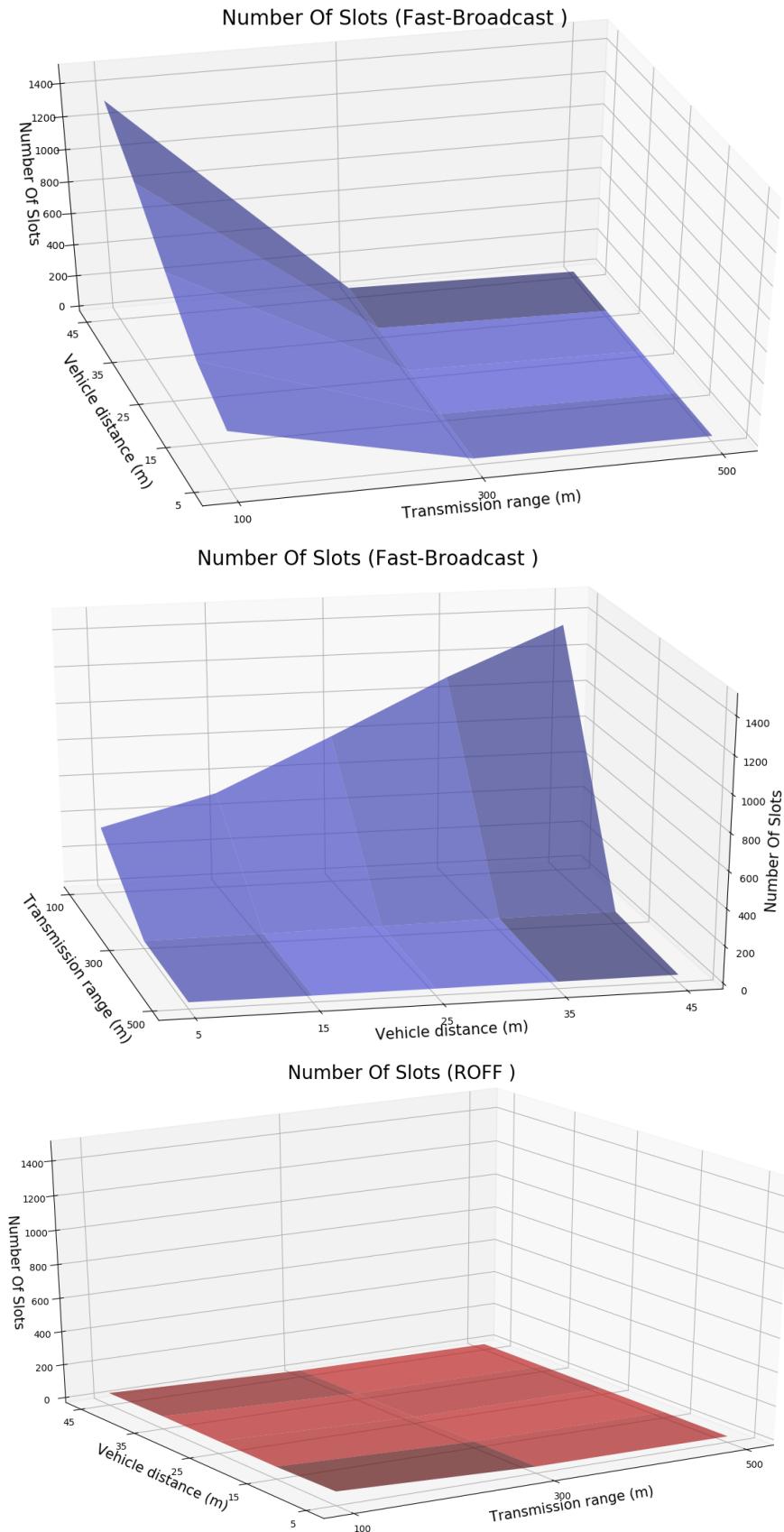
**Figure 6.38:** TDR for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges



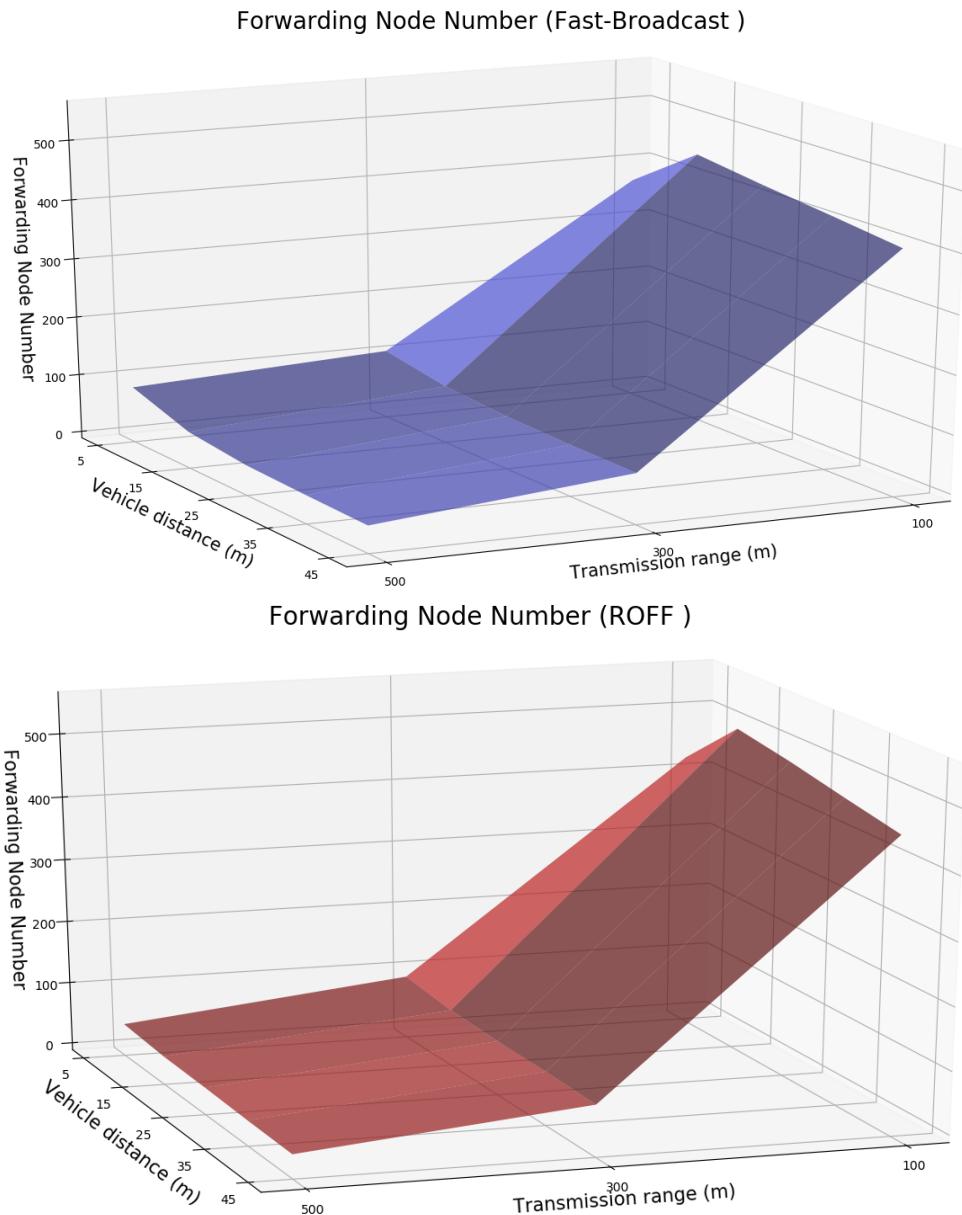
**Figure 6.39:** TDROC for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges



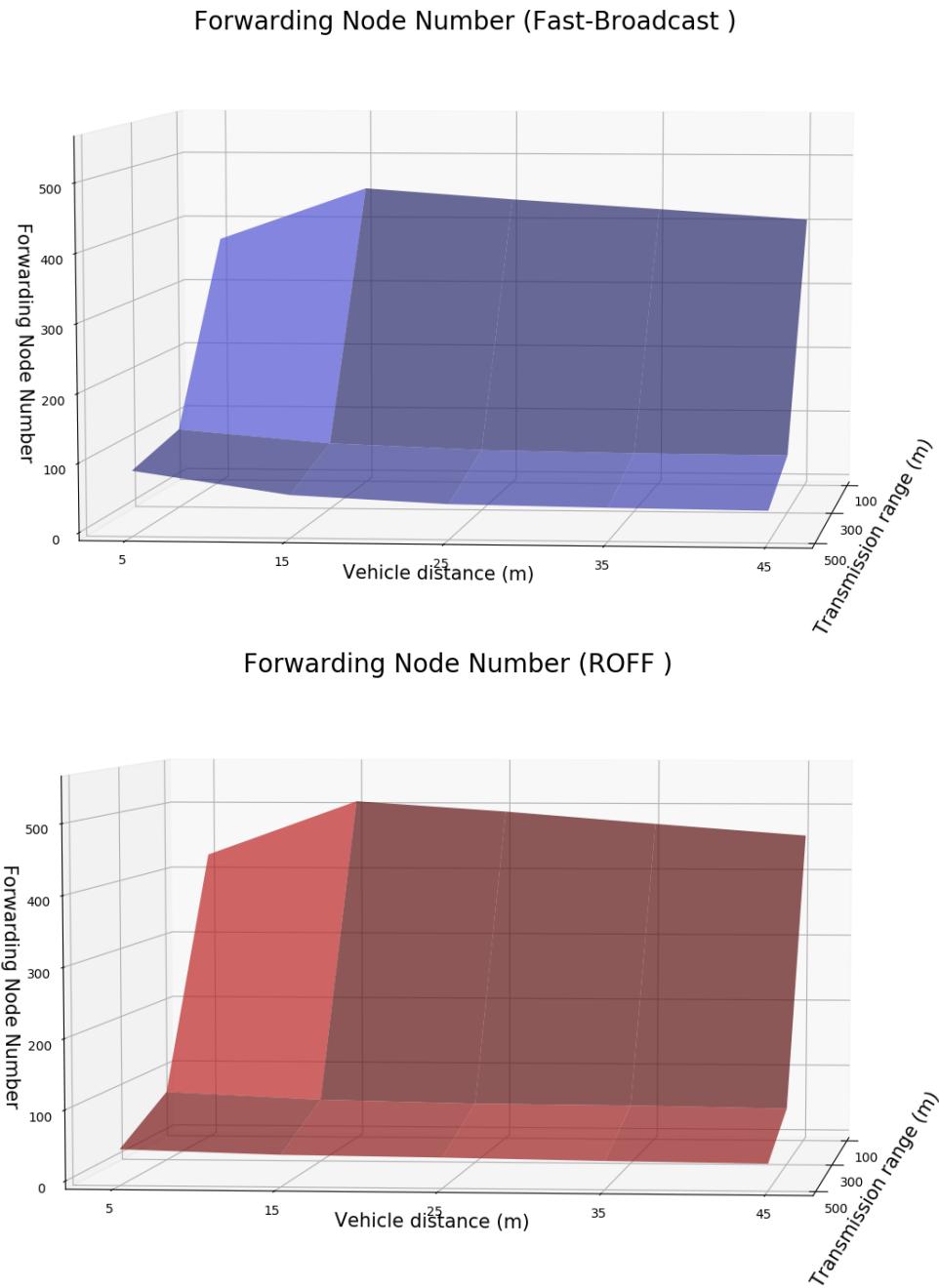
**Figure 6.40:** NOH for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges



**Figure 6.41:** *NOS for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges*



**Figure 6.42:** *FNN for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges*



**Figure 6.43:** FNN for Fast-Broadcast (top) and ROFF (bottom) with different distance between vehicles and transmission ranges (different angle)

Figure 6.38 and 6.38 show optimal delivery ratios both total and on the circumference. We can see a small drop in the metrics' values for the 100m transmission range and 5m vehicle distance configurations. This might be due to higher collisions near the forwarders, which cause the signal to propagate unevenly to all areas of the scenario.

Despite this, the values are still acceptable for both algorithms.

Concerning NOS (Figure 6.40), we notice that ROFF achieves better results in every case. For both algorithms, different vehicle densities do not seem to affect the number of hops by much. This means that the algorithms manage to designate a distant PFC as next forwarder in both high and low density scenarios, with ROFF designation working better as noticed in previous tests.

Figure 6.41 show that *NOS* decreases as the transmission range decreases for Fast-Broadcast. This phenomenon is also due to the decrease in *NOH* noticed in the previous paragraph. We notice how *NOS* increases as the vehicle distance increases (hence the density decreases) under the 100m transmission range condition. This is expected, since the FFC can be quite far from the 100m mark which guarantees the minimum waiting time for each hop. Hence, the designated PFC could end up waiting a lot longer than a scenario with higher density. ROFF is not affected by either transmission range and vehicle distance due to its waiting time calculation.

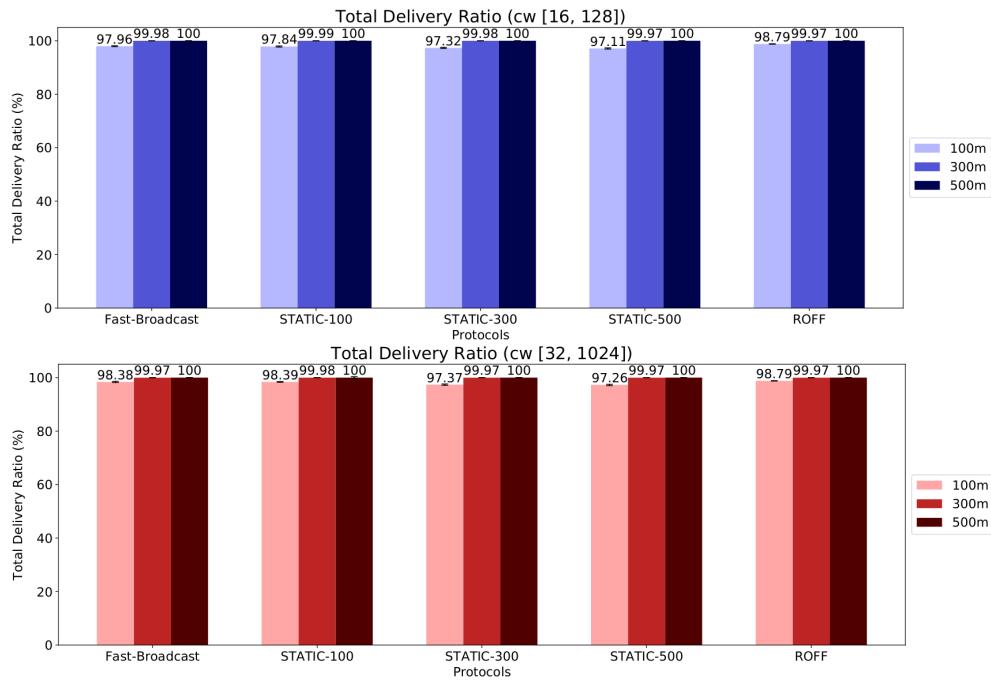
With 500m transmission range a positive correlation can be noticed for Fast-Broadcast's FNN (Figure 6.42 and 6.43). In fact, the value increases as vehicle density increases. This can be explained by the greater number of node waiting the exact same waiting time after a forwarding, which results in a greater number of redundant transmissions. For both algorithms, a dramatic increase in *FNN* appears when decreasing the transmission range from 300m to 100m. With a reduced transmission range, every retransmission reaches a lower number of nodes.

## 6.9 Smaller contention window

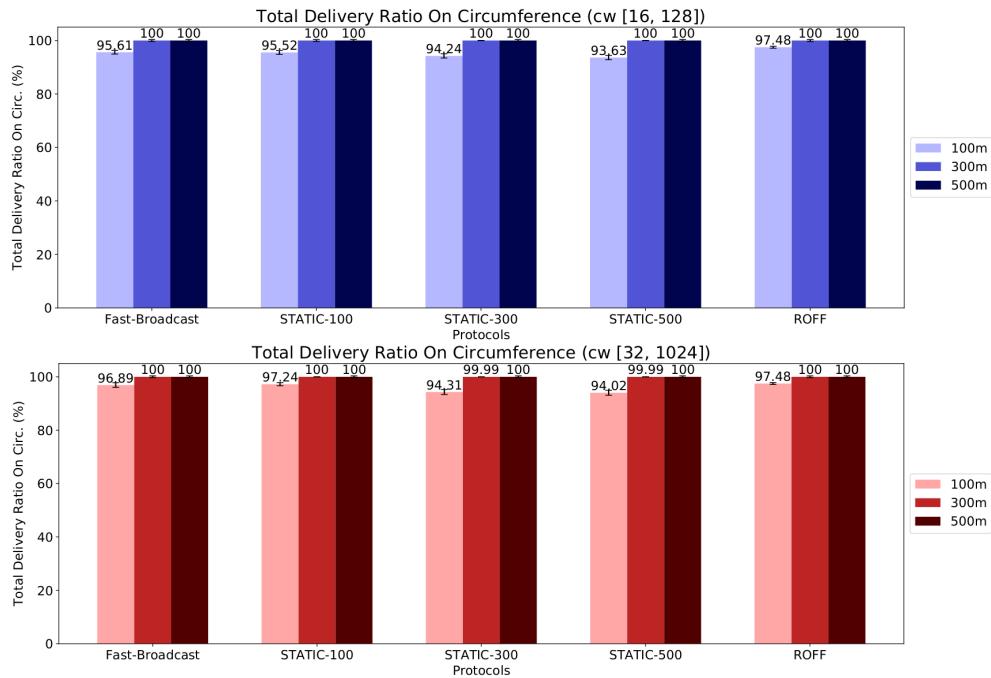
All previous tests involving Fast-Broadcast have been carried out with a contention window ranging from 32 to 1024 slots. This interval is so wide in order to guarantee correct waiting time calculation even in high vehicle density scenarios. Fast-Broadcast's results on Number Of Slots could benefit from using a narrower contention window. In this scenario, based on the Los Angeles urban scenario, a contention window ranging from 16 to 128 slots will be utilized. Parameters for this scenario are reported in Table 6.18.

| Parameter                               | Value                      |
|---|----------------------------|
| Simulator 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 | 600                        |

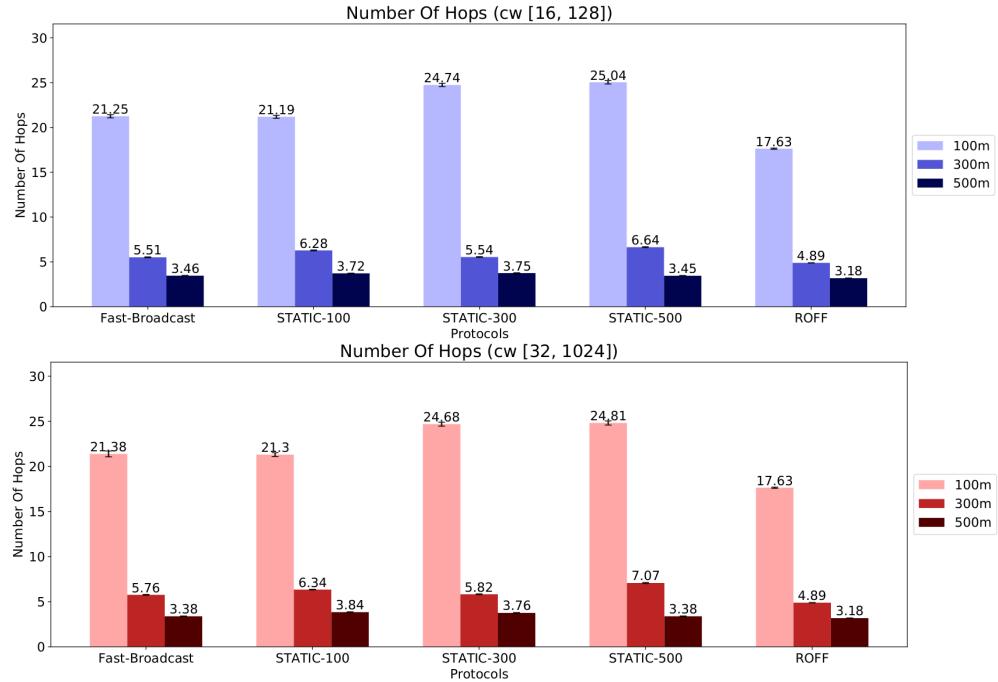
**Table 6.18:** Contention window scenario configuration based on Los Angeles urban scenario



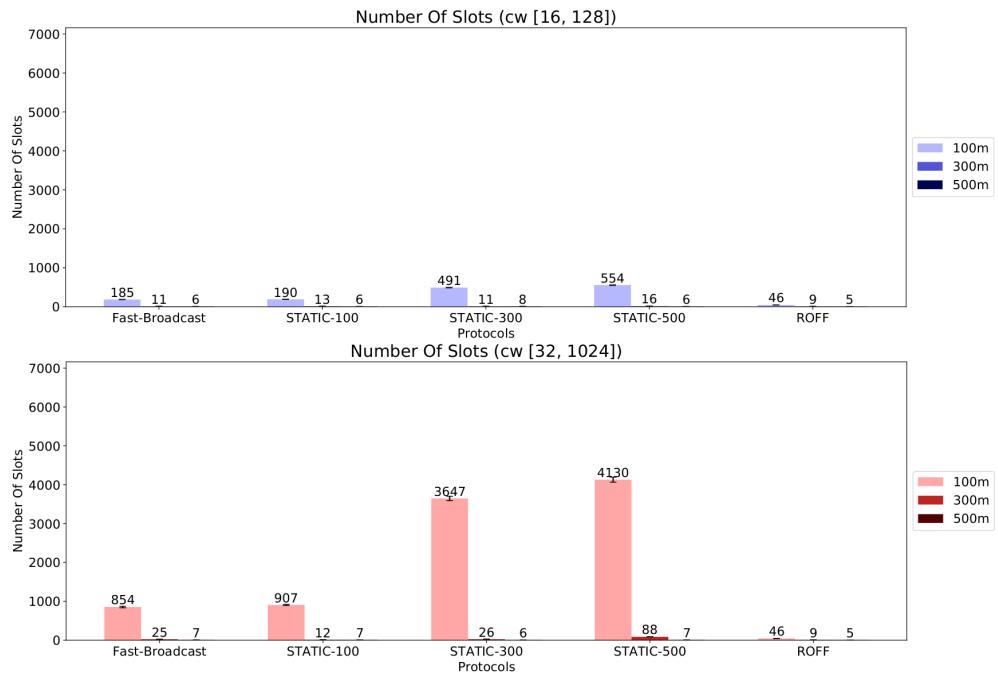
**Figure 6.44:** TDR for [16, 128] contention window (top) and [32, 1024] contention window (bottom)



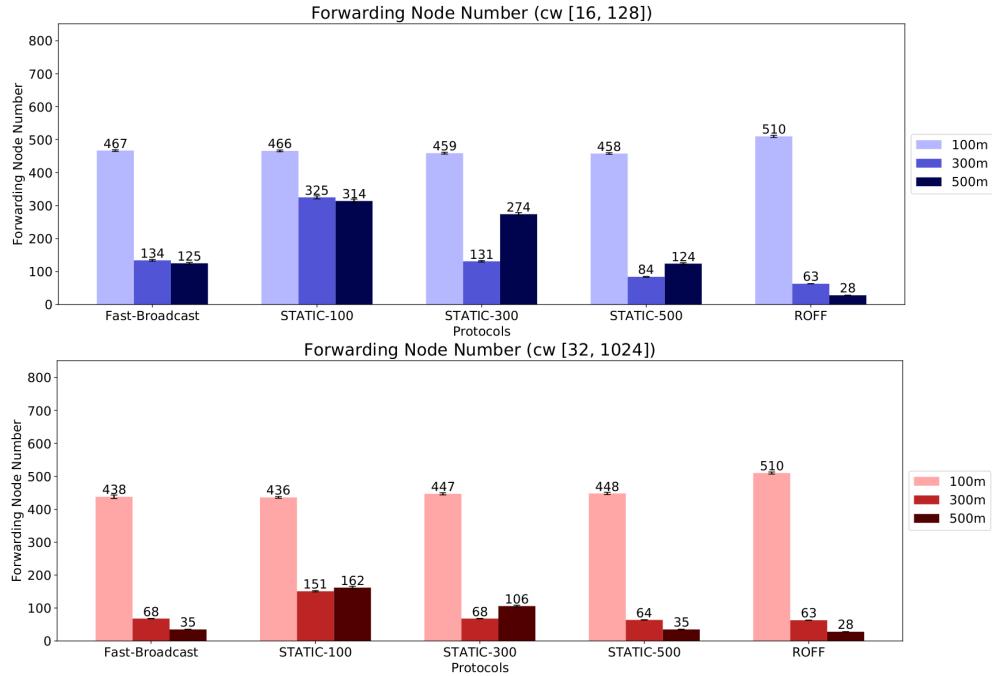
**Figure 6.45:** TDROC for [16, 128] contention window (top) and [32, 1024] contention window (bottom)



**Figure 6.46:** NOH for [16, 128] contention window (top) and [32, 1024] contention window (bottom)



**Figure 6.47:** NOS for [16, 128] contention window (top) and [32, 1024] contention window (bottom)



**Figure 6.48:** FNN for [16, 128] contention window (top) and [32, 1024] contention window (bottom)

Before starting to compare the metrics, we can notice that ROFF's results for both contention windows do not change: this is obvious, since the change in contention window affects only Fast-Broadcast (and its STATIC variants). Next, the same patterns between Fast-Broadcast and the STATIC protocols identified in previous tests regarding the underestimation, correct estimation and overestimation of transmission range continue to be true even for the lower contention window.

The two different contention windows do not cause any significant difference in delivery ratios and NOH. The smaller contention window brings about a big improvement in NOS, especially for the 100m transmission range configuration, where the metric improves by 361.62% for Fast-Broadcast. For the 300m transmission range, the metric improves by 56%, coming close to ROFF's value, equal to 9 slots. However, this improvement comes with a cost: looking at FNN (Figure 6.48), we can see that Fast-Broadcast's metric worsens by 97.05% and 254.29% for the two highest transmission range. This means that the waiting time difference between PFCs is not long enough to guarantee suppression of redundant transmissions for the current vehicle density. Given a smaller contention window, more vehicles could end up waiting the exact same number of slots, hence resulting in multiple transmissions being fired simultaneously and needlessly. However, FNN does not increase by much for the 100m transmission range, and the decrease in NOS is the greatest: this means that for a lower number of PFCs (lower transmission range means lower number of neighbors for each node, hence a lower number of PFCs) the smaller contention window actually brings benefits all across the board.

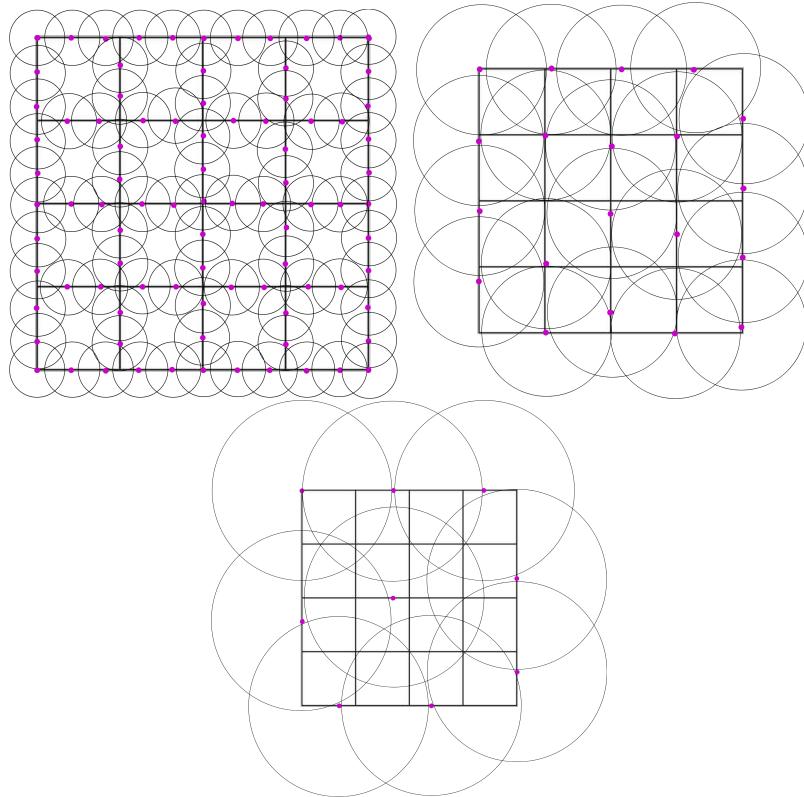
These observations could suggest that a dynamic contention window, function of node density, number of neighbors or both could benefit Fast-Broadcast greatly.

## 6.10 Number of Hello Messages computation

All previous tests focused on metrics concerning the Broadcast Phase, i.e., the phase after the event whose information has to be delivered to a certain area has already happened. We deemed also important the analysis of the Estimation Phase, i.e., the phase where nodes exchange information about their position, speed and other data in order to get to know their neighbors. In fact, both Fast-Broadcast and ROFF need nodes to exchange some Hello Message with each other in order to make the Alert Message delivery efficient. The focus of this analysis will mainly regard the number of Hello Messages sent, and will show how the number of Hello Messages required by Fast-Broadcast is a lot lower than those required by ROFF.

The following analysis is based on the theoretical formulation of the algorithms and on a simplified version of the Grid scenario (Section 6.3), with 100, 300m and 500m transmission range (*txRange*), as usual. For Fast-Broadcast, whose analysis has proven to be more difficult, we are interested in a rough estimation of the number of Hello Messages instead of the exact number.

Suppose we have a 1200x1200m grid with 5 north-south and 5 east-west roads, where vehicles are placed every 25m. In total, the scenario is composed of 455 vehicles. Since both algorithms utilize *turns* (or *Beacon Intervals*, as they are called in ROFF), i.e., vehicles wait on average a certain amount of time in between Hello Message transmissions, the comparison between the number of Hello Message sent can be carried out by comparing the number of messages sent for each turn/Beacon Interval. For ROFF, the calculation is pretty simple: each vehicle sends a Hello Message every turn. As a consequence, in the scenario at hand, ROFF sends 455 Hello Messages for each turn. For Fast-Broadcast, the estimation depends on how transmission ranges intersect with each other. The maximum number of Hello Messages sent each turn is bounded from above by the number of circumferences with radius *txRange* such that the intersection between the circumferences' area is maximum and the center of each circumference do not lie inside another circumference. Based on this observation, a rough estimation of the number of Hello Message can be seen in Figure 6.49, where the purple dots represent vehicles sending Hello Messages.



**Figure 6.49:** Number of Hello Messages sent for Fast-Broadcast with 100m (top), 300m (middle) and 500m (bottom) transmission range

By enumeration it is possible to count the number of nodes which send the Alert Message. Table 6.19 shows the number of Hello Messages sent for every turn for different transmission ranges. As we can see, ROFF transmits a constant number of messages for every turn and that number does not decrease with transmission range increase. Instead, Fast-Broadcast transmits a much lower number of messages, with values decreasing with greater transmission ranges. The huge number of Hello Messages sent by ROFF could cause problems such as greater overhead and more collisions.

In any case, further experimental tests are required to confirm the outcomes of this qualitative analysis.

| Protocol       | Number of Hello Messages |      |      |
|----------------|--------------------------|------|------|
|                | 100m                     | 300m | 500m |
| Fast-Broadcast | 89                       | 20   | 9    |
| ROFF           | 455                      |      |      |

**Table 6.19:** Number of Hello Messages sent for every turn



# Chapter 7

## Conclusions

After thorough testing in several scenarios of increasing complexity, Fast-Broadcast, ROFF and their Smart Junction variants have been compared. While the original algorithms lacked in delivery ratios, especially in more complicated scenarios, such as Padua, where the shadowing effects of the Obstacle Model were more pronounced, the SJ variants managed to reach good and comparable levels of coverage across all the network at the cost of increasing the number of forwarding nodes. Given the similar increase in metrics for both algorithms when junctions are introduced, the final comparison will focus on the original ROFF and Fast-Broadcast algorithms. As is often the case in these kind of analysis, no algorithm emerges as clear winner across the board. Instead, each scheme has its own pros and cons, which will be analyzed in the following paragraphs.

The comparison between the number of Hello Messages sent makes Fast-Broadcast a clear winner, with a decreasing number of messages sent when transmission range increases. Instead, ROFF requires every vehicle to transmit its own Hello Message during the estimation phase in order to reach good performances. A good knowledge of the neighborhood is hence paramount for ROFF. Fast-Broadcast can work with a lower number of transmitted messages since the only information needed is the maximum estimated transmission range.

After this consideration about the Estimation Phase, the comparison can now consider the Broadcast Phase after the initial sending of the Alert Message. Several metrics have been utilized in order to test the effectiveness of the algorithms under different points of view. As already explained earlier, both schemes manage to score comparable results from the delivery ratios point of view. The real advantage of ROFF concerns the number of hops and slots to reach the area of interest. ROFF achieves much better results than Fast-Broadcast especially for lower transmission ranges. With higher transmission ranges, the gap is much narrower. The reason that causes ROFF's higher performances is the much more reliable farthest forwarder candidate selection, which reduces hops, and the lower waiting time due to a more precise scheduling and waiting time computation, which reduces the number of slots. This comes with a cost: the number of nodes who forward the Alert Message is higher for ROFF compared to Fast-Broadcast for the majority of 2D and 3D scenarios. One possible cause which concerns collision areas due to a "too perfect" scheduling has been theorised in Section 6.3. A greater number of forwarders could contribute to a dissemination-hindering congestion across the network, especially when paired with other bandwidth consuming applications, which is obviously an undesirable property

for a multi-hop broadcasting protocol.

Another point against ROFF is its vulnerability to Hello Message forging, which causes the number of slots waited to increase when the percentage of affected vehicles increases. After a certain threshold, which depends on the severity of the attack, ROFF performs worse than Fast-Broadcast in terms of waited slots.

Table 7.2 summarises all the algorithms' pros and cons.

| Fast-Broadcast |        | Feature                    | ROFF |   |
|----------------|--------|----------------------------|------|---|
| ✓              | Medium | Number of slots            | Low  | ✓ |
| ✓              | Medium | Number of hops             | Low  | ✓ |
| ✓              | Low    | Number of Hello Messages   | High | ✗ |
| ✓              | Medium | Number of forwarding nodes | High | ✗ |
| ✓              | Low    | Vulnerability to forging   | High | ✗ |

Table 7.2: Fast-Broadcast and ROFF's pros and cons

Considering all the abovementioned advantages and disadvantages of the two algorithms, and considering also the necessity of quick propagation which characterises alert situations, the algorithm which seems to offer more advantages is ROFF. The low number of hops and slots helps reducing the end-to-end delay, delivering the information quickly to the area of interest.

## 7.1 Future works

The simulations carried out in Section 6.9 showed how a smaller contention window could benefit Fast-Broadcast's performances on the number of slots waited. A possible extension of Fast-Broadcast which could increase its performances could employ a dynamic contention window based on vehicle density.

The smart junction extension to the algorithms in this work operates correctly under the assumption that vehicles know junctions' locations, for example via a GPS system. Even though these kind of information is widespread nowadays, having a working broadcasting protocol even in emergency situations where GPS system are not available would be desirable. Hence, one possible proposal could identify junctions based on how many directions a vehicle receives other Hello Messages from. For example, a vehicle  $f$  receiving three different messages from three different directions such that the angle between  $f$  and the senders is greater than  $270^\circ$  could identify itself as a vehicle inside a junction. This backup mode could prove itself useful even when more accurate data about a location are not available or have not been updated in a long time (e.g., smaller cities, rural areas, etc.).

Lastly, the current work employed only multi-hop broadcasting protocols in the comparison. An extension of this work could compare the achieved results with probabilist or single-hop protocols (Section 1) to test the effectiveness of other state of the art algorithms.

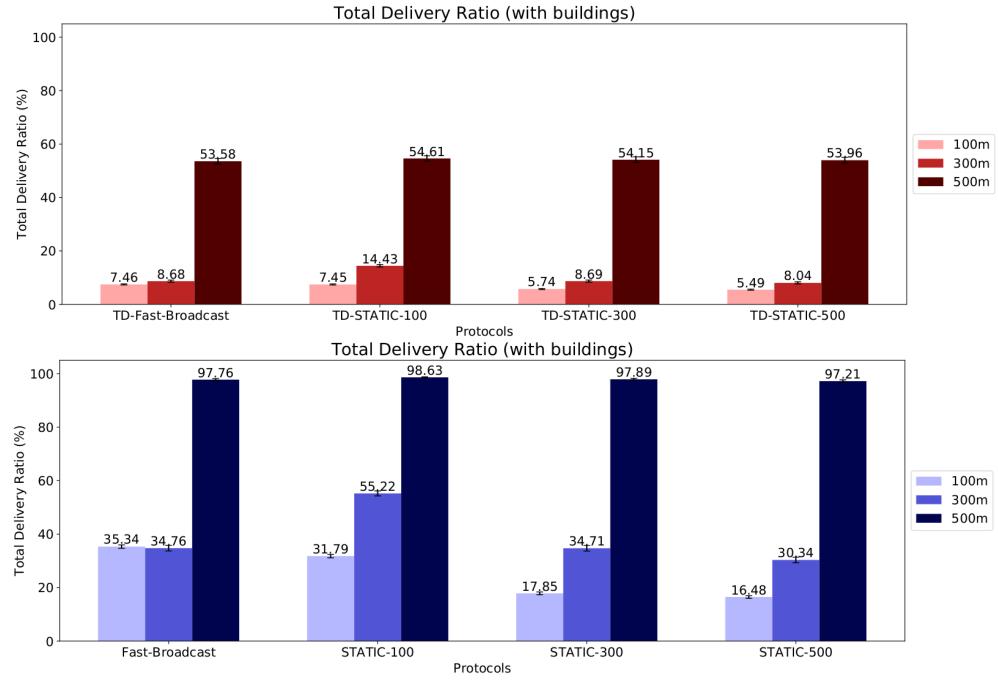
## Appendix A

# Fast-Broadcast modifications

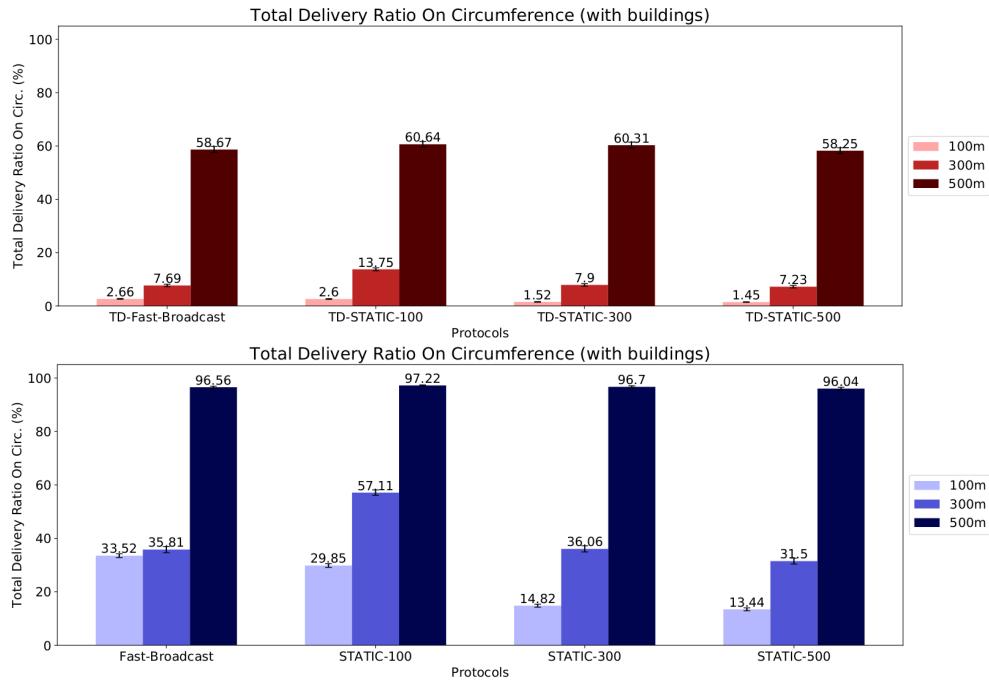
The previous work [19] utilized a version of Fast-Broadcast in which vehicles having origin-vehicle distance smaller than origin-sender distance suppressed their scheduled transmission since they do not contribute in the outward propagation of the message. In the following graphs, the old version will be called *TD-Fast-Broadcast* (as in *Triangular-Distance-Fast-Broadcast*). The aim of this appendix is to show the reasons why this additional check has been removed in the version of the Fast-Broadcast algorithm used in this work thanks to simulations which employed two scenarios. More in detail, the aim of these simulations has been twofold:

- \* show that TD-Fast-Broadcast performs worse than Fast-Broadcast in terms of delivery ratios in difficult scenarios with buildings, particularly the Padua urban scenario;
- \* show that Fast-Broadcast is non-pejorative in all metrics in scenarios where TD-Fast-Broadcast did not show delivery ratio problems, such as the Los Angeles urban scenario.

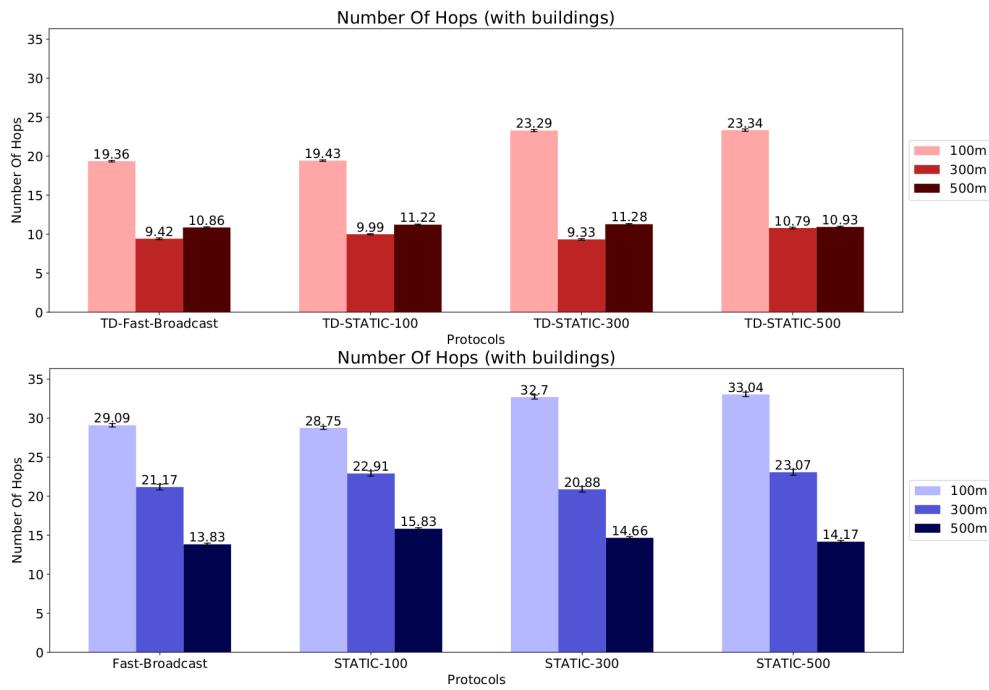
## A.1 Fast-Broadcast's improvement on delivery ratios



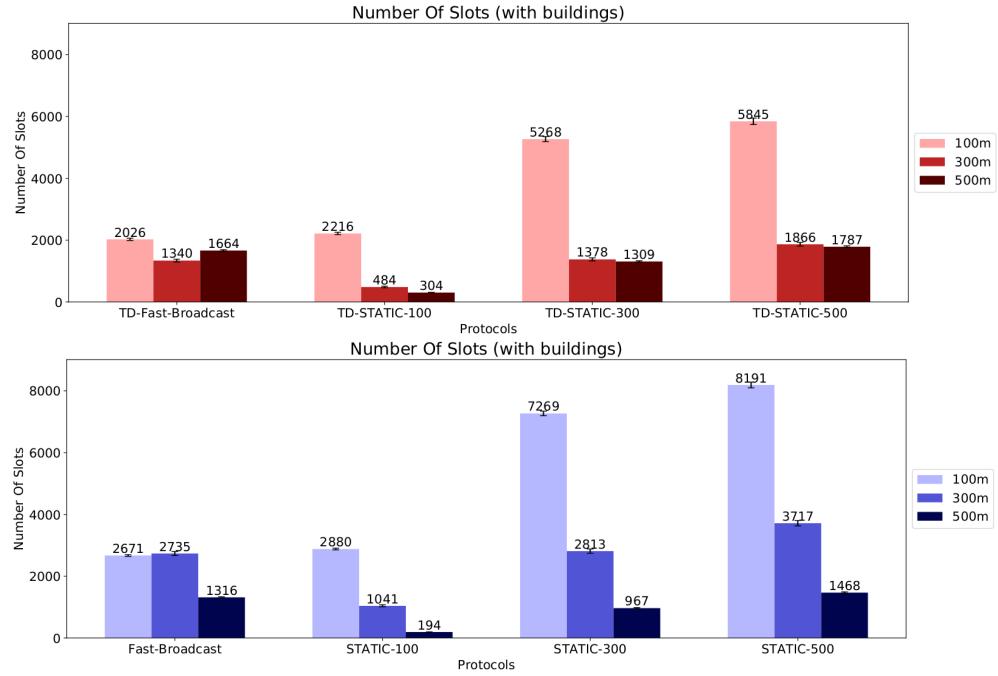
**Figure A.1:** *TDR for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario*



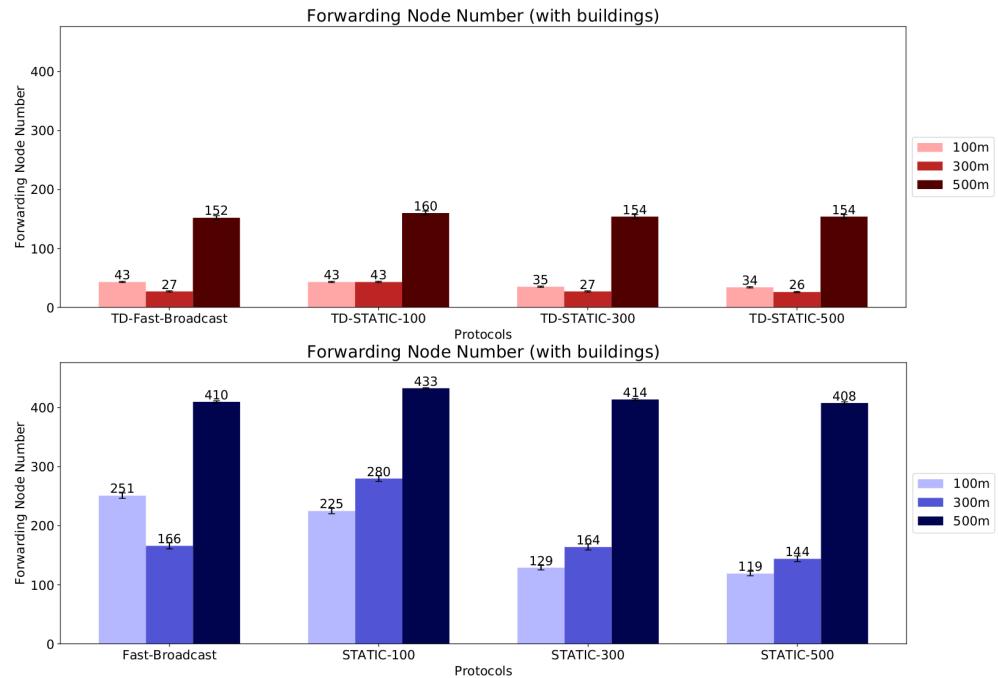
**Figure A.2:** TDROC for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario



**Figure A.3:** NOH for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario



**Figure A.4:** *NOS* for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario

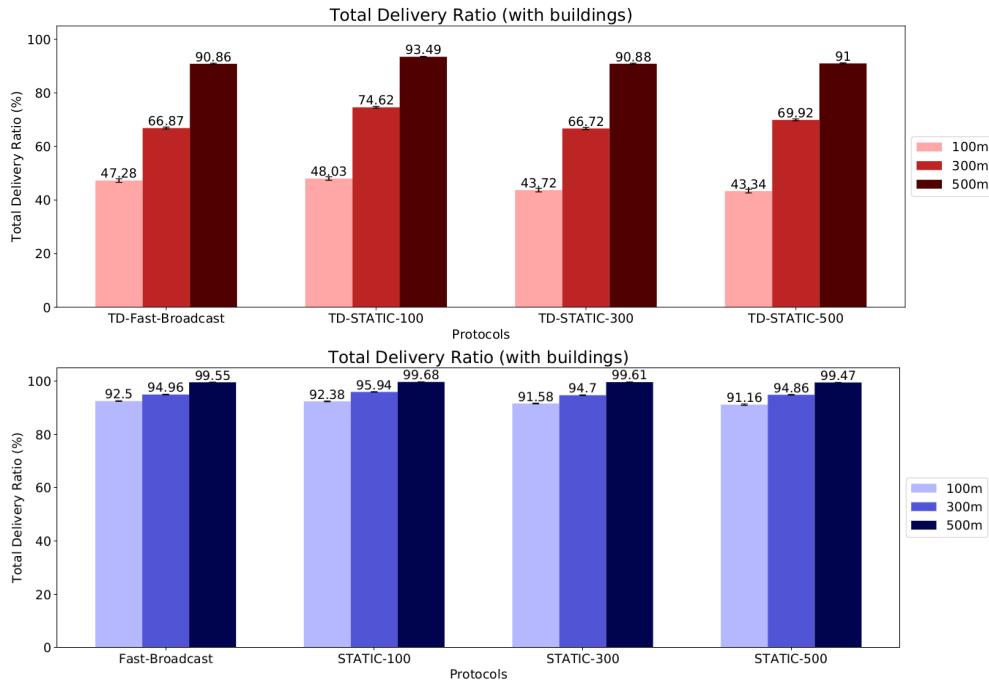


**Figure A.5:** *FNN* for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Padua urban scenario

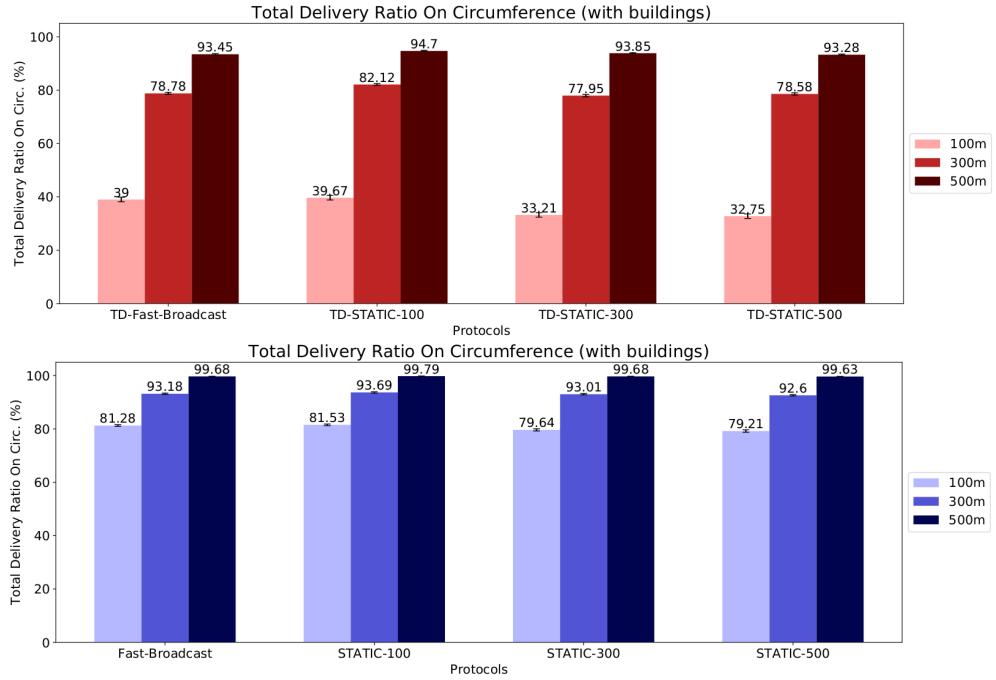
Figure A.1 and A.2 show that TD-Fast-Broadcast struggles to deliver message across the network, even with 500m transmission range. The results are poor especially for the lowest transmission ranges, where the achieved delivery ratios are under 10%. The removal of the suppression based on distance improve Fast-Broadcast's delivery ratios for all three transmission range configurations.

Figure A.3, A.4 and A.5 show an increase in all three metric's value (except for *NOS* with 500m transmission range), but it is important to remember how those metrics are calculated (Section 6.1). *NOS* and *NOH* take into consideration the nodes on the circumference reached by the Alert Message. Fast-Broadcast reaches a greater number of nodes on the circumference, hence the metric is calculated on different sets of vehicles. Moreover, the increase in *FNN* results in a greater coverage, which is a desirable property. Overall, Fast-Broadcast brings about a great increase in delivery ratios. The non pejorative effects of Fast-Broadcast on the other metrics in a scenario where TD-Fast-Broadcast delivery ratio is not poor will be analyzed in the next section.

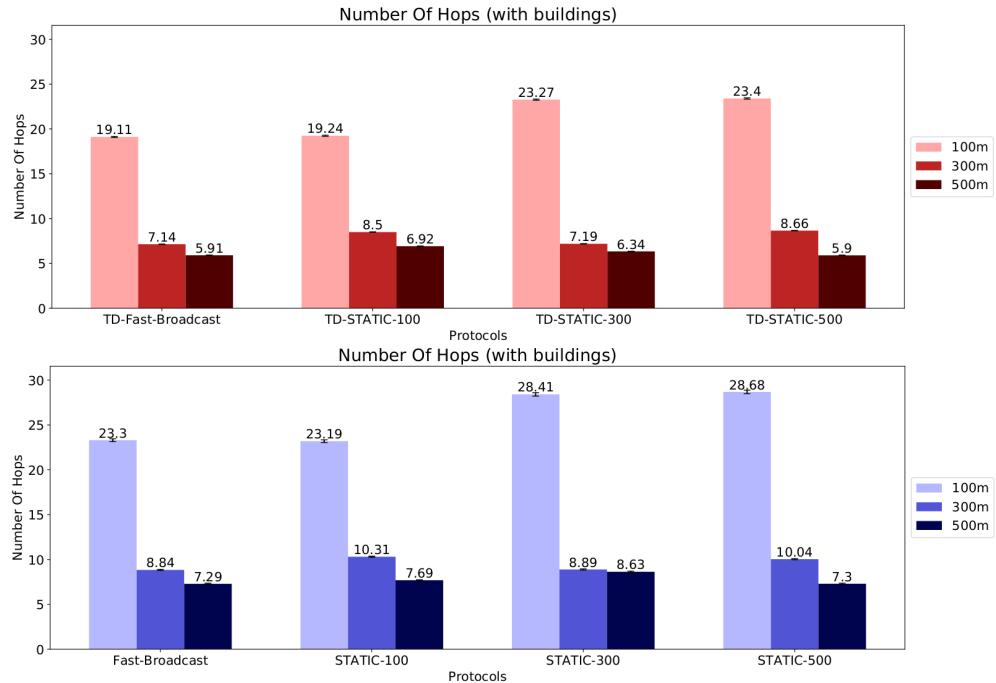
## A.2 Fast-Broadcast's non-pejorative effects



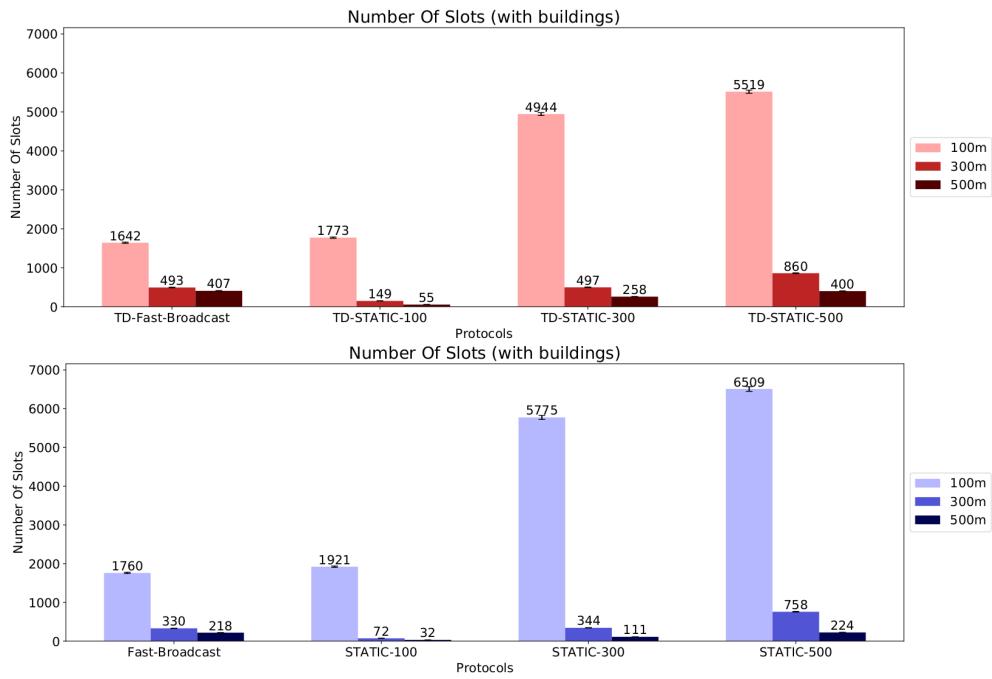
**Figure A.6:** *TDR* for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario



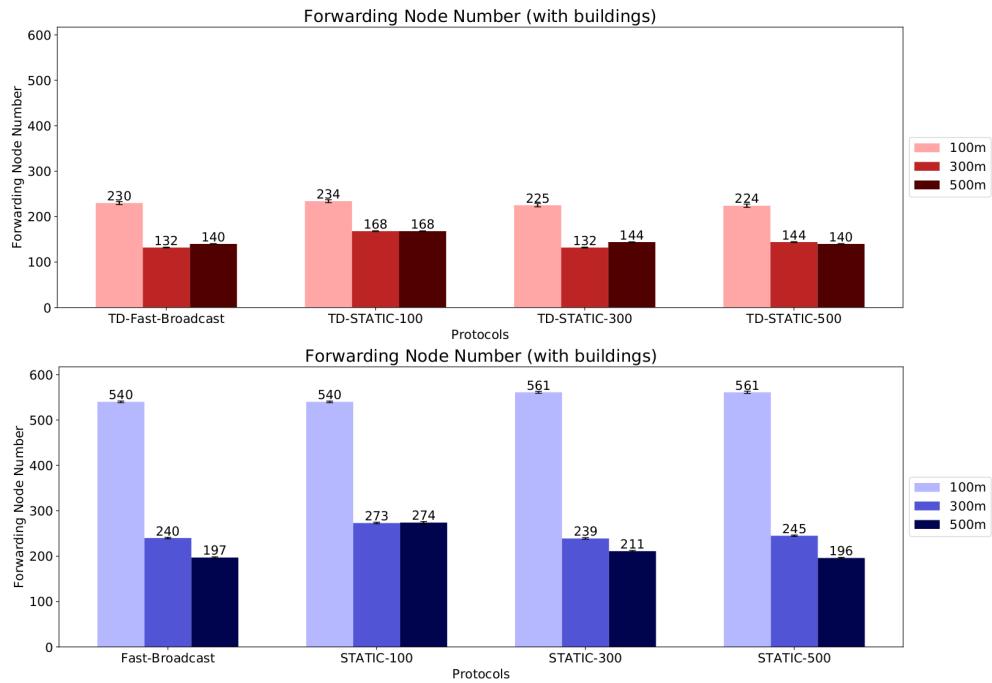
**Figure A.7:** TDROC for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario



**Figure A.8:** NOH for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario



**Figure A.9:** NOS for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario



**Figure A.10:** FNN for TD-Fast-Broadcast (top) and Fast-Broadcast (bottom) for Los Angeles urban scenario

The increase in delivery ratios brought about by Fast-Broadcast is visible also in this scenario. If we focus on the 500m transmission range delivery ratios in Figure A.6 and A.7, the increase is not so severe (from 90.86% to 99.55% for *TDR* and from 93.45% to 99.68% from *TDROC*), so we will utilize this configuration to notice the non pejorative effects on the other metrics. *NOH* (Figure A.8) increases by 23.35%, but this is coupled with a decrease in *NOS* of 46.44%. Basically, the paths taken from the source to the circumference are slightly longer in terms of hops, but the number of slots waited is a lot smaller. Overall, the decrease in *NOS* outweighs the slight increase in *NOH*, making Fast-Broadcast a winner under this point of view.

Regarding *FNN*, Fast-Broadcast's *FNN* increases by 40.71% compared to TD-Fast-Broadcast. Considering the slight increase in delivery ratios and the considerable increase in *NOS*, such increase in the number of forwarding nodes has been deemed acceptable.

This observation, paired with the one exposed in the previous Section, make Fast-Broadcast a clear winner compared to TD-Fast-Broadcast. Hence, all the algorithm improvements, extensions and tests of Chapter 3 and 6 will be based on Fast-Broadcast after the removal of the check on distances.

# Acronyms

**AM** Alert Message. [45](#)

**FFC** Farthest Forwarder Candidate. [42](#)

**LOS** Line of sight. [5](#)

**NIC** Network Interface Controller. [34](#)

**PFC** Potential Forwarder Candidate. [42](#)

**RPM** Radio Propagation Model. [5, 7](#)

**VANET** Vehicular Ad-Hoc Network. [1, 11](#)



# Bibliography

- [1] S. Panichpapiboon and W. Pattara-atikom, “A review of information dissemination protocols for vehicular ad hoc networks”, *IEEE Communications Surveys Tutorials*, vol. 14, no. 3, pp. 784–798, Third 2012, ISSN: 1553-877X. DOI: [10.1109/SURV.2011.070711.00131](https://doi.org/10.1109/SURV.2011.070711.00131).
- [2] Ting Zhong, Bo Xu, and Ouri Wolfson, “Disseminating real-time traffic information in vehicular ad-hoc networks”, in *2008 IEEE Intelligent Vehicles Symposium*, Jun. 2008, pp. 1056–1061. DOI: [10.1109/IVS.2008.4621303](https://doi.org/10.1109/IVS.2008.4621303).
- [3] T. Nadeem, S. Dashtinezhad, Chunyuan Liao, and L. Iftode, “Trafficview: A scalable traffic monitoring system”, in *IEEE International Conference on Mobile Data Management, 2004. Proceedings. 2004*, Jan. 2004, pp. 13–26. DOI: [10.1109/MDM.2004.1263039](https://doi.org/10.1109/MDM.2004.1263039).
- [4] T. Fujiki, M. Kirimura, T. Umedu, and T. Higashino, “Efficient acquisition of local traffic information using inter-vehicle communication with queries”, in *2007 IEEE Intelligent Transportation Systems Conference*, Sep. 2007, pp. 241–246. DOI: [10.1109/ITSC.2007.4357748](https://doi.org/10.1109/ITSC.2007.4357748).
- [5] Q. Yu and G. Heijenk, “Abiding geocast for warning message dissemination in vehicular ad hoc networks”, in *ICC Workshops - 2008 IEEE International Conference on Communications Workshops*, May 2008, pp. 400–404. DOI: [10.1109/ICCW.2008.81](https://doi.org/10.1109/ICCW.2008.81).
- [6] L. Wischhof, A. Ebner, and H. Rohling, “Information dissemination in self-organizing intervehicle networks”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 90–101, Mar. 2005, ISSN: 1524-9050. DOI: [10.1109/TITS.2004.842407](https://doi.org/10.1109/TITS.2004.842407).
- [7] G. Korkmaz, E. Ekici, F. Özgürner, and Ü. Özgürner, “Urban multi-hop broadcast protocol for inter-vehicle communication systems”, in *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, ser. VANET ’04, Philadelphia, PA, USA: ACM, 2004, pp. 76–85, ISBN: 1-58113-922-5. DOI: [10.1145/1023875.1023887](https://doi.org/10.1145/1023875.1023887). [Online]. Available: <http://doi.acm.org/10.1145/1023875.1023887>.
- [8] E. Fasolo, A. Zanella, and M. Zorzi, “An effective broadcast scheme for alert message propagation in vehicular ad hoc networks”, in *2006 IEEE International Conference on Communications*, vol. 9, Jun. 2006, pp. 3960–3965. DOI: [10.1109/ICC.2006.255700](https://doi.org/10.1109/ICC.2006.255700).
- [9] D. Li, H. Huang, X. Li, M. Li, and F. Tang, “A distance-based directional broadcast protocol for urban vehicular ad hoc network”, in *2007 International*

- Conference on Wireless Communications, Networking and Mobile Computing*, Sep. 2007, pp. 1520–1523. DOI: [10.1109/WICOM.2007.383](https://doi.org/10.1109/WICOM.2007.383).
- [10] Y. Tseng, R. Jan, C. Chen, C. Wang, and H. Li, “A vehicle-density-based forwarding scheme for emergency message broadcasts in vanets”, in *The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)*, Nov. 2010, pp. 703–708. DOI: [10.1109/MASS.2010.5663803](https://doi.org/10.1109/MASS.2010.5663803).
  - [11] S. Khakbaz and M. Fathy, “A reliable method for disseminating safety information in vehicular ad hoc networks considering fragmentation problem”, in *2008 The Fourth International Conference on Wireless and Mobile Communications*, Jul. 2008, pp. 25–30. DOI: [10.1109/ICWMC.2008.74](https://doi.org/10.1109/ICWMC.2008.74).
  - [12] T. Osafune, L. Lin, and M. Lenardi, “Multi-hop vehicular broadcast (mhvb)”, in *2006 6th International Conference on ITS Telecommunications*, Jun. 2006, pp. 757–760. DOI: [10.1109/ITST.2006.289011](https://doi.org/10.1109/ITST.2006.289011).
  - [13] M. M. I. Taha and Y. M. Y. Hasan, “Vanet-dsrc protocol for reliable broadcasting of life safety messages”, in *2007 IEEE International Symposium on Signal Processing and Information Technology*, Dec. 2007, pp. 104–109. DOI: [10.1109/ISSPIT.2007.4458046](https://doi.org/10.1109/ISSPIT.2007.4458046).
  - [14] N. Wisitpongphan, O. K. Tonguz, J. S. Parikh, P. Mudalige, F. Bai, and V. Sadekar, “Broadcast storm mitigation techniques in vehicular ad hoc networks”, *IEEE Wireless Communications*, vol. 14, no. 6, pp. 84–94, Dec. 2007, ISSN: 1536-1284. DOI: [10.1109/MWC.2007.4407231](https://doi.org/10.1109/MWC.2007.4407231).
  - [15] H. Alshaer and E. Horlait, “An optimized adaptive broadcast scheme for inter-vehicle communication”, in *2005 IEEE 61st Vehicular Technology Conference*, vol. 5, May 2005, 2840–2844 Vol. 5. DOI: [10.1109/VETECS.2005.1543865](https://doi.org/10.1109/VETECS.2005.1543865).
  - [16] A. Wegener, H. Hellbruck, S. Fischer, C. Schmidt, and S. Fekete, “Autocast: An adaptive data dissemination protocol for traffic information systems”, in *2007 IEEE 66th Vehicular Technology Conference*, Sep. 2007, pp. 1947–1951. DOI: [10.1109/VETECF.2007.409](https://doi.org/10.1109/VETECF.2007.409).
  - [17] S. Panichpapiboon and G. Ferrari, “Irresponsible forwarding”, in *2008 8th International Conference on ITS Telecommunications*, Oct. 2008, pp. 311–316. DOI: [10.1109/ITST.2008.4740277](https://doi.org/10.1109/ITST.2008.4740277).
  - [18] S. Busanelli, G. Ferrari, and S. Panichpapiboon, “Efficient broadcasting in ieee 802.11 networks through irresponsible forwarding”, in *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, Nov. 2009, pp. 1–6. DOI: [10.1109/GLOCOM.2009.5426212](https://doi.org/10.1109/GLOCOM.2009.5426212).
  - [19] M. Romanelli, C. Palazzi, and A. Bujari, “Propagazione di messaggi tra veicoli con modello urbano realistico”, Master’s thesis, University of Padua, 2017.
  - [20] C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, and M. Gerla, “How do you quickly choreograph inter-vehicular communications? a fast vehicle-to-vehicle multi-hop broadcast algorithm, explained”, in *2007 4th IEEE Consumer Communications and Networking Conference*, Jan. 2007, pp. 960–964. DOI: [10.1109/CCNC.2007.194](https://doi.org/10.1109/CCNC.2007.194).

- [21] H. Yoo and D. Kim, “Roff: Robust and fast forwarding in vehicular ad-hoc networks”, *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1490–1502, Jul. 2015, ISSN: 1536-1233. DOI: [10.1109/TMC.2014.2359664](https://doi.org/10.1109/TMC.2014.2359664).
- [22] M. Stoffers and G. Riley, “Comparing the ns-3 propagation models”, in *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug. 2012, pp. 61–67. DOI: [10.1109/MASCOTS.2012.17](https://doi.org/10.1109/MASCOTS.2012.17).
- [23] K. R. Schaubach, N. J. Davis, and T. S. Rappaport, “A ray tracing method for predicting path loss and delay spread in microcellular environments”, in */1992 Proceedings/ Vehicular Technology Society 42nd VTS Conference - Frontiers of Technology*, May 1992, 932–935 vol.2. DOI: [10.1109/VETEC.1992.245274](https://doi.org/10.1109/VETEC.1992.245274).
- [24] S.-C. Kim, B. J. Guarino, T. M. Willis, V. Erceg, S. J. Fortune, R. A. Valenzuela, L. W. Thomas, J. Ling, and J. D. Moore, “Radio propagation measurements and prediction using three-dimensional ray tracing in urban environments at 908 mhz and 1.9 ghz”, *IEEE Transactions on Vehicular Technology*, vol. 48, no. 3, pp. 931–946, May 1999, ISSN: 0018-9545. DOI: [10.1109/25.765022](https://doi.org/10.1109/25.765022).
- [25] I. Stepanov and K. Rothermel, “On the impact of a more realistic physical layer on manet simulations results”, *Ad Hoc Networks*, vol. 6, no. 1, pp. 61–78, 2008, ISSN: 1570-8705. DOI: <https://doi.org/10.1016/j.adhoc.2006.08.001>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1570870506000801>.
- [26] E. Giordano, R. Frank, G. Pau, and M. Gerla, “Corner: A step towards realistic simulations for vanet”, in *Proceedings of the Seventh ACM International Workshop on VehiculAr InterNETworking*, ser. VANET ’10, Chicago, Illinois, USA: ACM, 2010, pp. 41–50, ISBN: 978-1-4503-0145-9. DOI: [10.1145/1860058.1860065](https://doi.org/10.1145/1860058.1860065). [Online]. Available: <http://doi.acm.org/10.1145/1860058.1860065>.
- [27] K. N. Sridhar, S. Hao, M. C. Chan, and A. L. Ananda, “Egress: Environment for generating realistic scenarios for simulations”, in *2006 Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, Oct. 2006, pp. 15–24. DOI: [10.1109/DS-RT.2006.16](https://doi.org/10.1109/DS-RT.2006.16).
- [28] C. Sommer, D. Eckhoff, R. German, and F. Dressler, “A computationally inexpensive empirical model of ieee 802.11p radio shadowing in urban environments”, in *2011 Eighth International Conference on Wireless On-Demand Network Systems and Services*, Jan. 2011, pp. 84–90. DOI: [10.1109/WONS.2011.5720204](https://doi.org/10.1109/WONS.2011.5720204).
- [29] S. E. Carpenter and M. L. Sichitiu, “An obstacle model implementation for evaluating radio shadowing with ns-3”, in *Proceedings of the 2015 Workshop on Ns-3*, ser. WNS3 ’15, Barcelona, Spain: ACM, 2015, pp. 17–24, ISBN: 978-1-4503-3375-7. DOI: [10.1145/2756509.2756512](https://doi.org/10.1145/2756509.2756512). [Online]. Available: <http://doi.acm.org/10.1145/2756509.2756512>.
- [30] C. E. P. e. A. B. M. Barichello, “Propagazione rapida di messaggi in scenario veicolare urbano”, Master’s thesis, University of Padua, 2017.
- [31] U. of Padua. (2019). High performance computing, [Online]. Available: <http://computing.math.unipd.it/>.

- [32] E. P. for Intelligent Cities. (2019). European platform for intelligent cities, [Online]. Available: <http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm>.
- [33] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, “Recent development and applications of SUMO - Simulation of Urban MObility”, *International Journal On Advances in Systems and Measurements*, International Journal On Advances in Systems and Measurements, vol. 5, no. 3&4, pp. 128–138, Dec. 2012. [Online]. Available: <http://elib.dlr.de/80483/>.