Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Radio Propagation Models

A Radio Propagation Model (RPM) is an empirical mathematical formulation used to model the propagation of radio waves as a function of frequency, distance, transmission power and other variables. Over the years various RPMs have been developed, some aiming at modelling a general situation, and others more useful in specific scenarios. For example, implementations range from the more general free space model, where only distance and power are considered, to more complex models which account for shadowing, reflection, scattering, and other multipath losses. Moreover, it is important to keep into consideration the computational complexity and scalability of the model: some have poor accuracy but are scalable, while others have very good accuracy but can only work for small sets of nodes. As always, it is very important to find the right tradeoff between complexity and accuracy.

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

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

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

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

1.2 Obstacle Shadowing propagation loss model

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

The authors of **5720204** designed the model as an extension of well-established fading models, which can be expressed by Equation 1.1, where:

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

$$P_r[dBm] = P_t[dBm] + G_t[dB] + G_r[dB] - \sum L_x[dB] \tag{1.1}$$

Common RPMs can be written as components L of 1.1 and chained to obtain the compound attenuation. For example, Equation 1.2 and 1.3 represent respectively the Two-Ray Ground and Log-Normal models.

$$L_{TwoRayGround} = 10 \lg \left(\frac{d^4 L}{h_t^2 h_t^2} \right) \qquad [dB] \tag{1.2}$$

$$L_{LogNorm} = 10 \lg \left(X_\sigma \right) \qquad [dB] \tag{1.3}$$

The authors extended the general model shown in 1.1 adding a L_{obs} term for each obstacle in the line of sight between sender and receiver. The term is described by Equation 1.4, where:

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

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

$$L_{obs} = \beta n + \gamma d_m \tag{1.4}$$

Figure 1.1 shows an example of transmission where the signal encounters $n = 4$ walls.

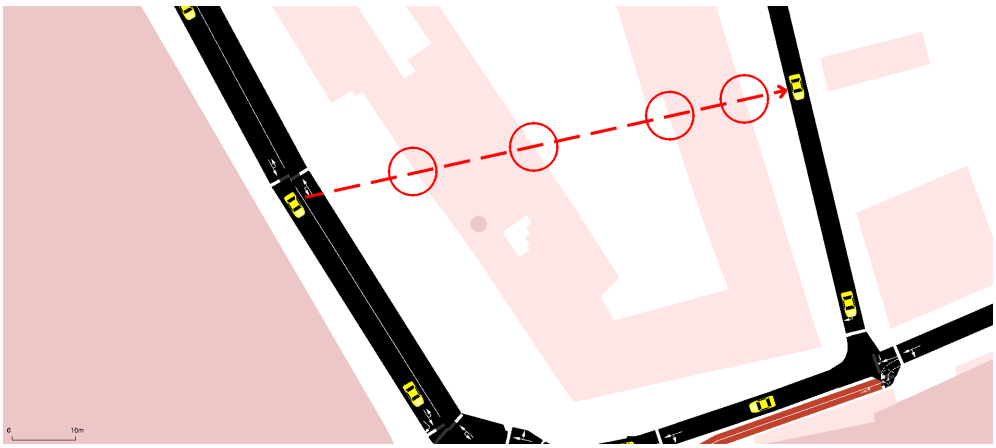

Figure 1.1: Example of obstacle shadowing in vehicle-to-vehicle communication. Walls encountered by the signal are surrounded in red circles

1.3 Emergency Message Dissemination and broadcasting protocols

1.4 sec:emd

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

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

This choice leads to some advantages, such as:

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

The idea behind existing algorithms consists in designating the next forwarder in the multi-hop chain from the source of the alert to the target region where the sensitive data has to be delivered. Ideally, the farthest vehicle from a previous forwarder in the dissemination direction should be given priority when designating the next forwarder.

However, due to unreliable wireless channel the designation of farthest vehicle can fail and interrupt the message dissemination. Due to this, next forwarder designation keeps into consideration vehicles (called potential forwarder candidates, PFCs) which have received an Alert Message. The PFCs participate in contention to elect the farthest forwarder candidate (FFC) who will continue disseminating the message.

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

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

* Multi-hop Broadcasting Protocols, in which packets are transmitted through the network via flooding by some of the neighbors of the source. It is of utmost importance to reduce the number of redundant transmission in order not to waste bandwidth.

* 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 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)* **Korkmaz:2004:UMB:1023875.1023887**, designed to solve the broadcast redundancy, hidden node and reliability problems in multi-hop broadcasting using *Request-to-Broadcast (RTB)* and *Clear-To-Broadcast (CTB)* packets;
   - *Smart Broadcast (SB)* **4025102** and *Efficient Directional Broadcast (EDB)* **4340158**, which try to reduce the delay introduced by UMB and remove the RTB and CTB packets, respectively;
   - *Vehicle-density-based Emergency Broadcasting (VDEB)* **5663803**, a slotted broadcasting protocol which keeps vehicle density into consideration when computing waiting time slots;
   - *Reliable Method for Disseminating Safety Information (RMDSI)* **4591259**, which aims to offer better performances when the network becomes fragmented by making a forwarder keep a copy of the packet it has broadcasted until it hears a retransmission (or until the packet lifetime expires). If no retransmission is heard within a certain time limit, the forwarder tries to find the next node which can relay the message using a small control packet;
   - *Multi-hop Vehicular Broadcast (MHVB)* **4068699**, a protocol that keeps traffic congestion into consideration by checking whether the number of neighbors of a vehicle is greather than a certain threshold and its speed is less than another threshold. When a node detects congestion, it increases its broadcast interval in order to try to reduce the network load;

- *Reliable Broadcasting of Life Safety Messages (RBLSM)* **4458046**, whose main objective is reliability, and a higher priority is given to the vehicle nearest to the sender instead to the one furthest 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 kind of protocols is similar to the one behind Delay based protocols, but instead of assigning a different rebroadcast delay to each vehicle, a different rebroadcast probability is assigned. Each protocol differs in the function that assigns probabilities. Some examples of probabilistic-based protocols are:

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

$$p_{ij} = \frac{D_{ij}}{R} \tag{1.5}$$

     where D_{ij} is the distance between transmitter i and PFC j and R is the transmission range. Given this function, the probability to rebroadcast is proportional to the distance between the PFC and the transmitter. The abovementioned formula does not keep into account vehicle density and also assumes that the transmission range is fixed and known to all vehicles.
   - *Optimized Adaptive Probabilistic Broadcast (OAPB)***1543865** *and AutoCast (AC)* **4350058**, which both keep the vehicle density into consideration when computing the forwarding probability by making vehicle periodically exchange Hello messages. Thanks to those messages, each vehicle can compute the number of neighbors and then use this information accordingly.
   - *Irresponsible Forwarding (IF)* **47402775426212**, a protocol that considers vehicle density like OAPB and AC, but the formula used is not a simple linear function. In fact, the rebroadcast probability assignment function is the following:

$$p = e^{-\frac{\rho_s(z-d)}{c}} \tag{1.6}$$

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

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

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

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

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

(b) Adaptive Broadcast Interval, which adapt the Broadcast Interval based on dynamic information. Some examples are:
   - *Collision Ratio Control Protocol (CRCP)***4357748**, a scheme according to which vehicles exchange information about location, speed and road ID. The Broadcast Interval is dynamically controlled based on the amount of detected collisions and bandwidth efficiency: the protocol tries to maintain the number of collisions under a certain threshold by doubling the Broadcast Interval every time the threshold is exceeded. Otherwise, the Broadcast Interval is decreased by one second when the bandwidth efficiency decreases too much.
     Moreover, the authors propose three different methods for selecting the data to be transmitted:
     - *Random Selection*: a vehicle selects a random information in its database and broadcasts it;
     - *Vicinity Priority Selection*: vehicles give priority to information of nearby areas;
     - *Vicinity Priority Selection with Queries*: similar to Vicinity Priority Selection, with the possibility of querying information for a certain area.
   - *Abiding Geocast:***4531929**, which aims to deliver an Alert Message to a specific area where the warning is still relevant. Only vehicles that are travelling towards the effective area can participate in contention to broadcast the message. Moreover, broadcast is dynamically adjusted based on transmission range, speed, and distance between the potential forwarder and the destination area, increasing when such distance increases or the potential forwarder's speed decreases.

- *Segment-oriented Data Abstraction and Dissemination (SODAD)***1402433**, a protocol according to which roads are divided into segments and each vehicle can both discover information itself and collect it from neighbor's transmissions. Whenever a vehicle receives a transmission from another vehicle, the information received will be classified as either one of two events:
  - a *provocation* event that will decrease the Broadcast Interval;
  - a *mollification* event that will increase the Broadcast Interval.

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

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

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

ROFF's solutions to these problems and the implementation of the protocol will be analyzed in Chapter 3

Chapter 2

Fast Broadcast

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

Fast Broadcast employs two different phases:

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

2.1 Estimation Phase

During this phase, cars 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

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

Algorithm 2 Hello message receiving procedure

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

2.2 Broadcast Phase

This phase is activated once a car 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 2.1. It is quite easy to see that the higher the sender/forwarder distance is, the lower the contention window is.

$$\left\lfloor \left(\frac{\text{MaxRange} - \text{Distance}}{\text{MaxRange}} \times (\text{CWMax} - \text{CWMin}) \right) + \text{CWmin} \right\rfloor \tag{2.1}$$

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 further 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

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

Algorithm 4 Alert Message generation procedure

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

2.3 Two dimensions extension

The original work **4199282** considered only a strip-shaped road, where it was easy to define directions and establish when a message came from the front or the back. In **BAR2017** an extension considering two dimensions was proposed.

The modifications to the Fast Broadcast algorithm are the following:

1. Utilizing only one parameter between CMBR and CMFR (thus considering only CMR):
2. Including the position of the vehicle which originally generated the Alert Message in addition to the position of the sender of the message.

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

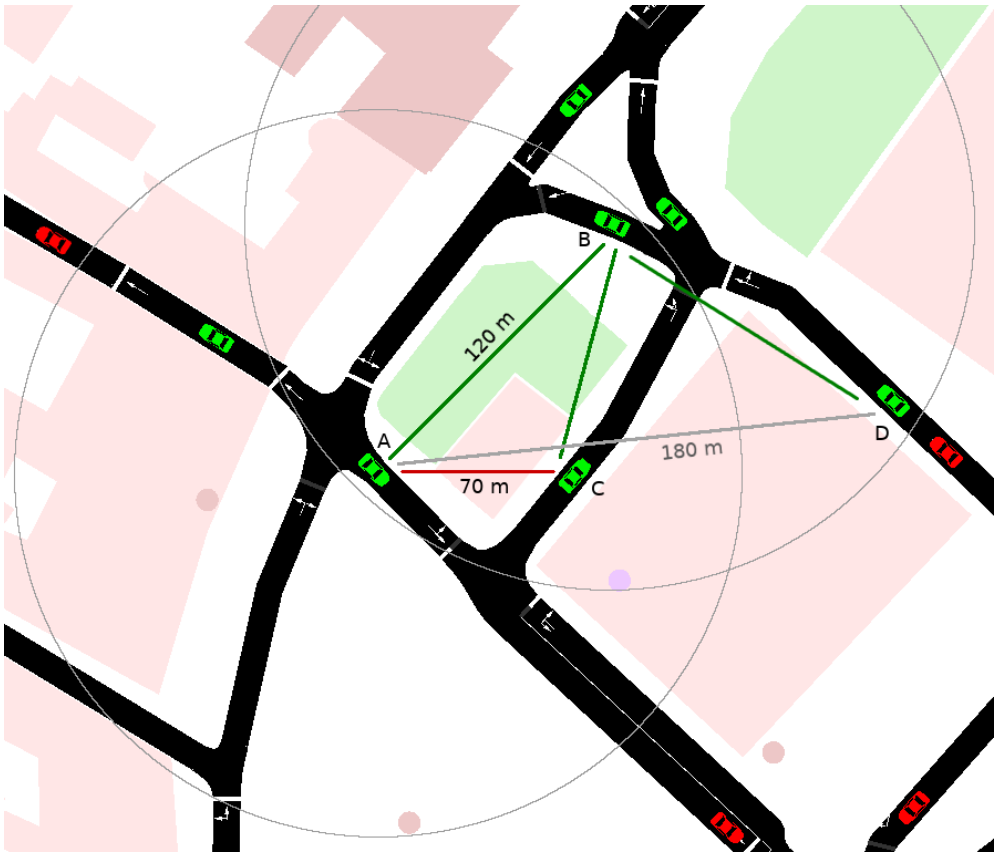

Figure 2.1: Example of Fast Broadcast in 2d scenario

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

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

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

Chapter 3

ROFF

RObust and Fast Forwarding scheme (ROFF) is a protocol proposed by Hongseok Yoo and Dongkyun Kim in **6906275**. This chapter will present the two main problems tackled by ROFF already introduced in Section **??**, namely the perfect suppression of redundant transmissions, which will be explained in 3.1.1 , and the disuniformity and the costant change in spatial vehicle distribution in VANETs, addressed in section 3.1.2.

3.1 Forwarder Selection Problem

3.1.1 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 **??**, 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.

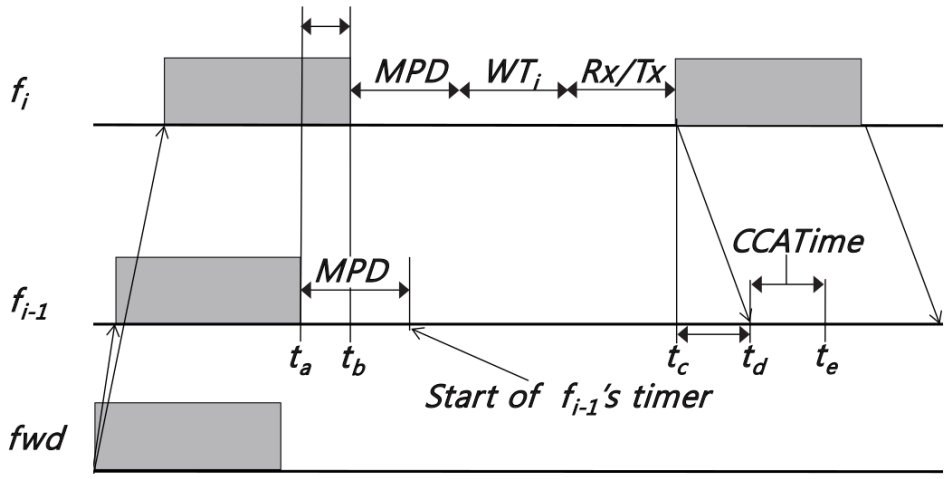

Figure 3.1: Definition of $minDiff$ between f_N and f_{N-1} (**6906275**)

Figure 3.1 depicts a situation where fwd is the forwarder and f_i (farther from fwd than f_{i-1}) relays the message before f_{i-1}. The two vehicles f_i and f_{i-1} complete receiving the message at different times (t_a and t_b) due to propagation delay (calculated as $pd = d/s$, where d is the distance between two points in space and s is the wave propagation speed of the medium, i.e. $s = c$, the speed of light, in wireless communication). After reception we have additional amount of time in order to process and retransmit the message:

* Each PFC spends MAC Processing Delay (MPD) to process the message and then waits for WT_i calculated according to whichever multi-hop algoritm is being used. As explained previously, this waiting time is inversely proportional to the distance between the PFC and fwd;

* After timer expiration, each PFC wait for Rx/Tx turnaround time (Rx/Tx), in order to switch their interface from reception to transmission mode.

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

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

3.1.2 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.

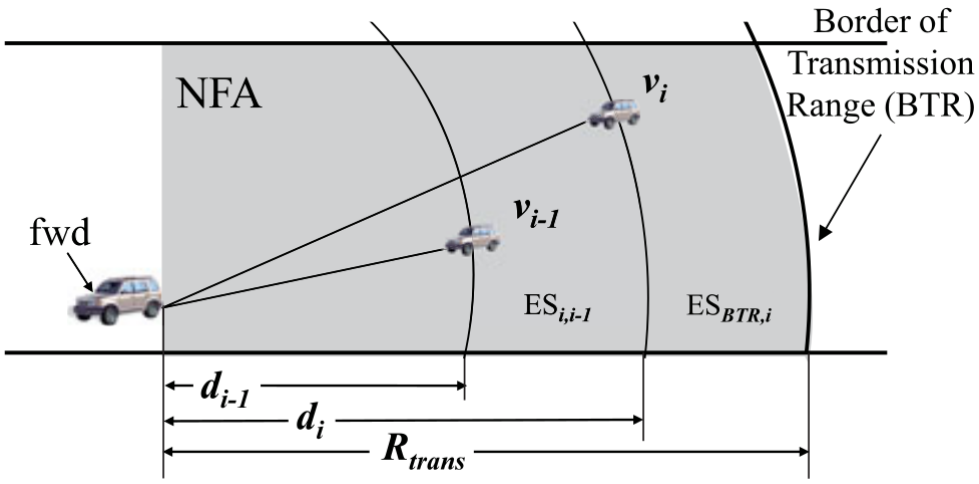

Figure 3.2: Definition of empty space (**6906275**)

Suppose that $S_v = \{v_i | 0 < i \leq M\}$ is a set of M vehicles inside NFA with vehicles ordered by distance between the vehicle and the previous forwarder in ascending order. Referring to Figure 3.2 , the empty space $ES_i, i-1$ between v_i and v_{i-1} is the segment within the two circles centered in fwd with radius d_{i-1} and d_i respectively. Hence, the size of $ES_i, i-1$ is equal to $d_i - di - 1$.

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

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

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

3.2 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 3.3.

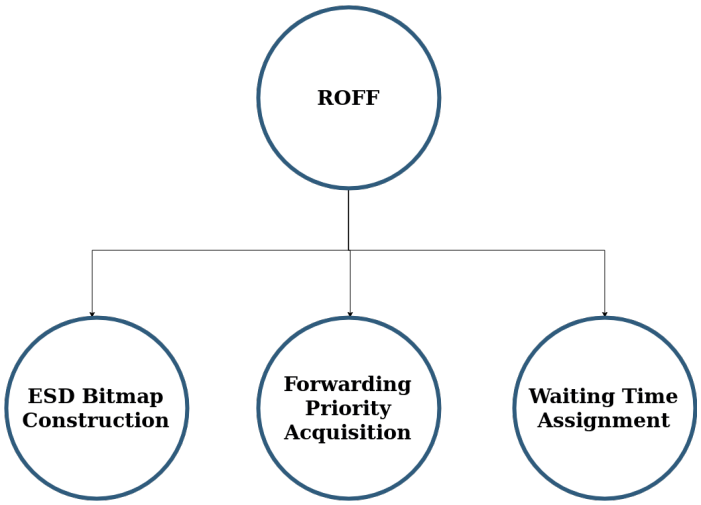

Figure 3.3: Components of ROFF algorithm

Chapter 4

Tools and applications

4.1 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.

4.1.1 Modules and module structure

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

* WiFi;
* AODV;
* CSMA.

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

Modules follow a prototypical structure in order to promote clarity and offer built-in documentation. Figure 4.1 shows the typical module structure.

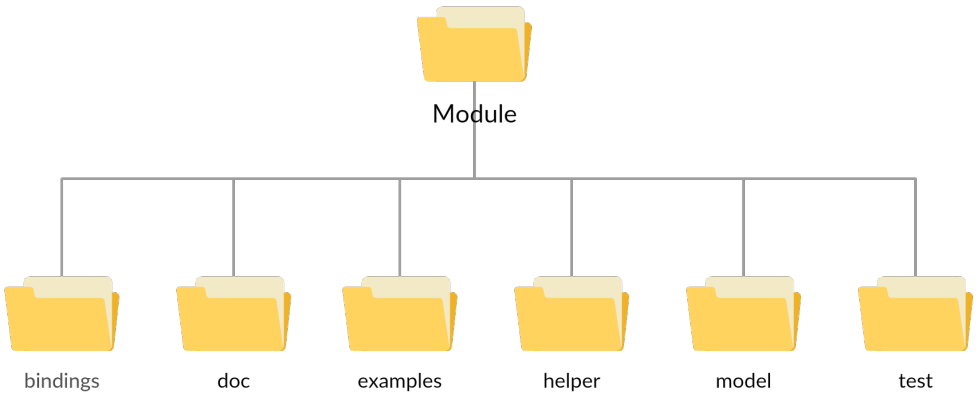

Figure 4.1: ns-3 module structure

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.

4.1.2 Key elements

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

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

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

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

4.1.3 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.

4.1.4 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.

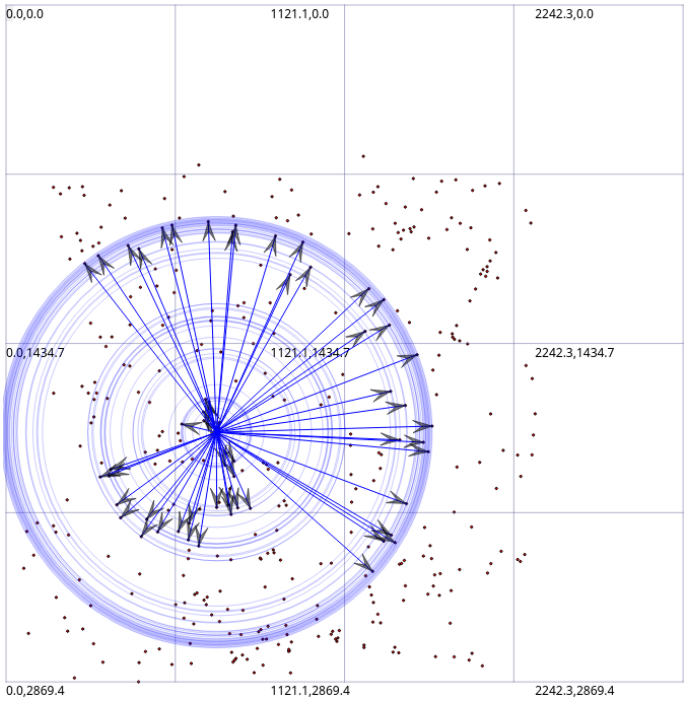

Figure 4.2: Packet transmission in NetAnim

4.2 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 **ROM2017** utilized SUMO to produce, starting from real maps obtained from OpenStreetMap (OSM), 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 4.3.

The original work considered a only a distance of 25 meters between vehicles; this work considers various distances, ranging from 15 to 45 meters. Figure 4.4 shows the same scenario (Padua) with different distances between vehicles.

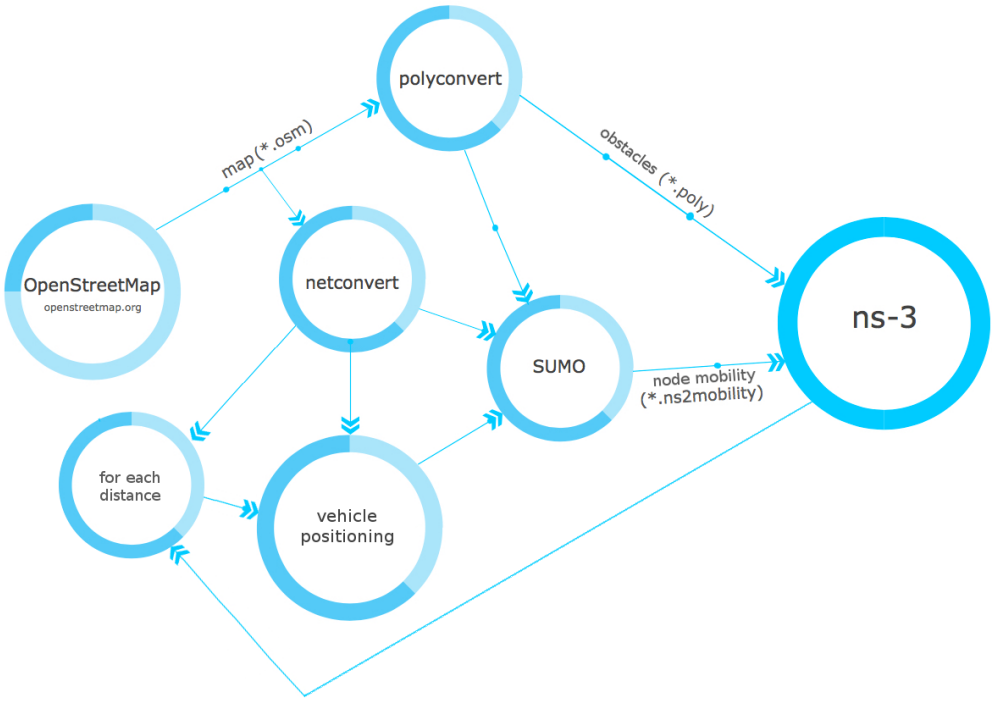

Figure 4.3: Steps to generate necessary files using SUMO (**ROM2017**)

Figure 4.4: Padua scenario with vehicle distance equals to 15 meters (top) and 45 meters (bottom)

Acronyms

LOS Line of sight. 1

NIC Network Interface Controller. 12

RPM Radio Propagation Model. 1, 2

VANET Vehicular Ad-Hoc Network. 3, 7