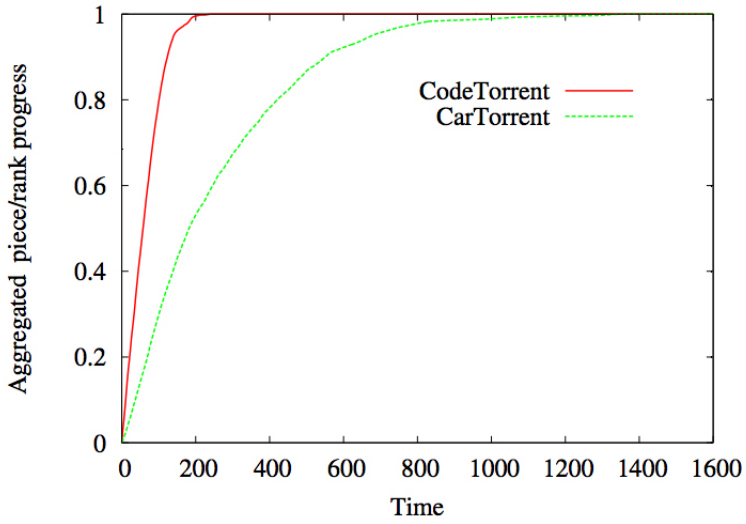

Figure 5.8: CodeTorrent vs. CarTorrent. The figure shows the aggregated downloading progress (200 nodes moving with the maximum speed of 20 m/s). The number of interested nodes is 80 (which means that the *popularity index* is 40%.

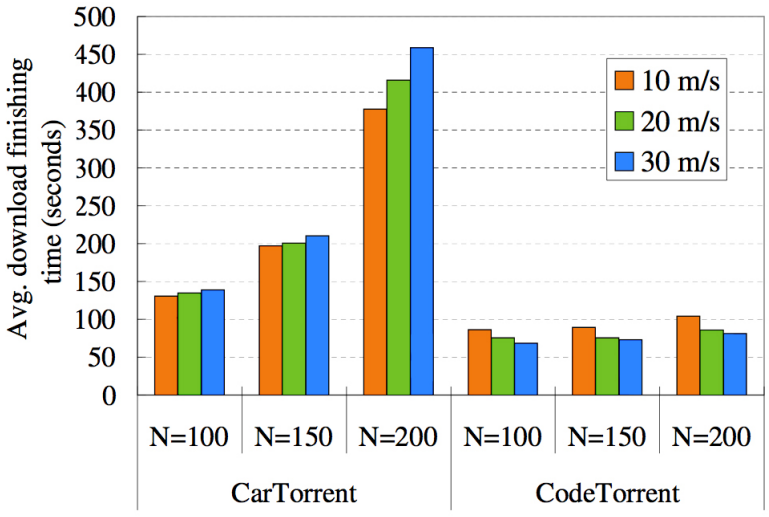

Figure 5.9: CodeTorrent vs. CarTorrent. Impact of mobility on average download delay.

the velocity of mobile nodes. In fact, the number is important for information dissemination but congestion is inevitable as the number of nodes participating gossiping increases. As it concerns the mobility, if it can reduce the dissemination latency on one hand, on the other it can also affect the overall performance because the contact period between two nodes becomes too short to exchange a data fragment.

#	Data Element	# of bits	Signal Violation	Curve Warning	Emerg. Brake	Pre-crash Sensing	Collision Warning	Turn Assist.	Lane Warning	Stop-Sign Assist.	# of uses
1	acceleration	12	•	•	•	•	•	•	•	•	8
2	acceleration precision	3	•	•	•		•	•		•	6
3	airbag count	7					•				1
4	ambient air temp.	8									0
5	anti-lock brake state	2	•		•		•				3
6	brake applied pressure	4	•	•	•		•	•		•	6
7	brake applied status	4	•	•	•		•	•		•	6
8	brake boost applied	1		•	•	•					3
9	driving wheel angle	8		•	•			•			3
10	DSRC message ID	12	•	•	•	•	•	•	•	•	8
11	elevation confidence	4		•							1
12	elevation	20		•		•					2
13	exterior lights	3			•						1
14	heading	16	•	•	•	•	•	•	•	•	8
15	heading precision	3	•	•	•	•	•	•	•	•	8
16	headlights	2						•			1
17	lateral acceleration	12	•		•	•		•	•	•	8
18	latitude of center of vehicle	32	•	•	•	•	•	•	•	•	8
19	longitude of center of vehicle	32	•	•	•	•	•	•	•	•	8
20	obstacle direction	16		•		•	•				3
21	obstacle distance	10	•	•	•	•	•	•	•	•	8
22	longitudinal acceleration	12	•	•	•	•	•	•	•	•	8
23	positioning precision	4	•	•	•	•	•	•	•	•	8
24	rain sensor	3		•				•		•	3
25	siren in use	2	•								1
26	speed	13	•	•		•	•	•	•	•	7
27	speed precision	3	•	•		•	•	•	•	•	7
28	stability control status	3		•	•	•	•	•			5
29	steering wheel angle	16	•	•			•		•		4
30	steering wheel angle precision	2	•	•			•		•		4
31	steering wheel rate of change	8			•	•	•	•	•		8
32	sun sensor	10									0
33	system health	4				•					1
34	throttle position	8	•	•			•	•		•	5
35	throttle precision	3	•	•			•			•	4
36	time precision	4	•	•		•	•		•	•	6
37	temporary ID	48					•		•		2
38	traction control state	2		•					•		2
39	turn signal/hazard signal	2	•				•	•	•	•	5
41	UTC time	40	•			•	•	•	•		5
42	vehicle length	14				•	•	•	•	•	5
43	vehicle width	10				•	•		•		3
44	vehicle heigth	8	•						•		2
45	vehicle mass	8		•	•	•	•				4
46	vehicle type	7		•		•		•			3
47	vertical acceleration	8									0
48	vertical acceleration threshold	4									0
49	wiper rate	8						•			1
50	wiper status	3						•			1
51	yaw rate	16		•							1
52	yaw rate precision	3		•							1

Table 5.2: A subset of the SAE common message set (more than 70 data elements) and their usage in vanet safety applications.

Chapter 6

Data dissemination with rateless codes

6.1 Before you start: digital fountain codes

Digital Fountain (DF) codes are random sparse-graph codes developed for erasure channels[1] and they can be compared to a running tap: when a cup is filled it is not important which droplets land in the cup, but only that enough water is required to fill the cup. The above metaphor also highlights another important property of digital fountain codes namely its ability to generate an infinite amount of encoded packets from the original source.

Actually, a packet layer communication can be modeled as an erasure channel, so that these codes can be used as FEC[2] codes on data packets. Therefore, Fountain codes can be used to enhance performance of the system in lossy channels, where classical approaches typically work poorly. In fact, in these channels, TCP protocol is not efficient because of the high amount of retransmissions required to receive the source. In

[1] An erasure channel is a channel in which each codeword symbol is lost with a fixed constant probability p in transit independent of all the other symbols.

[2] In telecommunication and information theory, Forward Error Correction is a system of error control for data transmission, whereby the sender adds redundant data to its messages, also known as an error correction code. This allows the receiver to detect and correct errors (within some bound) without the need to ask the sender for additional data. The advantage of forward error correction is that a back-channel is not required, or that retransmission of data can often be avoided, at the cost of higher bandwidth requirements on average. FEC is therefore applied in situations where retransmissions are relatively costly or impossible. In particular, FEC information is usually added to most mass storage devices to protect against damage to the stored data.

similar system conditions, UDP protocol might also perform poorly because it does not check the source integrity, that will be then granted by the upper layers. A solution to this problem can be a previous rateless encoding of the source data and a followed by a transmission using UDP. This would eliminate the need for retransmission since it is only critical that the decoder receives enough symbols. By using these codes the source data can be recovered from any subset of encoded packets, given that enough packets are received. A DF encoder generates can be thought of as a fountain that produces an endless supply of water drops. The water drops are encoded symbols (ESs). Similarly, a DF decoder can be thought of as a bucket that collects water drops until it reaches capacity. Given that the number of symbols that can be generated from the source data is potentially innite and that every symbol can be generated on the y, it is not possible to determine the rate in advance. Thus, these codes are also called rateless.
The generation of an ES is based on GF(2). In particular, each EC is obtained by XOR operations (ex-oring) on a certain number of source symbols chosen uniformly at random. The number of source symbols that that are chosen is given by a outer degree distribution that relies on the particular code. Examples of this distribution can be found in (58; 66). The probability of generating two or more ESs with the same information is negligible. Consequently, each ES has equal importance and, on average, it carries the same amount of information as the others. The number of different symbols to be collected from the receiver in order to recover the source info is $(1 + \epsilon) \cdot k$, where k is the number of source symbols and ϵ is the coding inefficiency (usually, we have that $\epsilon \ll 1$). However, the impact of this inefficiency on the overall performance of the system is the lower the more lossy is the channel. In fact, It can be demonstrated that for $k \to \infty$ these codes have $\epsilon \to \infty$, so they are potentially optimal(58). This means that, apart from the decoding inefficiency, it is possible to recover the source information as soon as enough ESs are received. Moreover, these codes are universal because the symbol length for the codes can be arbitrary. In fact, a symbol can extend from one-bit to general l bits, without affecting the coding and decoding effciency.

6.1.1 LT codes

Luby Transform (LT) codes were the first rateless code used to approximate a fountain. They are a class of asymptotically optimal rateless erasure correcting codes introduced by Michael Luby in 2002 (58). Similarly to LDPC (low density parity check) codes,

LT codes rely on a sparse bipartite graph representation to trade reception overhead for encoding and decoding speed. These codes transmit random linear combinations of message symbols, where the number of message symbols contributing to each transmitted symbol is chosen at random with a given probability distribution.

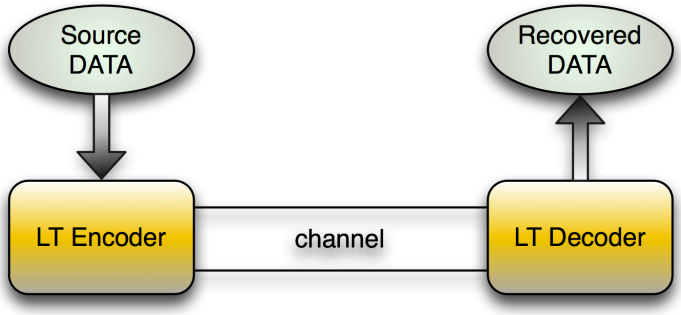

Figure 6.1: LT code. Functioning scheme of LT codes.

LT codes are rateless, which means that the number of encoded symbols that can be generated from the data is potentially limitless. In other words, their design does not depend on the estimate of erasure probability of the channel, so they can simultaneously serve a heterogeneus population of receivers efficiently. Furthermore, distinct encoding symbols can be generated on the fly by the server as needed. As a consequence, in order to let the decoder know which message nodes are the neighbors of a particular encoding symbol, Luby suggests that one solution is to explicitly include this information as an additional overhead in the packet. Another possibility is to replicate the pseudorandom process at the receiver by supplying it with the suitable seed and/or keys (i.e, as applied in CORP protocol). The k original data blocks are obtained by partitioning the original data into k uniform segments of l bits each. An encoded symbol is generated in the following manner:

1. *choose the degree d, that is the number of source symbols that form the encoded symbol. The selection of the degree is random, given a degree distribution ρ;*

2. *uniformly choose the d symbols that will contribute to the coded info. This symbols will be named neighbors;*

3. *the coded symbol info is obtained from a bitwise exclusive-or of the d neighbors.*

Description	**Expression**
message symbols	n
probability to reconstruct message symbols	$1-\delta$
symbols to receive	$n + O(\sqrt{n} \cdot \log^2(n/\delta))$
time for decoding each symbol	$\propto O(\log(n/\delta))$

Table 6.1: Main characteristics of an LT code.

The effect of the said procedure is exactly the same as multiplying a message with a dynamic random G matrix.

The choice of the degree i has to follow a particular probability distribution ρ. In particular, the probability of selecting a degree i is ρ_i, so that the distribution can be dened as $\rho = (\rho_1, \rho_2, \rho_3, ..., \rho_k)$, where $\sum_{i=1}^{k} \rho_i = 1$.

The design of a good degree distribution is very important in order to achieve good performance. Luby proposed to use the *Robust Soliton Distribution*, like presented in [(58)] that depends on the block size k and is characterized by the two parameters δ and c. By using this distribution, the creation of each encoded symbol requires $O(log(k/\delta))$ logical ex-or operations per each generated symbol.
The Robust Soliton distribution ensures that the ripple size, i.e. the number of degree-one encoded symbols, is large enough to not extinguish, but it also assures that the ripple is not too large in order to avoid redundant packets. In this way, in order to decode the original data with a $1-\delta$ chance of success, any $k + O\left(\sqrt{k} \cdot \log^2(k/\delta)\right)$ encoded symbols are sufficient.

In order to decode a fountain coded message, we have to go through the following steps:

1. find a parity bit p_n that is connected to only one message bit m_k (if there is no such parity bit, the decoding algorithm stops and fails to recover all message bits);

(a) Set $m_k = p_n$;

(b) Add m_k to all parity bits $p_{n'}$ that are connected to m_k:
$p_{n'} := p_{n'} + m_k$ for all n' such that $G_{n'k} = 1$;

(c) remove all the edge connected to m_k;

2. Repeat step 1 until all m_k has been recovered.

In order to better understand the said procedure, please consider the example of Fig. 6.2, which represents the decoding process of a very simple LT code on erasure channel. In the image, there are three message bits $(b1, b2, b3)$ and four parity bits $(p1, p2, p3, p4)$, which at the very beginning of the decoding phase are set to the vector [1011]. According to the algorithm, during the rst iteration, the only parity bit that is

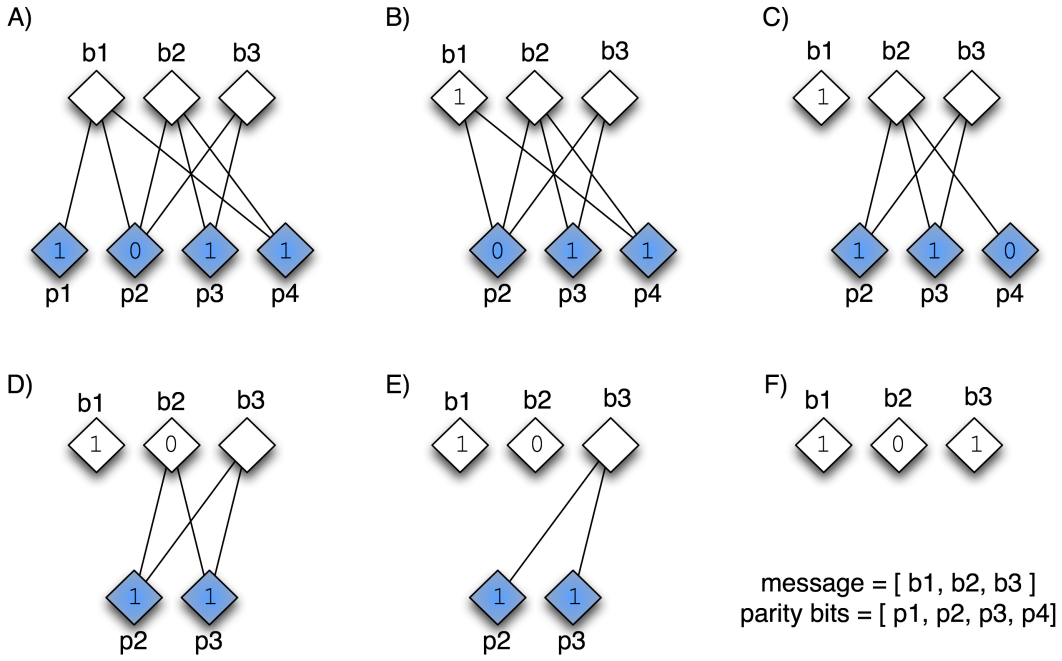

Figure 6.2: LT example. Decoding of a simple LT code on an erasure channel.

connected to one message bit is $p1$ (see Figure 6.2-A), whose value is then copied to $b1$ (see Figure 6.2-B) and then the relevant parity bit removed from the graph. The new value of $b1$ is added (xor-operation) to $p2$ and $p4$ and thus becoming disconnected from the graph. At the start of the second iteration (Fig. 6.2-C) the value of $p4$ is copied

into message bit $b2$ as it is the only bit connected to it (Figure 6.2-D), and then add this value to $p2$ and $p3$ (Figure 6.2-E). Finally, it is clear that the parity bits connected to $b3$ are equal as expected and can be used to restore $b3$ (see Figure 6.2-F).

An important property of DF codes is that the probability of generating two identical ESs is negligible. In particular, for LT codes we can compute this probability as follows. Let us suppose that N ESs are generated and that the ratio of symbols with degree i is equal to ρ_i . Then, we can define the number of ESs of degree i equal to $h_i = N \cdot \rho_i$. Obviously, we have that $\sum_{i=1}^{k} h_i = N$.
Since the possible number of encoded symbols of degree i are $\binom{k}{i}$, thus *the probability that two i-degree ESs are identical* is:

$$\frac{\binom{k}{i}!}{\binom{k}{i}^{h_i} \cdot \left(\binom{k}{i} - h_i\right)!} = \frac{\prod_{j=0}^{h_i-1}\left(\binom{k}{i} - j\right)}{\binom{k}{i}^{h_i}} \tag{6.1}$$

From (6.2) we can easily evaluate *the probability that non of the generated ESs is repeated*:

$$\prod_{i=0}^{k} \frac{\prod_{j=0}^{h_i-1}\left(\binom{k}{i} - j\right)}{\binom{k}{i}^{h_i}} \tag{6.2}$$

6.1.2 Tornado codes

Tornado codes are erasure block codes based on irregular spare graphs. Given an erasure channel with loss probability p, they can correct up to $p \cdot (1 - \epsilon)$ errors. They can be encoded and decoded in time proportional to $n \cdot \log(1/\epsilon)$. Thus, Tornado codes has been primarly designed to speed up erasure codes over the internet. These codes can be designed over arbitrary alphabet size.
Clearly, the main benefit of Tornado codes, is the linear time decoding, which is the result of a very simple decoding algorithm. At each step, a right node is selected whose all but one neighbors are known. The missing neighbor is then computed by performing a XOR operation beetwen the check bit and the known inputs bits. If at any step, no such node is found, then an error occurs. The algorithm terminates successfully when all the input bits are recovered.
Obviously, the algorithm shows a linear time, since at each step we recover one lost

input bit and it is also not optimal, since it can fail even when it would have been possible to recover the input bits through say gaussian elimination.

6.1.3 Raptor codes

Raptor codes, presented by Shokrollahi in (89), extend the idea of LT codes one important step further. In fact, one of the features of LT codes is that they need an average degree $O(\log k)$ to code at least once every source symbol with high probability. The solution implemented by Raptor codes is to pre-code the message M by means of a fixed length erasure code, such as a Tornado code (17) or LDPC codes (83), leading to a new encoded message M' which is encoded with LT codes. Hence, Raptor Codes are LT codes with a preventive encoding as shown in fig.6.3.

The main advantage of the precoding stage is that, for correctly decoding, it is not

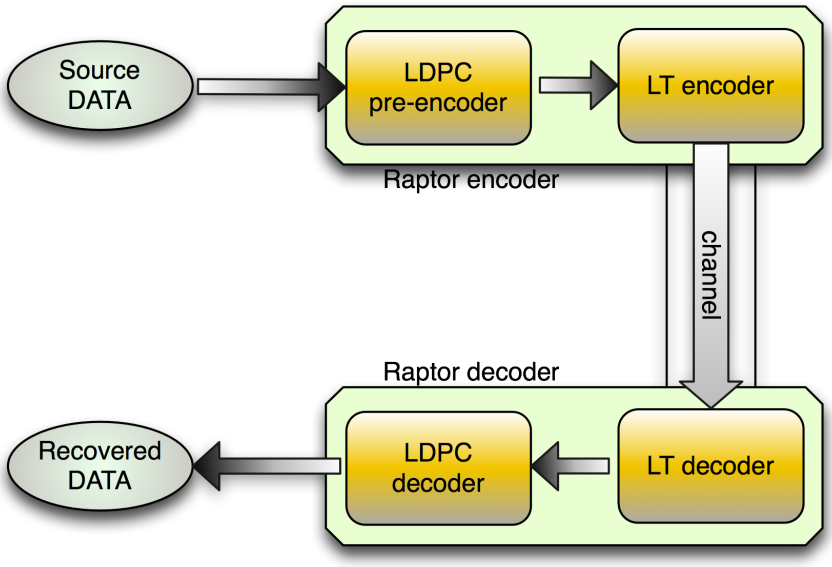

Figure 6.3: RAPTOR coding. Functioning scheme of RAPTOR codes.

necessary to decode all symbols of the message M' , but only a constant fraction of these. Thus, it is possible to use a simpler degree distribution that does not recover all the symbols but makes the decoding process faster. By using a precoding stage and a simpler degree distribution, the bound $O(ln(k))$ on the average degree of LT codes no longer applies. Indeed, thanks to pre-coding with an appropriate design, for any

constant $\epsilon > 0$ and sufficient large k, the original message M can be correctly decoded given that $(1 + \epsilon) \cdot k$ symbols have been received.
It can be proved that the value of ϵ tends to zero as k increases. This means that the property of optimality of digital fountains is veried also for these codes.
Shokrollahi proposed in (89) a new degree distribution that depends only on the overhead and has a maximum degree far lower than k. In particular, the average degree is no longer dependent on the logarithm of k, but it is constant, being $O(\ln(1/\epsilon))$. The constant degree leads to a total decoding time linear with k, i.e. $O(k \cdot \ln(1/\epsilon))$. In the following, we will refer to the new distribution as *Shokrollahi distribution* and to the new LT code as *weakened LT code (wLT)*. In fig.6.4 we can observe a comparison between Robust Soliton Distribution and Shokrollahi Distribution. As we can observe, the second peak is lower and the probability of symbols with degree one or two is much larger than using RSD.

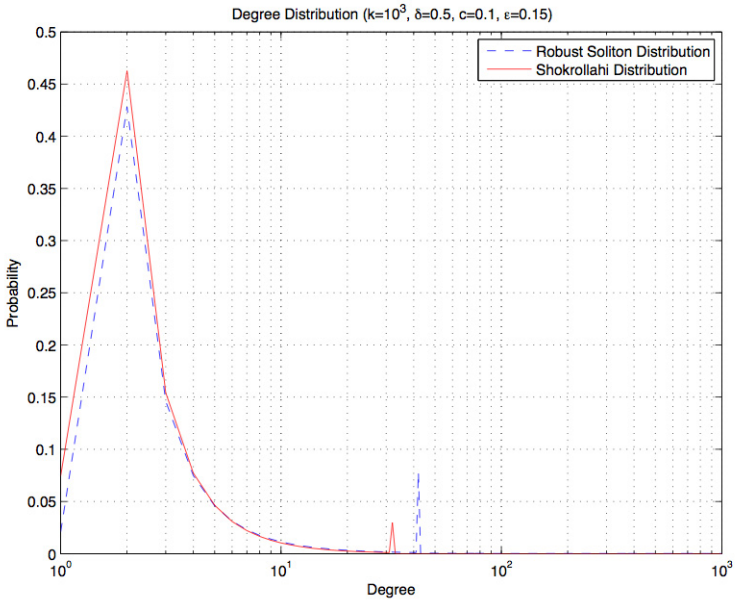

Figure 6.4: RS-SD comparison. Comparison between Robust Soliton and Shokrollahi Distributions.

The decoding algorithm is composed of two steps. The inner LT decoder returns a hard bit-reliability vector. This latter is processed by the outer LDPC decoder, based on belief propagation algorithm. The main drawback of Raptor codes is that the total

overhead is lower bounded by the overhead of the pre-code. On the contrary, LT codes have no overhead asymptotically.

Code	**Negatives**	**Positives**
RS	− Limited distinct decoding symbols − Quadratic decoding algorithm	− Can decode with k encoded packets
Tornado	− Need more than k encoded packets to decode − Limited distinct decoding symbols	− Decoding time proportional n
LT	− Need more than k encoded packets to decode	− Decoding time proportional to $k \cdot \log(n)$ − Unlimited distinct decoded symbols
Raptor	− Need more than k encoded packets to decode	− Decoding time proportional to k (due to pre-coding)

Table 6.2: Comparison of different coding schemes used to approximate a digital fountain code.

Moreover, similarly to LTcodes, Raptor Codes are not systematic. This means that the input symbols are not necessarily reproduced among the output symbols. The straight forward idea of transmitting the input symbols as a preamble of the symbols can be easily shown to be awed, since this does not guarantee a high probability of decodability from any subset of received output symbols (89). Shokrollahi introduced a new systematic version of Raptor Codesin (89) that has then further developed in (59).
In streaming applications, the effects of even small levels of packet loss are indeed apparent to the end user. In ahead-to-head comparison of Raptor and Reed-Solomon (RS) erasure codes, used to protect streaming applications, the following conclusions can be drawn:

- Raptor codes are more efficient than comparable RS code in protecting against packet loss.
- Raptor codes require significantly less processing power than Reed-Solomon erasure codes for encoding and decoding; the required processing power for Raptor

codes increases linearly with the level of provided protection, whereas, for Reed-Solomon erasure codes, it increases quadratically.

- Raptor codes allow a given application to be optimally addressed in terms of the degree of packet loss protection, bandwidth expansion and processing demands.

6.2 An application of fountain codes: CORP

6.2.1 Introduction

This chapter presents CORP (19) (Cooperative Rateless Protocol), a new rateless protocol for data dissemination designed for vehicular networks which exploits the reliability of the rateless coding approach while performing fast and efficient dissemination through cooperating nodes within the network. The application that we consider for CORP is e-commerce and e-advertising. When a car approaches a pre-dened area, information about local commercial offers is provided. Such information may range from common goods to sophisticated service offerings and may also include events and similar entities. Via vehicular communications, vehicle passengers can request more information and, if electronic commerce is supported, consumers can immediately offer and continue with the transaction. As an example, a car driver may inquire about the price of a particular good at various local businesses and selects the best price/value offering. Alternatively, a local business can advertise to car drivers in a certain area a new offer (e.g., restaurant menu, etc.). CORP protocol implements the information dissemination process between local businesses and vehicles.
Current data dissemination protocols for vehicular networks still lack the reliability and the efciency required by most applications, ranging from safe navigation to content delivery. One of the main problems is the unreliable connectivity. In fact, existing dissemination methods (e.g., BitTorrent) use TCP for data transmission among nodes. Unfortunately, due to high vehicular mobility, TCP does not perform well and cannot consistently guarantee efcient communications and as a result, frequent packet loss affects both window size and transmission efficiency.
On the other side, the use of UDP for data transfer may not be suitable for dissemination in vehicular networks. In particular, the lack of acknowledgements from the receiver may trigger repeated transmissions regardless of the status of the transfer. This

problem can be solved by allowing the receiver to send a packet to the sender to stop the data transmission when it has received the information. As is well known, UDP cannot guarantee the reliability of the data transfer. In fact, in the event that even a single packet is lost, the information will not be received correctly. The classic solution involves using *erasure-correcting* codes. By using these codes, a subset of encoded packets will allow to recover the original content. Among all the erasure-correction techniques presented in literature, rateless codes are the most interesting ones. In fact, they provide low encoding and decoding complexity and are especially suitable in networks where the channel conditions are unpredictable and unknown. These are often referred to as *channel oblivious* transmissions, because the number of encoded packets that can be generated is potentially infinite.
The use of erasure-correcting codes for the dissemination of data not only improves the reliability of the transmission but also helps to address the problem known as *coupon problem* (56). This problem is typical of P2P networks where the amount of time needed to obtain the last missing piece of the data can be very long. By generating encoded information and disseminating it cross the network, it is possible to increase the information availability.
Thanks to their unique characteristics, rateless codes have been widely used for data dissemination over the Internet. Yet, another problem that P2P systems face is how to efficiently exchange data among peers. At the beginning of the communication, peers have to exchange information about the packets they want to receive in order to efficiently use the available bandwidth by avoiding duplicates. In this work we will call this phase the content reconciliation phase, that is heavier, the larger the information is. *Speed* is a key factor in the design of a communication system in vehicular networks and is therefore critical that nodes exchange information about their contents as fast as possible. This means that the efficient techniques have to be developed in order to minimize the content reconciliation phase.
With CORP protocol, we aimed to address the following issues:

- reliable and fast data dissemination in vehicular networks;
- maximization of the content availability on the network;
- minimization of the content reconciliation phase cost.

In addition to our primary goals, we also tried to design a protocol that does not demand high computing resources, thus extending our solution to portable devices. In the following sections, a preliminary analysis of the performance of CORP is presentes. Basically, we focus on these three metrics:

1. delivery ratio after 15 minutes as a function of the nodes that collaborate in the dissemination;
2. delivery ratio as function of time;
3. average number and size of the sets of encoded packets that are exchanged among nodes.

For the analysis we adopted a first a random-waypoint mobility scenario (RWP) (22). Although this model is only an approximation of vehicles behavior, it is still useful to understand the asymptotical performance of the protocol. In order to provide more accurate results, additional tests have been performed with pseudo-real traces, generated with Vanetmobisim(35) on a real map imported from Open Street Map[1].

In section 6.2.7, CORP is compared to other two similar approaches. The first one, *Basic Rateless Protocol* (BRP) considers a rateless dissemination from a single access point to all the interested vehicles in its range, so that the nodes can recover the information if they have collected enough encoded packets. Since there is no cooperation among vehicles, the communication will be of the type infrastructure-to-vehicles (I2V). In the second one, mobile nodes can disseminate the source information as after having decoded it. Therefore, we will call this approach DDRP (i.e., *decode and disseminate rateless protocol*). Please, note that the latter is similar to the approach presented in (104).

6.2.2 Related work

The extant literature presents many proposed solutions for data dissemination over vehicular networks (introduced in (74) and (75)). The most interesting solution ex-

[1]Open Street Map (OMS) is a free editable map of the whole world which allows you to view, edit and use geographical data in a collaborative way from anywhere on Earth. The website is: http://www.openstreetmap.org.
OMS hosting is kindly supported by the UCL VR Centre and bytemark. Other supporters of the project are listed in the wiki.http://www.openstreetmap.org/

ploits network coding in order to obtain high bandwidth efciency. However, packet loss can deeply affect the performance of these methods. This problem has been recently addressed in (80), where errors and erasures can be tolerated. This method generates codes on a vector space and avoids the overhead of the encoded set description while simultaneously providing highly efcient transmissions. Unfortunately, this is not a desirable t for our scenario as the the recoding phase of network coding pproach is computationally more expensive than a smart selection of the received information to be forwarded to other nodes. Byers et al. in (43) study the content reconciliation problem in wired networks. In their study, two nodes compute the resemblance of their information in order to understand if the communication can be useful. The authors also present appropriate tools for an effective content reconciliation. Although the methods presented in their paper are very interesting, we argue that the performance in wireless vehicular networks would be poor. For example, to perform the content reconciliation methods, nodes would be required to periodically exchange packets to update the neighbors on its own representation of the information. Additional time would also be required to compute the information that is to be sent. In sum, the content reconciliation phase would require too much time.

Yet another solution, *Bullet*, is found in (50). This method can use either rateless or multiple description coding to efciently disseminate data. It efciently solves the problem of choosing the best nodes to talk to in an overlay mesh network. Moreover, it uses a simplied version of TCP to fulll the requirements of multimedia transmission that is also TCP friendly. Performance analysis shows that Bullet is efcient on wired networks but again performs poorly in vehicular networks for reasons mentioned earlier. The work in (67) simplies the content reconciliation phase by sequentially labeling the received packets and by exchanging the sequence number ranges in order to perform a simple content reconciliation. Unfortunately, this solution can work only in wired networks where packet loss is almost negligible. In fact, when loss happens, the number of subranges of packets to be transmitted grows, thus decreasing the efciency.

Finally, a method for P2P streaming in wired networks, rStream, is presented in (104). rStream is based on the encoding of the source using rateless codes. According to this method, every node forwards the information after having decoded it. Furthermore, availability of information is assured if each node generates a set of symbols that is independent of the others in the network. This allows symbols to be received from

several nodes at the same time without the need for content reconciliation. Unfortunately, a similar approach in vehicular networks would perform poorly because it is based on the implicit assumption that connections between source node and interested nodes have sufcient time to receive all the encoded information needed for recovery. However, this may not happen in vehicular networks because a node can be connected to APs or vehicles for only a limited time. Therefore, the transmission of the encoded information may not be quickly completed.

6.2.3 Proposed approach

The following Section presents the main ideas of the proposed protocol. The considered scenario is based on access points (APs) and mobile nodes (MNs). APs collect information from external sources and deliver it (or fractions of it) to interested MNs that will disseminate it to other vehicles. In other words, interested vehicles cooperate with each other to disseminate information. APs can be xed or mobile nodes, or a mix of those. However, for simplicity, this paper considers only fixed APs. The dissemination approach can be divided in two different phases:

1. The first phase is the *dissemination between AP and MN*. This phase starts when a MN communicates its interest in a certain information that an AP is advertising.

2. The second phase pertains to the *dissemination among MNs*, in which nodes cooperate to disseminate the received data.

The dissemination of the information is performed by exploiting the characteristics of rateless codes. These codes have been presented in (58) and in (66) and implement the digital fountain idea.

From a block of K source information symbols, the rateless encoder can produce a potentially innite amount of encoded symbols (ESs). The generation of the ESs is a based on random ex-oring of source symbols that depends on a random sequence generator that is assumed to be known to both encoder and decoder (see section 5.1). The ESs generated from a random sequence (that is created from a particular *generation seed*) form a set of ESs that we call Λ. Given the random nature of the encoding process, the probability of generating two identical ESs is negligible for sufciently large information blocks. This means that two different sets (i.e., created from two different

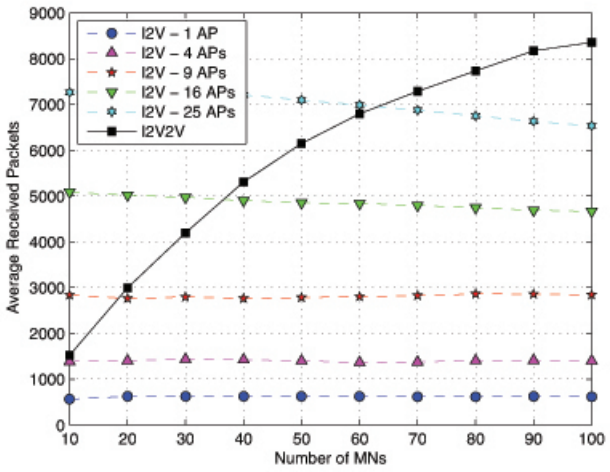

Figure 6.5: I2V vs. I2V2V The figure shows the benefit of an I2V2V approach (like in CORP) over the classical I2V, where data are disseminated only by APs.

seeds) contain different ESs with high probability, i.e., the sets are *disjoined*. As an example, we can consider a le of 512 KB composed of 8000 source symbols of 64 bytes. The block is encoded with different generation seeds, S_a, S_b, S_c, etc., thus generating two disjoined sets of ESs $\Lambda_a, \Lambda_b, \Lambda_c$, etc. Given that enough symbols have been received, the rateless decoder is able to recover the source block with high probability starting from any subset of distinct received ESs. In particular, the received ESs can belong to different sets of ESs without any loss of performance. The number of symbols N that have to be received is slightly greater than K. This means that the decoding process introduces an *inefficiency* $\epsilon = N/K - 1$. Usually, this inefficiency is very small and it can be demonstrated that it tends to be zero when the number of source symbols tends to innite.

The proposed approach is based on the dissemination of sets of ESs, rather than on single packets. This means that for a given block, nodes try to exchange all the ESs belonging to a set that is not in common (e.g., Λ_a), before exchanging ESs from another set (e.g., Λ_b). This method is advantageous in that ESs are transferred in groups, thus reducing in this way the content reconciliation complexity. In fact, since sets are univocally identied by the generation seeds, the content reconciliation algorithm is reduced to the research of the seeds that are not in common between two communicating nodes. For example, let us suppose that a node has received the ESs belonging to the

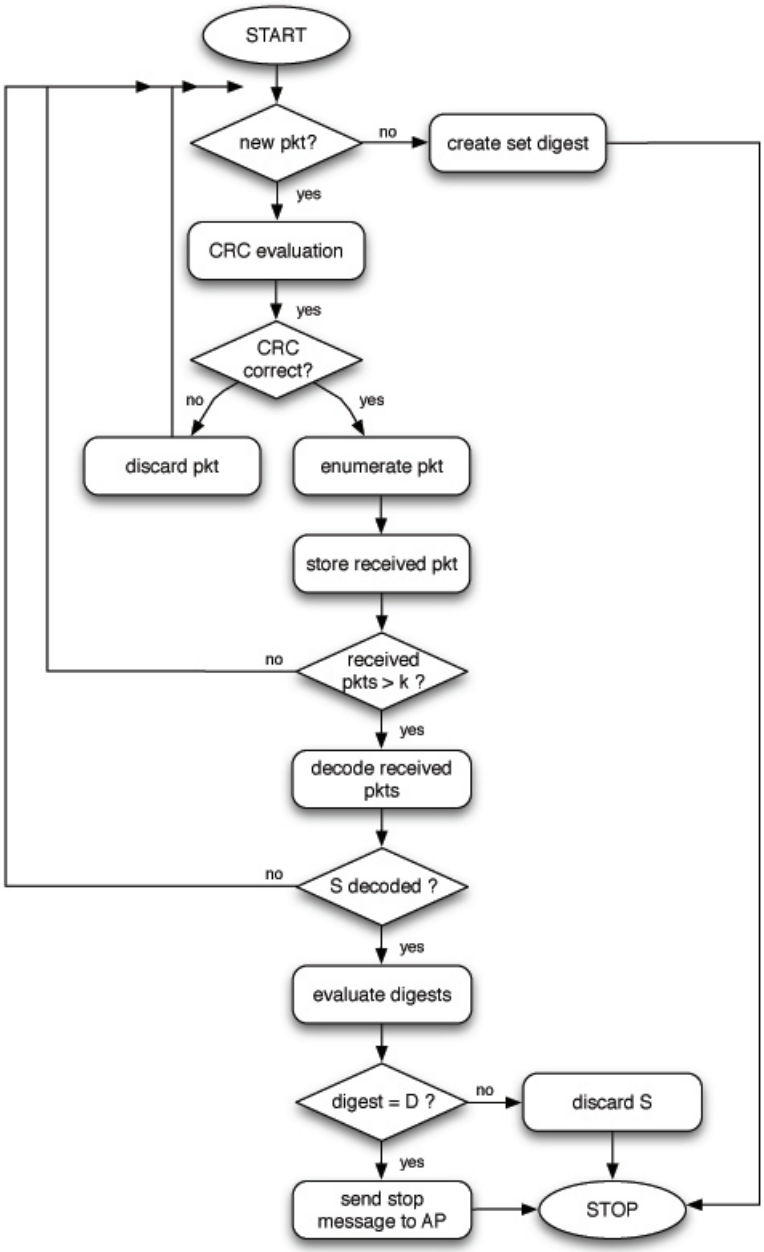

Figure 6.6: CORP Algorithm. The diagram shows the receiving phase of the CORP algorithm.

sets Λ_a and Λ_b . Instead of understanding which packets can be received, the node can just require ESs belonging to sets different than the ones associated to S_a and S_b.
The dissemination process has to follow four basic rules in order to be efficient and fast. The first rule is that a node that owns the whole information disseminates it by generating new disjoined sets of ESs and delivering them with unicast connections. This allows APs to increase the information availability by generating unique sets of ESs (i.e., generated from a new generation seed) per each new connection. By providing always

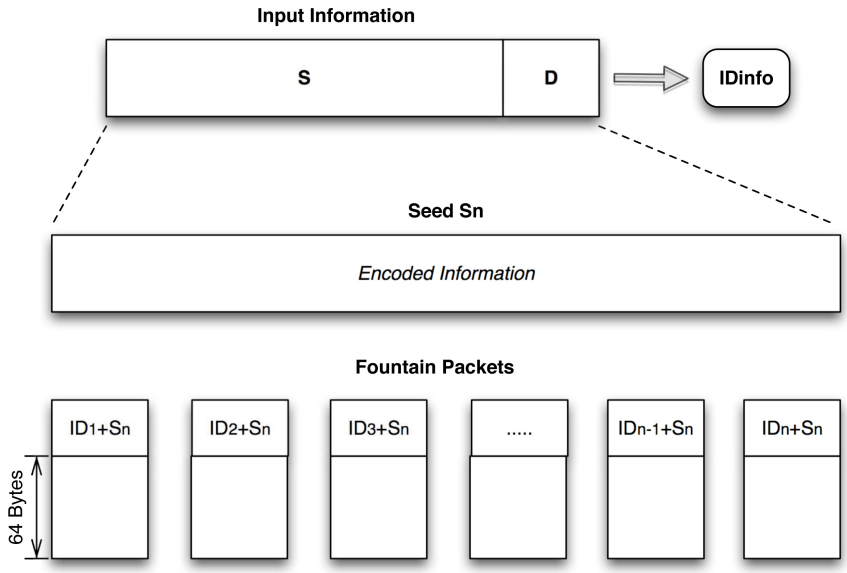

Figure 6.7: Coding in CORP. The figure shows the encoding symbols generation process in CORP protocol.

disjoined sets of ESs, the content reconciliation phase is simplied. In our approach, the use of unicast connections is more efficient than using broadcasting. In fact, despite its simplicity, broadcasting would not increase the information availability, and an expensive content reconciliation phase would be necessary.
The second rule we introduce is that a node cannot disseminate ESs belonging to the set it is receiving. This rule avoids the drawbacks of broadcast transmissions, restraining the uncontrolled spread of ESs that would yield to a long content reconciliation phase. For example, let assume that node A is receiving Λ_a from the AP and that the rule is not used. At a certain moment of the transfer, node B arrives close to node A and

receives some of the ESs belonging to Λ_a before leaving. In this way, node B will have only a subset of Λ_a . If in the future node B want to receive the missing ESs belonging to that set, then it will have to perform the content reconciliation phase on the ESs.
A particular situation in the dissemination scenario is when a node can receive ESs belonging to a certain set from two or more neighbors. The scenario requires a rule to dene how the downloading activity is performed.
The third rule is that a node cannot download information of a certain set of ESs by multiple nodes at the same time. According to this rule, a node can receive multiple sets at the same time only if these are disjoined. In this way, there is no need for coordination among the sending nodes. This is useful because it avoids the case in which nodes that are not within each others radio coverage serve the same requesting node with the same set. Although we stated that only one node can service a certain set, we still have to establish which of them will do it. In fact, because of poor channel conditions, nodes might not have received the same number of ESs belonging to that set. The fourth rule that we introduce is called certication strategy. According to this strategy, a node can disseminate a set only if it owns all the ESs of that set that are present on the network. Thus, a received set of ESs is said to be certied if the receiving node has indeed received all the ESs that are present on the network. For simplicity, we also say that a node is certied for a set if only if it owns all the ESs generated from a certain generation seed that are present in the network.
By using the certication strategy, we further simplify the content reconciliation strategy. In fact, the content reconciliation is performed only on certied sets. If the conditions of the channel are optimal, i.e., no packets are lost, then all sets will be certied. This would yield an optimal dissemination of the sets. However, when the channel conditions are poor, the certication strategy limits the transmission of sets only to nodes with enough information to deliver. Therefore, the strategy might yield a sub-optimal dissemination of the information. However, the content reconciliation time will be minimal, so that most of the connection time (that, in this case, would be very short) can be dedicated to data transmission. Moreover, even if a node cannot disseminate a set that is not certied, the ESs of that set are still useful for the decoding of the information block.

6.2.4 Packet definition

The CORP application packet is made up of an header and a payload. The header contains a field, Type, that indicates the payload type. The payload contains the data corresponding to the type indicated in the header and, optionally, a CRC to avoid data corruption. There are eight different types of packets:

1. *Beacon*
   This packet is broadcasted by AP nodes to publish the list of the available information files. Each file is identified by a unique tag (4 bytes).

2. *Request*
   This packet is broadcasted to request a specific file.

3. *Offer*
   This packet is sent either by AP or MN nodes as a response to a request for a file. In this packet, the seeds S of the certified sets Λ and the parameters needed for the decoding process are specified. For example, it is indicated that the block is composed of 8000 source symbols of 64 bytes and that the adopted code is a LT code with given parameters δ and c. The packet has also a field that indicates if the offering node owns the whole file.

4. *Ack*
   This packet is sent by the requesting node to communicate the set of ESs it wants to download by indicating the generating seed set. In addition, the requesting node can also specify the Starting Sequence Number (SSN), which is the first ES of the set that it wants to receive.

5. *Data*
   This packet contains the encoded data and a Sequence Number (SN). The encoded data is a group of one or more ESs belonging to the same. SN indicates the number of the packet in the generated set of ESs.

6. *Stop*
   This packet is sent by the requesting node to terminate the download of a certain set.

7. *EoS*
   This packet is sent by the offering node to communicate the end of the transmission of the set of ESs it is providing to a requesting node. The packet includes also a digest computed on the certified set, useful to the receiving node to check if the set can be tagged as certified.

6.2.5 Communications

The communication protocol is made up of two phases: the communication between AP and MN (Phase 1) and between two MNs (Phase 2). The information file is composed by the data information and a digest computed on the data file. The whole information file is then disseminated according to the CORP protocol. For simplicity, in the following we present the case where a file is composed of one block.

6.2.5.1 Phase 1: AP to MN

The AP periodically advertises the list of the available information files by broadcasting Beacon packets. If a MN within the radio coverage area of AP is interested in receiving one of the files, it broadcasts a Request packet specifying the relative tag. In particular, starting from this moment, the MN periodically sends a broadcast Request packet for the same file until it will complete the download.

When the AP receives the Request, it chooses a unique generation seed for the generation of the set of ESs to be transmitted. In addition, the packet also contains all the parameters that the rateless decoder needs to recover the information. After receiving the packet offer, the MN sends an ACK packet to conrm the settings of the AP for the transmission. Then, the AP can begin the data transmission and will continue until one of the following stop conditions occur:

a) the MN is out of the radio coverage;

b) the MN issues a stop packet.

If the MN realises that it is out of the APs radio coverage, then it assumes that the transmission is nished. As a consequence, if it has not received yet enough packets to decode, it computes a digest of the received ESs set and considers the set certified, because it has all the ESs present in the network. Starting from this moment, the MN

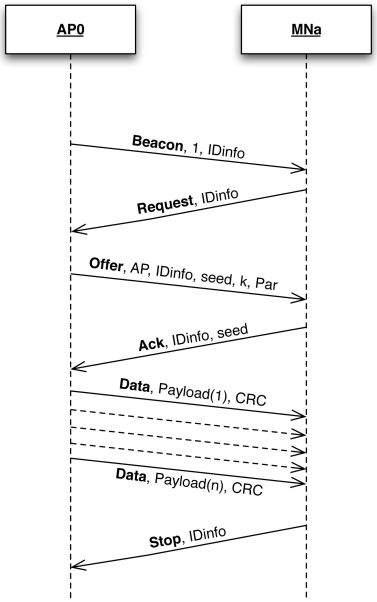

Figure 6.8: AP to MN communication. The figure shows the sequence diagram of the Access Point to Mobile Node communication.

can disseminate the received set of ESs.

If the same MN associates again to the AP and requests the same file, a new session will be opened and a new generation seed will be created. *In this way, the AP increases the availability of the information on network and there is no need for a content reconciliation phase.*

If the MN has received enough packets to decode, then it computes the digest of the recovered information and checks if it corresponds to the one present in the recovered file. If so, then the node has successfully received the source and, if still connected to the AP, then a Stop packet is generated. Moreover, the node will be able to provide the received file to other MNs generating new disjoined sets of ESs. However, if the digests o not match, then the data file is corrupted and will be immediately discarded.

6.2.5.2 Phase 2: MN to MN

In the dissemination between two MNs, there are two possibilities: The first one is when both nodes do not have enough ESs in order to decode the original information, the second one is when one of the nodes has the whole file. In the first case, once a MN