Work In Progress
Reducing Average Retrieval Delay in DTN via Partial Inter-data Coding

Giuseppe Bianchi, Lorenzo Bracciale

Università degli Studi di Roma - Tor Vergata
Dipartimento di Ingegneria Elettronica
Roma, Italy (name.surname@uniroma2.it)

***Abstract*—**

I. Introduction

Erasure codes are great tools for improving performance in several network communication scenarios. One popular example are Digital fountains [1] that allow reliable and efficient communication between two network nodes avoiding the usage of ARQ, making this technique very appealing when a back channel is unavailable or its usage is inconvenient as in the case of video streaming over a wireless medium. In Digital Fountains sources produce a potentially infinite number of encoding packets generated from the K input packets (e.g. a video file) so that a receiver can recover the *whole* original data by decoding a collection of *any* amount of packet whose size is typically slightly bigger than K. Digital fountains can be obtained by means of several erasure codes such LT codes or Raptor codes.

In these rateless codes encoding packets are typically generated by sum of d random input packets in a Galois Field $GF(q)$, where d is the *degree* of an encoding packet. The distribution of encoding packet degrees plays a fundamental role for decoding performance. For instance, LT encoders produce encoding packets by taking d random input packets and summing them modulo 2, where d must be chosen according to the Robust Soliton Distribution [3] so that the decoding procedure is optimized. Growth codes [5] instead increase the degree of codewords during time to the aim of maximizing the probability that data produced by independent sources nodes in a wireless sensor network can be recovered at any time by a sink, preventing data loss due to sensors failure. However, to the best of our knowledge, all the proposed approaches focus their attention on transmission of *all* K input data packets to a receiver, or, in the case of Growth codes, they relies upon a kind of temporal coordination between nodes.

In this work we want investigate which is the best we can do if we are interested in *only one* of the K different data available and there is no coordination at all between the sender and the receiver of the coded data. Being more formally, suppose we have a sender producing a very large amount N of codewords generated from K initial data packets and a receiver that is interested only in one random input data. Receiver collects a random amount of the produced codewords $x < N$ and wants to decode only the data it is interest in. Which is the optimal degree distribution of the encoding packets for minimizing the average decoding delay of receiver? And how much this average delay can be decreased respect to a no-code approach? In this paper we will try to give these questions an answer.

A. DTN reference scenario

For what concern the application field, we presents a Delay Tolerant Network (DTN) seen as distributed storage system, where nodes are capacity constrained and there are K different input data with equal popularity. Suppose we are interested only in one single data over the K available and we wander in the network[1] encountering one random node each time T until we can reconstruct the data we want. If each node can store at most one single packet (or a linear combination of some packets), which is the best distribution encoding packet degree to the aim of minimizing the *average time* we can find the data we are interested in?

II. Related Work

A. Erasure Codes

Since their first use for network communication, erasure codes come a long way.

Reed-Solomon [2] codes are perhaps the first code that can be used -at least in theory- to create a "digital fountain". With this term we refer to the tecnique allowing a sender to produce a potentially unlimited number of encoding data packets from initial K source data packets so that the decoders could reconstruct the whole original data by receiving any collection of N encoding packets slighly greater than K. However the severe limitation on the number of distinct encoding symbols together with the quadratic time require to decode, make Reed-Solomon codes less attractive for this kind of usage.

In his foundational work, M. Luby [3] proposes the LT code, a sparse random linear fountain code with a low complexity

[1]This model is often presented in literature under the name of random encounter, [**?**]

decoding algorithm. LT codes are rateless codes that works as follows: the encoder chooses a degree d_n randomly and according to a degree distribution $\rho(d)$, then it produces a data packet by taking d_n distinct input packets and summing them modulo 2. Decoding is performed by taking encoded packet of degree 1. Those packets actually contain one unencoded input packet so their decoding is trivial. Then the decoder subtracts the found symbols in all the other packets with degree > 1 containing the previously decoded symbols. This could bring some packets to degree 1 and so that can be decoded as well. This procedure iterates till the amount of the received encoded packets allows the full decoding of all the origin symbols. The encoding and decoding computational cost scales as $KlogK$ where K is the file size. Luby found out a robust solution distribution from which the encoder should take the degree of emitted packets to optimizing the decoding procedure. This lead to the desired "digital fountain" result: if the encoded file is composed by K data packet, a little decoding is possibile until slightly more than K packets have been received, then an avalanche of decoding takes place.

LT codes mean degree is $logK$ in reason of the following trade-off: a small degree packets assures rapid decoding but large degree packets lead a better coverage of all the source symbols in the decoded symbols. Raptor Codes [4] fill this gap achiving a linear time encoding and decoding cost by concatenating a LT code with an inner coded so that the original file is coded with the inner code and then coded again using the LT code with a very small average degree 3. During LT decoding, when only a small part (e.g. 5%) misses, Raptor Codes use the inner code to recover the missing part.

Growth Codes [5] are explicitly designed to increase data "persistence" in wireless sensor network defined as the amount of information that can be recovered from a sink at any point of time. The reference scenario is a WSN where nodes at each round exchange their sensed information with their nearby nodes and a sink periodically polls its neighbours trying to decoding as much encoding symbols it can. To this aim, at each round, each network node decides what information it must transmit and how the incoming information from its neighbouring nodes are stored. The stored information and the encoded packet can be a XOR between different input symbols. The degree of encoding stored packets changes during time, so that codewords in the network start with degree one and grow over time as they travel through the network en-route to the sink.

Decentralized Erasure Codes [6] are random linear codes over a finite Galois field whose best application field in a distributed storage. The reference scenario considers a sensor network with k indipendent sources of data and n storage nodes that can store at most one data packet or a linear combination of those. The goal is to minimize the number of data collection queries that a collector should issue towards random nodes for retrieving all the k data pieces. Authors demonstrates they can efficiently diffuse the data by prerouting $O(ln(k))$ packets per data node to randomly selected storage nodes. Despite previous cited works, here each data source acts independently, but the collector is interested in all the data.

B. DTN replication

Content replication is a strategy extensively used in several data-centric networking fields [8], [9], [10]. The benefits of copying a same data item in multiple storage points range from fault tolerance and reliability to performance improvements. In certain scenarios, such as Delay Tolerant Networks, data replication across the moving terminals is at the core of most proposed data access or data delivery solutions [11], [12], as the likelihood that an user interested in a specific data item "physically" meets only the single data producer becomes rapidly negligible as the network size scales.

Which strategy should be employed for replicating data depends on several system-specific aspect extensively addressed in several research works, including coordination among nodes and mobility patterns [13], [14], energy consumption and scalability [15], storage capability of nodes [16], [17], interest in the data by users [18], [19], performance metrics being tackled, and so on.

In this work, rather than further extending the scope of such prior works, we investigate a very basic, foundational, aspect: how data should be best spread across network nodes. We specifically seek to understand what can be done to reduce the *average* data retrieval delay in the assumption that data placement is a static, once-for-all, function (i.e. data not freshly generated and no data forwarding across nodes), no coordination among nodes is possible, no mobility predictions are available, and user interests for data items are uniform.

III. Problem Statement, Notation, and Baseline Results

The elementary system model addressed in the reminder of this work is formalized as follows.

Assume that a large (infinite) number of nodes are available. Let us focus on one of such nodes, which we call the *tagged node*. Such tagged node is assumed to *encounter* any other new node at random, and the difference in time between two consecutive encounters is assumed to be an independent and identically distributed random variable which does not depend on the previous encounters. We conveniently (and non restrictively) assume that the mean time between two consecutive encounters is unitary, meaning that we normalize time and delay measurements in terms of number of encounters.

Assume now that N distinct data items I_i ($i \in 1 \cdots N$) are spread in the network. To avoid formal complications (which do not change the nature of the problem tackled, but would only make the presentation harder to follow), we assume that each node carries exactly one data item. Obviously, since the number of nodes is assumed to be much larger than the number of data items, these items are replicated over multiple nodes. It is convenient to model the way data items are spread in the network through the probability distribution ψ_i that a data item is found over a randomly chosen node. For instance, uniform distribution of the N data items in the network implies that,

for each data item I_i, the probability that such item is found on a randomly encountered node is $\psi_i = 1/N$.

An important remark is that, in spreading data items, we do assume worst case conditions of *lack of correlation* between items' placement on nodes and encountered nodes. More precisely, we assume that the probability to encounter a node carrying a given data item does not depend on the previous encounters, i.e. that the probability distribution ψ_i is constant over time and holds for any newly encountered node irrespective of the past history.

In such a scenario, we are interested in quantifying the *average retrieval delay* $E[D]$ which characterizes a given spreading distribution of the data items. This metric, in principle, further depends on the interest of the tagged node for *specific* data items. If we denote with η_i (with $i \in 1 \cdots N$ and $\sum \eta_i = 1$) the probability that the tagged node is interested in, and hence generates a request for, the data item I_i, the average retrieval delay is expressed as the weighted average

$$E[D] = \sum_{i=1}^{N} \eta_i E[D(I_i)]$$

where $E[D(I_i)]$ is defined as the delay elapsing between the instant of time a tagged node becomes interested in retrieving the data item I_i, and the time that such *specific* data item is actually retrieved. In the assumption of uniform interest, $\eta_i = 1/N$ and hence

$$E[D] = \sum_{i=1}^{N} \frac{E[D(I_i)]}{N}$$

A. Average Retrieval Delay for replicated data items

As shown in what follows, under the above assumptions, the average retrieval delay achievable if we replicate data items over nodes cannot be lower than N, the total number of available data items. In fact, let us first compute such performance metric in the assumption of uniform spreading of data items across nodes. In this case, the average delay $E[D(I_i)]$ does not depend on the specific data item i chosen. Since the probability to encounter a node carrying exactly the data item i looked for is $1/N$, the average number of encounters elapsing before such data is retrieved is the mean of the geometric distribution with probability $1/N$, i.e.,

$$E[D] = \sum_{i=1}^{N} \frac{E[D(I_i)]}{N} = E[D(I_i)] =$$

$$= \sum_{k=1}^{\infty} k \cdot \left(1 - \frac{1}{N}\right)^{k-1} \cdot \frac{1}{N} = N$$

Furthermore, it is straightforward to prove that[2] any non uniform spreading of the data items yield a result worse than N. Indeed, if we assume that every item i is found over a randomly chosen node with probability ψ_i,

$$E[D] = \frac{1}{N} \sum_{i=1}^{N} E[D(I_i)] =$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{\infty} k \cdot (1 - \psi_i)^{k-1} \cdot \psi_i = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\psi_i}$$

which is minimized (e.g. through usual Lagrange multipliers, and recalling that the constraint $\sum_{i=1}^{N} \psi_i = 1$ holds) when $\psi_i = 1/N$ for all i.

B. Average Retrieval Delay for Random Linear Coded data items

From the above discussion, it appears that no spreading policy directly enforced on the original data items can outperform the average retrieval delay bound N. The natural next step is thus to consider whether such limit can be overcome by using *coded combinations* of the original data items.

Indeed, techniques based on Random Linear Coding (RLC) of stored data have been shown in the past to achieve performance improvements in some specific distributed storage settings [20]. We recall that RLC is a coding scheme where every element is envisioned as a vector in a Galois Field $\mathbf{F_q}$ of size q. A possible idea could be to store in each node, instead of the original data items I_i, some linear combination of such data. More specifically, we call *RLC replica* (or, in short, coded replica), a linear combination

$$R_j = \sum_{i=1}^{N} \beta_{j,i} I_i$$

where the coefficients $\beta_{j,i}$ are randomly drawn from the field $\mathbf{F_q}$ with uniform probability $1/q$.

However, as indeed well expected[3], this operation is not deemed to provide *any* performance improvement on the average retrieval delay. In fact, in the best case assumption that all linear combinations of the coded replicas found at any encountered node are independent each other, the original data item I_i will be decoded only when N coded replicas will be received, i.e. only when N nodes will be encountered (we recall our assumption that every node can carry a single data item, which in this case means a single coded replica - in favor to RLC we neglect to account for the fact that a coded replica requires some extra storage capability, namely $N \cdot \log_2(q)$ bits for keeping track of the β coefficients, than an uncoded data item).

Note that, in term of delay performance, there are other reasons to promote RLC. A first potential advantage is that the tagged node, after N encounters, will have the ability

[2]Owing to the assumption of uniform interests; otherwise the best spreading distribution would result to be different from the uniform one, and specifically would be the one proportional to the square root of the interest probability.

[3]note that most work done in RLC, including [20], assume that a requested message is subdivided into m chunks and coding is done among those chunks, i.e., the application scenario is that of *intra-data* coding. Our *inter-data* coding scenario is instead different: we code *different* data items together, and we seek to retrieve just one requested item (versus the case of *all* the chunks forming a message) among those combined through coding.

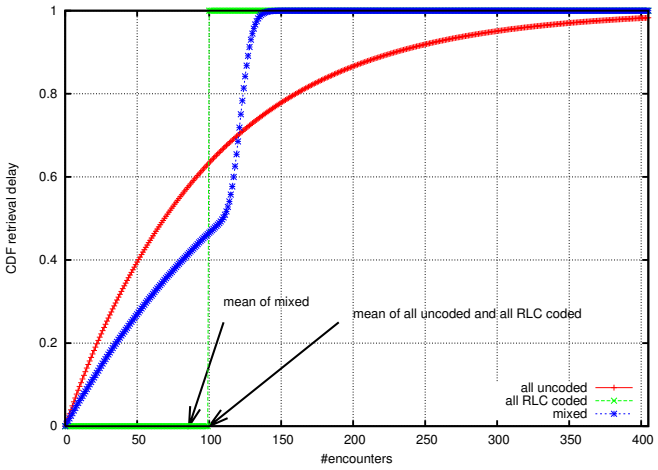

Fig. 1. Comparison of the retrieval delay Cumulative Distribution Functions for the cases of uncoded items (labeled as uniform), RLC-coded items (labeled as RLC), and a mixed strategy, discussed next, where we have both coded and uncoded items. N=100.

to retrieve *all* the N data items at once, and not only the specifically requested item. Moreover, since any data item will be retrieved after exactly N encounters, the variance of the delay, and hence the delay distribution tail (i.e., the peak delay) largely decreases with respect to the uncoded case. This is highlighted in Figure 1, which compares the Cumulative Distribution Function of the retrieval delay, measured in number of encountered nodes, for $N = 100$ data items, and for the cases of uncoded (previous section - curve labeled as "uniform") and coded replicas (curve labeled as RLC). We postpone the discussion of the curve labeled as "mixed" to section IV-A. As the figure shows, in the case of uncoded replicas, even if a data is retrieved in less than 100 encounters in more than 60% of the overall cases, and even if the average retrieval delay is in both cases equal to 100, the probability that a data will be retrieved after 200 or more encounters remains greater than 10%.

However, besides these advantages which may be considered useful in specific scenarios, we remark once again that such plain application of RLC is unable to reduce the average retrieval delay.

IV. Partial Inter-data coding

A. Motivation and Intuition

Let us reconsider figure 1. This figure can be in fact re-interpreted in a different manner, and specifically as the probability that a given percentage of the whole data items are received after a given number of encounters.

Clearly, in the case of coded replicas, no data item can be decoded until N linearly independent replicas are collected, and in this case all data items, hence including the one specifically requested, can be decoded at once. In the case of uncoded replicas instead, each encountered node may either carry a new, not yet previously encountered, data item, or carry a duplicated, already collected, data.

Conversely, in the case of uncoded items, the probability to receive a new data item decreases with the number of encounters (as quantified by the decrement in the slope of the curve plotted in figure 1). More specifically, at the first encounter, the tagged node will find a new item with probability 1 (as no additional items were earlier collected), which implies that the tagged node will have collected one Nth of all the available items in the network (this being the y-axis value in the plot corresponding to the x-axis value 1). During the second encounter, the probability that a new item is encounter is no more one, but decreases to $(N-1)/N$, as there is an $1/N$ chance that the next encountered node will carry the formerly collected data. And so on. The overall result is that the "rate" at which new items are encountered greatly decreases as time elapses. In the unlucky extreme situation that $N-1$ distinct items were encountered, but the specific item looked for was not in this set, the remaining average time to meet the looked for item is still N (by memoryless of the encounter process).

Now, all this discussion could appear irrelevant in the case of uncoded replicas, as any received information prior to the actual looked for data item is not used (indeed, in practice it obviously does not require to be stored). But such considerations suggest that if we were able to devise a mechanism capable of *using* such prior received information (whose rate of reception is high at the start of the retrieval process, and gradually decreases with time), and, at the same time, *avoiding to incur in the long delays emerging from the tail of such distribution*, we could achieve delay performance gains.

It hence becomes straightforward to conclude that such mechanism is trivially accomplished by simply combining uncoded replicas, distributed on a fraction of nodes, with coded replicas, distributed on the remaining nodes (and eventually combining only a subset of data items among all available ones, as the formal definition provided in the next section will permit, for greater generality). This is readily understood by the example illustrated in figure 2. $N = 4$ data items (A, B, C, D) are spread in the network. A tagged node looking for item D first encounters a coded combination of the four items, and then encounters two nodes carrying items A and B. At the fourth encounter, the tagged node will be able to retrieve item D either if i) the encountered node carries D, or ii) the new node carries a (linearly independent) coded combination of all four items, or iii) the new node carries C. In the last two cases, in fact, the tagged node succeeds to gather four linearly independent combinations (an uncoded item being a special case of combination) of the data items.

This operation has a key advantage over a "pure" RLC operation: the tagged node does not necessarily need to always wait for N encounters, as the specifically requested item may be received in uncoded form earlier. At the same time, it has the disadvantage that more than N encounters might be needed, as the probability to encounter a node carrying an already received item is not anymore negligible. But, as we will quantify in the following sections, the overall balance yields an improvement in the *average* retrieval delay for the requested

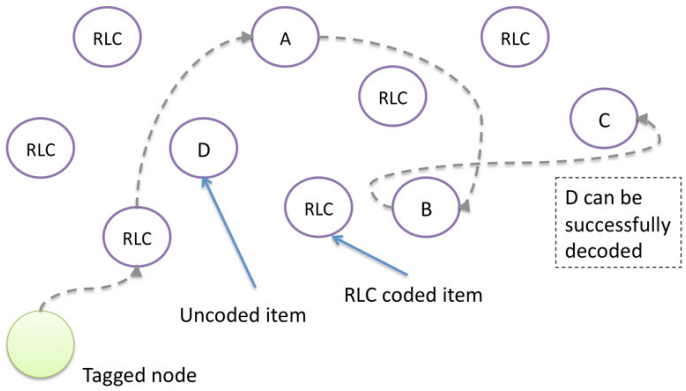

Fig. 2. After having encountered a coded instance combining all four available items, and three nodes carrying items A, B, and C, a tagged node can decode item D.

item. The curve labeled as "mixed" in figure 1 quantitatively backs up such insight. It shows the CDF of the retrieval time for a requested item among $N = 100$ available items, when 37.2% of the nodes[4] carry a RLC combination of the 100 items, and the remaining ones carry uncoded items (this strategy is later on named "All or Nothing"). From the figure we note that the probability of retrieving the target item in less than 100 encounters is relatively high (e.g. after 50 encounters, an item is retrieved in almost 30% of the cases) even if lower than the case of uncoded items, as, in addition to duplicated receptions, some encountered nodes carry RLC combinations which cannot be immediately exploited. Moreover, as long as about 110 encounters are made, the probability of retrieving the target item abruptly increase, as the number of collected linearly independent combination permits decoding of all the items, thus including the target one.

B. Proposed Approach: general case

In most generality, we call *Partial Inter-Data Coding* any approach which makes use of partially combined data items, thus including also the case of uncoded items (i.e. items whose combination is composed by themselves only). We formally define such class of approaches as follows.

- each node carries exactly one data instance.
- each data instance is the random linear coding of a number $k \in 1 \cdots N$ original data items; we call coding degree of each instance with the number k of combined data.
- a coding degree $k = 1$ is permitted (actually, recommended, see below); in this special case the data instance coincides with an uncoded original data item.
- we formally define a *Partial Inter-Data coding strategy* through the vector $[P_g(1), P_g(2), \cdots, P_g(N)]$ which represents the probability that an encountered node carries a data instance of given coding degree $1 \cdots N$; obviously $\sum_{i=1}^{N} P_g(i) = 1$.
- we call "average-delay-optimal partial coding strategy" the one characterized by the probability vector which minimizes the average retrieval delay.

Determining such optimal strategy appears feasible only for very small values of N. For instance, in the case $N = 2$, such derivation is straightforward. For compact notation, let us define the probability $P_g(1)$ that a node carries an uncoded item (say A or B) as p. Hence, $p/2$ is the probability to encounter a node carrying item A (equivalently, B), and $1 - p$ is the probability to encounter a node carrying a coded combination of A and B, hereafter referred to as AB. If the tagged node is interested in data item A (equivalently, B), the following cases occur:

- A is retrieved in 1 encounter is and only if the encountered node carries the item A in uncoded form (probability $p/2$);
- A is retrieved in 2 encounters if and only if the first encounter was *not* A, and either:
  - the first encounter was AB, in this case any second encounter (i.e., either A, B, or AB) provides the second linear equation needed to determine A;
  - the first encounter was B and the second encounter is either A (in this case A would be received in uncoded form) or AB (in this case A would be determined through decoding);
- A is retrieved in 3 encounters (and, more generally, in i encounters) if and only if the two previous encounters (more generally, $i - 1$) were B, and the last encounter is either A or AB;

The average delay to retrieve item A (equivalently, item B) is thus expressed as:

$$\text{Delay} = 1 \cdot \frac{p}{2} + 2 \cdot \left((1-p) \cdot 1 + \frac{p}{2} \cdot \left((1-p) + \frac{p}{2} \right) \right) +$$

$$+ \sum_{i=3}^{\infty} i \cdot \left(\frac{p}{2} \right)^{i-1} \cdot \left((1-p) + \frac{p}{2} \right)$$

After algebraic simplifications, we obtain

$$\text{Delay} = 1 - p + \frac{1}{1 - p/2}$$

The delay versus p is shown in figure 3. As expected, if $p = 0$ (all coded instances) or $p = 1$ (all uncoded replicas), the average retrieval delay results equal to the value 2. However, the curve of the shows a minimum, which holds when $p = 2 - \sqrt{2} \approx 0.586$, and which corresponds to a minimum average retrieval delay equal to $2\sqrt{2} - 1 \approx 1.828$, indeed about 9% lower than the delay achieved by any of the baseline techniques.

Extending such analysis to larger values of N is very complex as i) the number of optimization variables (the vector probabilities) linearly increase, and, especiall, ii) the number of possible partial combinations of items, and the need to track the corresponding encounters, do explode.

[4]This value is the asymptotically optimal one, and will be derived in section IV-E.

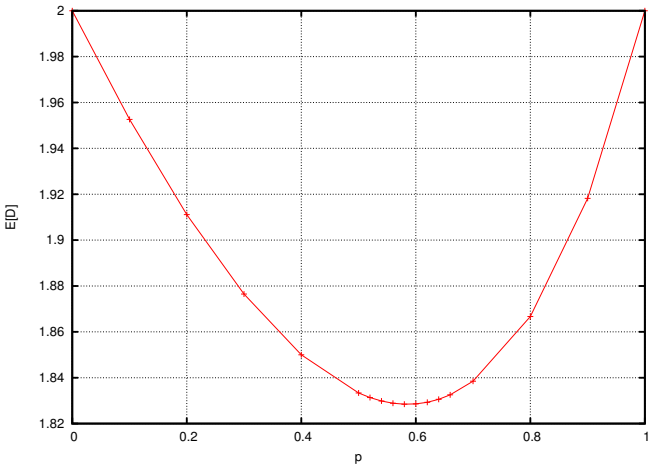

Fig. 3. Average retrieval delay in the case of $N = 2$ versus the probability p of encountering an uncoded instance

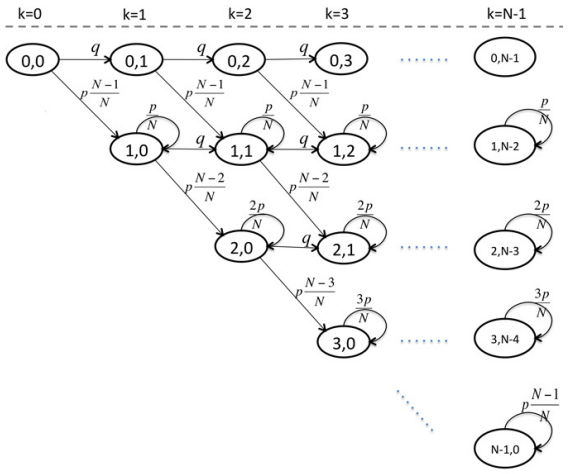

Fig. 4. Markov Chain model for the "all or nothing" partial coding approach. For improved readability, the absorption state "requested data item encountered/decoded" is not represented in the picture. That state is connected to all the other states with a probability that is the 1 complement of the sum of all the outgoing transition probabilities from each state. The average delay from the initial state $(0, 0)$ to the absorption state represents the desired "all or nothing" average delay.

C. Special case: All or Nothing

Since the analysis of the general case appears extremely complex, to gain some further performance insights, we consider a special case, simplified, partial coding strategy which we refer to as "All or Nothing". In this approach, we assume that every node can store only two types of instances:

- uncoded data instances, with probability $p = P_g(1)$;
- data instances achieved by random linear coding of *all* the N data items, with probability $q = 1 - p = P_g(N)$

In other words, we assume that nodes do not carry combinations of a *subset* of the N items, but can only carry either coded combinations of *all* the N data, or no combinations, i.e. uncoded data items.

this approach can be envisioned as a worst-case, upper bound, scenario of the performance achievable by the more general partial coding approach. In what follows, we first show that this approach can be somewhat easily analyzed. Second, and most interestingly, we show that the performance gains of such a simple approach are of *asymptotic* nature, and yield a 14% average retrieval delay reduction.

D. Average delay performance evaluation

The "All or Nothing" approach can be modeled through a relatively simple Markov Chain, depicted in figure 4, where the state of the chain is completely defined by the pair (N_u, N_c) of collected *distinct* uncoded (N_u) and fully coded (N_c) data instances.

The Markov process shall be studied as a transient process and the chain is represented in figure 4. The initial condition is that the tagged node does not store any data instance, i.e. the initial state of the chain is $(0, 0)$. The process ends whenever either an uncoded version of the seeked data item is receives, or the sum of the distinct uncoded instances plus the collected coded instances reaches the value N (as in this case decoding may occur through N linear independent equations). As such, this chain is conveniently analyzed as an *absorption process*, where we assume that the absorption state models the case when the requested data item is retrieved.

The Markov chain evolves as follows. If we assume it to be in the state (i, j), with $i + j < N - 1$, the following state transition probabilities hold (see also figure 4):

- with probability p/N the next encountered node carries the requested data item in uncoded version, hence the process ends;
- with probability $p \cdot i/N$ the next encountered node carries an uncoded data instance already received in the past, hence the process remains in state (i, j);
- with probability $p \cdot (N - i - 1)/N$ the next encountered node carries a new uncoded data instance which is not the one looked for and it is new with respect to those already received, hence the process moves in state $(i + 1, j)$;
- with probability $q = 1 - p$ the next encountered node carries a coded instance, hence the process moves in state $(i, j + 1)$.

Conversely, if we assume the chain to be in state (i, j) with $i + j = N - 1$, then

- with probability $p \cdot (N - i)/N + q$ the next encountered node carries either the requested data item in uncoded version, or a new uncoded data item, or a coded instance; since in the last two cases N linear combinations equations are now available, the process ends;
- with probability $p \cdot i/N$ the next encountered node carries an uncoded data instance already received in the past, hence the process remains in state (i, j);

We can now define the function $D_k(i)$, with $0 \leq k \leq N-1$ and $0 \leq i \leq k$, which represents the average delay elapsing between the chain being in state $(i, k - i)$ and the absorption state being reached. Note that the index k represents the

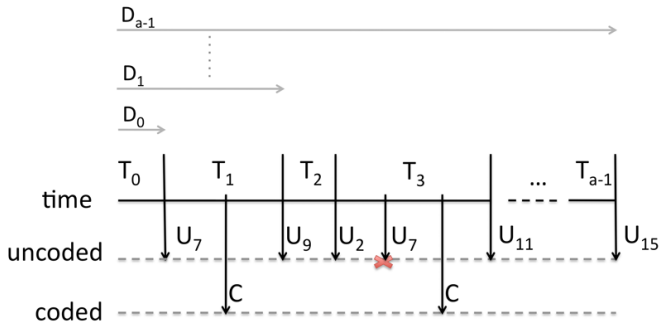

Fig. 5. A cycle is divided in sub cycles of different duration; Each sub cycle ends when a new uncoded instance is received. The cycle ends when the sum of distinct coded and uncoded collected instances reaches the value N.

number of linearly independent combination received, i.e. the sum of distinct uncoded and coded instances collected. Obviously, if $N-1$ distinct instances are collected, the time to absorption is 1 step, except in the case an already collected uncoded instance is encountered, case which happens with probability $p \cdot i/N$. In formulae,

$$D_{N-1}(i) = 1 + p\frac{i}{N}D_{N-1}(i) \rightarrow D_{N-1}(i) = \frac{1}{1 - p \cdot i/N}$$

Through similar considerations, it is possible to recursively express the delay function $D_k(i)$, for the case $k < N-1$, as:

$$D_k(i) = 1 + p\frac{i}{N}D_k(i) + qD_{k+1}(i) + p\frac{N-i-1}{N}D_{k+1}(i+1)$$

which can be rewritten as

$$D_k(i) = \frac{1 + qD_{k+1}(i) + p\frac{N-i-1}{N}D_{k+1}(i+1)}{1 - p \cdot i/N}$$

Finally, we note that the average retrieval delay looked for is provided by $D_0(0)$. Once the probability p is provided (and hence also $q = 1-p$), such delay can be numerically found starting from the values $D_{N-1}(i), i \in (0, N-1)$, and going backward using the above recursion.

E. Asymptotic Delay Analysis

The computation of the above recursion is very efficient, once the configuration parameter (the probability p or its complement $q = 1-p$) is numerically provided as input. However, the above recursion fails to explicitly provide insights on the asymptotic behavior of the proposed system operation. Indeed, a key question we wish to address is whether the emerging gain in the average delay consistently stands as the number of data items increase, and what is the resulting trend.

To face this issue we propose the following approximate expected-value analysis. Rather than focusing on a specifically requested data item, it appears convenient to start by modeling a "full" cycle which concludes when *all* the N data items can be decoded. This cycle can be described as composed of a number of *sub cycles*. As illustrated in figure 5, a sub cycle terminates whenever a *new* uncoded data item is received. During a sub cycle, two other types of events may occur: i) reception of zero or more coded instances, and ii) reception of zero or more duplicated uncoded items.

Let us now focus on the special case of the whole cycle concluding with the reception of an uncoded item. With this assumption which will be later on removed, we circumvent the need to consider the last sub cycle as a special case one, as the cycle could terminate with the reception of a coded instance (yielding the N-th, final, linearly independent equation needed) instead of an uncoded instance as per our definition of sub cycle.

Let a be the number of sub cycles composing such a whole cycle, and let us number sub-cycles starting from the index 0. We define:

- T_i: expected duration of sub cycle $0 \leq i \leq a-1$;
- $D_i = \sum_{k=0}^{i} T_k$: expected time elapsing between the start of the process and the end of sub cycle $0 \leq i \leq a-1$. It follows that, according to this notation, the whole cycle duration is D_{a-1}.

The average duration of the i-th sub cycle is readily determined as the average time elapsing between the end of the former sub cycle (if any) and the time in which a *new* uncoded instance is encountered. Since this event occurs with probability $p \cdot (N-i)/N$ depending only on the considered sub cycle i, T_i is expressed as:

$$T_i = \frac{N}{(N-i)p}. \tag{1}$$

If we assume that a, namely the overall number of sub cycles composing a whole cycle, were known, we could derive the average retrieval delay for a *specifically considered* data item as follows:

$$E[D] = \frac{\sum_{k=0}^{a-1} D_k + (N-a)D_{a-1}}{N} \tag{2}$$

This equation takes into account that the data item specifically requested by the tagged node can be either received in uncoded form at the end of each deployed sub cycle (this occurs with probability $1/N$ that the uncoded instance terminating a sub cycle is the requested one among the N available), or it can be retrieved (along with all the other $N-a$ items not yet available in uncoded form) through decoding at the end of the whole cycle, i.e., after the occurrence of all the a sub cycles.

Equation (2) can be effectively simplified as follows:

$$\begin{aligned} E[D] &= \frac{\sum_{k=0}^{a-1} D_k + (N-a)D_{a-1}}{N} = \\ &= \frac{1}{N}\left(\sum_{k=0}^{a-1}\sum_{i=0}^{k} T_i + (N-a)\sum_{j=0}^{a-1} T_j\right) = \\ &= \frac{1}{N}\left(\sum_{j=0}^{a-1}(a-j)T_j + (N-a)\sum_{j=0}^{a-1} T_j\right) = \\ &= \frac{1}{N}\sum_{j=0}^{a-1}(N-j)T_j = \\ &= \frac{1}{N}\sum_{i=0}^{a-1}(N-j)\frac{N}{(N-j)p} = \\ &= \frac{a}{p} = \frac{a}{1-q} \end{aligned} \quad (3)$$

To complete the analysis, we now need to find the value a which is then used in equation 3 to determine the average retrieval delay. For this purpose, we recall that q is the probability that an encountered node carries a coded instance. This probability is constant with time, i.e., it does not depend on the considered sub cycle. As such, during a whole cycle lasting D_{a-1} encounters, the average number C_a of collected uncoded instances is trivially given by

$$C_a = qD_{a-1}$$

. D_{a-1} can be computed as:

$$D_{a-1} = \sum_{i=0}^{a-1} T_i = \sum_{i=0}^{a-1} \frac{N}{(N-i)p} = \frac{N}{p}(H_N - H_{N-a}) \quad (4)$$

where H_n is the well known Harmonic Number arising from the truncation of the Harmonic series to the index n, i.e.:

$$H_n = \sum_{k=1}^{n} \frac{1}{k}$$

Recalling that $q = 1 - p$, we therefore conclude that

$$C_a = qD_{a-1} = N\frac{q}{1-q}(H_N - H_{N-a})$$

We are now ready to take the final step for deriving a. By removing the starting assumption of a being an integer value, and by extending the definition of the Harmonic Number to non integer values[5], we can compute the average number of sub-cycles that compose a full cycle by solving, in the unknown a, the following congruency equation:

$$a + C_a = N \quad \rightarrow \quad a + N\frac{q}{1-q}(H_N - H_{N-a}) = N \quad (5)$$

This equality follows from the fact that a full cycle ends whenever the number of uncoded instances received (namely, a) plus the number of coded instances received (namely, C_a) equals the total number of needed instances N. As are specifically interested in asymptotic insights, i.e. when the number of data items becomes large ($N \rightarrow \infty$), we can approximate the Harmonic Number H_i with $\ln i + \gamma$, being $\gamma = 0.57721$ the Euler-Mascheroni constant. Equation (5) can thus be rewritten as follows:

$$\frac{N-a}{N} = \frac{q}{1-q}\left(\ln\frac{N}{N-a}\right) \quad (6)$$

This non linear equation specifies an implicit function $a(q)$ which relates the value a to the configuration parameter q. Such function can be explicited by exploiting the Lambert $\mathcal{W}$ function[6], as:

$$a(q) = N\left[1 - \frac{\mathcal{W}(\frac{1-q}{q})}{\frac{1-q}{q}}\right] \quad (7)$$

Finally, owing to equation 3, we conclude that the delay versus the parameter q is given by

$$E[D] = \frac{N}{1-q}\left[1 - \frac{\mathcal{W}(\frac{1-q}{q})}{\frac{1-q}{q}}\right]$$

Taking the derivative with respect to q, and equating to zero, we ultimately obtain that the delay is minimized when the parameter q is set to the value q^* which satisfies the following equation (independent of N):

$$q - 1 + \mathcal{W}\left(\frac{1-q}{q}\right)\left[2q - 1 + (1+q)\,\mathcal{W}\left(\frac{1-q}{q}\right)\right] = 0 \quad (8)$$

Although not very handy, this equation can be solved numerically, and admits a solution when $q = q^* = 0.373588$. This corresponds to an optimal retrieval delay for the "All or Nothing" partial coding approach given by $E^*[D] = 0.85988N$, i.e., a reduction of about 14% with respect to the delay attainable by the baseline approaches.

F. Validation

For validation purposes, figure 6 compares, for N in the range 10 to 100, and for the optimal $q = q^*$ probability, the All or Nothing average retrieval delay results as provided by i) the exact recursive computation of the average delay (section IV-D), ii) its approximate asymptotic computation (section IV-E), and iii) the results obtained through simulation. Not only, as expected, the recursive computation matches the simulation results (this matching confirms that the proposed markov chain approach is rigorous), but also no difference can be noticed in the approximate asymptotic computation. This suggests that the asymptotic analysis is also an excellent approximation for relatively small values of N.

[5]The analytic extension of the Harmonic numbers is

$$H_z = \gamma + \frac{\Gamma'(z+1)}{\Gamma(z+1)}$$

Being $\gamma = 0.57721$ the Euler-Mascheroni constant, and $\Gamma(z)$ the Gamma Function.

[6]We recall that the Lambert W function is defined as the solution for the trascendent equation $z = \mathcal{W}(z)e^{\mathcal{W}(z)}$. It can be specifically shown that an equation in the form $x = -c\ln x$, indeed the form of equation (6) once we set $x = (N-a)/N$ and $c = q/(1-q)$, can be espressed through the Lambert W function as $x = c\mathcal{W}(1/c)$. Equation (7) now readily follows via algebraic manipulation.

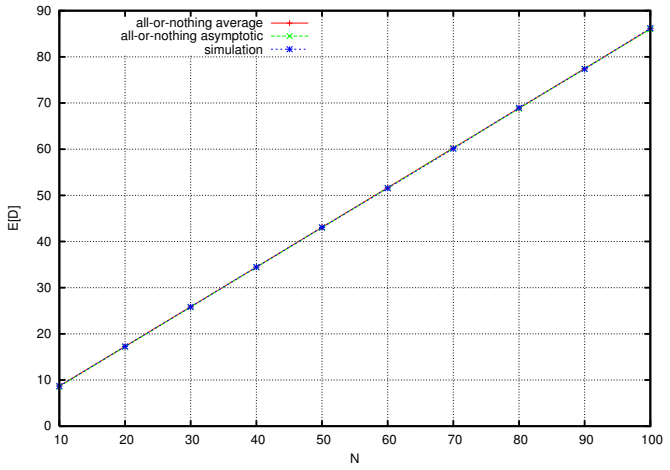

Fig. 6. Comparison of the results obtained by the exact recursive approach, the asymptotic approximation, and simulation.

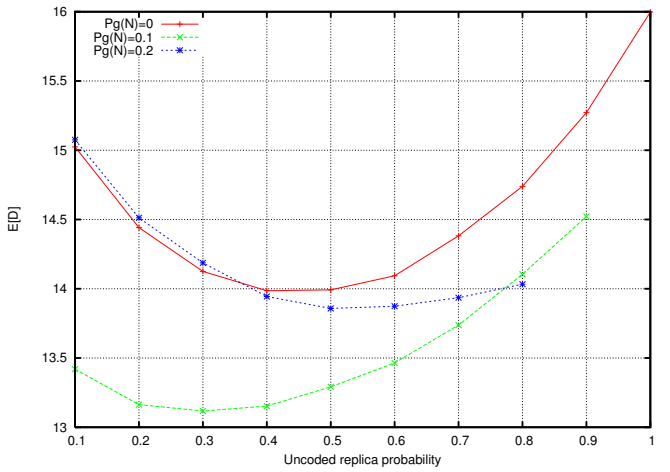

Fig. 7. Average retrieval delay for a partial coding strategy with further uses instances coding pairs of items. $N = 16$ data items. Three considered cases: $P_g(N) = 0$, 0.1, 0.2. The probability $P_g(1)$ of collecting an uncoded item is reported in the x-axis; the probability of collecting a pair is given by $P_g(2) = 1 - P_g(N) - P_g(1)$.

V. Further delay improvements

The "All or Nothing" partial coding approach introduced in section IV-C is a special case of partial coding. As such, techniques which additionally exploit coded combinations of a *subset* of data items might further reduce the average retrieval delay. Unfortunately, finding a lower bound valid for the general case appears to be an extremely challenging issue. In this section, we thus limit to show, through simulation, to what extent delay may be improved vith a simple extension of the partial coding strategy, where in addition to uncoded and fully coded instances, we also consider instances which combine randomly chosen *pairs* of items. This strategy, using the notation introduced in section IV-B, is described by a probability vector $[P_g(1), P_g(2), 0, \cdots, 0, P_g(N)]$ which has only three non null entries, two of them being independent (the third being the complement to 1).

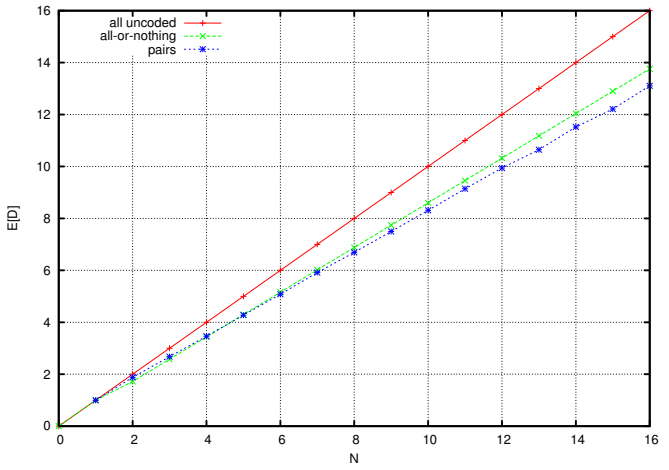

Fig. 8. Average retrieval delay varying N for the extended strategy involving coded pairs; Results for the uncoded baseline case and the All or Nothing approach are further plotted for comparison.

In what follow for reasons of space we focus on a single scenario of $N = 16$ items. The results presented are averaged over 10^5 different retrieval cycles. To gain some insights on how to set the configuration parameters for such strategy further involving the coding of pairs, figure 7 fixes the probability $P_g(N)$ of encountering a node carrying a coded instance to the values $0, 0.1, and 0.2$, and varies the probability $P_g(1)$ of receiving an uncoded instance. First, the figure shows that, as expected, adding pairs improves performance with respect to the "All or Nothing" approach (whose perfomance are provided in correspondence to the rightmost point in every curve). Second, the figure shows that the lowest delay performance are achieved by the configuration $[0.3, 0.6, 0, \cdots, 0, 0.1]$. This optimum configuration point is quite far from the optimal parameters determined for the "All or Nothing" approach. For instance, here, only 10% of the instances should be fully coded, whereas in the All or Nothing approach optimal conditions were achieved by 37.3% of coded instances. This suggests that the identification of the optimal mix for a partial coding strategy requires a specific per-strategy separate study, and general rules are not easily inferred.

Finally, figure 8 aims at quantifying the delay improvements provided by the introduction of pairs. For reference purposes the figure also plots the results for the baseline strategy of no coding at all (in this case the average delay is equal to N), and the optimized performance for the All or Nothing approach. Results for the extended strategy using pairs are computed using the above optimized configuration $[0.3, 0.6, 0, \cdots, 0, 0.1]$ (although note that such optimization was carried out for the specific case $N = 16$). A linear trend appears to emerge also for the extended strategy. For the case $N = 16$ the delay gain is about 18.1% with respect to the uncoded reference case. The improvement with respect to the 14% gain of the All or Nothing case, even if not major, is nevertheless non negligible.

VI. Conclusive remarks

The main conclusion that can be drawn from this work is that there is margin to improve the average retrieval delay performance in a DTN beyond that accomplished by naive data replication of "pure" network coding techniques. Specifically, the new class of data placement approaches here introduced, called partial inter-data coding, appears capable to improve such average delay even in networks where no mobility predictions or data forwarding schemes are employed. Results, derived for a simple partial coding strategy called "All or Nothing", show asymptotic average delay improvements of 14%, and preliminary results for more sophisticated strategies suggest that there is supplementary margin for improvement.

Our work further poses a very challenging foundational question: to what extent the average delay can be improved if we persist in not assuming any coordination among nodes? We easily infer that such a theoretical lower bound cannot exceed the 50% gain, namely the asymptotic gain attainable in the ideal case of perfect coordination of all the nodes' encounters (proof omitted for reasons of space). However, our preliminary simulation results (in the order of about 20% gain) somehow imply that such a lower bound is likely to be far from 50%. And finding a tight lower bound appears to be a widely open, not easy to address, issue.

References

[1] M.Mizenmacher. Digital Fountains: A Survey and Look Forward. Information Theory Workshop, 2004.

[2] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. Jouranl of the Society for Industrial and Applied Mathematics, 8:300-304, June 1960.

[3] Luby. M. "LT codes". Proc. 43rd Ann. IEEE Symp. on Foundations of Computer Science, 16-19 November 2002, pp 271-282.

[4] Shokrollahi A. : "Raptor codes". Technical Report, Laboratoire d'algorithmique, Ecole Polytechnique Fdrale de Lausanne, Lausanne, Switzerland 2003.

[5] A. Kamra , V. Misra , J. Feldman , D. Rubenstein, Growth codes: maximizing sensor network data persistence, Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, September 11-15, 2006, Pisa, Italy

[6] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes. In IPSN, Apr. 2005.

[7] M. Luby, M. Mitzenmacher, M.A. Shokrollahi, D. Spielman. Efficient erasure correction codes. IEEE Transaction on Information Theory, 47(2):569-584, February 27, 2001

[8] P. Rodriguez, S. M. Tan, C. Gkantsidis, "On the feasibility of commercial, legal P2P content distribution", ACM SIGCOMM Computer Communication Review 36(1):75-78, 2006.

[9] N. Laoutaris, P. Rodriguez, L. Massoulie, "ECHOS: Edge Capacity Hosting Overlays of Nano Data Centers", ACM SIGCOMM Computer Communication Review 38(1):51-54, 2008.

[10] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, R. Braynard, "Networking named content", Proc. 5th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2009), Rome, Italy, December 2009, pp. 1-12.

[11] J. Reich, A. Chaintreau, "Replication schemes for opportunistic networks with impatient users", Proc 47th annual Allerton conference on Communication, control, and computing (Allerton 2009), Monticello, Illinois, USA, 2009, pp. 1446-1451.

[12] Y. Jiao, Z. Jin, Y. Shu, "Data Dissemination in Delay and Disruption Tolerant Networks Based on Content Classification", Proc. 5th International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2009), Fujian, Dec. 2009, pp. 366-370.

[13] G. Sandulescu, S. Nadjm-Tehrani, "Optimising Replication versus Redundancy in Window-Aware Opportunistic Routing", Proc. 3rd Int. Conf. on Communication Theory, Reliability, and Quality of Service (CTRQ 2010), June 2010, pp. 192-201.

[14] Y. Ishimaru, W. Sun, K. Yasumoto, M. Ito, "DTN-based Delivery of Word-of-Mouth Information with Priority and Deadline", Proc. 5th Int. Conf. on Mobile Computing and Ubiquitous Networking (ICMU2010), April 2010, pp. 179-185.

[15] A. Derhab, N. Badache, "Data Replication Protocols for Mobile Ad-Hoc Networks: A Survey and Taxonomy", IEEE Communications Surveys and Tutorial, Vol.11 No. 2, 2009.

[16] Q. Ayub, S. Rashid, "T-Drop: An optimal buffer management policy to improve QOS in DTN routing protocols", Journal of computing, Volume 2, Issue 10, October 2010.

[17] Y. K. Ip, W. C. Lau, O. C. Yue, "Forwarding and Replication Strategies for DTN with Resource Constraints", IEEE 65th Vehicular Technology Conference (VTC2007-Spring), April 2007, pp 1260-1264.

[18] G. Sollazzo, M. Musolesi, C. Mascolo, "TACO-DTN: a time-aware content-based dissemination system for delay tolerant networks", Proc. 1st international MobiSys workshop on Mobile opportunistic networking (MobiOpp 2007), San Juan, Puerto Rico, 2007, pp. 83-90.

[19] R. J. D'Souza, J. Jose, "Routing Approaches in Delay Tolerant Networks", Int. Journal of Computer Applications, Volume 1, No. 17, 2010.

[20] S. Acedanski, S. Deb, M. Medard, R. Koetter, "How Good is Random Linear Coding Based Distributed Networked Storage?" in Proc. of 1st Workshop on Network Coding, Riva del Garda, Italy, Apr. 2005.