Rateless Codes and Random Walks for P2P Resource Discovery in Grids

V. Bioglio, R. Gaeta, M. Grangetto, M. Sereno
Rateless Codes and Random Walks for P2P Resource Discovery in Grids
IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS
(2014) 25
DOI: 10.1109/TPDS.2013.141

Rateless codes and random walks for P2P resource discovery in Grids

Valerio Bioglio, Rossano Gaeta, Marco Grangetto, *Senior Member, IEEE,* Matteo Sereno

Abstract—Peer-to-peer (P2P) resource location techniques in Grid systems have been recently investigated to obtain scalability, reliability, efficiency, fault-tolerance, security, and robustness. Query resolution for locating resources and update information on their own resource status in these systems can be abstracted as the problem of allowing one peer to obtain a local view of global information defined on all peers of a P2P unstructured network.

In this paper, the system is represented as a set of nodes connected to form a P2P network where each node holds a piece of information that is required to be communicated to all the participants. Moreover, we assume that the information can dynamically change and that each peer periodically requires to access the values of the data of all other peers. A novel approach based on a continuous flow of control packets exchanged among the nodes using the random walk principle and rateless coding is proposed. An innovative rateless decoding mechanism that is able to cope with asynchronous information updates is also proposed. The performance of the proposed system is evaluated both analytically and experimentally by simulation. The analytical results show that the proposed strategy guarantees quick diffusion of the information and scales well to large networks. Simulations show that the technique is effective also in presence of network and information dynamics.

Index Terms—Resource discovery, Peer to Peer, Rateless codes, Random walks, GRID

1 Introduction

Peer-to-peer (P2P) resource location techniques in Grid systems have been investigated to obtain scalability, reliability, efficiency, fault-tolerance, security, and robustness. To this end, structured, unstructured, and hybrid P2P systems have been considered and the relative merits and drawbacks have been highlighted [1], [2].

Many proposals exploiting unstructured P2P systems share a common characteristic: Grid nodes within one administrative domain periodically query for locating resources and update information on their own resource status through one or more interface peers. The interface peers (usually those with the largest capacity) play two major roles: they are connected to other interface peers forming a P2P unstructured network that is used to forward (and respond to) queries on the behalf of nodes in its administrative domain. They collect and maintain data of all nodes in the local administrative domain.

Query resolution in these systems can be abstracted as the problem of allowing one peer to obtain a local view of global information defined on all peers of a P2P unstructured network. In particular, we assume that each peer holds a piece of information (the aggregate resource statuses of all nodes in its administrative domain) and that any peer requires to access the values of the data of all other peers periodically at rate λ queries/sec. The goals to be achieved are threefold: first, one wants to guarantee that every node is likely to collect the complete global information in a timely fashion. Moreover, the communication overhead must be kept as limited as possible to avoid congesting the network. Finally, the processing power of each node must be used parsimoniously.

Our contribution

The contribution of this paper is three-fold:

- we propose to use a practical form of network coding based on rateless codes, that being based on simple XOR operations are more computationally efficient than previous attempts based on random network coding in large Galois fields. The proposed approach is based on a continuous flow of control packets exchanged among the nodes using the *random walk* principle. Each node is allowed to start a limited number of control packets, thus limiting the overall number of random walkers traveling in the network, i.e. the communication overhead of the proposed technique. Every node enriches any incoming packet with its local information according to the rateless coding principle and forwards it randomly to one of its neighbors. As a consequence a continuous flow of packets carrying coded information is spread in the network and any node can use them to recover the global data;
- using network coding in a dynamic scenario where the information varies asynchronously is an issue, since the information combined by each node has to be the same to guarantee the inversion of the linear system of equations. Usually this issue is circumvented by constraining the nodes to change

• *V. Bioglio is with Politecnico di Torino, Dipartimento di Elettronica, Torino, Italia. R. Gaeta, M. Grangetto, M. Sereno are with Università degli Studi di Torino , Dipartimento di Informatica, Torino, Italia.*
E-mail: valerio.bioglio@polito.it {gaeta,grangetto,sereno}@di.unito.it

the information synchronously, i.e. the information is kept static in a period of time called generation. In this paper we present a novel decoding mechanism coping with asynchronous information updates. In particular a novel rateless codes decoding algorithm taking into account the variation of the combined information is proposed;
- we show that the efficiency of our solution grows as the size of the data held by each node increases in a scenario where communication among nodes is constrained by a limit on the size of the MTU.

The performance of the proposed solution is evaluated both analytically and experimentally by simulation. The analysis proves that the designed strategy guarantees a significant reduction of the time required to spread the information among all the participants. Moreover, the analytical model proves that our solution scales much better with the size of the network. The simulative study shows that the proposed solution is able to cope with a dynamic scenario where the node alternates between active and idle states and their information varies asynchronously. We also show that alternative approaches to allow peers to obtain a local view of global system information are less effective and efficient.

The paper is organized as follows: Section 2 discusses previous work related to ours, Section 3 defines the system we consider, Section 4 describes our approach for decoding rateless codes, Section 5 presents the analytical model we developed to evaluate the time required to spread the information among all nodes, Section 6 discusses simulation results for the dynamic scenario, Section 7 compare our distributed proposal with alternative centralized approaches and Section 8 summarized the paper contribution and outlines possible future developments.

2 Related Work

The problem of data gathering in distributed systems has been faced with many different tools and approaches.

A first class of techniques are those based on probabilistic gossiping [3], [4]. Probabilistic gossiping has been used both to compute a function of the global information, e.g. averages, and to actually spread local information across a network as in our settings although such techniques rely on a set of assumptions that are difficult to guarantee in practice [5]. Notable attempts to overcome some of these limitations in the area of epidemic dissemination are [6], [7] that result in close to optimum latency-bandwidth trade-of. In particular, [6] uses flow control on the the maximum rate at which a participant can submit updates without creating a backlog and devises content reconciliation mechanisms to reduce message redundancy. In [7], [8], [9], [10] exploitation/enforcement of topological properties of the network are proposed to improve the performance of the data dissemination process.

Algebraic Gossip, proposed in [11], is the first algorithm addressing data gathering with Network Coding (NC). In this paper a gossip algorithm based on NC is presented, and it is proved that the spreading time of this algorithm is $O(K)$, where $K < N$ is the number of nodes having some information to spread. This algorithm is very similar to classical NC: at every transmission opportunity, each node sends to another node a linear combination, computed in Galois Field $GF(q)$ with $q \geq K$, of the previously received packets. However, NC exhibits a high computational complexity [12], due to the cost of the coding and decoding operations performed in high-order GF. Moreover, each packet requires a padding of additional $Klog_2(q) > Klog_2(K)$ bits. Such padding turns to be infeasible for large networks; as an example, if $K = 1000$ each packet needs more than 10^4 padding bits. Finally, the authors suggest that the message size m should scale with the size of the network, since it is required that $m >> log(q)$.

A different approach is to store and create packets using rateless codes. In [13] distributed fountain codes are proposed for networked storage. To create a new encoded packet, each storage node asks information to a randomly selected node of the network. The receiver answers to the caller sending its information, that will be used by the caller to encode a new packet. A similar algorithm is proposed in [14], where the coded packet formation mechanism is reversed; in this case, the nodes that stores the information send random walkers containing the information. The storage nodes store this information and create encoded packets XORing some of the information they already received. At the end of the process, each storage node stores an encoded packet, and it is possible to retrieve the initial information querying any $K + \epsilon$ randomly chosen storage nodes.

Growth codes, proposed in [15], use a similar technique but propose a particular degree distribution for the rateless codes to maximize the data persistence in presence of a single data collector node.

In all the previously presented papers, the creation of the codes is node-centric, i.e. the nodes cope with the information gathering and the encoding operations; in [16] this responsibility is assigned to the packets. The aim of this work is to use particular random walkers, named as rateless packets, for distributed storage of information in WSN. Each node creates a certain number of rateless packets, that are initially empty packets that travel across the network as random walkers. The goal in [16] is to use packets encoded in a distributed fashion that will be stored at random locations in the network to maximize data persistence in the WSN. Each rateless packet is associated with a degree chosen following the standard LT degree distribution, and τ, the mixing time of the graph, is supposed to be known. Each rateless packet performs a random walk across the network and a novel information is combined only once every τ hops; when a new information is added the packet degree is reduced by one. When the degree becomes zero, the

rateless packet performs τ supplementary hops to hit the node that will store it. However, the focus of the paper is to increase data persistence; the time required for the distribution of the rateless packets is not studied.

It is worth pointing out that in all previous studies based on coding, the information to be spread is assumed to be static, or alternatively it is divided into generations. The nodes are allowed to update their information only when a new generation is initiated. The concept of generation represents a great limitation for practical distributed application since it requires synchronization. Moreover, the size of the information to be spread is not considered as an important parameter of the system.

Finally, another class of technique adopted for spreading data across a distributed system is based on the recent results in the area of compressive sensing [17], [18]. These approaches rely on the compressibility of the information and does not apply in the general case when no prior statistical knowledge of the data is available.

3 System Description

In this paper we model the interface peers of a Grid system and the connections among them as a graph $G(V, E)$, where V and E are the set of interface peers and edges connecting them, respectively. Each node of the network is uniquely identified by an identifier ID. The ID can be assigned by a fixed rendezvous node, e.g. a tracker, or can be represented by the IP, port address of the node. Each node $v_j \in V$ owns an m-bits information $x_{v_j}^{t_j}$, where t_j is a time-stamp or an integer that is incremented each time the information in v_j changes. To simplify the notation in the rest of the paper we assume that v_j coincides with the ID of node; t_j is usually referred to as the generation number. In our settings a node can update its information asynchronously with respect to the rest of the network, increasing the generation number associated with the information.

The goal of nodes is to communicate with one another the respective information, so as to realize a concurrent broadcasting of all the information collected by all the nodes in the network. This must be done indefinitely often at an arbitrary rate λ by each node. This observation rules out any centralized solution where all nodes report to a common monitoring node, that in turn must propagate the collected data to all the participants. This approach is clearly infeasible because it imposes a huge amount of traffic to and from the monitoring node, not to mention the issues related to the election and vulnerability of a centralized sink (see discussion in Section 7).

Therefore, in this paper we propose a fully distributed solution based on random walks. Each node is allowed to start a limited number w of packets that are the random walkers propagating the information in the network. The parameter w clearly allows one to control the amount of traffic injected in the network. On every reception by a node, the packet is forwarded to a random neighbor thus realizing a simple form of probabilistic gossiping. It is well known that network coding solutions, e.g. carrying linear combinations of the collected information, increases the performance in terms of throughput, robustness and persistence [11], [16]. On the other hand, coding approaches exhibit two main shortcomings. The first and most studied issue is represented by the added computational complexity. A possible solution that has already been proposed in the literature [16] is to simplify the original random network coding approach, that requires one to combine the data blocks in high order Galois Field, with systems based on simple binary combinations, e.g. XOR. Our work copes with the complexity issue using a simple class of rateless codes, known as Luby Transform (LT) codes [19]. The second most relevant shortcoming of NC is represented the impossibility for a node to update asynchronously the information it combines without catastrophically impacting on the decoding capability of all the other nodes. Indeed, the nodes keep collecting linear combinations of a set of unknowns until they successfully invert the corresponding system of equations. Clearly, the system of linear equations is meaningful if one keeps combining the same information. On the contrary, in this paper we propose a novel decoding approach for LT codes that is resilient to asynchronous changes of the information. In conclusion, we let each node propagate a fixed number of packets carrying coded information of the nodes that the packets have hit performing a random walk along $G(V, E)$. All the nodes use the received packets to solve a system of linear equations allowing them to retrieve the data associated with all the information collected by the network in a timely, complete and robust way.

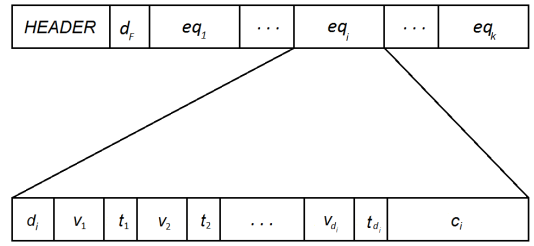


Fig. 1. Packet structure.

In the following the details of the proposed random walk coding strategy and the design of the novel LT decoding algorithm are presented.

3.1 Random walk LT coding

In this section the structure of the packet spreading the coded information using a random walk approach is described.

The packets format is shown in Fig. 1. Each packet is composed of a header followed by a set of equa-

tions $\{eq_1, \ldots, eq_k\}$. It is worth pointing out that one packet carries multiple coded information at a time. An equation eq_i, $i = 1, \ldots, k$, is characterized by a degree d_i, a set of d_i pairs v_j/t_j, $j = 1, \ldots, d_i$, and an encoded m bits message c_i. The degree d_i of the equation corresponds to the number of nodes that have encoded their information $x_{v_j}^{t_j}$ in c_i. The message c_i corresponds to all the information that has been progressively XORed by the nodes hit by the packet. In fact, we have

$$c_i = \sum_{j=1}^{d_i} x_{v_j}^{t_j},$$

where the summation is performed in $GF(2)$.

The coding strategy follows the standard LT approach [19], where the degree of the equations is constrained to follow the probability density function known as Robust Soliton distribution (RSD), that in turn guarantees asymptotically optimal coding performance. In other words, using RSD it is very likely that each node can recover N coded information from any set of $N' \geq N$ equations, with N' that is arbitrarily close to N in the limit $N \to \infty$. This property holds for any information size m. RSD depends on N, that should be known by the nodes; however, an approximation of N is sufficient to compute RSD without a great loss for the performance if a decoder based on Gaussian Elimination is used [20], [21]. This approximation could be calculated by the nodes observing the IDs of the nodes contained in the previously received equations. To cope with the creation of the equation according to the RSD, each packet carries in the header part the signaling of the degree d_F that must be achieved by the equation under formation in the packet (that in our settings is the first equation written in the packet body form left to right). When a node v_j at generation t_j receives a packet, it checks if the degree of the first equation stored in the packet has reached the requested degree. If $d_F > d_1$, and hence the target degree has not been reached yet, the node performs 3 operations: it XORs its information to the term c_1, i.e. $c_1 = c_1 \oplus x_{v_j}^{t_j}$. Then the degree d_1 of the equation is incremented and the corresponding field in the packet updated. Finally, the node v_j and the information timestamp t_j are appended to the equation. On the other hand, if $d_F = d_1$, the first equation has already achieved the requested degree, hence a new equation is created and stored as the new first equation, while the other equations are shifted, e.g. eq_i becomes eq_{i+1} for $i = 1 \ldots k$. To create a new equation eq_1 a node draws a random degree from RSD and stores it in the d_F field of the packet header. Then $d_1 = 1$ is set, its v_j, its actual timestamp t_j and information $c_1 = x_{v_j}^{t_j}$ are written in the proper fields. Every packet created or updated by a node is then forwarded to another node, randomly selected among the local neighbors.

The number of hops globally taken by a packet is not limited in our system. The only limitation is represented by the maximum packet size DIM, that is generally imposed by the maximum transfer unit (MTU) allowed by the underlying communication technology at the physical layer. When a packet approaches the maximum dimension DIM, the eldest equation carried by it is deleted since it is very likely to carry aged or already known information.

In case of a dynamic network, where nodes can randomly join and leave the graph $G(V, E)$ and/or in presence of unreliable links that turn into packet losses, a mechanism to acknowledge the presence of a given packet in the network must be devised. As an example, an acknowledgement timer (a Time-to-Live field) and the the address of the originator can be added to the message. As usual, The acknowledgement timer field is initialized to a constant value upon the packet creation, then each node decrements it on every hop. When the acknowledgement timer reaches 0 the receiving node acknowledges the originator that its random walker is still alive. The receiving node also re-sets the acknowledgement timer to the initial value. The originator of the packet uses a timer to detect packet losses; when a timer expires before the reception of the corresponding acknowledgement message the node is allowed to regenerate the packet.

4 Asynchronous LT decoding

The information spread by the random walkers can be recovered by any node in the network as soon as the number of equations required to perform an LT decoding algorithm [19], [20], [22], [21] has been collected. Since our goal is to reconstruct the information as quickly as possible, all the equations carried by each packet, including the ones that are still in progress, are buffered by each node. If we assume that the number of nodes $|V|$ in the network is equal to N, the decoder task can be formulated as the solution of the following system of linear equations $\mathbf{Gx} = \mathbf{c}$ where $\mathbf{G}$ is an $N \times N$ binary[1] matrix whose rows represent the N possible independent equations collected by the node, $\mathbf{x}$ and $\mathbf{c}$ are $N \times 1$ column vectors representing the N unknown pieces of information and the corresponding buffered linear combinations carried by the packets payloads. Both $\mathbf{x}$ and $\mathbf{c}$ contain m-bit elements. The node can recover all the information $\mathbf{x}$ using a progressive form of Gaussian Elimination [22], [21] to solve the system. Clearly, this will require all the nodes in the network to keep their information constant to avoid perturbing the solution of the system.

To circumvent this strong limitation we propose a novel decoding strategy that does not assume that the N pieces of information $\mathbf{x}$ are constant. To this end, the vector $\mathbf{x}$ is extended to the $(\nu+1) \cdot N \times 1$ vector $\tilde{\mathbf{x}}$, where ν represents the memory used to store the values of the past generations of the information. The bottom part of

1. The matrix is binary since LT coding is performed in GF(2) and it is analogous to the code generation matrix of standard forward error correction linear block codes.

the column vector (last N elements) refer to the most recent version of each piece of information, whereas the top $\nu \cdot N$ elements are related to past generations. In other words, vector $\tilde{\mathbf{x}}$ represents a snapshot of the information collected in the network with a sliding window mechanism including the $(\nu+1)$ most recent generations for the information of each node. It is worth pointing out that the generation of different nodes is not synchronized and proper mapping between a vector position and the respective node ID and generation timestamp must be stored as auxiliary data. In particular, two auxiliary vectors are used; a $N \times 1$ vector $\mathbf{a}$, whose i-th element is the ID of the node associated to the information in positions $i, i+N, \ldots, i+\nu N$ of $\tilde{\mathbf{x}}$, and a $N \times 1$ vector $\mathbf{t}$, containing the time-stamps of the most recent information collected for each node. According to the previous notation it follows that $\tilde{\mathbf{x}}[i+jN] = x_{\mathbf{a}[i]}^{\mathbf{t}[i]-(\nu-j)}$, $j = 0, \ldots, \nu$, where square brackets are used to index a single element of a vector. The vector $\mathbf{a}$ allows each node to map the ID of the nodes that are being progressively discovered in the received packet to a particular set of rows of the decoding matrix. Adopting this extended notation, the decoding task can be formulated as $\tilde{\mathbf{G}}\tilde{\mathbf{x}} = \tilde{\mathbf{c}}$, where $\tilde{\mathbf{G}}$ turns to be a $(\nu+1)N \times (\nu+1)N$ extended decoding matrix specifying the linear combinations among the $(\nu+1)$ most recent versions of information circulated in the network. Analogously vector $\tilde{\mathbf{c}}$ collects the results of such linear combinations. The idea is to keep the decoding as updated as possible aiming at reconstructing the last N elements of $\tilde{\mathbf{x}}$. Nonetheless, the equations referring to previous versions of a given node are not invalidated, provided that they fit in the window of the ν past generations with respect to the most recent timestamp observed for each information item. Indeed, given the fact that the nodes are not synchronous, some outdated information is usually still being propagated by packets and can be exploited by progressive decoding as described in the following.

The objective of the proposed decoder is twofold. First, given all the equations extracted from the packet we want to decode, the maximum number of information items $x_{v_j}^{t_j}$. Moreover, this should be done incrementally while processing the incoming packets. Second, we propose a strategy to manage the extended decoding matrix $\tilde{\mathbf{G}}$ so as to make the decoding process robust to asynchronous updates of the information.

At startup $\tilde{\mathbf{G}}$, $\tilde{\mathbf{x}}$, $\tilde{\mathbf{c}}$, $\mathbf{a}$ and $\mathbf{t}$ are initialized as empty vectors. At a high level the decoder operates as shown in Algorithm 1: every received packet RW is processed by function `Decode`(RW) that extracts all the transported equation/linear combination pairs (eq_i, c_i). Then the algorithm maps each pair using the extended notation, and finally runs an incremental version of the Gaussian Elimination algorithm to insert the new equation in the linear system.

The function $\tilde{eq}$ =`Extend_Equation` (eq), reported as Algorithm 2, is designed to progressively update the extended notation, according to the node identities and information timestamps observed in RW. Each collected equation eq_i is remapped as $\tilde{eq}_i$ using the actual extended notation, i.e. the incoming eq_i, that is represented as a list of IDs and timestamps, is formatted as binary vector with 1 in each position corresponding to the information XORed in c_i and 0 otherwise. Clearly $\tilde{eq}_i$ represents a potential row of $\tilde{\mathbf{G}}$. Every time a new identity is observed, the data structures $\tilde{\mathbf{G}}$ and $\tilde{\mathbf{c}}$ are extended and the information in vector $\mathbf{a}, \mathbf{t}$ updated accordingly. The collected equations referring to obsolete information, i.e. elder than ν past generations, are discarded. On the contrary as soon as a generation update is observed, i.e. if $t_j > \mathbf{t}[l]$ (using pseudo code notation), the `Update_Decoder` function described in the following is invoked to re-synchronize the decoding matrix.

Algorithm 1 `Decode`(RW)

```
for All (eq_i, c_i) ∈ RW do
  ẽq_i = Extend_Equation (eq_i)
  Insert_Equation(ẽq_i, c_i)
end for
```

Algorithm 2 $\tilde{eq}$ = `Extend_Equation` (eq)

```
Initialize ẽq as a vector of (ν+1)N zeros.
for (v_j, t_j) ∈ eq do
  if ∄l : a[l] = v_j then
    {New node is observed for the first time}
    N = N + 1
    Extend G̃, c̃, Update a, t
  else
    if t[l] − ν ≤ t_j ≤ t[l] then
      {Information in managed generation window}
      ẽq[l + (ν − (t[l] − t_j))N] = 1
    else if t_j > t[l] then
      {Update generation}
      Update_Decoder(l, t_j − t[l])
      t[l] = t_j
      ẽq[l + (ν − (t[l] − t_j))N] = 1
    else
      {Obsolete information}
      Discard eq
    end if
  end if
end for
```

The core of the decoding task is represented by Algorithm 3, that shows the details of the `Insert_Equation`$(\tilde{eq}, c)$ function used to populate the rows of $\tilde{\mathbf{G}}$. The goal of function `Insert_Equation` is to insert each extended equation $\tilde{eq}$ guaranteeing that $\tilde{\mathbf{G}}$ remains in upper triangular matrix and that the maximum number of $x_{v_j}^{t_j}$ are decoded. This latter constraint is achieved by guaranteeing that the decoding matrix is filled with linear independent combinations of the

information and performing partial back substitution on each row insertion. To this end, the algorithms proposed in [22], [21] have been extended to operate in the case of non synchronous information updates. At startup, matrix $\tilde{\mathbf{G}}$ is empty and the incoming equation is simply inserted as the row corresponding to the position of the leftmost 1 in the equation, thus guaranteeing that an upper triangular matrix is formed. The next equations may find the corresponding row is already taken. In such a case, the leftmost 1s are progressively canceled by XOR operations with the matrix rows. At the end of this process whether the equation hits a still empty row, or all its 1s are canceled out (the equation is linearly dependent on the collected rows of $\tilde{\mathbf{G}}$ and can be discarded). The described process guarantees the creation of an upper triangular $\tilde{\mathbf{G}}$. The last part of the algorithm, referred to as partial back-substitution, aims at progressively decoding the maximum amount of information items. This is achieved by combining the equation inserted as the l-th row of $\tilde{\mathbf{G}}$ with all the upper rows that have 1 in column l. As soon as a row with a single 1 on the diagonal is formed, the corresponding information item is known. This process has been demonstrated to yield optimal partial decoding performance in [21].

Algorithm 3 `Insert_Equation` $(\tilde{eq}, c)$

```
{Clean all possible 1s from ẽq}
for i from 1 to k(ν+1) do
  if ẽq[i] = 1 and G̃[i][i] = 1 then
    ẽq = XOR(ẽq,G̃[i][·])
    c = XOR(c,c̃[i])
  end if
end for
{Row insertion}
Find position l of leftmost 1 in ẽq
if l ≤ k(ν+1) then
  G̃[l][·] = ẽq {Insert ẽq as l-th row}
  c̃[l] = c
  {Partial back-substitution}
  for i from 1 to l−1 do
    if G̃[i][l] = 1 then
      G̃[i][·] = XOR(G̃[i][·],G̃[i][l])
      c̃[i] = XOR(c̃[i],c̃[l])
    end if
  end for
else
  delete ẽq, c
end if
```

Finally, the decoder requires the already mentioned `Update_Decoder`(l, Δ_t) procedure that is used to increment by Δ_t the generation of the node stored in the l-th row of the decoding matrix. The details of this procedure are described using pseudo code in Appendix A.

5 Recovery Time Model

In this section we provide an analysis of the time required to spread all the local information to all the participants in the network, that in the following is defined as *recovery time*. In particular, we are interested in modeling the recovery time as a function of the size of the local information m, the number of random walkers generated per node w and number of nodes in the network N, given the constraint on the maximum size of the random walk packets DIM. Moreover, the proposed analytical model permits to compare the coded approach versus an analogous system without coding, i.e. when the information is gossiped explicitly. In fact, the proposed approach degenerates into an uncoded system if one uses a constant equation degree equal to one; the packet format of Fig. 1 can hence be changed removing the d_i and d_F fields, that turn to be useless.

Given the size DIM (in bits) of the transmission packet, we can suppose that a packet is divided into two parts: the header and the free space, of size h and $f = DIM - h$ respectively. From now on, we refer to the memory occupation, in bits, of an object as its size. In every equation, we call g the size of the pair (v_l, t_l) and m the size of the combined message c_i. Following the packet specification in Fig. 1, we have that the size of a single equation turns to be $e_C = d_i \cdot g + m$, where d_i is the degree of the equation. In the following derivation we perform a mean value approximation assuming that all equations have the same degree $d_i = d$, where $d = 2 \ln N$ [19] is the average degree of LT codes. In the previous approximation we also considered as negligible the cost of signaling the value of d. Analogously, the size of the message gossiped by an uncoded system is simply $e_U = g + m$, i.e. on every hop a new identity along with the corresponding local information is added to the payload of the packet. If we call n_U and n_C the maximum number of equations storable in an uncoded and encoded packet respectively, we have that

$$n_U = \left\lfloor \frac{f}{e_U} \right\rfloor = \left\lfloor \frac{f}{g+m} \right\rfloor, n_C = \left\lfloor \frac{f}{e_C} \right\rfloor = \left\lfloor \frac{f}{d \cdot g + m} \right\rfloor.$$

If we assume to set $g = 2 \log_2 N$, i.e. twice the number of bits required to map N unique identifiers both n_U and n_C turn out to be functions of m and N.

In Appendix B we show that is possible to predict analytically the number of hops T_C required to distribute a certain number of equations R_C using the coded approach. Analogously, the number of hops T_U needed by the uncoded counterpart as a function of the number of equations R_U is derived. The result is as follows:

$$\begin{aligned} T_U &= \begin{cases} \sqrt{\frac{2R_U}{w}} & if\ R_U < \frac{n_U(n_U-1)}{2} \\ \frac{R_U}{wn_U} + \frac{n_U-1}{2} & otherwise \end{cases} \\ T_C &= \begin{cases} \sqrt{\frac{2dR_C}{w}} & if\ R_C < d\frac{n_C(n_C-1)}{2} \\ \frac{R_C}{wn_C} + d\frac{n_C-1}{2} & otherwise \end{cases} \end{aligned} \tag{1}$$

Our objective is to estimate the number of hops (i.e. the time) needed by any node to retrieve all the information

present in the network. Equations (1) can be used to calculate such a recovery time. In a network of N nodes, for the encoded case it is theoretically sufficient to receive $R_C = N$ equations in order to enable successful decoding. On the contrary, for the uncoded case the number of equations turns to be higher; in fact, this can be recognized as an instance of the coupon collector's problem. In the case of equally distributed coupons, every node needs about $R_U = N \cdot (\ln N + \frac{1}{2})$ equations to collect all the information. However, in our system the probability of receiving a certain equation is not uniform and depends on the length and the number of the paths that connect the nodes; in Sect. 6 we will show that the coupon collector's approximation is a lower bound for the actual performance of the uncoded system. Substituting R_C and R_U in (1) one gets the recovery time as a function of w, DIM, N and n_C (or n_U); we shall recall that in turn n_C and n_U depend on m and N. In Fig. 2(a) the recovery time evaluated according to (1) is reported as a function of m in the case $DIM = 1500$ bytes, i.e. a typical value of the maximum size of a UDP packet, and for a network with $N = 1000$ nodes. It turns out that the proposed approach is able to spread the information much faster than the uncoded counterpart and that the gap increases for larger information sizes. In Fig. 2(b) the recovery time is shown when m is fixed to 128 bytes and the network size is increased. These results point out that the coded approach scales much better with the network size, making our solution very attractive for large distributed systems. Additional analytical results are reported in Appendix B for conciseness.

6 Simulation results

In this section we present the results obtained through a simulative study of the proposed approach and the corresponding uncoded counterpart. The simulator creates a random network of N nodes; each node is initially connected to a fixed number N_{neigh} of neighbors, randomly sampled among all the participants, i.e., we create a random regular graph. This amounts to assume that a common rendez-vous point (e.g., a tracker node) is available to obtain information on the initial list of nodes to contact. The information $x_{v_i}^{t_i=1}$ of each node is initialized to a random value that can be updated asynchronously incrementing the corresponding timestamp t_i. Each node runs the asynchronous LT decoder described in Sect. 4 with memory parameter $\nu = 1$. Experiments will show that even such a low memory parameter permits to outperform the uncoded system.

The simulation time is slotted; at startup each node initiates w new random walkers, sending w packets to a set of randomly chosen neighbors. In each time slot T the packets received by any node at time $T-1$ are updated with a local piece of information and forwarded randomly. The maximum packet size is fixed to 1500 bytes in all our simulations. As predicted by the analytical model of Section 5, the parameter w has the only effect to change the time scale of the results. For conciseness, in the following we perform all the simulations in the case $w = 1$.

The overlay network is managed borrowing ideas from [23]. The first kind of overlay network we consider is static meaning that nodes do not churn; nevertheless we let their neighborhood change with time. In particular, we start with a random regular graph with N nodes where each node has N_{neigh} neighbors. For the first 3000 time slots, at each time slot 30 random nodes shuffle their neighborhood by exchanging one random neighbor. At that point we use the resulting graph to start the simulation for data delivery and once every two time slots 30 randomly chosen nodes shuffle their neighborhood.

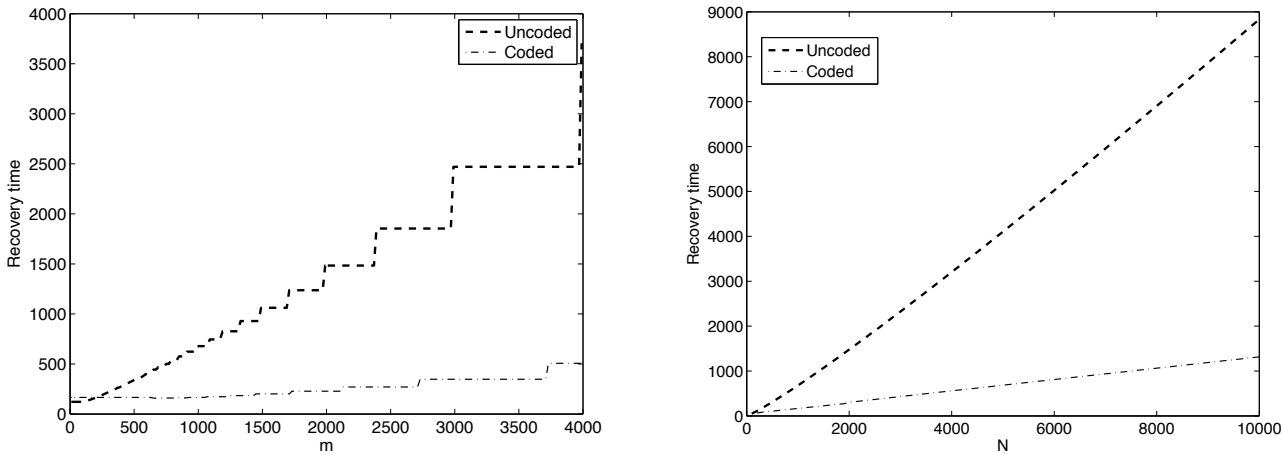
We also consider the case of a dynamic network where the nodes can randomly join and leave. In addition to the overlay management described for the static network case (i.e., network initialization and periodical neighborhood change) when a node joins it connects to a random set of N_{neigh} nodes. When a node leaves its neighbors replace it through the described shuffling mechanism.

Moreover, a node that turns off in time slot T does not forward any packet eventually received at time $T-1$. To keep constant the overall number of packets in the network ideal signaling is assumed, so as that the originators of the lost packets can restart new random walkers at time $T+1$. It is clear that the acknowledgement mechanism would be implemented according to a practical distributed protocol, adding some delay with respect to the ideal behavior of our simulator. Nonetheless, the goal of our analysis being a relative comparison between the performance of the coded and uncoded system, we are not interested to add the sub-optimality of an actual implementation that would equally affect both approaches.

The system performance is measured in terms of the amount of information that can be gathered by the nodes and the time required to achieve the result. In particular, for each node v_l we calculate the percentage of information retrieved by that node as a function of time T:

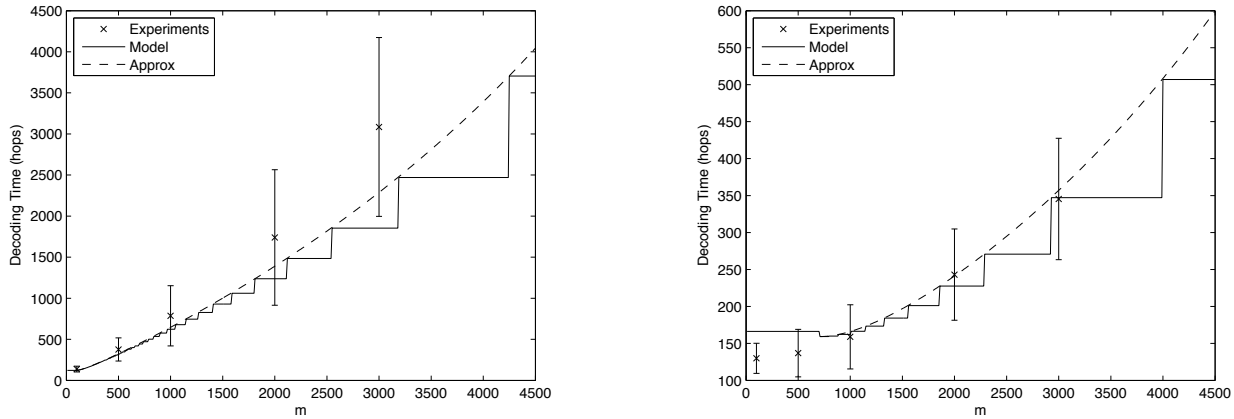
$$p^l(T) = \frac{\sum_{j=1}^{N} \left(\sum_{i=1}^{t_j(T)} \mathrm{dec}_l(i,j) \right)}{\sum_{i=1}^{N} t_i(T)}, \tag{2}$$

where $\forall j$, $t_j(T)$ is the timestamp achieved by node v_j at time T, and $\mathrm{dec}_l(i,j)$ is a logic variable identifying the pieces of information currently decoded by node v_l. In particular, $dec_l(i,j) = 1$ if v_l has recovered the generation $i \leq t_j(T)$ of node v_j, i.e. if $x_{v_j}^{t_j=i}$ is known by v_l at time T, and equals 0 otherwise. Since $\sum_{i=1}^{N} t_i(T)$ sums to the overall number of data items disseminated in the network from the beginning of the simulation, the metric in (2) represents the percentage of overall information



(a) (b)

Fig. 2. Recovery time as a function of m in bits (a) and N (b).



(a) U system (b) C system

Fig. 3. Experimental recovery time and analytical model for the coded and un-coded systems as a function of m (stable network with constant information).

collected by v_l at a given time instant. As an index of the performance of the whole network we use

$$P(T) = \frac{\sum_{l \in A(T)} p^l(T)}{|A(T)|}, \tag{3}$$

that is the average value of the previous index computed on the set of nodes $A(T)$ that are active, i.e. in the on state, at time T. In the case of a stable network with constant information, $P(T)$ can be used to calculate the recovery time defined in Section 5; indeed, the recovery time can be computed as $\{min(T)|P(T) = 1\}$, i.e. the first time instant when all the nodes of the network know the overall information. Finally, we recall that all the numerical results based on the previous definitions have been averaged over 30 independent trials so as to compute the confidence interval and guarantee statistically meaningful values. Every trial uses a different seed for random generators, thus resulting into statistically independent outcomes for both the network topology and nodes behavior.

We start comparing the coded (C) and uncoded (U) approaches in the simplest configuration, characterized by a stable network of $N = 1000$ nodes with $N_{neigh} = 50$ and constant information. In this case every node has to collect $N - 1$ pieces of information owned by the others. This particular case allows us to validate the analytical model presented in Section 5. In Fig. 3 we compare the experimental average recovery time for 5 values of information size $m = 100, 500, 1000, 2000, 3000$ bits and compare them with the analytical model of equation (1). The experimental results are shown as error bars reporting the 95% confidence interval whereas the full and dash lines represent the exact and approximated version of equation (1), respectively. First of all, it can be observed that the analytical model is quite accurate in the C case (right graph), with the exception for the smallest values of m. As anticipated, the analytical re-

sults represent a lower bound for the performance of the U system (left graph). This is due to the coupon collector approximation adopted for the U model, that assumes that all data $x_{v_l}^{t_l}$ can be gathered by a certain node with the equal probability. This assumption is clearly violated for large value of m, since a piece of information is soon shifted out of the packet body to make room for the new ones. In other words, every piece of information takes very few hops in these case. Under this circumstances, node v_l is not able to sample the pieces of information randomly and the coupon collector approximation does not hold any more. Another way to explain this phenomenon is that the number of hops performed by a given piece of information is less than the mixing time of the graph. In conclusion, this first set of experimental results from one hand validates our analytical model, from the other hand confirms the significant advantage of the coded approach in terms of recovery time. As already noted, the gap between the 2 approaches increases for larger information m, or equivalently when decreasing the maximum transfer unit (DIM). As an example, for $m = 2000$ bits the U solution exhibits a recovery time of 1739 ± 420 hops whereas the C solution achieves the same result in 243 ± 31 hops.

For the same scenario, in Fig. 4 the value of $P(T)$, i.e. the average percentage of collected information, is shown for various values of m. Looking at this index, it is possible to observe how the information propagates into the network as a function of time. The U system initially collects a great number of innovative pieces of information, but takes a long time to find the last information. On the contrary, the C system, that is initially penalized by the observation of encoded messages that do not allow to know the actual information, is able to recover the whole information much earlier.

More simulation results obtained in scenarios with dynamic network topology and asynchronous information updates are reported in Appendix C.

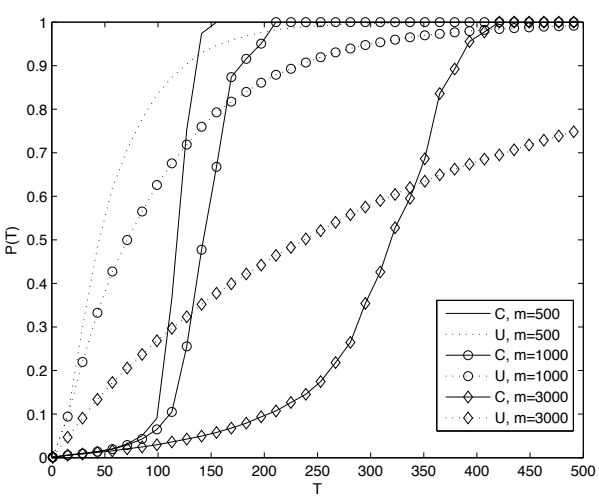


Fig. 4. $P(T)$ as a function of time for some values of m (stable network with constant information).

7 Other approaches

In the following we argue that alternative approaches to allow nodes of a distributed application to obtain a local view of some global system property are less effective and efficient.

The simplest solution to this problem is to have nodes report their local data to a well-known central repository node. Access to the global system property defined on all local data is obtained by sending queries to the repository that provides the requester node with one or more responses. This solution is neither scalable nor resilient: the aggregate request load is equal to $\lambda \cdot N$ and if we view the repository node as a server whose service capacity is equal to μ response/sec we obtain a load factor $\rho = \frac{\lambda \cdot N}{\mu}$ that quickly becomes greater than 1 leading to queries loss and unbounded delays [24]. A similar reasoning must be done for the load generated by asynchronous data updates: the load factor for this service is $\rho_c = \frac{\lambda_c \cdot N}{\mu}$ where λ_c is the rate of data update for a node.

Decentralized solutions might work better. Each node may spread its local updated information by means of flooding or gossiping. In flooding each node sends update messages to its neighbors containing the local data and the current generation number. These neighbors propagate the update messages to their neighbors up to a maximum number of hops (the message Time-to-Live). In gossiping activity is organized in *rounds*: each node stores a maximum number of update messages, each message is forwarded a maximum number of times, and each time a node forwards a message it randomly selects a subset of recipient nodes (the number of selected recipients is called *fanout*). To spread the current generation version of the local data a node starts a gossiping round. It is proved that atomic reliable broadcasting, i.e., all nodes receive the data disseminated by a round initiator, is achieved with high probability if the fanout is on average $O(logN)$ taking $O(logN)$ rounds to complete [3].

As a consequence, flooding and gossiping are characterized by high redundancy, i.e., the same update message can be received several times by the same node, especially by those with a high number of incoming connections. This translates into the possibility of nodes saturating their processing and communication capacities. Indeed, it is easy to show that in the best case the load factor at each peer is $\rho_c = \frac{\lambda_c \cdot N}{\mu}$, the same of the centralized solution. Furthermore, this and other issues in gossiping were discussed in [5] where the authors explicit numerous hidden assumptions that are necessary to ensure robustness of gossip-based protocols and that thus make gossiping not a viable option in the context we consider in this paper.

In our solution the load factor at each peer is not dependent on λ_c and N. To show this we note that

the probability that a random walk starting at any peer is at peer v in the long run (the so called *mixing time* of the random walk) is equal to $\frac{|N_{neigh}(v)|}{|E|}$ [25] where $N_{neigh}(v)$ denotes the number of neighbors of node v. This probability is equal to $\frac{1}{N}$ for random regular graphs, i.e., when all nodes have the same number of neighbors; this can be considered a good approximation when the variance of the number of neighbors is small, e.g., for classical Erdos-Renyi random graphs. Since the total number of random walkers is equal to wN it follows that the average number of walkers at each node in each time slot is equal to w. This represents a constant and tunable communication load that can be designed not to overload peers.

8 Conclusion

In this paper we have shown that the recent advances in rateless coding and decoding can be profitably exploited to achieve a robust and timely P2P resource location technique in Grid systems. The major novelty of the proposed approach lies in the use of network coding principles in a scenario where local data can be updated asynchronously. Moreover, as opposed to some forms of distributed storage proposed in the literature, our proposal realizes a continuous update of the global information across the whole distributed system, while keeping the amount of traffic under control.

From the algorithmic point of view, the major contribution is represented by the design of a novel decoder for rateless codes that is robust to asynchronous updates of the information. Another interesting result that we achieved is the development of a simple analytical model for the estimation of the time required to spread the information as a function of the network and information sizes, given a constraint on the MTU allowed by the available transmission protocol. Such a model can be exploited for the estimation of the performance and for the selection of some important parameters of the system. The analytical results show that the proposed coded approach reduces the time required to communicate all the information with respect to an analogous system without coding. Furthermore we prove that such gain increases with the size of the information to be spread, or analogously when the MTU shall be very limited. Another paramount result is that the encoded system scales better than the uncoded one when the number of nodes in the distributed system increases.

Finally, the designed simulator shows that the system is very efficient in many different scenarios characterized by network dynamics and information updates that cannot be analyzed with the analytical approach.

References

[1] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi, "Peer-to-peer resource discovery in grids: Models and systems," *Future Generation Computer Systems*, vol. 23, no. 7, pp. 864–878, 2007.

[2] V. Vijayakumar, R. S. Wahida Banu, and J. H. Abawajy, "An efficient approach based on trust and reputation for secured selection of grid resources," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 1, pp. 1–17, 2012.

[3] A. Kermarrec, L. Massoulié, and A. Ganesh, "Probabilistic reliable dissemination in large-scale systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 3, pp. 248–258, Mar. 2003.

[4] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems*, vol. 23, no. 3, pp. 219–252, Aug. 2005.

[5] L. Alvisi and et al, "How robust are gossip-based communication protocols?" *Operating Systems Review*, vol. 41, no. 5, pp. 14–18, Oct. 2007.

[6] R. van Renesse, D. Dumitriu, V. Gough, and C. Thomas, "Efficient reconciliation and flow control for anti-entropy protocols," in *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware, LADIS '08*. ACM, 2008.

[7] N. Carvalho, J. Pereira, R. Oliveira, and L. Rodrigues, "Emergent structure in unstructured epidemic multicast," in *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*. IEEE, 2007, pp. 481–490.

[8] J. Leitao, J. Pereira, and L. Rodrigues, "Hyparview: A membership protocol for reliable gossip-based broadcast," in *DSN*, vol. 7, 2007, pp. 419–429.

[9] M. Ferreira, J. Leitao, and L. Rodrigues, "Thicket: A protocol for building and maintaining multiple trees in a p2p overlay," in *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, 2010, pp. 293–302.

[10] C. Tang, R. Chang, and C. Ward, "Gocast: gossip-enhanced overlay multicast for fast and dependable group communication," in *DSN*, 2005, pp. 140–149.

[11] S. Deb, M. Medard, and C. Choute, "Algebraic gossip: a network coding approach to optimal multiple rumor mongering," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2486–2507, jun 2006.

[12] M. Wang and B. Li, "How practical is network coding?" in *IWQoS*, 2006.

[13] A. G. Bimakis, V. Probhakaran, and K. Ramchandran, "Bistributed fountain codes for networked storage," in *IEEE ICASSP*, 2006.

[14] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *IEEE Infocom*, 2007.

[15] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein, "Growth codes: Maximizing sensor network data persistence," in *ACM SIGCOMM*, 2006.

[16] D. Vukobratovic, C. Stefanovic, V. Crnojevic, F. Chiti, and R. Fantacci, "Rateless packet approach for data gathering in wireless sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 9, pp. 1169–1179, Sep. 2010.

[17] J. Haupt, W. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 92–101, Mar. 2008.

[18] R. Gaeta, M. Grangetto, and M. Sereno, "Local access to sparse and large global information in p2p networks: A case for compressive sensing," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, aug 2010, pp. 1 –10.

[19] M. Luby, "LT codes," in *IEEE FOCS*, Nov. 2002, pp. 271–280.

[20] S. Kim, K. Ko, and S. Chung, "Incremental gaussian elimination decoding of raptor codes over BEC," *IEEE Communications Letters*, vol. 12, no. 4, pp. 307–309, Apr. 2008.

[21] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "An optimal partial decoding algorithm for rateless codes," in *IEEE International Symposium on Information Theory (ISIT)*, aug 2011, pp. 2731 –2735.

[22] V. Bioglio, R. Gaeta, M. Grangetto, and M. Sereno, "On the fly gaussian elimination for LT codes," *IEEE Communication Letters*, vol. 13, no. 2, pp. 953–955, Dec. 2009.

[23] S. Voulgaris, D. Gavidia, and M. van Steen, "Cyclon: Inexpensive membership management for unstructured p2p overlays," *Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005.

[24] L. Kleinrock, "Queueing systems. volume 1: Theory," 1975.

[25] L. Lovász, "Random walks on graphs: A survey," *Combinatorics, Paul Erdos is Eighty*, vol. 2, no. 1, pp. 1–46, 1993.

Valerio Bioglio V.Bioglio was born in Torino in 1984. He received his Laurea degree in Mathematics in July 2008 at the University of Turin, Italy, and his Ph.D. degree in Computer Science from the University of Torino, in 2013. He currently is a temporary researcher at the Politecnico of Torino, Telecommunication Department. His main research interests include error correcting codes, with a particular interest in rateless codes, and compressed sensing.

Rossano Gaeta Rossano Gaeta received his Laurea and Ph.D. degrees in Computer Science from the University of Torino, Italy, in 1992 and 1997, respectively. He is currently Associate Professor at the Computer Science Department, University of Torino. He has been recipient of the Best Paper award at the 14-th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006) and at the 26th International Symposium on Computer Performance, Modeling, Measurements, and Evaluation (PERFORMANCE 2007). His current research interests include the design and evaluation of peer-to- peer computing systems and the analysis of compressive sensing and coding techniques in distributed applications.

Marco Grangetto M. Grangetto (S99-M03-SM09) received his Electrical Engineering degree and Ph.D. degree from the Politecnico di Torino, Italy, in 1999 and 2003, respectively. He is currently Associate Professor at the Computer Science Department, University of Torino. His research interests are in the fields of multimedia signal processing and networking. In particular, his expertise includes wavelets, image and video coding, data compression, video error concealment, error resilient video coding unequal error protection, and joint source channel coding. Prof. Grangetto was awarded the Premio Optime by Unione Industriale di Torino in September 2000, and a Fulbright grant in 2001 for a research period with the Department of Electrical and Computer Engineering, University of California at San Diego. He has participated in the ISO standardization activities on Part 11 of the JPEG 2000 standard. He has been a member of the Technical Program Committee for several international conferences, including the IEEE ICME, ICIP, ICASSP, and ISCAS.

Matteo Sereno Matteo Sereno was born in Nocera Inferiore, Italy. He received the Laurea degree in Computer Science from the University of Salerno, in 1987 and the Ph.D. degree in Computer Science from the University of Torino, in 1992. He is currently Full Professor at the Computer Science Department, University of Torino. His current research interests are in the area of performance evaluation of computer systems, communication networks, peer-to-peer systems, compressive sensing and coding techniques in distributed applications, game theory, queuing networks, and stochastic Petri net models.

APPENDIX A

In this section we describe the `Update_Decoder`(l, Δ_t) procedure, shown using pseudo code in Algorithm 4. This function is invoked when a generation update of the node corresponding to the rows $l, l+N, \ldots, l+\nu N$ of $\tilde{\mathbf{G}}$ is observed. The parameter $\Delta_t \leq \nu$ represents the gap between the last two observed generations and typically equals to 1. The procedure is used to implement a sliding window mechanism on the decoder data structures. In particular, the top Δ_t rows congruent modulo l are cleaned up while the remaining rows congruent modulo l are moved to the upper part of $\tilde{\mathbf{G}}$ so as to make room for the new Δ_t versions of the information. The same process is used to update the vector $\tilde{\mathbf{c}}$.

Algorithm 4 `Update_Decoder`(l, Δ_t)

```
for i = l, l+N, ..., l+(Δ_t − 1)N do
    G̃[i][·] = 0
    c̃[i] = 0
end for
for i = l+(Δ_t)N, l+(Δ_t+1)N, ..., l+νN do
    if G̃[i][i] = 1 then
        G̃[i − Δ_t N][·] = G̃[i][·]
        G̃[i][·] = 0
        c̃[i − Δ_t N] = c̃[i]
        c̃[i] = 0
    end if
end for
```

APPENDIX B

In this section we present an analytical approach for modeling the effectiveness of the proposed approach in disseminating the information collected by the progressively coded equations. We are interested in finding the cumulative number of equations distributed by every packet, as a function of the number of hops T, termed as $N_U(T)$ and $N_C(T)$ for the uncoded and the encoded case, respectively. Both the functions depend also on the number of equations stored in the packet; in fact, the number of collected equations saturates to the maximum number that in turn depend on the maximum packet size DIM. When this limit is reached, the packet is full and the number of equations remains constant (on average for the encoded case).

In our proposal, each node is allowed to insert in a packet only its information. This is due to the risk to spread not updated information; the only information that is surely updated is the the one owned by the node. For an uncoded packet, $N_U(T) = 1+2+\cdots+T = \frac{T(T+1)}{2}$ if $T < n_U$, i.e. until the packet is not full; this happens because at every hop a new piece of information is added at the packet until the packet is full. When the packet is full,

$$\begin{aligned} N_U &= 1+2+\cdots+n_U-1+\underbrace{n_U+\cdots+n_U}_{T-(n_U-1)} = \\ &= \tfrac{n_U(n_U-1)}{2} + (T-(n_U-1))\cdot n_U = \\ &= n_U \cdot (T \cdot \tfrac{n_U-1}{2}). \end{aligned}$$

Previous derivation is less trivial in the case of an encoded packet: in fact, an equation is completed only once every d hops on average. Hence the packet is full after $T = (n_C-1)\cdot d$ hops. By setting $h = \lceil T/d \rceil$ and taking into account that in our proposal partially formed equations are collected by the nodes, when $T < (n_C-1)\cdot d$ we have:

$$\begin{aligned} N_R &= \underbrace{1+\cdots+1}_{d}+\cdots+\underbrace{(h-1)+\cdots+(h-1)}_{d}+ \\ &\quad +\underbrace{h+\cdots+h}_{T-d(h-1)} = \\ &= d\tfrac{h(h-1)}{2} + h(T-d(h-1)) = \\ &= d\tfrac{\lceil T/d \rceil \lfloor T/d \rfloor}{2} + \lceil T/d \rceil (T - d\lfloor T/d \rfloor) = \\ &= \lceil T/d \rceil \left(T - \tfrac{T\lfloor T/d \rfloor}{2}\right) \end{aligned},$$

The coded packet is full when $T = (n_C-1)\cdot d$; in this case we have,

$$\begin{aligned} N_R &= \underbrace{1+\cdots+1}_{d}+\cdots+\underbrace{(n_C-1)+\cdots+(n_C-1)}_{d}+ \\ &\quad +\underbrace{n_C+\cdots+n_C}_{T-(n_C-1)\cdot d} = \\ &= d\cdot\tfrac{n_C(n_C-1)}{2} + (T-(n_C-1)\cdot d)\cdot n_C = \\ &= n_C\cdot(T - d\cdot\tfrac{n_C-1}{2}). \end{aligned}$$

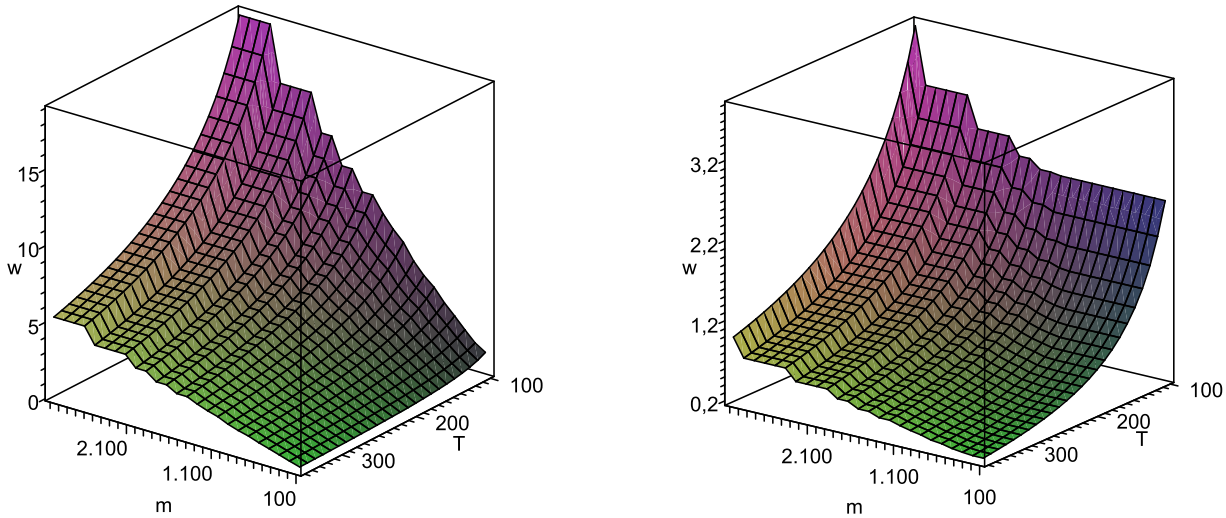
We can summarize the obtained results as follows:

$$\begin{aligned} N_U &= \begin{cases} \frac{T(T+1)}{2} & if\ T < n_U \\ n_U\left(T - \frac{n_U-1}{2}\right) & otherwise \end{cases} \\ N_C &= \begin{cases} \lceil T/d \rceil (T - \frac{d\lfloor T/d \rfloor}{2}) & if\ T < (n_C-1)d \\ n_C\left(T - d\frac{n_C-1}{2}\right) & otherwise \end{cases} \end{aligned} \tag{4}$$

Taking into account that every node creates w packets, it turns that the global number of equations spread in a network of N nodes after T hops is on average $w \cdot N \cdot N_C(T)$ ($w \cdot N \cdot N_U(T)$ in the uncoded case). If the $G(V,E)$ is regular, i.e. the distribution of the edges is peaked around its average, the equations hit any node with the same probability and therefore each node receives on average $R_C = \frac{w\cdot N\cdot N_C(T)}{N} = w\cdot N_C(T)$ equations ($R_U = w\cdot N_U(T)$ in the uncoded case). Summarizing we get

$$\begin{aligned} R_U &= \begin{cases} w\frac{T(T+1)}{2} \simeq w\frac{T^2}{2} & if\ T < n_U \\ wn_U\left(T - \frac{n_U-1}{2}\right) & otherwise \end{cases} \\ R_C &= \begin{cases} w\lceil T/d \rceil (T - \frac{d\lfloor T/d \rfloor}{2}) \simeq w\frac{T^2}{2d} & if\ T < (n_C-1)d \\ wn_C\left(T - d\frac{n_C-1}{2}\right) & otherwise \end{cases} \end{aligned} \tag{5}$$

where we introduce some linear approximations to simplify the following analysis. Using such approximations, it is possible to reverse the functions obtaining the number of hops needed to distribute a certain number of equations, i.e. T as a function of R_U and R_C, obtaining the result in Equation (1).



(a) W_U

(b) W_C

Fig. 5. W_U and W_C as a function of T and m.

The recovery time model can be used to fix the value of the parameter w (the number of packets created per node). Reversing (5) it is indeed possible to calculate the minimum value of w that probabilistically guarantees to retrieve all the information within a time interval T. Such minimum values turn to be:

$$W_U = \begin{cases} \frac{2R_U}{T(T+1)} & if\ T < n_U \\ \frac{2R_U}{n_U(2T-n_U+1)} & otherwise \end{cases}$$
$$W_C = \begin{cases} \frac{2R_C}{\lceil T/d \rceil (2T-d\lfloor T/d \rfloor)} & if\ T < (n_C-1)d \\ \frac{2R_C}{n_C(2T-d(n_C-1))} & otherwise \end{cases} \quad (6)$$

for the uncoded and encoded approaches respectively. Equations 6 are shown in Fig. 5 as a function of m and T under the hypothesis of $N = 1000$ and $DIM = 1500$ bytes. Again, for large values of m the advantage of the encoded system is apparent, i.e. the same performance in terms of recovery time requires much less network traffic.

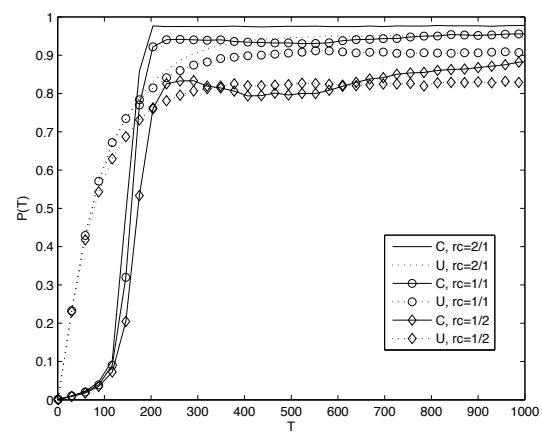
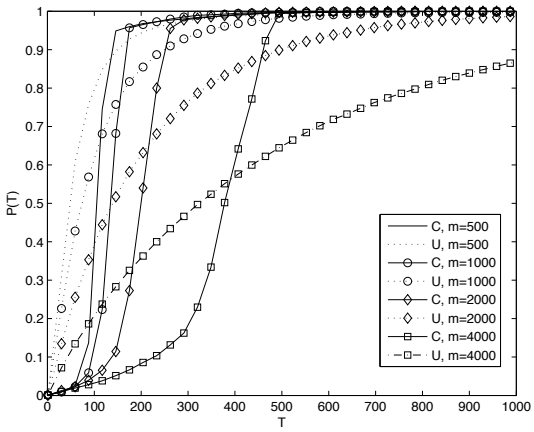


Fig. 6. $P(T)$ for different r_c, with $N = 1000$ nodes, $N_{neigh} = 50$ and $m = 1000$ (stable network and dynamic information).

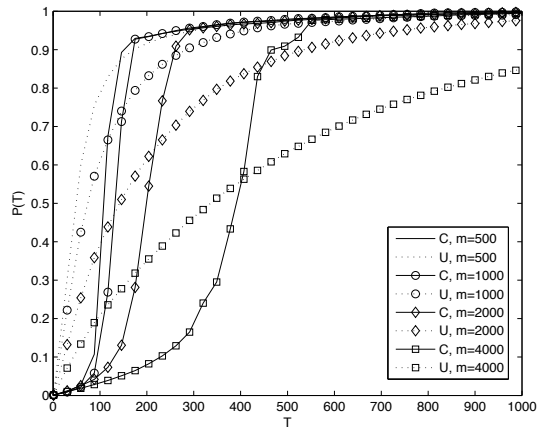
APPENDIX C

In the following we extend the simulative investigation to configurations characterized by a dynamic network topology and asynchronous information updates. In this case it is not possible to define the full recovery time. As a consequence, our analysis will be based on the simulative results, and in particular on the behavior of the index $P(T)$.

The first case that we are interested in is that of a stable network where the nodes can change their information. In this scenario, we suppose that every t_c time slots n_c randomly chosen nodes update their information. The system performance clearly depends on the rate of the information update $r_c = \frac{t_c}{n_c}$: if the information changes very frequently it is not possible to retrieve all the information on time unless we increase the network traffic, i.e. the value of w. In Fig. 6 $P(T)$ is shown for various values of the rate r_c, in a network of $N = 1000$ nodes, $N_{neigh} = 50$ and $m = 1000$. It can be noted that the C system is more robust to the information update that the U one for all the investigated rates r_c. As an example in the case $r_c = 2/1$, i.e. every 2 time slots a node change the information, the C system approaches a 100% recovery of the information in about 200 time slots; on the contrary, the U counterpart not only takes almost double of the time to saturate to its maximum performance, but the achieved recovery is limited to slightly more than 90%.



(a) $r_l = \frac{25}{20}$



(b) $r_l = \frac{50}{20}$

Fig. 7. $P(T)$ for various values of m and r_l with $N = 1000$, $N_{neigh} = 50$ (dynamic network and constant information)

.

Let us now investigate the effect of network dynamics. In this case we create a $G(V, E)$ with $N = 1000$ nodes willing to broadcast their local information. A randomly chosen set of i_l nodes is started in the off state; every t_l time slots n_l nodes pass from off to on state, i.e. join the overlay, and the same number of nodes do the opposite, thus keeping constant the number of active nodes $|A(T)| = N - i_l$. In all the following simulations we let the rate of network dynamics, defined as $r_l = \frac{t_l}{n_l}$, vary and we fix $i_l = 5n_l$.

In Fig. 7 $P(T)$ is reported, showing that the proposed system is robust also to network dynamics. It can be noted that when $r_l = 25/20$ and $m = 2000$ the recovery times for C and U are about 500 and 1000 hops, whereas increasing the network dynamic to $r_l = 50/20$ we get 750 and 1250 hops, respectively. As in all other scenarios, the advantage of the C approach is more significant for larger m.

Finally, in Figs. 8 and 9 network dynamics and asynchronous information updates are considered jointly. In Fig. 8 $P(T)$ is shown for the case of $m = 2000$ for various ratios r_l and fixing the information change rate to $r_c = \frac{50}{20}$. As expected, the more the information in the network changes the more the system has difficulty in retrieving the complete information. However, the C system consistently outperforms the U system. In Fig. 9, $r_c = \frac{50}{20}$ and $r_l = \frac{1}{1}$ are fixed, and we let m vary. As already noted, the gain of C over U increase for larger values of m.

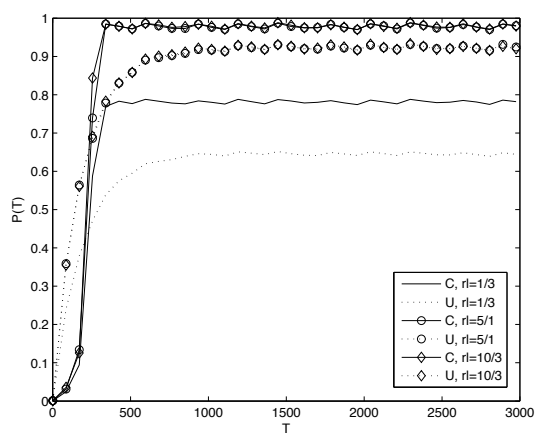


Fig. 8. $P(T)$ in the case $m = 2000$, $r_c = \frac{50}{20}$ for various values of r_l (dynamic network and information).

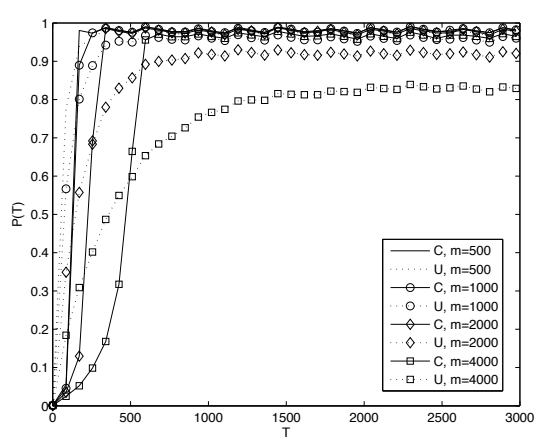


Fig. 9. $P(T)$ in the case $r_l = \frac{1}{1}$ and $r_c = \frac{50}{20}$ for various values of m (dynamic network and information).