J. Parallel Distrib. Comput. 74 (2014) 2029–2038

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Sporadic decentralized resource maintenance for P2P distributed storage networks☆

M. Martalò [a,b,*], M. Amoretti [c], M. Picone [b], G. Ferrari [b]

[a] E-Campus University, Novedrate (CO), Italy

[b] Department of Information Engineering, Università degli Studi di Parma, Italy

[c] Centro Interdipartimentale SITEIA.PARMA, Università degli Studi di Parma, Italy

HIGHLIGHTS

- We analyze a distributed storage architecture based on a DHT-based overlay.
- An innovative decentralized resource maintenance strategy has been developed.
- A complete theoretical mathematical framework and realistic simulations are provided.
- The proposed approach leads to a fully decentralized strategy.
- Maintenance bandwidth is kept very low.

ARTICLE INFO

Article history:
Received 22 August 2012
Received in revised form
29 October 2013
Accepted 6 November 2013
Available online 12 November 2013

Keywords:
Distributed storage
Decentralized maintenance
Erasure coding
Randomized network coding
Peer-to-peer (P2P)

ABSTRACT

In this paper, we propose a novel decentralized resource maintenance strategy for peer-to-peer (P2P) distributed storage networks. Our strategy relies on the Wuala overlay network architecture, (The WUALA Project). While the latter is based, for the resource distribution among peers, on the use of erasure codes, e.g., Reed–Solomon codes, here we investigate the system behavior when a simple randomized network coding strategy is applied. We propose to replace the Wuala regular and centralized strategy for resource maintenance with a decentralized strategy, where users regenerate new fragments *sporadically*, namely every time a resource is retrieved. Both strategies are analyzed, analytically and through simulations, in the presence of either erasure and network coding. It will be shown that the novel sporadic maintenance strategy, when used with randomized network coding, leads to a fully decentralized solution with management complexity much lower than common centralized solutions.

1. Introduction

Peer-to-peer (P2P) applications have attracted the interest of the research community because of their potential (e.g., high scalability, fault tolerance, etc.). An important application of the P2P paradigm is distributed storage, where information (e.g., a file) is stored with proper redundancy at different nodes of the network: this allows efficient retrieval and low probability of error. In these systems, users place their own data on Storage Nodes (StNs) using proper encoding functions, thus generating redundancy to counter-act possible faults. To protect data against theft, files are encrypted on the user's desktop, and the password never leaves his/her computer. It is necessary to have a public key infrastructure and strong node identities. This is very important to incentivize users in providing their unused disk spaces. On the other hand, it is important to include rewarding mechanisms, for users that share their storage resources, and penalties, for selfish users. A large amount of literature is available on this topic (see, e.g., [7,9]), which is, however, out of the scope of this paper.

The most popular (source) encoding strategy is based on the use of efficient maximum distance separable (MDS) *erasure codes*, e.g., Reed–Solomon codes, parity-array codes, and low-density parity-check (LDPC) codes [26,25]. With MDS coding, an information message with k symbols is encoded into a message with $n > k$ symbols (thus with coding rate $R_c = k/n$). The users can then retrieve the information by downloading a proper subset (of size k) of the encoded packets, owing the fact that this minimum size subset is sufficient to reconstruct the original information. When the

☆ This paper has been in part presented at the IEEE Int. Symposium on Network Coding (NetCod), Toronto, Canada, June 2010 and the Information Theory and Applications (ITA) Workshop, UCSD, San Diego, CA, USA, February 2011.

* Corresponding author at: Department of Information Engineering, Università degli Studi di Parma, Italy.

E-mail addresses: marco.martalo@unipr.it (M. Martalò), michele.amoretti@unipr.it (M. Amoretti), marco.picone@unipr.it (M. Picone), gianluigi.ferrari@unipr.it (G. Ferrari).

0743-7315/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jpdc.2013.11.001

information available in the overall network reduces below a critical level, a maintenance event needs to be scheduled in order to restore an amount of information sufficient to allow resource retrieval.

Since the appearance of the seminal paper [2] in 2000, network coding has held the promise to increase the throughput and reliability of a large class of distributed systems, such as wireless networks, sensor networks, P2P networks, etc. [14]. In particular, a significant (theoretical and applied) research activity in the field of network coding for P2P networks has recently spurred, thanks to [17], where, for the first time, the integration of randomized network coding [19] with the topology management of a classical P2P content distribution system (e.g., BitTorrent [6]) was proposed. In [11], the authors analyze a P2P distributed storage system where the data is not encoded with erasure codes but, rather, with random linear combinations. Results show that this approach allows to achieve the optimal redundancy–reliability tradeoff.

In this paper, we propose a decentralized network coding-based approach to P2P distributed storage, relying on the architecture of the Wuala project [29]. In particular, we reproduce at our best the architecture of the Wuala project [18], which is representative of the more general class of DHT-based overlay architectures. Unlike the Wuala-based system, where resource maintenance is periodic and centralized—and more precisely, is carried out by a few Super-Nodes (SuNs), we propose a novel proactive "sporadic" maintenance strategy, according to which a resource is regenerated each time a Client Node (CN) downloads it. This sporadic maintenance strategy can be implemented either in centralized (with erasure coding) or fully decentralized (with randomized network coding) ways. Although our DHT-based architecture has been preliminary proposed in our previous works [23,22], the key goal of this manuscript is to provide a comprehensive analysis, both with mathematical analysis and simulations, of this distributed storage architecture. To this end, the analytical performance evaluation framework is a properly extended version of the preliminary attempts in [23,22]. Previous works in this field focus on the resource availability, but we think that the usage of shared resources is important as well. Thus, we propose a simple but effective analysis of the required maintenance bandwidth, to highlight the conditions under which sporadic maintenance (SM) has to be preferred to periodic maintenance (PM). Analytical results are further confirmed by realistic simulation results. Our results show that the novel decentralized sporadic maintenance strategy, besides reducing maintenance complexity (at the SuNs), incurs a negligible performance loss, in terms of resource availability and used storage space, with respect to a centralized maintenance approach.

This paper is structured as follows. Section 2 summarizes related work. In Section 3, we present the network coding-based architecture used in our distributed storage system, which is applied to a Wuala overlay network architecture. Analytical and simulation results are shown in Section 4 and Section 5, respectively. Finally, in Section 6 concluding remarks are given.

2. Related work

2.1. P2P distributed storage systems

Distributed data storage has been widely studied by the file system and database community. A conceptually simple example is the Google File System, built to host the state of Google's internal applications [16]. This system uses a single master server for hosting the entire data. Moreover, the data is split into chunks and replicas of these chunks are stored in chunk servers. Conversely, the highly available key–value storage system Dynamo is based on a fully decentralized approach [10]—Dynamo is used by some of Amazon's core services to provide an "always-on" experience. It can be characterized as a zero-hop overlay scheme where each node locally maintains a sufficient routing information to route directly a request to the appropriate node. Malugo is a recently developed P2P storage system [8], where peers are clustered in groups. New inserted files are replicated to different groups and different files have different numbers of replicas according to the pre-specified replication policy. Additional copies are cached in different peers to balance the load of storage peers, which host popular files.

Wuala [29], which is our main reference, is a P2P distributed storage system, with more than 110 million stored files, where a node can simultaneously be supplier and client [29]. The main reason for the success of Wuala is the high resource availability (99.9% guaranteed file availability). Each new user is immediately allowed to use 1 GB of free storage space. Moreover, users can buy more storage space and, possibly, share their own local disk spaces.

The distributed architecture of Wuala is based on Chord [27], probably the most known P2P overlay scheme adopting the Distributed Structured Model (DSM) [3], i.e., implementing a Distributed Hash Table (DHT) to store data or, in general, information about resources, with uniform distribution of responsibility among peers. It can be proved that, with high probability, the number of nodes that must be contacted to find a desired data chunk in a network composed by N nodes is $O(\log N)$ [27]. With respect to the traditional Chord structure, Wuala defines three node classes: SuNs, StNs, and CNs. Data redundancy is achieved by means of MDS erasure coding and SuNs periodically generate maintenance events, in order to guarantee, for each stored file, the presence of a number of fragments sufficiently large to allow its reconstruction. Further details will be given in Section 3.

2.2. Network coding for distributed storage

Network coding is a recently proposed network-oriented channel coding paradigm, arising in the field of information theory, which generalizes the classical concept of routing in wired networks. With network coding, in fact, intermediate nodes, besides forwarding incoming packets, can also encode them. This allows to achieve the multicast capacity [2] and, therefore, leads to potential advantages in terms of bandwidth and computational efficiency, robustness, etc. Although network coding has been extensively studied from a theoretical point of view, several practical scenarios, where benefits can be observed, have been recently proposed [15,14].

Nowadays, distributed storage is one of the main applications where benefits of network coding have emerged [12,20]. In [11], the authors propose a solution to the repair problem using network coding and evaluate the fundamental tradeoff between the storage space and the amount of data required to repair a resource. After the publication of this pioneering work, a significant research activity has been devoted to this topic and, in particular, to the design of optimal codes which allow to achieve this tradeoff. An interesting survey on the main problems in distributed storage applications with network coding can be found in [13], whereas a complete bibliography on this topic is maintained at the Distributed Storage Wiki [28].

3. System architecture

As anticipated at the end of Section 2.1, there are three different groups of nodes with specific behaviors. CNs work as data publishers and consumers: they can publish or retrieve resources through the system, as will be discussed later. When joining the network, a CN has to find a SuN gateway to be used for its communication activities.

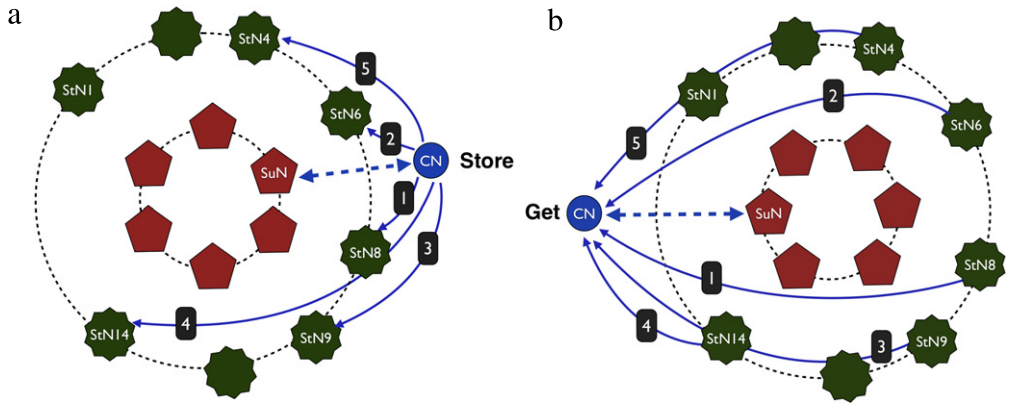


Fig. 1. The considered P2P overlay scheme with illustrative examples of (a) resource store and (b) resource retrieval process. In both cases, the resource consists of 5 fragments.

StNs guarantee high availability and great reliability. They store fragments published by CNs. However, they can (temporarily) disconnect from the network, thus leading to a possible resource unavailability. CNs can evolve and become StNs if their presence has been sufficiently continuous in the network—for instance, in Wuala they need to be online for at least 4 h a day and 10 consecutive days. However, if a promoted StN is unavailable for more than a given time (3 days in Wuala), it is marked as unreliable and downgraded to the CN level.

SuNs form the architecture kernel, implementing an overlay network based on the Chord scheme. The main purpose of this class of peers is to provide a mechanism for message routing that allows clients to find StNs. SuNs play a crucial role in the system and they must be online all the time. Therefore, it is expected that these peers are managed by the organization that provides the distributed storage service or by trusted parties. In order to guarantee storing operations also in unstable scenarios with a large number of disconnections, each SuN is also associated to a StN on the same machine. This is crucial when the system is in its transient phase, after being bootstrapped, as there may be only a few CNs which could not be immediately promoted to the StN level.

In Fig. 1, a logical scheme of the system structure is shown, with (i) an "inner ring" of Super-Nodes, (ii) an "outer ring" of StNs, and (iii) a CN. The figure provides illustrative representations of (a) resource publishing ("Store") and (b) resource retrieval ("Get") actions from a CN. In the remainder of this section, we detail the operations performed by nodes involved in the network.

One should observe that the proposed network architecture is composed by (and can be analyzed at) two different levels. The first level is the system architecture, which is composed, in our case, by the DHT-based overlay network where different classes of nodes are employed. The other level at which the system can be analyzed is associated to the operations that nodes can perform on resources, e.g., files. In this case, three different operations, performed on the overlay network, arise: publish, retrieval, and maintenance. The description of these operations is subject of the following sections.

3.1. Resource publishing

A file of size $\mathscr{M}$ (dimension: (Bytes)), which needs to be stored, is divided into N_g (adimensional) generations composed of k (adimensional) fragments $\{s_i\}_{i=1}^{k}$ each, so that $\mathscr{M} = N_g k d_F$, d_F being the common size (dimension: (Bytes)) of each fragment. We now focus on a single generation, since all the operations do not depend on the generation. All fragments can be equivalently described as symbols belonging to the Galois field GF(q), where $q = 2^m$ is the field size—note that, according to our definition, $m = 8d_F$ bits. The fragments of a generation can be collected in a column vector denoted as $\boldsymbol{s} = (s_1, \ldots, s_k)^T$, where T denotes the transpose operator. Each fragment (q-ary symbol) corresponds to the payload of a packet and the coefficients representing the linear combination of the original fragments $\{s_1, \ldots, s_k\}$ are stored in the so-called coding vector, which is contained in the packet header together with the information about the generation the fragment belongs to. Given that the generation has k fragments in GF(2^m), the coding vector has size $\ell_{cv} = mk$ bits. Typical values of ℓ_{cv} (as will be considered in the simulations in Section 5) are on the order of hundreds of bytes. Since the typical payload size is of the order of hundreds of kB, the overhead can be kept sufficiently small.

The n redundant (published) fragments can be collected in a column vector denoted as $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n)^T$, so that the following linear combination holds:

$$\boldsymbol{\delta} = \boldsymbol{B}\boldsymbol{s}$$

where $\boldsymbol{B}$ is a matrix whose rows $\{\boldsymbol{B}_j\}_{j=1}^{n}$ correspond to the coding vectors of the generated fragments. In the case with MDS erasure coding (namely, Wuala), the combined fragments $\{\delta_j\}_{j=1}^{n}$ are guaranteed to be linearly independent, whereas in the randomized network coding-based approach, each coefficient β_{ji} ($j = 1, \ldots, n$; $i = 1, \ldots, k$) is uniformly chosen among all possible values in GF(q). Note that the use of erasure coding incurs a larger computational complexity, since possible new fragments need to be linearly independent of those already published. This operation typically requires a computational complexity on the order of n^3 [5].

Resources and nodes are identified by binary keys in the same keyspace range. To publish a file, a unique key is assigned to the resource, allowing to route it to the responsible SuN (the one being identified by the nearest key). Moreover, a fixed number of linearly encoded fragments is generated. The SuN responsible for the resource, denoted as gateway SuN, assigns each fragment to a specific SuN, which selects the StN with the current largest portion of available free disk space. The SuN responsible for the resource maintains a list with fragments and corresponding StNs, and communicates that list to the CN. At the end of the lookup process, the CN can upload its generated fragments in parallel to its assigned StNs.

3.2. Resource retrieval

According to the previously illustrated lookup functionality, in order to retrieve a published file it is necessary to obtain, for each generation, k linearly independent fragments $\{\rho_1, \ldots, \rho_k\}$, which can be collected in the column vector $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_k)^T$. Note that $\boldsymbol{\rho}$ is one of the possible subsets of size k drawn from $\{\boldsymbol{\delta}_j\}_{j=1}^{n}$ ($n \geq k$).

After these fragments have been collected, the following system of linear equations can be solved using the classical Gauss elimination algorithm [5]:

$$\boldsymbol{\rho} = \boldsymbol{A}\boldsymbol{s} \tag{1}$$

where $\boldsymbol{A}$ is a size k submatrix of $\boldsymbol{B}$ whose rows $\{\boldsymbol{A}_i\}_{i=1}^{k}$ correspond to the coding vectors of the retrieved fragments $\{\rho_i\}_{i=1}^{k}$. In order to solve (1), $\boldsymbol{A}$ needs to be inverted. In the case with MDS erasure coding, it is guaranteed that $\boldsymbol{A}$ is full rank for every subset $\{\rho_i\}_{i=1}^{k}$. In the presence of randomized network coding, instead, there is a non-zero probability that $\boldsymbol{A}$'s rank is not full. Our simulations, not shown here for lack of space, show that in the considered settings this probability is below 10^{-4} and, therefore, the performance of randomized network coding is very close to that of MDS erasure coding [23].

Referring to the "GET" function in Fig. 1, the SuN gateway routes the CN's request to the responsible node and then to the StNs which have the fragments. Once the fragments have been retrieved, i.e., the file has been downloaded, the CN is able to trigger a regeneration process, as will be discussed in more detail in Section 3.3, thus publishing new (linearly independent) fragments.

3.3. Resource maintenance strategy

In the original Wuala system, resource maintenance is *reactively* carried out by SuNs that periodically (every T_M (min)) check the availability, in the StNs, of the resources under their responsibility. If, for each generation, the percentage of surviving fragments falls below a properly defined retrieval "guard threshold" (which represents the fraction of fragments below which the resource is likely to become soon unavailable), the SuN generates new fragments independent of the surviving ones, and distributes them to StNs. The threshold is denoted as τ and is equal to a fraction of the total number of fragments, i.e., $\tau \triangleq \epsilon n, \epsilon \in [R_c, 1]$. The number of new generated fragments is chosen so that the overall number of available fragments for a generation is equal to n, i.e., the published number of fragments.

Network coding can be efficiently employed to perform the following novel *proactive* resource maintenance strategy. Whenever a CN retrieves (on average, every T_s (min)) a resource, it generates a given number of new fragments for each generation. In particular, the CN decides how to generate fragments after checking, through its gateway SuN, if/how the network has evolved. More precisely: if one or more StNs have been downgraded to CNs within a given period (the last 24 h in our simulations), a whole new set of n linearly independent fragments are regenerated; otherwise, each fragment is generated with probability equal to 0.5. This leads to upload, on average, $n/2$ fragments. Note that this strategy is effective if CNs have a sufficiently large upload bandwidth. This is not a limitation, as final users can nowadays have access to sufficiently powerful Internet connections and, therefore, it may be possible to move the maintenance complexity from the core network towards them. It is obviously still true that the larger the file to be maintained, the higher the maintenance complexity, as will be shown in Section 4.3.

We remark that, in the proposed strategy, maintenance is performed only when a client has successfully completed a download and, therefore, a resource may not be regularly maintained if it is not shared between the owner and a sufficient number of other CNs. However, the complexity of the maintenance operations is significantly reduced, as will be shown later, with respect to that of the Wuala approach. Moreover, as the number of exchanged control messages is significantly smaller, bandwidth consumption is lower.

In the following, we will refer to the reactive maintenance strategy as "periodic maintenance" (PM), whereas the (novel) proactive one will be denoted as "sporadic maintenance" (SM). Note that, in both cases, according to the taxonomy in [11], the maintenance strategy aims at performing *functional* (not *exact*) repair. In both cases, in fact, the new generated fragments are not exactly the same as those lost by disconnected nodes, but they still allow the reconstruction of the same generations.

4. Analytical performance evaluation framework

We now present some results to characterize the performance of the distributed storage architecture. As already anticipated in Section 3, the proposed network architecture is composed by two different levels: (i) the DHT-based overlay network where different classes of nodes are employed, and (ii) the level associated to the operations that nodes can perform on resources. To this end, we characterize the performance in terms of resource availability, storage/repair tradeoff, and required maintenance bandwidth. Note that all the performance indicators are related to both the architecture levels at the same time. The preliminary analysis in [23,22] is properly extended. In particular, previous works in this field take care about the resource availability, but the usage of shared resources, e.g., bandwidth, is important as well. To this end, we propose a simple, but effective, analysis of the required maintenance bandwidth.

In particular, in this section we present an analytical framework that, although based on simple assumptions, is meant to give significant insights on the considered distributed storage systems' behavior. The analytical framework will be compared with simulation results in Section 5.

4.1. Resource availability

Resource availability is defined as the probability that, at any given time, it is possible to successfully download a given resource. For simplicity, we assume, as in [26], that each coded fragment is stored in a different StN. Although this assumption is not always verified in our DHT-based overlay architecture, in Section 5 we will show that simulations confirm, at least trend-wise, the results predicted by the analytical framework. Similar considerations for MDS codes and files divided into a single generation have also been carried out in [11].

Since all generations are independent, the availability of the entire resource can be written, regardless of the chosen coding strategy, as

$$A_X = \left(A_{X,g}\right)^{N_g}$$

where X = E, R, or NC in the presence of erasure, repetition, or randomized network coding, respectively. Note that, as in Wuala, a hybrid strategy can be considered, by also storing a full copy of the resource in a remote server, to make the system more fault-tolerant. This situation will not be considered, but the resource availability can be derived through considerations similar to those provided in [11]. In the following, we specify the value of $A_{X,g}$ for the above mentioned coding strategy.

If MDS codes are adopted, one can successfully recover a single generation of the file if k out of the n fragments are downloaded.[1] Assuming that (i) each StN is available with probability[2] p and (ii) disconnection events are independent, the availability of

[1] Note that, at a given time, the number of fragments for a given resource may be larger than n due to maintenance. This event is neglected, since we assume that the available number of fragments per resource is sufficiently close to n.

[2] The value of p will be characterized by means of simulations in Section 5 for realistic distributed storage networking scenarios.

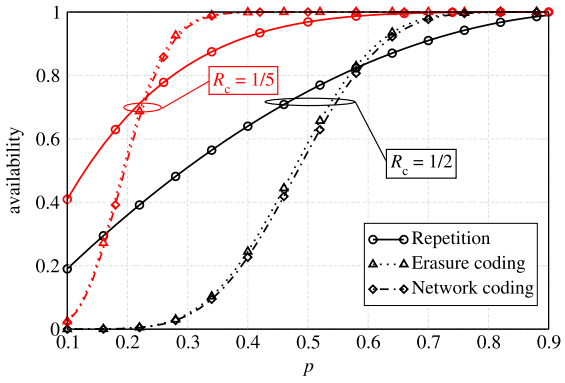


Fig. 2. Resource availability, as a function of the StN availability, in a scenario with $k = 10$ fragments and $N_g = 1$. The code rate is set either to 1/2 ($n = 20$) or 1/5 ($n = 50$). Different coding strategies are considered.

a single generation can be computed as

$$A_{\mathrm{EC,g}} = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{2}$$

It is well known that, for a given value of n and sufficiently large p, the availability (2) is larger than that of a repetition code [26], which is given by

$$A_{\mathrm{R,g}} = 1 - (1-p)^{\frac{1}{R_c}}.$$

In a randomized network coding-based scheme, a node can successfully recover a resource if, for each generation, at least k fragments can be downloaded *and* at least k of these are linearly independent. Let us denote as $\mathscr{E}_i$ the event "i ($i \geq k$) fragments are recovered" and as $\mathscr{B}$ the event "at least k of the recovered fragments are linearly independent". Therefore, one can write

$$A_{\mathrm{NC,g}} = P\left(\bigcup_{i=k}^{n} \{\mathscr{E}_i \mathscr{B}\}\right) = \sum_{i=k}^{n} P\left(\mathscr{B}|\mathscr{E}_i\right) P\left(\mathscr{E}_i\right)$$

where mutual exclusivity of the events $\{\mathscr{E}_i\}$ has been used.

Note that $P(\mathscr{E}_i)$ equals the ith term in (2) and $P(\mathscr{B}|\mathscr{E}_i) \leq 1$ ($i = k, \ldots, n$). Therefore, the availability with randomized network coding is not higher than that with erasure coding. Moreover, from well-known results in the field of network coding [15], it is expected that the larger is the field size, the closer are the availabilities for the two systems. In particular, if one defines as D_i the dimension of the subspace spanned by i vectors, it is possible to write

$$P\left(\mathscr{B}|\mathscr{E}_i\right) = P\left(D_i \geq k\right) = \sum_{j=k}^{i} P\left(D_i = j\right)$$

where the term $P(D_i = j)$ can be written as [1]

$$P\left(D_i = j\right) = \frac{\prod_{\ell=0}^{j-1} \left(q^i - q^\ell\right)\left(q^k - q^\ell\right)}{q^{ki} \prod_{\ell=0}^{j-1} \left(q^j - q^\ell\right)}$$

and q is the field size.

In Fig. 2, the resource availability is shown, as a function of the StN availability, in a scenario with $k = 10$ fragments, $N_g = 1$, and $q = 2^8$. The code rate is set either to 1/2 ($n = 20$) or 1/5 ($n = 50$). Different coding strategies are considered: (i) repetition, (ii) erasure, and (iii) randomized network coding. As can be observed from the results in Fig. 2, the use of randomized network coding leads to a performance loss with respect to the use of erasure coding. However, for sufficiently large values of p, this loss is negligible and the use of a sufficiently large value of the field size q allows to approach the performance with erasure coding. Finally, the smaller the code rate, the lower the performance loss incurred by network coding-based schemes.

4.2. Storage/repair tradeoff

The storage/repair tradeoff has been first introduced in [11], focusing on a single resource composed by a single generation (i.e., $N_g = 1$). Suppose that the n coded fragments of a file of size $\mathscr{M} = N_g k d_F$ are stored at n active StNs, each of which contains a single fragment formed by α bits (denoted as "storage size"). Whenever a StN fails, a new StN is placed in the network. In order to generate a new coded fragment, the newcomer needs to receive $\beta \leq \alpha$ bits from $d \leq n-1$ surviving StNs. Therefore, the number of bits that are necessary to generate a new coded fragment is $\gamma = d\beta$ (denoted as "repair size").[3] In [11], the authors show that, given n, k, d, and γ, there exists an optimal tradeoff storage size (with closed-form expression) $\alpha^*(n, k, d, \gamma)$ such that a storage size $\alpha \geq \alpha^*$ is feasible, whereas a storage size $\alpha < \alpha^*$ is not. Moreover, the so-called *regenerating codes* [28], based on the use of randomized network coding, allow to achieve every point of the optimal tradeoff curve.

In order to estimate the storage/repair tradeoff in our schemes, assume that each StN has, on average, $N_g \overline{n}_F$ fragments for each resource and each fragment has an average size $\overline{d}_F$ (the size of each fragment depends on the total size of the resource). Thus, the average storage size is $\overline{\alpha} = N_g \overline{n}_F \overline{d}_F$. Since new fragments are generated only when the node responsible for the maintenance (a CN with SM or a SuN with PM) can recover the entire resource, the repair size can be computed as $\overline{\gamma} = N_g \overline{k}_F \overline{d}_F$, where $\overline{k}_F$ is the average number of fragments that need to be retrieved in order to find k linearly independent fragments per generation. Note that, in general, $\overline{k}_F \geq k$, i.e., it would be necessary to download more fragments before obtaining a set of k linearly independent fragments. In Section 5, we will evaluate $\overline{\alpha}$ and $\overline{\gamma}$ by simulations and compare them with the optimal storage curve (α^* as a function of $\overline{\gamma}$) in [11].

4.3. Maintenance bandwidth

In this subsection, we investigate the bandwidth required by maintenance operations. In particular, we define this bandwidth as follows:

$$B \triangleq \frac{\#\text{of exchanged bits}}{\text{maintenance period}}.$$

We now compute the number of exchanged bits for both PM and SM strategies. In a scenario with PM, for each resource the SuN interrogates the StNs with a packet[4] containing the resource ID. Since a few bits are transmitted, they can be neglected. At this point, $n_{\mathrm{R},i}$ ($n_{\mathrm{R},i} \leq n$) packets of size d_F are downloaded by the SuNs for the ith generation ($i = 1, \ldots, N_g$). If $k \leq n_{\mathrm{R},i} \leq \tau$, where τ is the maintenance threshold, packets' regeneration is triggered and $n - n_{\mathrm{R},i}$ fragments are injected in the network. This happens with a probability denoted as $P_{\mathrm{R},i}$. It is reasonable to assume that $P_{\mathrm{R},i}$ is constant, i.e., $P_{\mathrm{R},i} = P_{\mathrm{R}}$ ($i = 1, \ldots, N_g$). The parameter P_{R} is related to network conditions and depends on traffic statistics

[3] Note that, in [11], the authors use the term "repair bandwidth" for γ (dimension: [bits]). Since in this paper the term bandwidth will be used, in Section 4.3, to refer to the maintenance bandwidth B (dimension: (bits/s)), we refer to γ as "repair size".

[4] As already said in Section 3.1, a fragment corresponds to a packet payload.

(e.g., network dynamics, stability, etc.). In particular, regeneration happens if some network conditions are matched. In the proposed framework, the parameter P_R is not a design parameter and is used in the analytical framework to determine the corresponding required maintenance bandwidth. In other words, given specific network operational conditions, the realistic value of P_R can be computed by simulations. An interesting extension of our work consists in the derivation of an analytical framework which, by taking into account a proper model of network evolution, allows to compute the regeneration probability. However, this goes beyond the scope of this paper. In the presence of erasure coding, more than $n_{R,i}$ packets may be generated in order to obtain $n_{R,i}$ linearly independent packets, whereas this is not the case with randomized network coding. Therefore, the maintenance bandwidth can be written as

$$B_{PM} = \frac{N_{res}}{T_M}\left[\sum_{i=1}^{N_g} n_{R,i} d_F + \sum_{i=1}^{N_g}(n - n_{R,i}) d_F P_R\right] \tag{3}$$

where N_{res} is the number of resources in the network. Since $n_{R,i}$ can be expressed as $k + \phi_i$ ($\phi_i \in \{0, \ldots, n-k\}$) and using the fact that $k/n = R_c$, after a few manipulations, not shown here for lack of space, (3) becomes

$$B_{PM} = \frac{N_{res} d_F}{T_M}\sum_{i=1}^{N_g}\left\{k\left[1 + \left(\frac{1}{R_c} - 1\right)P_R\right] + \phi_i(1 - P_R)\right\}.$$

At this point, recall (from Section 4.2) that kd_F is equal (for erasure coding) to γ/N_g and approximately equal (for randomized network coding) to γ/N_g. Therefore, in the randomized network coding case it holds that

$$B_{PM} \simeq \frac{\gamma\left[(1 - P_R)\left(1 + \frac{\overline{\phi}}{k}\right) + \frac{P_R}{R_c}\right]}{T_M} N_{res} \tag{4}$$

where $\overline{\phi} \triangleq \sum_{i=1}^{N_g} \phi_i / N_g$. Eq. (4) is exact for erasure coding. Note that the parameters P_R, $\overline{\phi}$, and N_{res} depend on the particular network configuration, e.g., number of nodes, connection/disconnection period, etc. Moreover, since $0 \le \overline{\phi} \le n - k$, the following upper and lower bounds for B_{PM} are obtained:

$$\frac{\gamma\left[1 - P_R\frac{R_c - 1}{R_c}\right]}{T_M} N_{res} \le B_{PM} \le \frac{\gamma\left[1 - P_R\frac{R_c^2 - 1}{R_c}\right]}{T_M} N_{res}. \tag{5}$$

In a scenario with SM, a CN downloads γ bit in order to reconstruct the resource. If the resource is available (this happens with probability A, as stated in Section 4.1), a given number of fragments is regenerated for each generation. According to the considerations carried out in Section 3.3, if the network is dynamic n fragments are generated; otherwise (i.e., the network is static), the average number is $n/2$. Since it may be reasonable to assume that the two network statuses (e.g., static or dynamic) are equally likely, the average number of regenerated packets is

$$\frac{1}{2}\left[n + \frac{n}{2}\right] N_g = \frac{3}{4} n N_g.$$

Therefore, the maintenance bandwidth with SM can be written as

$$B_{SM} = \frac{\gamma + \frac{3}{4} A N_g n d_F}{T_s} \simeq \frac{\gamma\left[1 + \frac{3}{4R_c} A\right]}{T_s} \tag{6}$$

where, as in the case with PM analyzed above, we have used the fact that $\gamma \simeq N_g k d_F$ and $k/n = R_c$. Comparing (6) with (5) allows to clearly identify the set of parameters' values for which one should prefer one strategy or the other. Comparing expression (6) with expression (4), it can be concluded that, unlike B_{PM}, B_{SM} does not depend on $\overline{\phi}$. Note also that, unlike the corresponding PM scenario, (6) does not clearly depend on A. However, this dependence is hidden by the fact that the SM strategy performs fragment regeneration only if the considered resource can be downloaded and, therefore, it is available in the network. The bandwidth for the PM strategy is instead computed assuming that N_{res} resources are available in the network at the time of the maintenance. Obviously, if the total number of injected resources is N_{tot}, it follows that $N_{res} \le N_{tot}$ depending on the particular resource availability.

Table 1
Main framework parameters.

Parameter	Description
n_F	Number of fragments for a given generation
d_F	Fragment size
k_F	Number of per generation fragments needed to find k linearly independent fragments
p	Availability probability of a StN
P_r	Probability of packets' regeneration

In Fig. 3, the maintenance bandwidth is shown, as a function of $\overline{\phi}$, in a scenario with $k = 50$, $R_c = 1/2$, $\gamma = 10$, with either PM (with $T_M = 30$ min) or SM (with $T_s = 3$ min)—the selected values of T_M and T_s are chosen as they correspond to the realistic parameters used in the simulations in Section 5. Different values of P_R and N_{res} are considered. As observed at the end of the previous paragraph, B_{SM} does not depend on $\overline{\phi}$ and, hence, is constant. For a small number of resources (e.g., 2), the maintenance bandwidth required by PM is always smaller than that of SM, since in the latter case a larger number of fragments is uploaded after regeneration. When the number of resources increases, instead, the performance of SM becomes better than that of PM. Therefore, when there is a large number of resources in the network, SM is to be preferred to PM. Our results, not shown for lack of space, suggest that SM is preferable if $T_M = T_s$. In other words, when several resources are stored in the network one should perform more maintenance operations to guarantee that the resources cannot be lost. In particular, if resource searches have sufficiently small frequency so that the availability is high but the scheduled maintenance events are lower than those with PM, SM guarantees to reduce the bandwidth occupation in the network. In Section 5, we will validate these considerations by estimating the values of the parameters P_R, $\overline{\phi}$, and N_{res} through realistic simulation results relative to practical P2P applications.

4.4. Discussion

The proposed analytical framework depends on several system parameters, which are summarized for brevity in Table 1. In particular, some of them are related to the resources exchanged within the network and can be obtained according to the chosen coding strategy and the type of traffic (i.e., what is stored). Other parameters (e.g., p or P_r) are more related to network conditions and depends on traffic statistics (e.g., network dynamics, stability, etc.). However, all of them are strictly related to each other, to derive the performance. As an example, resource availability depends on data characteristics, e.g., the number of fragments and the coding rate, but also on the network situation, e.g., the availability of storage nodes p.

Moreover, the analytical models above presented are characterized by a few assumptions. The main inaccuracies of the model are the following.

- More than one coded fragment may be stored in a given StN.
- At a given time, the number of fragments for a given resource may be larger than n, due to maintenance.
- StN disconnections may not be independent.
- All generations may not behave in the same way—for instance, packet regenerations may happen with different probabilities.

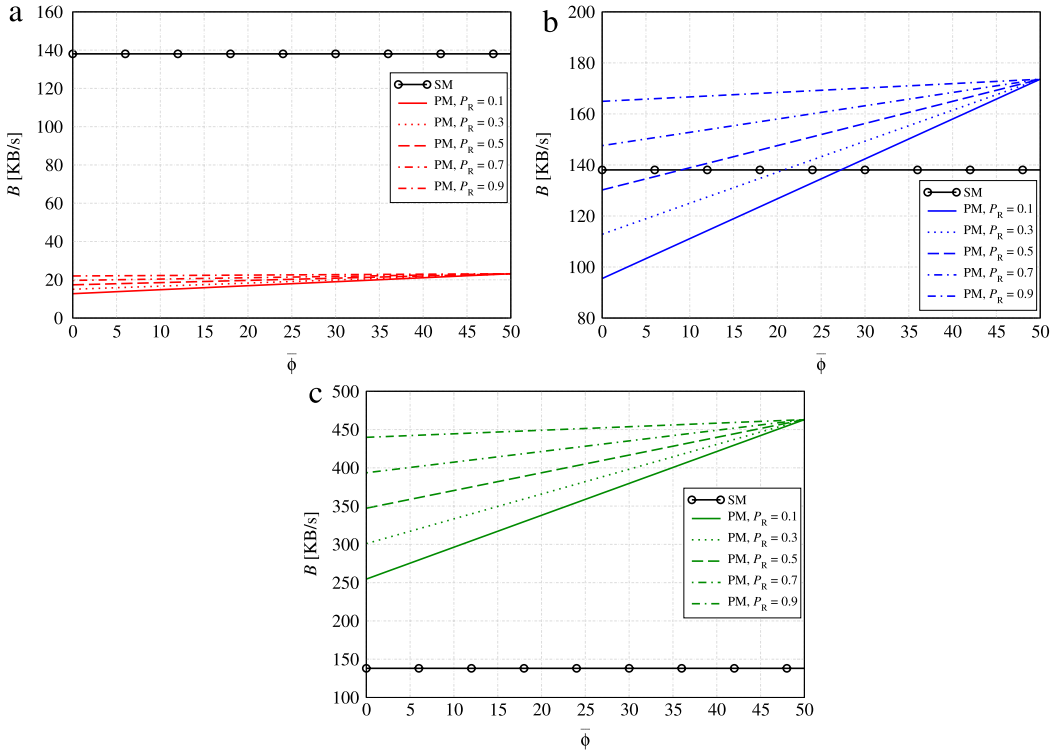


Fig. 3. Maintenance bandwidth (dimension: (kB/s)), as a function of $\overline{\phi}$ (adimensional), in a scenario with $k = 50$, $R_c = 1/2$, $\gamma = 10$, with either PM (with $T_M = 30$ min) or SM (with $T_s = 3$ min). In case (a) $N_{res} = 2$, in case (b) $N_{res=15}$, and in case (c) $N_{res} = 40$. Different values of P_R are considered in the PM scenarios.

However, these assumptions cannot be considered critical, as the simulation results (especially those regarding resource availability and maintenance bandwidth) will be shown to be in good agreement with analytical results.

5. Simulation-based performance analysis

The simulation tool used in this paper is DEUS, a discrete event universal simulator, based on Java and XML, created by the Distributed Systems Group (DSG) of the University of Parma [4].[5] In [24], simulation results, based on DEUS, and experimental results, based on PlanetLab, are compared in a traffic information system application, showing the effectiveness of this simulator. In order to evaluate the performance of the proposed P2P distributed storage systems, the scenario summarized in Table 2 has been simulated. This scenario is representative of a realistic dynamic network, where many users connect, share, and look for resources. The rationale behind the chosen simulation settings is the following. The simulation length is such that 30 days of network functioning are considered. During this time, a sufficient number of events is scheduled: in particular, if 100 events per hour are scheduled a total amount of 72,000 events is generated. The churn verification period equal to 1 h is set as a reasonable tradeoff between the storage node availability and a realistic node functioning. In real networks, in fact, some nodes may be disconnected for longer times and some others for shorter time. The storage node capacity instead is compliant with simulations in [18]. The file size distribution has been chosen as a reasonable value to obtain simulation results with not too lengthy simulations; however, in [21] a more realistic distribution of the space occupied by files, according to Wuala data, is derived. Finally, the search time is given so that each client performs, on average, 1 lookup per day. At the end of this section, we will investigate the impact of a different (more realistic) file size distribution and different values of T_s.

Table 2
Main simulation parameters.

Parameter	Value
Number of SuNs	10
Number of StNs	50 (10 disconnected at $t = 0$)
Number of CNs	500 (80 disconnected at $t = 0$)
Disk space for each StN	10 GB
Resource Size $\mathscr{M}$	Uniform in [1, 35] MB
Simulation duration	30 days
Number of events per hour	100
Number of independent runs	100
Resource search period T_s	3 min
Connections/disconnections period	12 min
Churn verification period	1 h
Number of regenerated fragments	$k + 0.15k$ (without churn)
with SM	$k + 0.25k$ (with churn)
τ (regeneration with PM)	$0.7n$

We first investigate the storage/repair tradeoff discussed in Section 4.2. In Fig. 4, the storage/repair tradeoff is shown for resources of average size $\overline{\mathscr{M}} = 18$ MB. Each resource corresponds to $N_g = 1$ generation composed by $k = 50$ fragments, coded with rate R_c equal to 1/2 or 1/5. The operating points (obtained by simulations) with erasure coding and randomized network coding, considering either PM or SM, are shown in the figure as symbols (●, ■, ★, ▲). One can observe that, for a given value of the coding rate, the different coding/maintenance strategies entail minor differences. In particular, the repair size $\overline{\gamma}$ is approximately equal to $\overline{\mathscr{M}}$. Moreover, for decreasing values of R_c, a larger amount of storage $\overline{\alpha}$ is required, since the average resource availability is larger (as will be shown in more detail in Table 3). However, the SM strategy is to be preferred, since it requires a lower computational complexity and network overhead. We remark that Fig. 4 is

[5] This simulator has been used instead of classical network simulators (e.g., ns-2 or Opnet), since it is optimized to analyze P2P networks at a higher level (up to the overlay network).

Table 3
Average resource availability (denoted as μ) and its standard deviation (denoted as σ) for $k = 50$ fragments. Different coding and maintenance strategies are considered.

	$R_c = 1/2$			$R_c = 1/5$		
	μ	σ	An.	μ	σ	An.
EC-PM ($T_M = 30$ min)	0.9891	0.0082	0.9979	0.9999	0.00001	1
EC-PM ($T_M = 45$ min)	0.9863	0.0099	0.9979	0.9999	0.00009	1
EC-SM	0.9883	0.00825	0.9979	0.9999	0.00005	1
NC-PM ($T_M = 30$ min)	0.9918	0.0053	0.9967	0.9999	0.00002	1
NC-PM ($T_M = 45$ min)	0.9862	0.00985	0.9967	0.9999	0.00013	1
NC-SM ($q = 2^3$)	0.9899	0.0048	–	0.9999	0.00008	–
NC-SM ($q = 2^8$)	0.9901	0.00485	0.9967	0.9999	0.00008	1
NC-SM ($q = 2^{16}$)	0.9899	0.00476	–	0.9999	0.00008	–

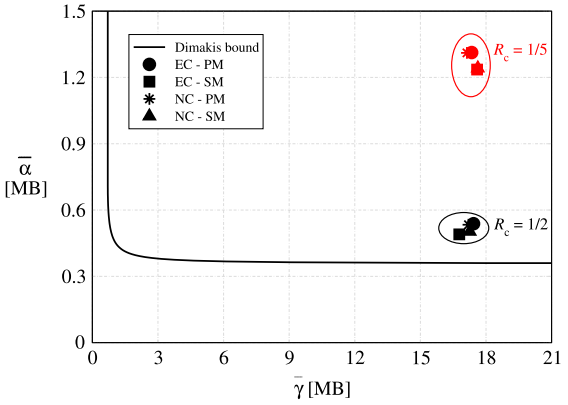


Fig. 4. Storage/repair tradeoff for resources of average size $\overline{\mathscr{M}} = 18$ MB divided into 50 fragments coded with rate $R_c = 1/2$ or $R_c = 1/5$. The operating points of different strategies are shown.

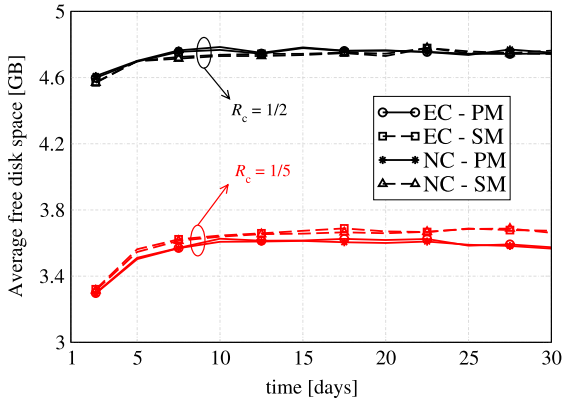


Fig. 5. Average free available disk space per StN, as a function of time, in a scenario with $k = 50$, $T_M = 30$ min, $T_s = 3$ min, and $q = 2^{16}$. Different coding and maintenance strategies are considered.

meant to compare the (theoretical) bound with single points are associated with simulations. In particular, the bound depends on two parameters $\overline{\alpha}$ which can vary $\overline{\gamma}$, but during simulations these two parameters assume a given value. Therefore, one single point is obtained for a given simulation scenario.

We now analyze the resource availability, defined as the success rate of downloading a resource by a CN. In Table 3, the average resource availability (denoted as μ) and its standard deviation (denoted as σ) are shown in a scenario with generations divided into $k = 50$ fragments. Different coding and maintenance strategies are considered. The value of μ is computed by averaging the availability over all simulation times and all possible resources. First, one can observe that the lower the coding rate, the higher the availability. In this case, in fact, the number of published fragments n is larger and a larger number of disconnections is needed to make a resource unavailable. In particular, for a fixed coding rate, all strategies have approximately the same performance (almost 99% when $R_c = 1/2$ and 99.99% when $R_c = 1/5$). Finally, the simulation results are compared with those predicted by the analytical framework in Section 4.1. With the considered simulation settings, the average probability with which a StN is available in the network (over time and across nodes) is approximately equal to $p = 0.77$. The analytical results in Fig. 2 with $p = 0.77$ are very close but slightly larger than those in Table 3. This is due to the fact that in the analytical framework we have considered a "more benign" scenario, where a StN disconnection leads to the loss of only one fragment.

In Fig. 5, the average free available disk space per StN is shown, as a function of time, in a scenario with $k = 50$, $T_M = 30$ min, $T_s = 3$ min, and $q = 2^{16}$. Different coding and maintenance strategies are considered. As previously observed for the resource availability, for a fixed coding rate, the performances of all considered strategies are approximately the same. Unlike the results in Table 3, the higher the coding rate, the smaller the storage occupancy. In

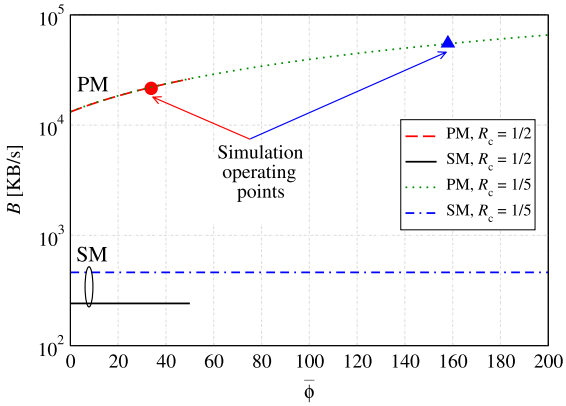


Fig. 6. Maintenance bandwidth, as a function of $\overline{\phi}$, in a scenario with $k = 50$ with either PM with $T_M = 30$ min or SM with $T_s = 3$ min. Two values R_c for the coding rate are considered: 1/2 or 1/5. Other parameters are obtained through simulations.

fact, as one can see, the case with $R_c = 1/2$ allows to save, on average, about 1.2 GB with respect to the case with $R_c = 1/5$. From our simulations, we have observed that this is due to the fact that scenarios with higher coding rates generate a smaller number of fragments. Therefore, there exists a tradeoff between the resource availability and the free disk space.

Since all performance indicators considered so far show that, for a given coding rate R_c, all strategies are almost equivalent (with erasure/randomized network coding and PM/SM strategies), we now evaluate the maintenance bandwidth. As the simulated system is operating in the presence of a large amount of resources, we expect, according to the results in Section 4.3, that SM will guarantee bandwidth savings. In Fig. 6, the maintenance bandwidth is shown, as a function of $\overline{\phi}$, in a scenario with $k = 50$ with either PM (with $T_M = 30$ min) or SM (with $T_s = 3$ min). Two values

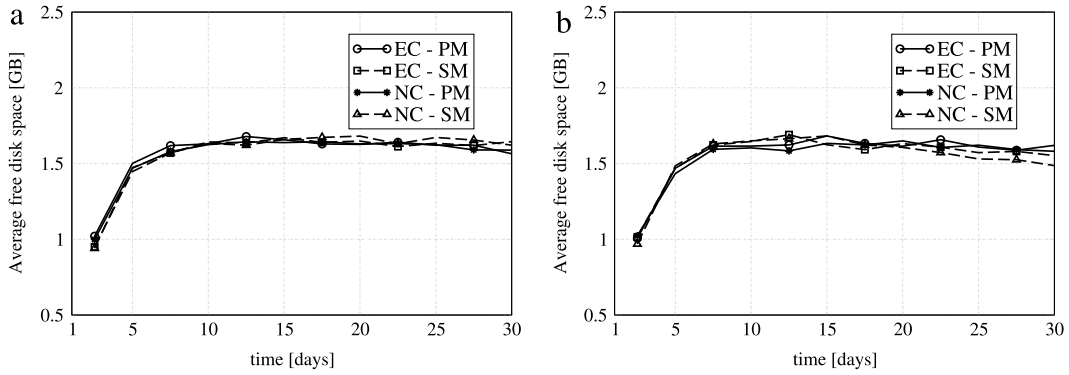


Fig. 7. Average free available disk space per StN, as a function of time, in a scenario with $k = 50$, $R_c = 1/2$, $T_M = 30$ min, $q = 2^{16}$, and the non-uniform file size distribution. The performance is analyzed for both $T_s = 3$ min (case (a)) and $T_s = 36$ s (case (b)). Different coding and maintenance strategies are considered.

for the coding rate R_c are considered: 1/2 or 1/5. Other parameters (i.e., γ, P_R, A, and N_{res}) cannot be evaluated analytically and, therefore, are obtained through simulations. Note that the curves associated with $R_c = 1/2$ stop at $\overline{\phi} = 50$, since this corresponds to the maximum allowed value. The results in Fig. 6 show clearly that, in the simulated scenarios, the maintenance bandwidth with SM is significantly smaller than that with PM. In particular, operating points corresponding to the average simulated values of $\overline{\phi}$ are also shown. Therefore, one can conclude that SM is the best choice for the considered P2P distributed storage system, since the same performance of PM (considered in the Wuala system) can be achieved with a saving of two orders of magnitude in terms of maintenance bandwidth.

Finally, we evaluate the impact of a different (more realistic) file distribution and more frequent file searches. According to [21], a more realistic file size distribution may be the following step-wise function

$$f(\mathscr{M}) = \begin{cases} 0.04 & 0 \le \mathscr{M} \le 1 \\ 0.18 & 1 \le \mathscr{M} \le 16 \\ 0.78 & 16 \le \mathscr{M} \le 256. \end{cases}$$

Note that values are given in MB. In Fig. 7, average free available disk space per StN is shown, as a function of time, in a scenario with $k = 50$, $R_c = 1/2$, $T_M = 30$ min, $q = 2^{16}$, and the non-uniform file size distribution. The performances are analyzed for both $T_s = 3$ min (case (a)) and $T_s = 36$ s (case (b)). Different coding and maintenance strategies are considered. The same trend of Fig. 5 can be observed (similar considerations hold for other performance indicators). Therefore, the previous performance analysis is not limited by the chosen simulation settings and can be considered sufficiently general. The only difference with respect to Fig. 5 is the fact that the average free available disk space is reduced, due to the fact that larger file sizes are generated and, as a consequence, more storage space is occupied. Moreover, one should observe that more frequent searches do not entail any performance loss (the available space is approximately the same). Therefore, the proposed approach is scalable with more dynamic networking scenarios.

6. Concluding remarks

In this paper, starting from an existing P2P distributed storage scheme (namely, the one in the Wuala project) with PM and erasure coding, we have proposed a novel scheme based on the use of a SM strategy and randomized network coding. According to our approach, CNs generate new fragments to be stored in the network "sporadically", namely every time they successfully download a resource. We have proposed an analytical framework for performance characterization and validated this framework by means of simulations. Our analytical and simulation results have shown that, in the presence of a large number of resources to be stored, SM guarantees the same performance, in terms of storage/repair tradeoff, resource availability, and storage occupancy, of PM with savings, in terms of maintenance bandwidth, of two orders of magnitude. Randomized network coding is essential to design a fully distributed SM scheme. The validation of our analytical framework with simulation results makes it an efficient tool for the design of effective distributed storage systems for P2P systems. An experimental validation of the proposed approach is an interesting future research direction.

Acknowledgments

The authors would like to thank Riccardo Bussandri for his help in the derivation of the presented simulation results. The authors would also like to thank Prof. Riccardo Raheli (Università degli Studi di Parma, Italy) for useful discussions.

References

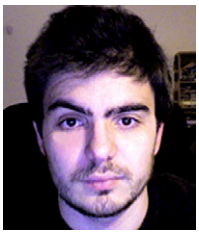
[1] S. Acedanski, S. Deb, M. Medard, R. Koetter, How good is random linear coding based distributed networked storage? in: Proc. IEEE International Symposium on Network Coding (NetCod), Riva del Garda, Italy, 2005.
[2] R. Ahlswede, N. Cai, S.-Y.R. Li, R.W. Yeung, Network information flow, IEEE Trans. Inform. Theory 46 (4) (2000) 1204–1216.
[3] M. Amoretti, A survey of peer-to-peer overlay schemes: effectiveness, efficiency and security, BSP Recent Pat. Comput. Sci. 2 (3) (2009) 195–213.
[4] M. Amoretti, M. Agosti, F. Zanichelli, DEUS: a discrete event universal simulator, in: 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools), Rome, Italy, 2009.
[5] A.K. Atkinson, An Introduction to Numerical Analysis, John Wiley & Sons, New York, NY, USA, 1989.
[6] BitTorrent, URL: http://www.bittorent.com.
[7] A. Blanc, Y.-K. Liu, A. Vahdat, Designing incentives for peer-to-peer routing, in: Proc. IEEE Conf. on Computer Commun. (INFOCOM), Vol. 1, Miami, FL, USA, 2005, pp. 374–385.
[8] P.-C.S.Y.-W. Chan, T.-H. Ho, Y.-C. Chung, Malugo: A peer-to-peer storage system, Int. J. Ad Hoc and Ubiquitous Comput. 5 (10) (2010) 209–218.
[9] C. Courcoubetis, R. Weber, Incentives for large peer-to-peer systems, IEEE J. Select. Areas Commun. 24 (5) (2006) 1034–1050.
[10] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, Dynamo: Amazon's highly available key-value store, ACM SIGOPS Oper. Syst. Rev. Arch. 41 (6) (2007) 205–220.
[11] A.G. Dimakis, P.B. Godfrey, Y. Wu, M.O. Wainwright, K. Ramachandran, Network coding for distributed storage systems, IEEE Trans. Inform. Theory 56 (9) (2010) 4539–4551.
[12] A.G. Dimakis, V. Prabhakaran, K. Ramchandran, Decentralized erasure codes for distributed networked storage, IEEE Trans. Inform. Theory 52 (6) (2006) 2809–2816.
[13] A.G. Dimakis, K. Ramchandran, Y. Wu, C. Suh, A survey on network codes for distributed storage, Proc. IEEE 99 (3) (2011) 476–489.
[14] C. Fragouli, E. Soljanin, Network Coding Applications, Now Publisher Foundations and Trends in Networking, Hanover, MA, USA, 2007.
[15] C. Fragouli, E. Soljanin, Network Coding Fundamentals, Now Publisher Foundations and Trends in Networking, Hanover, MA, USA, 2007.
[16] S. Ghemawat, H. Gobioff, S. Leung, The Google file system, in: Proc. ACM Symp. on Symposium on Operating Systems Principles, SOSP, Landing, NY, USA, 2003, pp. 29–43.

[17] C. Gkantsidis, P. Rodriguez, Network coding for large scale content distribution, in: Proc. IEEE Conf. on Computer Commun., INFOCOM, Miami, FL, USA, 2005, pp. 2235–2245.
[18] D. Grolimund, Wuala—a distributed (P2P) storage system, in: NASA Information Science & Technology Colloquium Series, NASA Goddard Space Flight Center, Greenbelt, MD, USA, 2009, available at http://istcolloq.gsfc.nasa.gov/fall2009/speaker/grolimund.html.
[19] T. Ho, M. Medard, R. Koetter, D.R. Karger, M. Effros, B.J.S. Leong, A random linear network coding approach to multicast, IEEE Trans. Inform. Theory 52 (10) (2006) 4413–4430.
[20] A. Kamra, V. Misra, J. Feldman, D. Rubenstein, Growth codes: maximizing sensor network data persistence, ACM SIGCOMM Comput. Commun. Rev. 36 (2006) 255–266.
[21] T. Mager, K. Biersack, P. Michiardi, A measurement study of the Wuala on-line storage service, in: Int. Conf. Peer-to-Peer Computing (P2P), Tarragona, Spain, 2012, pp. 237–248.
[22] M. Martalò, M. Picone, M. Amoretti, G. Ferrari, R. Raheli, Randomized network coding in distributed storage systems with layered overlay, in: Information Theory and Applications Workshop, ITA, UCSD, San Diego, CA, USA, 2011, pp. 1–7, invited paper.
[23] M. Martalò, M. Picone, R. Bussandri, M. Amoretti, A practical network coding approach for peer-to-peer distributed storage, in: Proc. IEEE International Symposium on Network Coding (NetCod), Toronto, Canada, 2010, pp. 103–108.
[24] M. Picone, M. Amoretti, F. Zanichelli, A decentralized smartphone based traffic information system, in: Proc. IEEE Intelligent Vehicles Symposium, Alcalá de Henares, Spain, 2012, pp. 523–528.
[25] J.S. Plank, Erasure codes for storage applications, in: 4th USENIX Conf. on File and Storage Technologies (FAST), San Francisco, CA, USA, 2005.
[26] R. Rodrigues, B. Liskov, High availability in DHTs: Erasure coding vs. replication, in: Proc. Int. Workshop on Peer-to-Peer Systems (IPTPS), Ithaca, New York, USA, 2005, pp. 226–239.
[27] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for Internet applications, in: Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Communications, San Diego, CA, USA, 2001, pp. 149–160.
[28] StorageWiki, URL: http://csi.usc.edu/~dimakis/StorageWiki/doku.php.
[29] The WUALA Project, URL: http://www.wuala.com.

Marco Martalò was born in Galatina (LE), Italy, on June 1981. He received the "Laurea" degree (3-year program) and the "Laurea Specialistica" (3+2 year program) degree (summa cum laude) in Telecommunications Engineering on September 2003 and December 2005, respectively, from the University of Parma, Italy. On March 2009, he received the Ph.D. degree in Information Technologies at the University of Parma, Italy. From October 2007 to March 2008, he was a "Visiting Scholar" at the School of Computer and Communication Sciences of the Ecole Polytechnique Federale De Lausanne (EPFL), Lausanne, Switzerland, collaborating with the laboratory of Algorithmic Research in Network Information, directed by Prof. Christina Fragouli. From January 2009 to April 2012, he was a Post-Doc Researcher at the Information Engineering Department of the University of Parma, Italy. From May 2012, he has been an Assistant Professor at the E-Campus University, Novedrate (CO), Italy, and a Research Associate at the Department of Information Engineering (DII) of the University of Parma, Italy, working with Prof. Gianluigi Ferrari on design of digital communication systems and, in particular, of wireless sensor networks.

Dr. Martalò is co-author of the book "Sensor Networks with IEEE 802.15.4 Systems: Distributed Processing, MAC, and Connectivity" edited in 2011 by Springer, Germany. Dr. Martalo' was a co-recipient of a "best student paper award" (with his tutor Dr. Gianluigi Ferrari) at the 2006 International Workshop on Wireless Ad hoc Networks (IWWAN'06). He also won the first prize award, together with the WASNLab team, at the first Body Sensor Network (BSN) Contest, organized in conjunction with the 2011 Body Sensor Networks (BSN'11) conference. He has been TPC member of the International Workshop on Performance Methodologies and Tools for Wireless Sensor Networks (WSNPERF 2009), the International Conference on Advances in Satellite and Space Communications (SPACOMM 2009–2010), and the IEEE Global Communications Conference (GLOBECOM 2011), Communication Theory Symposium. He also serves as a reviewer for many international journals and conferences.

Michele Amoretti received the Dr. Ing. (Master) degree in Electronic Engineering in 2002, and the Ph.D. degree in Information Technologies in 2006 from Università degli Studi di Parma (Italy). Currently, he is a Research Associate and Contract Professor at the Centro Interdipartimentale SITEIA.PARMA of the same university. His research focuses on modeling and simulation of large-scale distributed systems, in particular those based on the peer-to-peer paradigm; complex adaptive systems and autonomic computing; design and development of service-oriented architectures. He has published over 70 technical papers in refereed international journals and conferences.



Marco Picone currently is a Post-Doctoral Research Associate at the University of Parma. He received from the same University the Dr. Ing. degree (Master) in Computer Engineering "cum laude" in 2008 and the Ph.D. degree in Information Technologies in 2012. Between January 2011 and June 2011 he was a research visitor in the Network and Operating Systems group at the Computer Laboratory, University of Cambridge, where he worked on mobile based sensing systems and sensor networks interaction. His research activity focuses on distributed and peer-to-peer systems, with particular interest in solutions that involve mobile devices. Application fields include: neighbor position discovery in peer-to-peer networks, vehicle-to-vehicle and vehicle-to-infrastructure communications, a P2P approach for inter-vehicular networks, vehicular networks simulation and mobility models, mobile based sensing systems and vertical handover algorithms & applications.

Gianluigi Ferrari (http://www.tlc.unipr.it/ferrari) was born in Parma, Italy, in 1974. He received his "Laurea" and Ph.D. degrees from the University of Parma, Italy, in 1998 and 2002, respectively. Since 2002, he has been with the University of Parma, where he currently is an Associate Professor of Telecommunications. He was a visiting researcher at USC (Los Angeles, CA, USA, 2000–2001), CMU (Pittsburgh, PA, USA, 2002–2004), KMITL (Bangkok, Thailand, 2007), and ULB (Brussels, Belgium, 2010). Since 2006, he has been the Coordinator of the Wireless Ad-hoc and Sensor Networks (WASN) Lab (http://wasnlab.tlc.unipr.it/) in the Department of Information Engineering of the University of Parma.

As of today he has published more than 180 papers in leading international journals and conferences, and 19 book chapters. He is the co-author of 7 books, including "Detection Algorithms for Wireless Communications, with Applications to Wired and Storage Systems" (Wiley: 2004), "Ad Hoc Wireless Networks: A Communication-Theoretic Perspective" (Wiley: 2006—technical best seller), "LDPC Coded Modulations" (Springer: 2009), and "Sensor Networks with IEEE 802.15.4 Systems: Distributed Processing, MAC, and Connectivity" (Springer: 2011). He edited the book "Sensor Networks: where Theory Meets Practice" (Springer: 2010). His research interests include digital communication systems analysis and design, wireless ad hoc and sensor networking, and adaptive digital signal processing. He participates in several research projects funded by public and private bodies.

Prof. Ferrari is a co-recipient of: a best student paper award at IWWAN'06; a best paper award at EMERGING'10; an award for the outstanding technical contributions at ITST-2011; the best paper award at SENSORNETS 2012; the best paper award at EvoCOMNET 2013. The WASNLab team won the first Body Sensor Network (BSN) contest, held in conjunction with BSN 2011. He acts as a frequent reviewer for many international journals and conferences. He acts also as a technical program member for many international conferences. He currently serves on the editorial boards of several international journals. He was a Guest Editor of the 2010 EURASIP JWCN Special Issue on "Dynamic Spectrum Access: From the Concept to the Implementation."