Randomized Network Coding in Distributed Storage Systems with Layered Overlay

M. Martalò, M. Picone, M. Amoretti, G. Ferrari, and R. Raheli
Department of Information Engineering, University of Parma, Italy
{marco.martalo, michele.amoretti, gianluigi.ferrari, raheli}@unipr.it, picone@ce.unipr.it

***Abstract*—In this paper, we analyze the performance of a peer-to-peer (P2P) distributed storage architecture based on a layered overlay scheme, where some nodes provide their disk capacities for hosting data fragments generated by other active users. This system has been introduced and studied, by means of simulations, in a previous paper, where the use of randomized network coding has been proposed as an appealing alternative to classical erasure coding to generate the required redundancy in a distributed storage system. In fact, the proposed network coding-based scheme provides *proactive* network maintenance to guarantee sufficient resource redundancy: this leads to a reduction of maintenance complexity and to a larger amount of free disk space on storage nodes. In the current paper, we provide a preliminary analytical framework that encompasses the use of a few characteristic parameters of the considered distributed storage system, such as the resource availability and the available free disk space portion. Our results show a good agreement between analysis and simulations, thus confirming the effectiveness of our network coding-based approach.**

***Index Terms*—Distributed storage, erasure coding, randomized network coding, peer-to-peer (P2P).**

I. Introduction

The last decade has seen an exponential growth of innovative Internet-based applications. In particular, peer-to-peer (P2P) applications have attracted the interests of the research community because of their high scalability, fault tolerance, etc. Many practical systems have been developed for content distribution (e.g., BitTorrent [1]), receiving significant attention world-wide. An important application of the P2P paradigm is large-scale distributed storage, where information, e.g., a file, is stored with proper redundancy at different nodes of the network: this allows efficient retrieval with low probability of error. Relevant examples of P2P distributed storage systems are OceanStore [2], Total Recall [3], and Wuala [4].

In order to generate redundant information, popular encoding strategies are based on the use of repetition coding or maximum distance separable (MDS) erasure coding, with code rate R_c. In the case of repetition coding, a given number of replicas (namely, $1/R_c$) of the k information symbols are generated. An MDS erasure code, which allows the best redundancy-reliability tradeoff with respect to classical replication-based (i.e., repetition coded) systems, is such that the original information message can be recovered from any subset of size k of the n coded symbols. Possible erasure codes for storage applications are Reed-Solomon codes, parity-array codes, and low-density parity-check (LDPC) codes [5].

Since the seminal paper in 2000 [6], network coding has held the promise to increase the throughput and reliability of a large class of *distributed* systems with multiple information flows. Relevant examples of such systems are wireless networks, sensor networks, P2P networks, etc. [7]. In [8], the authors, leveraging on the topology management of a classical P2P content distribution system, e.g., BitTorrent, propose a novel content propagation system. In this approach, the information is encoded at each node by means of randomized network coding [9], [10]. 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. The obtained results show that this approach achieves the optimal redundancy-reliability tradeoff.

In this paper, we consider the practical network coding-based P2P distributed storage scheme, based on the architecture of the Wuala project, presented in [12]. We extend the simulation analysis of [12] with new analytical results which show the effectiveness of our scheme, in particular concerning maintenance of published resources. Unlike the erasure coding-based Wuala system, where super nodes need to know exactly the number of available fragments and generate new fragments when this number reduces below a "guard" retrieval threshold, in the network coding-based approach maintenance is done proactively and super nodes do not need to check the number of available fragments. The results presented here show that our approach, besides reducing the maintenance complexity (at the super nodes), increases the amount of free available space (at the storage nodes) and reduces the bandwidth usage (network-wide). This gain is shown to come at the price of a smaller resource availability.

II. Related Work

In OceanStore [2], each version of a data object is stored in a permanent read-only form, which is encoded with an erasure code and spread over hundreds (or thousands) of servers. A small subset (of size k) of the encoded fragments is sufficient to reconstruct the archived object: only a global-scale disaster could hinder such a large number of machines to prevent retrieval of the archived object. OceanStore is constructed from interacting resources (such

as permanent blocks of storage or processes managing the consistency of data). These resources are virtual in the sense that they are not permanently tied to a particular piece of hardware and can move at any time. Virtualization is enabled by a structured P2P overlay scheme called Tapestry [13].

The Google File System (GFS) [14] is a distributed file system built for hosting the state of Google's internal applications. GFS uses a simple design with a single master server for hosting the entire metadata and where the data is split into chunks and stored in chunk-servers.

Another distributed storage system, Total Recall [3], is composed of the following three layers. The Storage Manager layer (TRSM) handles file requests from clients and maintains file availability for those files for which it is the master. The Block Store layer reads and writes data blocks on storage hosts. The Distributed Hash Table (DHT) layer is used to maintain the ID space and to provide scalable lookup and request routing. The TRSM dynamically adapts its mechanisms and policies for efficiently maintaining data according to the availability of hosts in the system.

Dynamo is a highly available key-value storage system that some of Amazon's core services use to provide an "always-on" experience [15]. Dynamo is built for latency sensitive applications that require at least 99.9% of read and write operations to be performed within a few hundred milliseconds. For this reason it can be characterized as a zero-hop DHT where each node maintains enough routing information locally to route a request to the appropriate node directly.

Malugo is a recently developed P2P storage system, consisting of two layers [16]. The first one is the bottom layer in which peers are clustered together to form groups and construct a ring topology within the Chord framework [17]. The other is the upper layer in which groups of different regions are connected with each other by utilizing a tree-like topology. In Malugo, 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.

III. Wuala Distributed Storage System

Unlike traditional client-server distributed storage systems, where storage space is supplied only by servers, Wuala [4] is a P2P system, with more than 110 million stored files, where each node can be supplier and client at the same time. The main reason of the success of Wuala is the guaranteed high resource availability (99.99% guarantee of file availability). Each new user is immediately allowed to use 1 GB of free storage space. Moreover, users may buy more storage space and possibly sell their own local disk space.

The distributed architecture of Wuala is based on Chord [17], probably the most diffused P2P overlay scheme adopting the Distributed Structured Model (DSM) [18] i.e., implementing a DHT to store data or, in general, information about resources. In Chord, like in every other decentralized structured overlay schemes, the responsibility of storing data (fragments, in our case, or general information about shared resources) is uniformly distributed among peers.

In Chord each data block is identified by a unique *key* (a m-bit hash of the block's name) and described by a *value* (typically a pointer to the block's owner). Each node n is assigned a random identifier in the same space of keys, and it is responsible for storing key/value pairs for a limited subset of the entire key space. Each node maintains a routing table with up to m entries, called the *finger table*. A scalable lookup algorithm based on finger tables is defined for contacting a successor in this kind of network [12]. It can be proved that, with high probability, the number of nodes that must be contacted to find a successor in a network composed by N nodes is $O(\log N)$ [17].

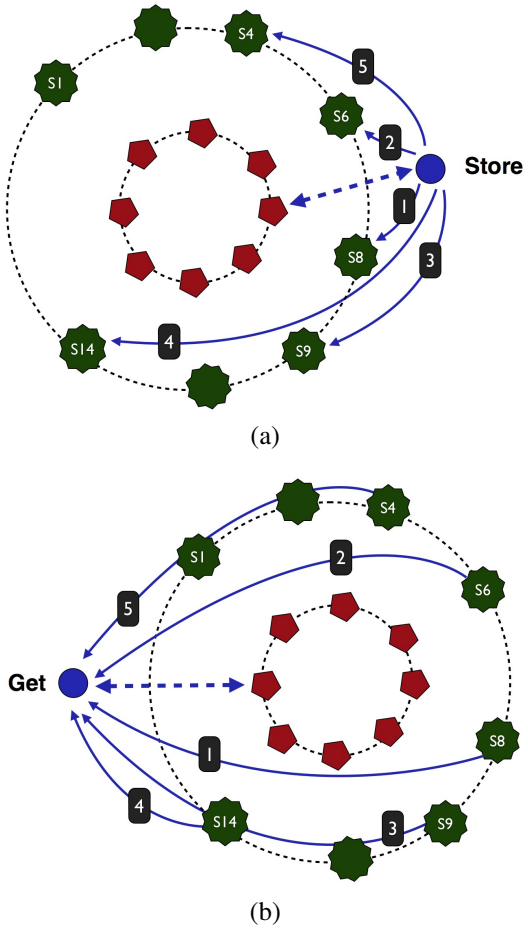
With respect to the traditional Chord structure, illustrated above, Wuala defines three node classes: *Super Nodes*, responsible for message routing in the whole network, *Storage Nodes*, whose duty is to provide at least 1 GB disk space for storing data, and *Client Nodes*, which publish and retrieve files. More details are given in Section IV.

Data redundancy is achieved, by means of MDS erasure coding, as follows. Each file (whose dimension can vary), is split into generations formed by k fragments, from which n coded fragments are generated. The original file can be reconstructed by retrieving, for each generation, any k-subset of the n coded fragments. Different values for k will be considered, through simulations, in Section VI. In all cases, however, $n = 2k$, i.e., the coding rate is $R_c = 1/2$. After a file has been published by a client node, the super nodes periodically generate maintenance events with possible generation of new fragments for the resources they are responsible for, in order to guarantee the presence of enough fragments for reconstructing the file. More details will be given in Section IV-E.

IV. System Architecture

The architecture of the proposed distributed storage system builds upon the P2P approach adopted by Wuala, where peers are both suppliers and consumers of resources. Unlike Wuala, we adopt randomized network coding instead of erasure coding, obtaining two main benefits: reduced storage occupancy and distributed file maintenance (with a consequent reduced bandwidth waste). As anticipated at the end of Section III, the overlay scheme used by Wuala and by our distributed storage architecture is layered, since there are three different groups of nodes with specific behaviors.

Super Nodes form the architecture kernel, providing a mechanism for message routing that allows clients to find storage nodes. Super Nodes play a crucial role in the system and they must be online at every time. Therefore, it is expected that these peers are managed by the organization that provides the distributed storage service or by trusted parties. A Super Node disconnection or malfunctioning can create a serious damage to the overall system functionality.



(a)

(b)

Fig. 1. The considered P2P overlay scheme with an example of (a) resource store and (b) resource retrieval process.

Storage Nodes guarantee a high availability with long sessions and great reliability in the network. They store fragments published by client nodes.

Client Nodes work as data publishers and consumers. They can publish or retrieve resources through the system, as will be discussed later.

Fig. 1 shows a logical scheme of the system structure, with (i) an "inner ring" of Super Nodes, (ii) an "outer ring" of Storage Nodes, and (iii) a Client Node. The figure provides illustrative representations of (a) resource publishing ("Store") and (b) resource retrieval ("Get") actions from the Client Node.

In the remainder of this section, we detail the main actions involving the nodes in the system. Note that the join of a client node (see Section IV-A) and its promotion to storage node (see Section IV-B) are the same of the Wuala system. In all other operations, we will distinguish between erasure coding (as in Wuala) and randomized network coding.

A. Network Join

In order to join the network, a Client Node needs to know at least one Super Node that will manage the creation of a new key, through a consistent hashing function, for the incoming peer. This key is used to find a Super Node gateway to be used for the communication activities. This operation corresponds to the placement of a resource in the Chord ring, where in this case the resource is the new peer.

B. Evolution from Client Node to Storage Node

The constraint for the evolution of a Client Node to a Storage Node depends on the daily online presence, which must be at least of 4 hours. This presence must be also continuous, being monitored by the gateway for 10 days. After this period, the associated Super Node checks if the client can be moved to the Storage class. If a Storage Node is unavailable for a time superior to 3 days, it is marked as unreliable and downgraded to the Client Node class.

C. Resource Publishing

In the proposed scheme, when a file must be stored, it is divided into generations composed by k fragments, which can be collected in a column vector denoted as $\boldsymbol{s} = (s_1, \ldots, s_k)^T$, where T denotes the transpose operator. These fragments belong to GF(q) and are linearly combined according to coefficients belonging to the same field, with a pre-determined overhead, and published to storage nodes. In the case with MDS erasure coding, the combined packets are guaranteed to be linearly independent, whereas in the randomized network coding-based approach, the coefficients are chosen randomly and uniformly. This implies that there exists a non-zero probability that two coded packets are linearly dependent; however, it is well known that this probability goes to zero if q is sufficiently large [10] and this is confirmed by our simulation results [12]. Each packet flowing in the network contains both the coded payload and a small header which carries information about the generation to which the fragment belongs and the global coding vector of dimension k. For instance, if $k = 50$ and $q = 2^{16}$, the extra header has size equal to

$$16\,\text{bits} \times 50 = 100\ \text{Bytes}.$$

If the payload is equal to 500 KB, then the overhead is equal to 0.02%.

At this point, a unique key is assigned to the resource, allowing it to route to the responsible Super Node (the one having the nearest key). For each fragment, this Super Node searches for the Super Node responsible for this fragment. The latter selects the most suitable Storage Node, among those for which it acts as gateway, applying a load balancing strategy, and the Super Node responsible for the resource saves, in a list, the references about the fragments and the corresponding Storage Nodes.

D. Resource Retrieval

To retrieve a published file, according to network coding theory, it is necessary to retrieve, for each generation, k linearly independent fragments, using the previously illustrated lookup functionality. After these fragments have been collected, the following system of linear equations is solved using the classical Gauss elimination algorithm:

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

where $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_k)^T$ is a column vector containing the k linearly independent fragments and $\boldsymbol{A}$ is a matrix whose rows correspond to the coding vectors of each retrieved fragment.

Referring to the "GET" function in Fig. 1, the Super Node gateway routes the Client Node's request to the responsible node and then to the Storage Nodes which have the fragments. In the presence of randomized network coding, once the file has been downloaded, we propose that the Client Node triggers a re-generation process, as will be discussed in more detail in the next section.

E. Resource Maintenance Strategy

According to the definitions given in [11], the maintenance strategy aims to perform *functional repair* and not *exact repair*. In fact, the new generated fragments during the maintenance operations are not exactly the same as those lost by disconnected nodes, but they have the same characteristics.

In the Wuala system, resource maintenance is carried out by the Super-Nodes *reactively* as follows. Super-Nodes periodically (i.e., every T_{M} seconds) check the availability, in the Storage Nodes, of the resources for which they are responsible. If the percentage of surviving fragments falls below a properly defined retrieval "guard threshold" (set in our simulation to 70% of n), generates new fragments independent from the surviving ones, and distributes them in Storage Nodes. The number of new generated fragments is chosen so that the overall number of available fragments for a resource is above the "retrieval guard threshold," which represents the fraction of fragments below which the resource is likely to become soon unavailable.

In the proposed system, resource maintenance is carried out *proactively* as follows. Whenever a client node retrieves a resource, it generates a new amount of fragments, set in our simulations to 15% of k. Owing to the fact that randomized network coding guarantees, with very high probability, that every new generated fragment will be independent from the ones already distributed, the Super-Node does not need to check the existing fragments. In other words, the complexity of the maintenance operations reduces. Moreover, as the number of exchanged control messages is significantly smaller, bandwidth waste is more limited. One should observe that this maintenance strategy may be applied to erasure coding-based techniques. However, in MDS erasure coding schemes the client nodes should check if the fragments are linearly independent from all other in the networks. Therefore, randomized network coding is a more suitable technique for the distributed maintenance strategy.

V. Analytical Framework for the Resource Availability

The resource availability is defined as the probability that, at any given time, it is possible to successfully download a given resource. For simplicity, let us assume, as in [19], that each coded fragment is stored in a different storage node. Note that in our Wuala-compliant system, each storage node can store more than one fragment associated with the same resource, due to the considered DHT-based overlay architecture. Although this assumption is not always verified in our scheme, simulation results will show that the results predicted by the analytical framework are still meaningful. Note that similar considerations for MDS codes have also been done in [11].

First, let us consider the erasure coding-based system. If MDS codes are adopted, one is able to successfully recover a file if k out of the n fragments can be downloaded.[1] Supposing that each storage node is available with probability p and that disconnection events are independent, the availability with erasure coding can be computed as

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

It is well known [19] that, for a given value of n and sufficiently large p, the availability (2) is larger than that of a repetition code, which is given by

$$A_{\mathrm{R}} = 1 - (1-p)^{\frac{1}{R_{\mathrm{c}}}}.$$

In the Wuala system, a hybrid strategy is used, by also storing a full copy of the resource in a remote server for backup purposes.[2] In this case, it is impossible to download the resource if the backup server is not connected and less than k fragments are available. Assuming that the two events are independent, this occurs with probability

$$p_{\mathrm{HEC}} = (1-p) \sum_{i=0}^{k-1} \binom{n}{i} p^i (1-p)^{n-i} = (1-p)(1-A_{\mathrm{EC}}).$$

Therefore, for the hybrid strategy (erasure coding & full copy) used in Wuala, availability can be written as

$$A_{\mathrm{HEC}} = 1 - p_{\mathrm{HEC}} = 1 - (1-p)(1-A_{\mathrm{EC}}). \tag{3}$$

Let us now consider the scheme with randomized network coding. In this case, one is able to succesfully recover a resource if 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 recovered" and $\mathscr{B}$ the event "at least k of these are linearly independent." Therefore, one can write

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

where mutual exclusivity of the events $\mathscr{E}_i$ has been used. Note that $P(\mathscr{E}_i)$ equals the i-th term in (2) and $P(\mathscr{B}|\mathscr{E}_i) \leq 1$ ($i = k, \ldots, n$). Therefore, it is expected that the availability with randomized network coding is smaller than that with erasure coding (or erasure coding + copy). Moreover, from the well-know results in the field of network coding, it

[1] Note that, at a given time, the number of fragments for a given resource may be greater 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] This situation has not been simulated in our environment, since we are interested in a complete distributed system where a user is able to download a resource without the help of any central entity.

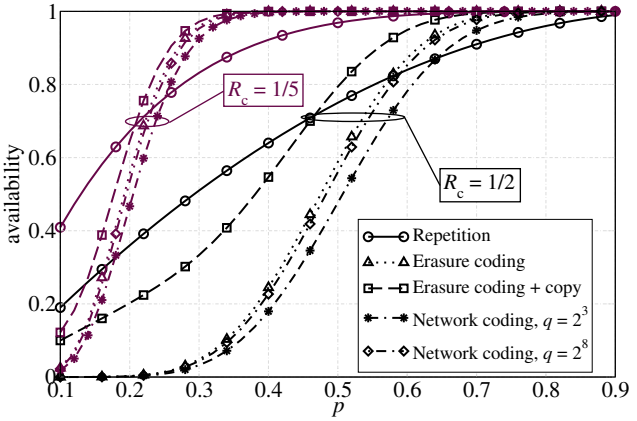


Fig. 2. Resource availability, as a function of the storage node availability, in a scenario with $k = 10$ fragments. The code rate is set either to 1/2 ($n = 20$) or 1/5 ($n = 50$). Different coding strategies are considered.

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

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

where the term $P(D_i = j)$ can be written as [20]

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

where q is the field size.

In Fig. 2, the resource availability is shown, as a function of the storage node availability, in a scenario with $k = 10$ fragments. 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, (iii) erasure + copy, and (iv) randomized network coding with $q = 2^3$ or $q = 2^8$. As predicted by our theoretical framework, the best strategy is the hybrid with erasure coding and one full replica of the resource. Moreover, the use of randomized network coding leads to performance degradation. However, for sufficiently large values of p, this loss is negligible and the use of larger field size q allows to approach the performance of optimal erasure codes. Finally, the smaller is the code rate, the lower is the performance loss. However, since the performance reduction is sufficiently low for $R_c = 1/2$, in the following we will consider this code rate for both erasure coding and network coding.

VI. Simulation Results

The simulation tool used in this paper is DEUS, a discrete event universal simulator, based on Java and XML, created at the University of Parma [21].[3] The simulator is based on the concept of *event* and each new time step is characterized by the generation of an event. The network's time evolution is given in terms of *virtual time* (VT), i.e., each virtual time unit in the network evolution corresponds to the generation of an event. In the following, we will refer, without distinction, to time and VT.

A. Set-up

The following scenario has been considered in order to evaluate the performance of the proposed P2P distributed storage system:

- 10 super nodes with the associated storage nodes;
- 50 storage nodes, randomly associated with the super nodes, with a maximum 10 GB disk space each;
- 500 client nodes which can generate resources with a size $\mathscr{M}$ uniformly distributed between 1 MB and 35 MB (the average size is $\overline{\mathscr{M}} = 18$ MB).

At the beginning of the simulation, 10 storage nodes and 80 client nodes are randomly disconnected from the network and the network evolution is analyzed from this point on. The analysis is carried out for 30 days and during each hour 100 events (corresponding to generations and searches of resources in the network) are scheduled, so that the overall simulation time is

$$30\,\text{days} \times 24\,\text{hours} \times 100\,\text{events} = 72000\ \text{VTs}.$$

Since events are periodically generated, it means that 1 VT corresponds to 36 seconds. The resources to be published are divided into generations of k elements, so that the overhead in the packet header containing the coding vector is k bytes (if the operations are in $GF(2^8)$). The simulation results are averaged over 100 independent runs in order to reduce statistical fluctuations. In particular, during every run our simulator generates different connections, disconnections, and publishing/download events.

We have performed simulations by scheduling searches every 5 VTs (corresponding to 3 minutes), whereas connections and disconnections are scheduled every 20 VTs (corresponding to 12 minutes). Note that during each event, the simulator randomly selects the client node which performs the operation (search, connection, disconnection) and eventually the resource to be searched. We also set a maximum percentage of 40% of client nodes which can be promoted to storage nodes, according to the protocol described in Subsection IV-B. The publication overhead is set to 100%, i.e., $R_c = 1/2$. The maximum overhead of the regeneration after each download is set to 15%. Moreover, *churn*, i.e., departures of peers from the P2P network, is evaluated by allowing super nodes to count the number of expired connections. This value is updated each hour. If churn is detected, i.e., there is a high rate of connections/disconnections of peers from the network, the overhead increases by 10% of the generation size, in order to compensate for this phenomenon. Finally, in a scenario with erasure coding maintenance is performed every T_M VTs: for instance, $T_M = 48$ VTs means that maintenance is scheduled approximately every 30 minutes.

[3] 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 I
AVERAGE (μ) RESOURCE AVAILABILITY AND ITS STANDARD DEVIATION (σ) FOR $k = 50$ FRAGMENTS. ERASURE CODING WITH DIFFERENT VALUES OF T_M IS COMPARED WITH NETWORK CODING WITH DIFFERENT SIZE OF GALOIS FIELD.

	A_E ($T_M = 48$ VTs)	A_E ($T_M = 72$ VTs)	A_{NC} ($GF(2^3)$)	A_{NC} ($GF(2^8)$)	A_{NC} ($GF(2^{16})$)
μ	0.999965	0.99997	0.987047	0.988985	0.987047
σ	0.000052	0.000042	0.008565	0.007413	0.008565

B. Results

The first analysis carried out in this paper concerns the resource availability, defined as the success rate of downloading a resource by a client node. In Table I, the average (μ) resource availability and its standard deviation (σ) are shown in a scenario with $k = 50$ fragments. The availability with erasure coding (with different values of T_M) is compared with that with network coding (with different size of Galois field). The value of μ is computed by averaging the availability over the whole simulation time and all possible resources. As one can see, the scheme with erasure coding allows to obtain, as expected from Fig. 2, a larger average availability with respect to the scheme with randomized network coding. This is due to the different distributed redundancy maintenance performed by client nodes, as illustrated in Section IV-D. In fact, periodic maintenance allows to regularly check the status of all resources, whereas distributed maintenance is only performed when a client has finished a succesful download and, therefore, a resource may not be regularly maintained if it is not sufficiently "popular." One should note that the use of larger finite size Galois field does not lead to significant performance improvement. This means that in the proposed network coding approach a finite field size equal to 2^3 is sufficient to obtain good performance. However, randomized network coding leads to an average availability close to 99% instead of 99.99%.

Although the average resource availability is close to 100%, it may be of interest to evaluate the time evolution of this quantity. In Fig. 3, the resource availability is shown, as a function of the virtual time, for a scenario with generation size $k = 100$. Erasure coding is compared to network coding with $GF(2^{16})$. First, one can observe that erasure coding allows to have approximately constant availability close to 100%. This is due to the fact that the resource availability can be determined through (2) and statically preserved through maintenance (in this scenario every $T_M = 48$ VTs). In the presence of network coding the resource availability is, on average, around 99%, as previously shown in Table I. However, the time evolution in the presence of network coding is different and more oscillating. Finally, we remark that in the presence of network coding, there may be a drastic reduction in resource availability, as preliminary observed in [12].

In Fig. 4, the performances with erasure (with $T_M = 48$ VTs) and network (with $GF(2^{16})$) coding are compared in a scenario with $k = 50$, considering two performance indicators: (a) the number of generated fragments and (b) the average free available disk space per storage node

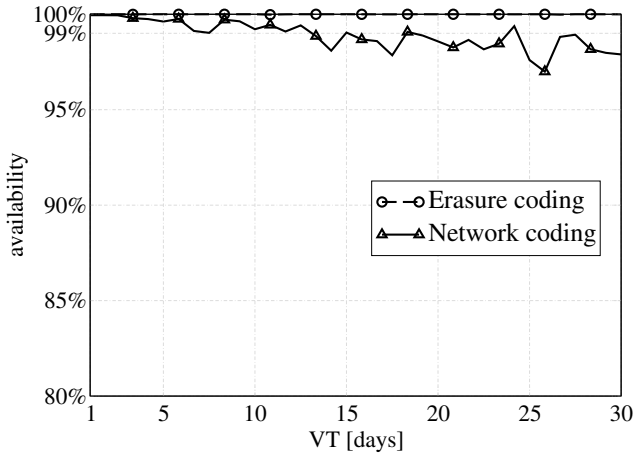


Fig. 3. Resource availability as a function of the virtual time. The generation size is $k = 100$ and the considered field size is 2^{16} for the scenario with network coding.

as a function of time. One can directly observe that our "simple" network coding strategy allows us to drastically reduce the number of generated fragments from almost 5×10^5 to almost 2×10^5 (Fig. 4 (a)). This is due to the fact that in the presence of erasure coding a large number of maintenance events are scheduled to have a sufficient number of fragments in the network to retrieve the complete resource. Each maintenance event generates new fragments to be stored in the network and increases the total number of fragments in the network. However, as already observed in Table I, this reduction in the number of generated fragments comes at the price of a smaller resource availability.

This is confirmed by the results in Fig. 4 (b), where the evolution of the average available free disk space per storage node is shown as a function of the time. As one can see, network coding allows us to save, on average, about 1.2 GB with respect to the classical erasure coding technique, since it generates less fragments. Therefore, with randomized network coding it is possible to achieve an average saving of 23%. Moreover, although both techniques are applicable to this scenario with acceptable results in terms of resource availability, our approach based on network coding could be preferred. In fact, it is able to dynamically trace the network evolution, guaranteeing, at the same time, lower storage space occupation. Moreover, the resource maintenance is easier for the Client Nodes, since they do not regularly check the resource availability, but re-generate fragments every time the resource is downloaded, as explained in Section IV-D.

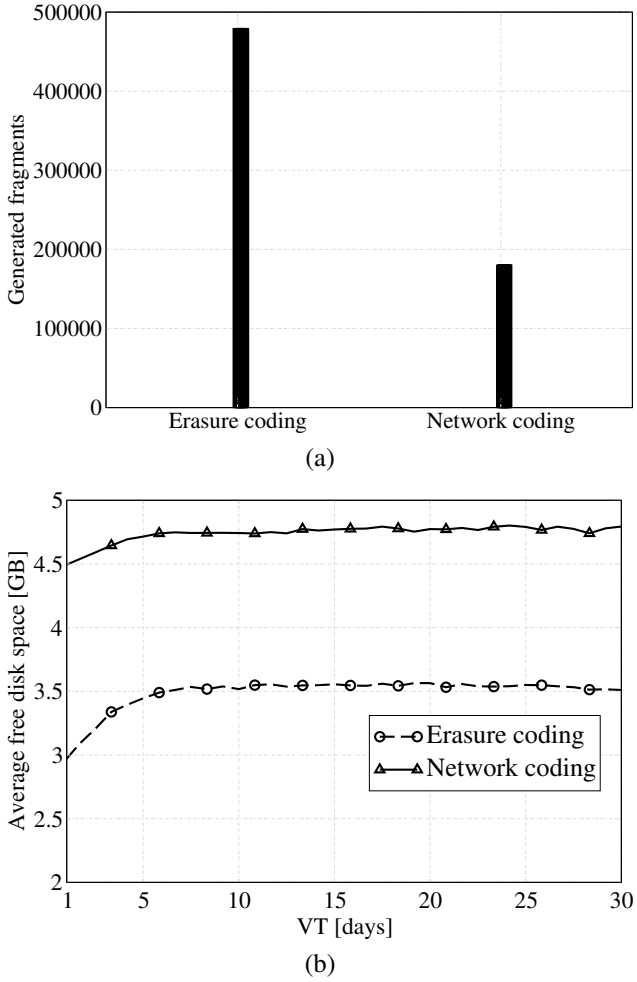


Fig. 4. Comparison between the performance of erasure (with $T_M = 48$ VTs) and network coding (with $GF(2^{16})$) in a scenario with $k = 50$. In case (a), the number of generated fragments is considered, whereas in case (b) the average free available disk space per storage node is shown as a function of time.

VII. Concluding Remarks

In this paper, we have analyzed the performance of a peer-to-peer (P2P) distributed storage architecture based on the layered overlay scheme defined by the Wuala project, where the use of randomized network coding has been proposed, as an appealing alternative to classical erasure coding, to generate and maintain the required redundancy in a fully decentralized distributed storage system. Motivated by a simulation study presented in a previous paper of ours, we have presented here a preliminary analytical framework which allows to evaluate the resource availability as a function of key system parameters. A good agreement between analytical and simulation results has been observed, thus confirming the effectiveness of our network coding-based approach.

References

[1] "Bit Torrent Project," URL: `http://www.bittorrent.com`.

[2] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. D. Kubiatowicz, "Pond: the OceanStore prototype," in *2nd USENIX Conf. on File and Storage Technologies* (FAST), San Francisco, CA, USA, March 2003, pp. 2235–2245.

[3] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. M. Voelker, "Total Recall: Systems support for automated availability management," in *Proc. USENIX Symp. on Networked Systems Design and Implementation* (NSDI), San Francisco, CA, USA, March 2004.

[4] "The WUALA Project," URL: `http://www.wuala.com`.

[5] J. S. Plank, "Erasure codes for storage applications," in *4th USENIX Conf. on File and Storage Technologies* (FAST), San Francisco, CA, USA, December 2005.

[6] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[7] C. Fragouli and E. Soljanin, *Network Coding Applications*. Hanover, MA, USA: Now Publisher Foundations and Trends in Networking, 2007.

[8] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE Conf. on Computer Commun.* (INFOCOM), Miami, FL, USA, March 2005, pp. 2235–2245.

[9] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, and B. J. S. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, October 2006.

[10] C. Fragouli and E. Soljanin, *Network Coding Fundamentals*. Hanover, MA, USA: Now Publisher Foundations and Trends in Networking, 2007.

[11] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. O. Wainwright, and K. Ramachandran, "Network coding for distributed storage systems," *IEEE Trans. Inform. Theory*, vol. 56, no. 9, pp. 4539–4551, September 2010.

[12] M. Martalò, M. Picone, R. Bussandri, and M. Amoretti, "A practical network coding approach for peer-to-peer distributed storage," in *Proc. IEEE International Symposium on Network Coding* (NetCod), Toronto, Canada, June 2010, pp. 103–108.

[13] K. Hildruw, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao, "Distributed object location in a dynamic network," in *40th ACM Symposium on Parallel Algorithms and Architectures* (SPAA), San Francisco, CA, USA, March 2002, pp. 41–52.

[14] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," in *Proc. ACM Symp. on Symposium on Operating Systems Principles* (SOSP), Landing, NY, USA, October 2003, pp. 29–43.

[15] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *ACM SIGOPS Operating Systems Review Archive*, vol. 41, no. 6, pp. 205–220, December 2007.

[16] P.-C. S. Y.-W. Chan, T.-H. Ho and Y.-C. Chung, "Malugo: A peer-to-peer storage system," *Int. J. Ad Hoc and Ubiquitous Computing*, vol. 5, no. 10, pp. 209–218, April 2010.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and 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, March 2001, pp. 149–160.

[18] M. Amoretti, "A survey of peer-to-peer overlay schemes: Effectiveness, efficiency and security," *BSP Recent Patents on Computer Science*, vol. 2, no. 3, pp. 195–213, September 2009.

[19] R. Rodrigues and B. Liskov, "High availability in DHTs:Erasure coding vs. replication," in *Proc. Int. Workshop on Peer-to-Peer Systems* (IPTPS), Ithaca, New York, USA, February 2005, pp. 226–239.

[20] S. Acedanski, S. Deb, M. Medard, and 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, April 2005.

[21] M. Amoretti, M. Agosti, and F. Zanichelli, "DEUS: a discrete event universal simulator," in *2nd ICST/ACM International Conference on Simulation Tools and Techniques* (SIMUTools), Rome, Italy, March 2009.