

MIT Open Access Articles

Implementation and performance evaluation of distributed cloud storage solutions using random linear network coding

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: Fitzek, Frank H.P., Tamas Toth, Aron Szabados, Morten V. Pedersen, Daniel E. Lucani, Marton Sipos, Hassan Charaf, and Muriel Medard. "Implementation and Performance Evaluation of Distributed Cloud Storage Solutions Using Random Linear Network Coding." 2014 IEEE International Conference on Communications Workshops (ICC) (June 2014).

As Published: <http://dx.doi.org/10.1109/ICCW.2014.6881204>

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Persistent URL: <http://hdl.handle.net/1721.1/100941>

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike



Implementation and Performance Evaluation of Distributed Cloud Storage Solutions using Random Linear Network Coding

Frank H.P. Fitzek^{1,2}, Tamas Toth^{2,3}, Aron Szabados^{2,3}, Morten V. Pedersen¹, Daniel E. Lucani¹,
Marton Sipos^{2,3}, Hassan Charaf³, Muriel Medard⁴

¹Aalborg University, Denmark

²desk.io GmbH, Germany

³BME, Budapest University, Hungary

⁴RLE - MIT, USA

Abstract—This paper advocates the use of random linear network coding for storage in distributed clouds in order to reduce storage and traffic costs in dynamic settings, i.e. when adding and removing numerous storage devices/clouds on-the-fly and when the number of reachable clouds is limited. We introduce various network coding approaches that trade-off reliability, storage and traffic costs, and system complexity relying on probabilistic recoding for cloud regeneration. We compare these approaches with other approaches based on data replication and Reed–Solomon codes. A simulator has been developed to carry out a thorough performance evaluation of the various approaches when relying on different system settings, e.g., finite fields, and network/storage conditions, e.g., storage space used per cloud, limited network use, and limited recoding capabilities. In contrast to standard coding approaches, our techniques do not require us to retrieve the full original information in order to store meaningful information. Our numerical results show a high resilience over a large number of regeneration cycles compared to other approaches.

I. INTRODUCTION AND MOTIVATION

The use of coding for providing reliability in storage solutions dates back to the introduction of Redundant Array of Independent Disks (RAID) systems [1]. These storage solutions have been widely used for back up solutions in the past, but have been characterized by a highly planned and inflexible coding process due in part to the type of error correcting codes used. In fact, early RAID systems needed hard disks to have the same storage space and speed. Although successful, these approaches may not be well suited for more distributed storage solutions, e.g., cloud storage, which are characterized not only by heterogeneity but also by high costs to transmit data across the network during the regeneration of storage units. This can be particularly taxing when using standard coding techniques, which require enough data to be gathered in order to first decode and then re-encode the missing parts.

Network coding has been identified as a viable technology for enabling distributed storage by reducing the system's requirements for regenerating redundancy. It was originally proposed in [2] in the context of sensor networks. Network coding has a series of advantages over standard end-to-end coding, including the possibility to recode already encoded data without destroying the code properties. Since that initial

work, research has focused strongly on code regeneration, e.g., [3], with [4] providing a comprehensive survey on the topic. More recently, an initial research implementation of distributed storage with network coding was reported in [5], while [6] focused on auditing systems for network coding storage.

Finally, [7] used random linear network coding (RLNC) to implement a coded cloud storage provider that spread coded data over several clouds. Coding was shown to reduce download times of stored files while requiring less storage space in total (and also per cloud) compared to state-of-the-art approaches. However, these approaches still rely on a fairly static structure, a moderate storage size, and careful planning process in order to yield the desired benefits. This contrasts with the current and future trends of storage systems, which envision highly dynamic conditions. One example includes edge caching in communication networks or peer to peer networks that need to adapt to highly requested content. Another example are cloud centers, where hot storage disks, used for highly demanded content, cannot be accessed and only the cold storage disks can be used to retrieve the data. Flexibility is thus critical as cloud systems grow in size, need to recover from physical hard disk failures, outages due to software updates, or need to react efficiently to sudden load balancing activities. These scenarios inherently limit the ability of the system to plan and optimize the data regeneration process.

This paper aims to address these new challenges by developing network coding approaches for distributed and highly dynamic storage systems. These approaches allow the system to trade-off traffic costs and storage costs while maintaining a high reliability over time. By developing a simulator based on the Kodo C++ library [8], we show that state-of-the-art approaches based on other coding mechanisms are unable to provide reasonable reliability after several loss/recovery cycles while our techniques can be highly reliable with little planning or coordination for a wide range of operating conditions.

II. NETWORK CODING BASICS

In this work, we consider random linear network coding (RLNC) to generate coded data in our cloud storage systems.

RLNC linearly combines uncoded packets into any number of coded packets using random coding coefficients from a finite field. The coded packet has the same size as the uncoded packets plus some additional information referred to as the encoding vector, which comprises the values of the random coefficients used to generate that particular coded packet. In contrast to any other end-to-end coding approach, RLNC allows for recoding, i.e. any intermediate node is able to code over any number of already code packets without needing to decode the data. This feature is vital for our distributed cloud approach in order to reduce the amount of data to be conveyed to a newly added cloud storage device without compromising reliability.

III. SYSTEM MODEL

In the following, we describe the different strategies under investigation for distributed cloud storage. Our main focus is on the different strategies for network coding but we also introduce two comparison approaches relying on uncoded data and Reed–Solomon (RS) coding.

A. Network Coding and Storage Model

As given in Figure 1, we consider that data is packetized into G segments (packets). These G segments are coded into J coded segments using RLNC with a finite field of size F . We define R as the redundancy factor given by the ratio of J and G , i.e., $R = \frac{J}{G}$. The coded segments are stored afterwards into C clouds. For simplicity, we assume each cloud can store at most Q coded segments, but the model can be extended to cover clouds with heterogeneous storage sizes. In order to store all J packets, the number of clouds C with capacity Q needs to be at least

$$C = \lceil J/Q \rceil. \quad (1)$$

B. Cloud Loss and Recovery Model

We assume L out of C clouds are randomly lost in each simulation round. In order to restore the lost clouds, P out of the remaining $C - L$ clouds can be used to fill L new empty clouds according to the strategies explained in Sections III-D and III-E. This simple model that removes and adds L clouds is mimicking dynamics in a cloud storage system attempting to preserve the same number of coded storage devices over time. But our approach can also be used to control the *hot* and *cold* storage areas in a cloud center. In general clouds will be characterized by adding more and more storage to handle to increasing service requests. Without going into detail, the presented approaches using network coding are highly suited for such changes, while traditional codes would need to recall all data in order to change the coding strategy.

C. Performance Metrics

- **Success Probability:** After filling up the new L clouds, the integrity of the data is checked over all C clouds. If successful, the procedure starts from the beginning until H rounds are completed. The tests are repeated

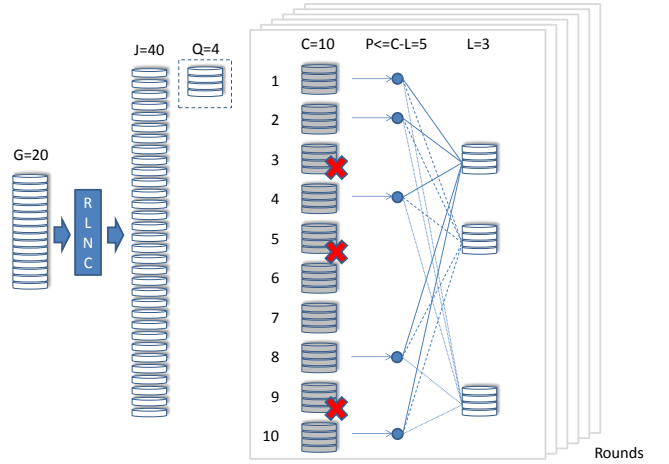


Fig. 1. Model of the dynamic distributed storage system losing L clouds and recovering by exploiting data from a subset of the remaining clouds.

several times to derive the success probability p_s , which is defined by the ratio of successful test to the overall number of tests carried out.

- **Traffic Cost:** The number of transmitted segments T_s for a specific strategy s to regenerate the missing clouds is a key metric predicting network use. In order to keep the traffic cost low, T_s should be low.
- **Storage Cost:** R is the value specifying how much more storage will be used compared to the original size. In order to keep the storage cost low, R should be low.

Ultimately, our goal is to minimize the amount of storage over all clouds and traffic among the clouds while being still able to retrieve the original data without losses.

D. Recovery Strategies

In the following we explain the different strategies how the lost clouds L are filled by their remaining P parents. The strategies will differ in the success probability p_s and the related traffic that is needed to repair for the lost cloud storage.

1) *Post-Recoding Approach:* P clouds are conveying all available $(P \cdot Q)$ information to the new L clouds. Each cloud out of L will recode over all $P \cdot Q$ received packets storing only Q segments into the cloud, where the remaining $(P-1) \cdot Q$ are simply discarded. Due to the large number of received packet and the possibility to recode, each of the L clouds will store different coded versions and the likelihood to store redundant information across the new clouds is reduced with an increased field size. The resulting traffic T_{post} per round equals

$$T_{post} = P \cdot Q \cdot L. \quad (2)$$

2) *Pre-Recoding Approach:* In order to reduce the traffic involved in filling the new clouds, pre-recoding is introduced. Each of the P clouds recodes over its own Q packets sending only $\lceil Q/P \rceil$ to each new cloud. For each new cloud, different coded versions are generated. The receiving cloud will only store the received coded versions without any further actions.

The overall traffic T_{pre} is drastically reduced in comparison with the previous approach

$$T_{pre} = \left\lceil \frac{Q}{P} \right\rceil \cdot P \cdot L = Q \cdot L + \theta \cdot P \cdot L, \quad (3)$$

where $\theta \in [0, 1)$.

Here the traffic is reduced significantly by a factor of up to P , but the diversity in the code regeneration is small. Therefore, we now propose hybrid approaches combining the benefits of the former two.

3) *Hybrid Approach*: While the post-recoding approach sends the maximum traffic but achieving the maximal mixing of the coded segments, the pre-recoding is clearly sending the minimum traffic with limited mixing capabilities across different clouds, i.e., only recoding within each existing cloud. As the name already implies, the hybrid solution provides a simple mechanism to trade-off traffic and quality of the coding (mixing) produced.

In the hybrid approach, each cloud out of P recodes over its own packets sending $\lceil \alpha \cdot Q/P \rceil$ to each of the L clouds. The values for α are between 1 and P , where $\alpha = 1$ corresponds to the pre-recoding and $\alpha = P$ to the post-recoding approach. Each new cloud receives now $\lceil \alpha \cdot Q/P \rceil \cdot P$ over which the new cloud recodes again in order to store Q segments into the new cloud, while the unused segments are discarded. The overall traffic T_{hybrid} for the pre-recoding approach is given by

$$T_{hybrid} = \left\lceil \frac{\alpha \cdot Q}{P} \right\rceil \cdot P \cdot L = \alpha \cdot Q \cdot L + \theta \cdot L \cdot P, \quad (4)$$

with $\theta \in [0, 1)$.

E. Reed–Solomon Coding Approach

In order to compare our network coding approaches with the state of the art, we use Reed–Solomon (RS) coding. There are two ways of using Reed–Solomon in this context and the first one will result in more traffic but makes sure the integrity of the information will be always intact, while the second is prone to loose information but using use less traffic, which is comparable to our network coding approaches.

1) *Fully controlled RS approach*: Each of L clouds has to retrieve any G segments from the remaining P parents, decode the received segments if possible, and store the information that was lost beforehand. This approach needs some overlay control entity to organize that the L clouds are filled in the correct way. The resulting traffic $T_{RS,control}$ in this case is

$$T_{RS,control} = G \cdot L, \quad (5)$$

without taking into account for the extra signaling that would be needed among the overlay control entity and all C clouds.

An optimization in the amount of traffic generated is to retrieve all G segments in of the L clouds and create the missing $L \cdot Q$ segments and distribute the missing pieces to the $L - 1$ clouds, so that the traffic equals

$$T_{RS,control} = G + Q \cdot (L - 1). \quad (6)$$

This requires that the one of the L cloud storage devices has also the capability to perform the required coding and has enough computational power to do so.

2) *Random RS approach*: In contrast to the controlled approach, we could forward randomly selected pieces to the new clouds. Following the pre- and the post-recoding approach for network coding, Reed–Solomon could also perform a traffic aggressive or a traffic careful policy, sending either $P \cdot Q$ or only Q segments to each of the L new clouds, respectively. The resulting traffic $T_{RS,random}$ here is therefore similar to the traffic of the post and pre-recoding approach of network coding given in Section III-D1 and III-D2, respectively, that is

$$T_{RS,random} = \beta \cdot P \cdot Q \cdot L, \quad (7)$$

with β being either 1 (large traffic) or $1/P$ (low traffic).

F. Uncoded Approach

The uncoded approach used here is also used using random filling as described for Reed–Solomon. Clearly, a carefully planned scheduling approach would do better, but this would require that we would always have access to sufficient number of segments and ensure coordination. The approach is used for comparison and not as a recommendation for use in a storage system.

IV. IMPLEMENTATION

In order to undergo a performance evaluation the approaches introduced beforehand have been implemented in a simulator. The core of the simulator is the KODO library [8], an SDK for network coding that supports different field sizes, namely binary, linear extension fields 2^8 as well as 2^{16} , and one optimal prime field (OPF) with $2^{32} - 5$. It also includes an extension for Reed–Solomon (RS) codes, an important feature to compare RLNC with codes used in state-of-the-art systems. The simulator then is a flexible tool to visualize the dynamics of the distributed clouds as well as running the simulations with the given set of parameters.

The simulator is a platform independent application written in Java. Since the KODO SDK is written in C++, method calls and data access from Java are performed using the JNI framework. We have implemented an intuitive user interface to enable getting quick result for a particular set of parameter values. The state of the system can be tracked on a round-by-round basis using a graphical illustration. This gives a more in-depth understanding of the processes in effect and allows the replay of a certain scenario for closer inspection.

V. NUMERICAL RESULTS

In this section we provide numerical results for the approaches presented beforehand using the analytics and the simulation tool introduced beforehand. If not specified otherwise, the field size F is chosen to be 2^8 and the number of storage

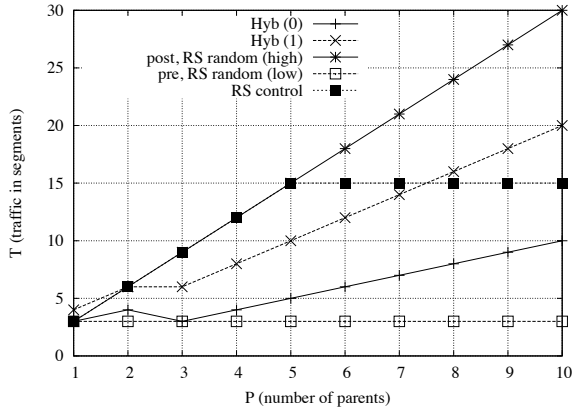


Fig. 2. Traffic T versus number of available parents P for the different strategies ($L=1$ $Q=3$ $F=2^8$ $G=15$).

entities to be replaced L is one. Furthermore the number of segments for the original data is $G = 15$. Furthermore, the number of clouds is $C = 15$, the number of storage segments per cloud is $Q = 10$.

A. Traffic

Figure 2 shows the traffic T versus the number of parents P for the different strategies described in Section III. We assume again only one lost storage entity, thus $L = 1$. Therefore, the controlled RS scheme has to retrieve any $G = 20$ segments in order to restore the lost packets successfully. If too few parents are available the controlled RS scheme may fail. In our example to retrieve 20 segments we need at least 5 parents as each parent contributes with $Q = 4$ segments. If fewer parents are available the controlled RS scheme cannot be applied and one of the RS random approaches should be used.

The post-recoding NC and the full random RS increase the traffic T linearly with any additional parent. The pre-recoding and the low random RS approach always ask for a bare minimum to fill up the own storage Q and is therefore independent from the number of parents in terms of traffic. Obviously there is no gain in having more parents than actually storage places. If $P > Q$ only Q out of P parents will send one segment.

B. Cloud integrity

In Figure 3 the cloud integrity versus the number of available parents and the number of storage place per cloud Q is given for different coding strategies.

For the given results we assume to have only one cloud loss $L = 1$ per round that needs to be repaired, which is the most common case in data centers. With respect to Figure 3, the first, second, and third column show the cloud integrity after 10, 100, or 1000 rounds, respectively.

The first row shows the uncoded approach. In Figure 3(a) the cloud integrity can only be guaranteed for a large value

of the storage per cloud Q , which means high costs in maintaining the cloud infrastructure. The number of parents have a smaller impact. For larger rounds the probability to retrieve the data is going towards zero.

The second row shows the random RS approach. For 10 rounds the cloud integrity can be assured by a small number of parents ($P \geq 1$) and a small number of storage ($Q \geq 2$). For 100 rounds the cloud integrity is significantly decreased and can only be assured by a large number of parents ($P \geq 3$) and a very large number of storage per cloud ($Q \geq 8$). For 1000 rounds the cloud integrity can not be guaranteed anymore.

In the third and forth row the post and pre-recoding network coding approaches are presented. Obviously both approach yield better performance for the cloud integrity. Even for large round numbers such as 1000 both approaches are still robust. The differences for 10 rounds is not visible and both clouds are intact for $P \geq 1$ and $Q \geq 2$. For 100 rounds the pre-recoding approach can assure the integrity with $P \geq 2$ and $Q \geq 7$ or with $P \geq 4$ and $Q \geq 4$ (optimal working point with respect to cloud storage).

The post-recoding approach can assure the integrity with $P \geq 2$ and $Q \geq 5$ (optimal working point with respect to traffic) or with $P \geq 5$ and $Q \geq 2$ (optimal working point with respect to cloud storage) after 100 rounds.

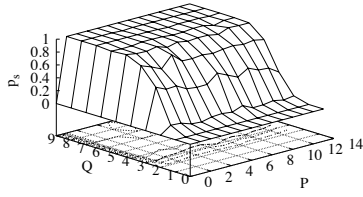
C. Impact of α

In order to optimize for the traffic the right choice of α is crucial for the NC approaches. While post-recoding is achieving the best success probability p_s , it comes at the cost of large traffic. Pre-recoding on the other side is producing less traffic but does not yield good performance in terms of cloud integrity. In this section results for different values of α are presented and compared with the post and the pre-recoding as given in Figure 4 (with a fixed value of $Q = 4$ and 10k rounds).

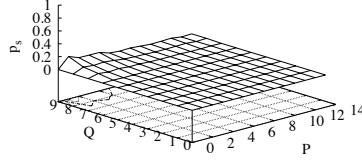
Pre-recoding approach yields a very bad performance with no chance of providing cloud integrity for any value of P . But increasing the traffic slightly (as given in Figure 2 Hyb (0)) achieves significantly better results, i.e. with seven parents $P = 7$ the cloud integrity could be achieved with $T = 7$. Hyb (1) and Hyb (2) are improving the situation even slightly requiring only six parents to achieve cloud integrity with $T = 7$ and $T = 8$, respectively. Hyb (3) is then performing as good as the post-recoding approach requiring only five parents to assure cloud integrity with traffic value $T = 8$, which is significant smaller than the full post-recoding approach with $T = 20$.

D. Impact of the field size

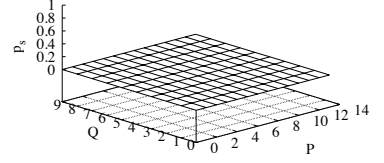
In this section we shortly underline the impact of the field size F . In Figure 5 the success probability p_s is given versus the number of parents for different field sizes for 10k rounds. While the large field sizes 2^8 , 2^{16} , and OPF ($2^{32} - 5$) are achieving nearly the same performance, the binary field size is significantly worse. Larger field sizes need five or more parents to restore the cloud information successfully, while the



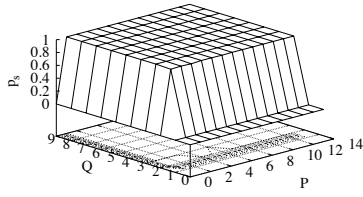
(a) NO CODING 10 rounds



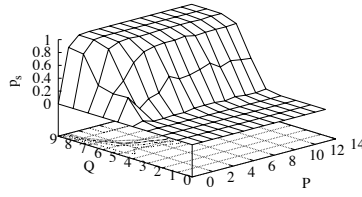
(b) NO CODING 100 rounds



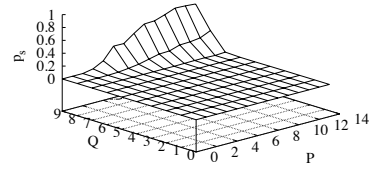
(c) NO CODING 1000 rounds



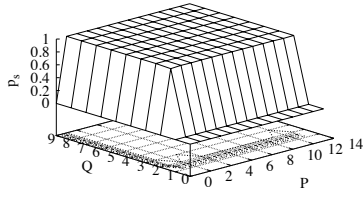
(d) REED SOLOMON 10 rounds



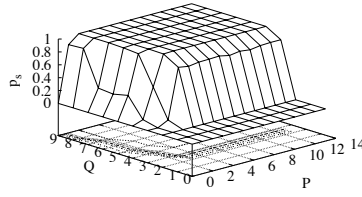
(e) REED SOLOMON 100 rounds



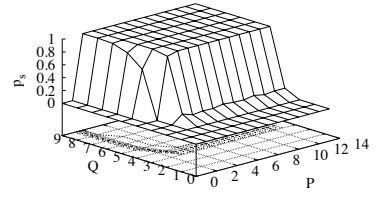
(f) REED SOLOMON 1000 rounds



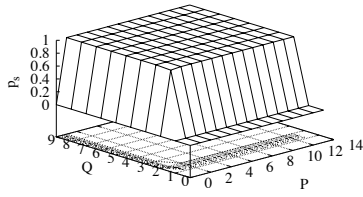
(g) NC PRE-RECODING 10 rounds



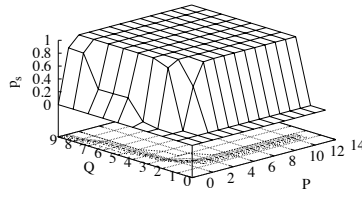
(h) NC PRE-RECODING 100 rounds



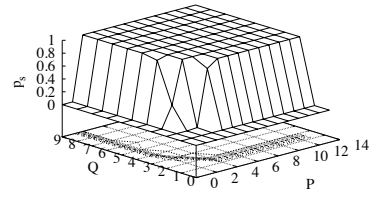
(i) NC PRE-RECODING 1000 rounds



(j) NC POST-RECODING 10 rounds



(k) NC POST-RECODING 100 rounds



(l) NC POST-RECODING 1000 rounds

Fig. 3. Success probability p_s to retrieve successfully the stored data in dependency of the number of parents P (equivalent to the cost in traffic) and the redundancy factor R (equivalent to the cost in storage).

binary approach would need more than ten parents in the given scenario. The reason to chose the binary field size anyway is based on the low complexity it provides [9].

VI. DISCUSSION

With respect to the overall traffic that is conveyed among the clouds, the uncoded, the random RS, and the pre-recoding are using the minimum amount of traffic, barely enough to

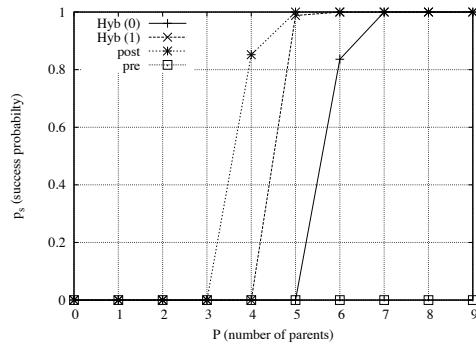


Fig. 4. Impact of α on the success probability p_s ($L=1$ $Q=3$ $F=2^8$ $G=15$).

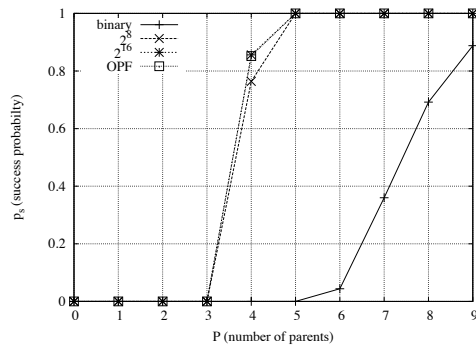


Fig. 5. Impact of the field size F on the success probability p_s ($L=1$ $Q=3$ $F=2^8$ $G=15$).

fill the new clouds. The post-recoding is using P -fold times more traffic, while the hybrid approach is using only α times more traffic than the bare minimum. The fully controlled RS approach is using also more than the bare minimum if we assume that $G \geq Q$ holds. In case of $G = Q$ we would just have a repetition code and we would not exploit the full benefits of a real distributed cloud. In [7] we have shown that distributing the data over four clouds lead to a speed up in accessibility of 50% and therefore Q should be always smaller than G .

The random linear network coding approaches better results for the cloud integrity. The uncoded approach yields unacceptable results, while the random RS approach at least is stable over the first rounds. The reason why the network coding approaches are so stable even after a large number of rounds is the recoding capability described in Section II. While the other approaches are creating redundancy by introducing copies, the network coding approach is *refreshing* the data in each round.

All discussed approach will yield better results if we allow an overlay control unit to check each round which packets have been stored and clean up the redundant information. Such an overlay approach was proposed in [5], but this will have

a negative impact in delay and imposes more traffic to the system.

VII. CONCLUSION

In this paper we introduced the implementation of a simulation tool for distributed storage investigating the dynamics of a distributed storage system. Applying several coding strategies a performance evaluation has been carried out showing the benefits of random linear network coding over traditional codes and uncoded approaches. For a dynamic setting it was shown that random linear network coding can preserve the cloud integration even for a large number of changes without any overlay entity in a distributed fashion. The paper presents an in-depth evaluation of the traffic and storage cost to preserve cloud integration. Results in this paper show that random linear network coding outperforms Reed–Solomon and uncoded approaches using a minimum in terms of traffic and storage. For the network coding approaches different strategies are investigated to optimize the cost of storage and traffic. Furthermore parameters related to network coding are investigated.

ACKNOWLEDGEMENT

This work was partly supported by desk.io GmbH and partially financed by the Green Mobile Cloud project granted by the Danish Council for Independent Research (Grant No. DFF-060201372B), by the European Union and the European Social Fund through project FuturICT.hu (Grant no. TAMOP-4.2.2.C-11/1/KONV-2012-0013) organized by VIKING Zrt. Balatonfüred, and the Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fund (grant no.: KMR_12-1-2012-0441).

REFERENCES

- [1] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," *j-SIGMOD*, vol. 17, no. 3, pp. 109–116, Sep. 1988.
- [2] A. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 111–117.
- [3] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *IEEE Int. Conf. on Computer Comm. (INFOCOM)*, 2007, pp. 2000–2008.
- [4] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.
- [5] W. Lei, Y. Yuwang, Z. Wei, and L. Wei, "Ncstorage: A prototype of network coding based distributed storage system," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 12, 2013. [Online]. Available: <http://iaesjournal.com/online/index.php/TELKOMNIKA/article/view/3709>
- [6] A. Le and A. Markopoulou, "Nc-audit: Auditing for network coding storage," in *Int. Symp. on Network Coding (NetCod)*, 2012, pp. 155–160.
- [7] M. Sipos, F. Fitzek, D. Lucani, and M. Pedersen, "Distributed cloud storage using network coding," in *IEEE Cons. Comm. and Netw. Conf.*, 2014, pp. 6–19.
- [8] M. V. Pedersen, J. Heide, and F. Fitzek, "Kodo: An Open and Research Oriented Network Coding Library," *Lecture Notes in Computer Science*, vol. 6827, pp. 145–152, 2011.
- [9] A. Paramanathan, M. Pedersen, D. Lucani, F. Fitzek, and M. Katz, "Lean and Mean: Network Coding for Commercial Devices," *IEEE Wireless Communication Magazine*, Dec. 2013.