Distributed Cloud Storage Using Network Coding

Sipos, Marton A.; Fitzek, Frank; Roetter, Daniel Enrique Lucani; Pedersen, Morten Videbæk

Published in:
IEEE Consumer Communications and Networking Conference (CCNC) 2014

DOI (link to publication from Publisher):
10.1109/CCNC.2014.7056318

Publication date:
2014

Document Version
Accepted author manuscript, peer reviewed version

Distributed Cloud Storage Using Network Coding

Márton Sipos[1,3], Frank H.P. Fitzek[2,3], Daniel E. Lucani[2] and Morten V. Pedersen[2]
[1]Budapest University of Technology and Economics, Budapest, Hungary
[2]Department of Electronic Systems, Aalborg University, Denmark
[3]desk.io GmbH, Germany
Email: sipos.marton@aut.bme.hu; {ff | del | mvp}@es.aau.dk

***Abstract*—Distributed storage is usually considered within a cloud provider to ensure availability and reliability of the data. However, the user is still directly dependent on the quality of a single system. It is also entrusting the service provider with large amounts of private data, which may be accessed by a successful attack to that cloud system or even be inspected by government agencies in some countries. This paper advocates a general framework for network coding enabled distributed storage over multiple commercial cloud solutions, such as, Dropbox, Box, Skydrive, and Google Drive, as a way to address these reliability and privacy issues. By means of theoretical analysis and real–life implementations, we show not only that our framework constitutes a viable solution to increase the reliability of stored data and to ensure data privacy, but it also provides a way to reduce the storage costs and to increase the download speed significantly. Our measurements show that the download time could be reduced up to six fold in some scenarios exploiting four commercial cloud solutions.**

I. Introduction and Motivation

Cloud solutions have become a commodity for commercial users over the last years. These services provide a wide range of advantages, from allowing users to manage files over different devices, to simplify collaboration among colleagues, and even to simply back up personal data. More recently, video streaming solutions are available as part of the cloud. These video services have also the need for fast delivery times, raising new challenges for each individual cloud. Each cloud thus aims to provide a service that guarantees that data be reliably stored, readily available, private, and at high data rate.

There are some issues with the single cloud approach. First, a cloud is still vulnerable to outage due to system overload or failure. Second, storing the entire data in a single cloud brings forth data privacy issues if the system is compromised by an attack or by governmental laws that force the provider to release a user's data. Spreading the content over multiple clouds would force attackers to compromise multiple clouds in order to gain access to the data. Thus, a solution that can harness the power of multiple commercial cloud storage systems to provide higher data rates, higher storage capacity, higher resiliency and data availability, and increased privacy is key to meeting end–users' increasing expectations of cloud services.

The main idea of this paper is to store data in a distributed fashion over multiple cloud providers. This should help to increase reliability and resolve the privacy issues to some extent. Additionally, using random linear network coding makes storage more efficient in terms of storage space and time to retrieve the distributed data.

Network coding for distributed storage was originally proposed in [1] in the context of sensor networks. It has a series of advantages over standard end–to–end coding, including the possibility to recode already encoded data without destroying the code properties in terms of complexity and efficiency. Since that initial work, research has focused strongly on code regeneration, e.g., [2], with [3] providing a comprehensive survey on the topic. Recently, an initial research implementation of distributed storage with network coding was reported in [4]. Research work has also focused on auditing systems for network coding storage, e.g., [5].

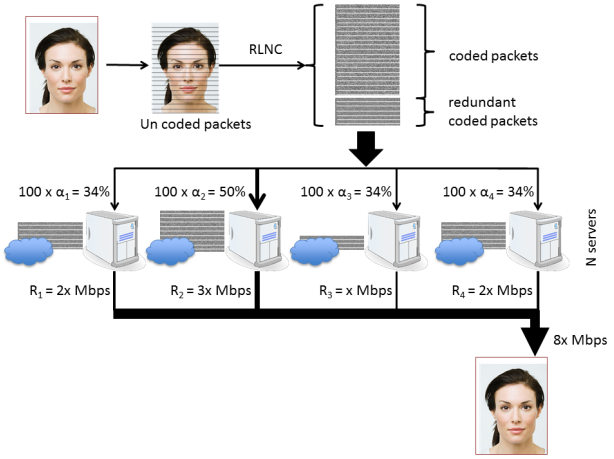

Fig. 1. Main idea of distributed clouds with network coding

This paper is inherently different from previous network coding research in the topic in terms of our goals. We are interested in exploiting commercial systems with inherently heterogeneous characteristics (e.g., round–trip delay, throughput, storage cost) to provide improved services to the end–user. The key is to exploit network coding to allocate resources in a flexible manner in the clouds to increase the download speed for end–users, while providing security guarantees. More specifically, our approach is to store data in a network coded fashion on distributed storage entities, e.g., Dropbox, Box, Skydrive, Google drive. Network coding stores different linear combinations of the original data in each cloud, which allows us to avoid an overwhelming book keeping process when downloading the data and also to avoid additional delays from a highly loaded cloud. As given in Figure 1 the original data, which is composed out of several uncoded packets, is coded and stored in the distributed clouds. As network coding is a rate–less code, more coded packets than uncoded can be generated. In the given example enough data is stored such that one cloud failure can be tolerated.

This paper presents detailed benefits for both the user and the cloud operator and mathematical analysis to support these intuitions. Then, we describe our demonstrator implementation and provide measurement results using real–life cloud

providers under a variety of scenarios.

II. Theoretical Analysis

In this section we derive performance measures for our approach. We consider a set of N cloud providers, $\mathcal{C}$, where $\mathcal{C} = \{c_1, c_2, ..., c_N\}$, where c_i represents the i–th cloud provider. We consider that a user, u, uses this set of clouds to upload and download data using network coding. The size of the file is F. The upload and download rate from the user u and the i–th cloud is $R_{(u,c_i)}$ and $R_{(c_i,u)}$, respectively.

A. Reliability

If we consider that a cloud provider, c_i, has a probability p_i of being unavailable (e.g., service is down, data is lost), then the probability of all data being unavailable to the user given that it has used T redundant clouds is

$$\mathcal{P}\left(T, N, \{p_i\}\right) = 1 - \sum_{a_1+...+a_N \geq N-T} \prod_{i=1}^{N} (1-p_i)^{a_i} p_i^{1-a_i}, \quad (1)$$

where $a_i \in \{0, 1\}$ indicates that cloud i is down or not for $a_i = 0$ and $a_i = 1$, respectively. For the case of $p_i = p, \forall i = 1, ..., N$, this simplifies to

$$\mathcal{P}(T, N, p) = 1 - \sum_{t=0}^{t=T} \binom{N}{t} \cdot (1-p)^{N-t} \cdot p^t. \quad (2)$$

For this case, we assume that each cloud contains a fraction of packets $P \geq 1/(N-T)$.

B. Download Speed

In order to maximize the download speed from the clouds, we need to consider that the amount of data stored need not be the same as provided by the reliability criteria. The latter is an indication of the minimum amount of data that can be stored, but it does not consider the benefits of downloading more data from faster clouds. We consider that each cloud c_i provides a download rate of $R_{(c_i,u)}$ to u and that F large enough so that the download time is much larger than the individual round trip times from each server. Our goal is then to request a fraction α_i from cloud c_i such that all clouds complete the delivery of the data simultaneously for the user. Thus, the fraction of the file requested from cloud c_i is

$$\alpha_i = \frac{R_{(c_i,u)}}{\sum_{i=1}^{N} R_{(c_i,u)}}. \quad (3)$$

If both reliability and download speed are important factors in the design, then the fraction of the data stored, P_i, stored in cloud c_i needs to be

$$P_i = \max\left(\frac{1}{N-T}, \alpha_i\right) = \max\left(\frac{1}{N-T}, \frac{R_{(c_i,u)}}{\sum_{i=1}^{N} R_{(c_i,u)}}\right). \quad (4)$$

If we consider there is asymmetric round trip delays until the packets are received, where D_i represents the round trip delay from cloud c_i, then

$$\alpha_i = \frac{R_{(c_i,u)}}{\sum_{i=1}^{N} R_{(c_i,u)}} \cdot \left(1 + \sum_{c_j \in \mathcal{C}} R_{(c_j,u)}(D_j - D_i)/F\right). \quad (5)$$

C. Privacy and Security

We can consider different conditions for data privacy. The most stringent is to force attackers to break in at least $N-T$ clouds to obtain the data assuming that the link to the user is encrypted during download/upload. This condition is of course fulfilled if we only store enough for guaranteeing reliability. If we store additional data for boosting download speed performance, then we need to set a different condition. A simple way of thinking about it is that any combination of the fractions of data of a subset of $N-T-1$ clouds cannot amount to 1, i.e., enough degrees of freedom to decode.

Of course, $P_i < 1/(N-T-1), \forall i$ is a sufficient, but not necessary condition to preserve privacy. A necessary condition for preserving privacy is stated in the following. Let us define $\mathbf{C}_m$ as the set of distinct subsets of $\mathcal{C}$ missing exactly m cloud elements each. For example, $\mathbf{C}_1 = \{\mathcal{C} \setminus c_1, \mathcal{C} \setminus c_2, ..., \mathcal{C} \setminus c_N\}$, and $\mathbf{C}_2 = \{\mathcal{C} \setminus \{c_1, c_2\}, \mathcal{C} \setminus \{c_1, c_3\}, ..., \mathcal{C} \setminus \{c_{N-1}, c_N\}\}$

Then, as long as

$$\max_{C \in \mathbf{C}_{T+1}} \sum_{c_k \in C} P_k < 1, \quad (6)$$

our condition for security is preserved. This provides an interesting result, namely, that there is room for providing higher download speeds without compromising security albeit with some additional storage cost. From a practical perspective, a simple approach to test the condition is to sort P_i's in increasing order. Then, we can add the $N-T-1$ greatest values. If the result is lower than 1, then privacy is preserved. Otherwise, it is compromised.

D. Storage Costs

For the case in which reliability is the only concern, the overall storage size, H, that is needed for the distributed approach when a file of size F bytes is used equals

$$H(N, T, F, P) = \frac{N}{N-T} \cdot F = F + F \cdot \frac{T}{N-T}, \quad (7)$$

where we assume that each cloud will contain a fraction $P = 1/(N-T)$ of the data. Clearly if $N >> T$, the storage cost becomes negligible.

If more factors are important, then

$$H(N, T, F, \{P_i\}) = \sum_{i=1}^{N} P_i \cdot F. \quad (8)$$

E. Overall costs

Let us consider that we have a storage cost of $H = \frac{N}{N-T} \cdot F$ and a cost of having the data unavailable U_c. The optimal choice for T will be given by the problem:

$$\min_T H(N, T, F, P) + U_c \cdot \mathcal{P}\left(T, N, \{p_i\}\right). \quad (9)$$

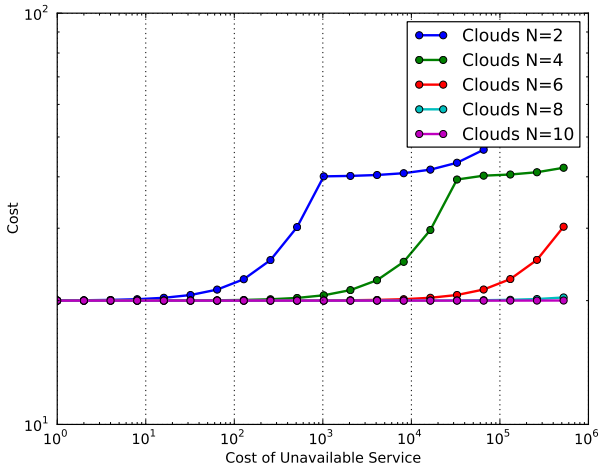

Fig. 2. Overall Cost

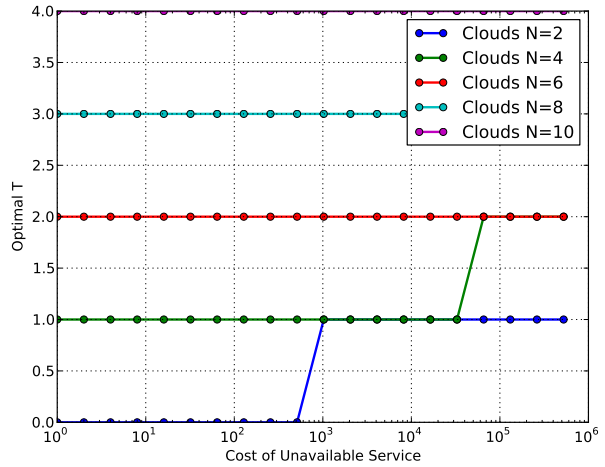

Fig. 3. Optimal number of redundant clouds

Figure 2 and Figure 3 show that the overall cost can be maintained low by using a higher number of cloud providers, even if the number of redundant ones is high. For example, $N = 10, T = 4$ provides not only a low overall cost but also allows for a configuration that is stable for a wider regime of U_c.

III. Cloud Testbed

A. Testbed Setup

To evaluate and showcase the previously presented analytic results, we have created an implementation to conduct measurements in a real–life environment. This subsection discusses the implementation details as well as the technical specifications of our testbed. The implementation was done in C++ using the KODO library [6] (fully fledged network coding software library) and the Qt framework. It was compiled with GCC 4.6.2 set to O2 level optimization. We chose four cloud storage services based on their market penetration: Box, Dropbox, Google Drive and Skydrive. All provide free storage space and have publicly available, well documented REST–based APIs.

Measurements were carried out in two locations, Hungary and Germany. The first represents an ideal scenario where the only limitation is the bandwidth provided by the cloud providers, the second is a more realistic small–office or home environment where the Internet connection is the bottleneck. Unless explicitly stated, the results show the measurements from Hungary, where we used a fiber optic network connection. Several provisional measurements were conducted using servers located in different cities in Europe (Amsterdam, Rome, Madrid, Paris, London, Oslo). This was necessary because the location of the servers used by the cloud services has not been disclosed in every case. Ping times were under 55ms and download bandwidth over 94Mbps in all cases. In Germany we used an ADSL connection, which limited our achieved results. Bandwidth was over 24 Mbps, ping under 45ms. In both cases a machine equipped with an AMD Athlon II X3 450 CPU at 3.2GHz, 4GB of RAM and a clean installation of Windows 7 was used.

B. Measurement Campaign and Metric

We distributed and retrieved a 16MB file containing random data. First, a simple replication based approach was examined, then one based on random linear network coding (RLNC) [7]. A subset of packets covering the entire data was distributed to all providers in both cases. For the RLNC approach, we used a generation size of 32 and operated over GF(2^8), resulting in packets of size 512k (and a few extra bytes overhead for the coding coefficients). Decoding was done once enough packages were downloaded. For the non–encoded replication–based approach, we used the same packet size of 512k. Each provider was artificially limited in the number of parallel downloads it could sustain. This constraint was achieved by using a sliding window–like technique. Initially, a set number of packets was requested from each provider with a new request being issued once one was served. We decided to use this simple retrieval technique, because in theory this ensured that download times were optimal in case an approach using network coding was employed. This results from the fact that every provider is used at its maximum capacity all the time. Each measurement was repeated one hundred times.

We employed several metrics, the two more important ones are presented here: the time taken to retrieve a 16MB file and the number of useful packets received from each provider. The first one is the variable we chose to optimize for, whilst the second provides insight into the processes taking place. Our goal is two-fold, show that distributing data to several cloud storage providers shortens retrieval time and that employing random linear network coding provides consistent results in a changing environment.

C. Testbed Results

In the following we present the testbed results for the download time and the number of redundant packets. By presenting the results we also motivate the use of network coding and discuss its complexity.

1) Download Time: We begin with the measurements of the download times for the individual clouds. We do not wish to disclose which provider is which, so we shall refer

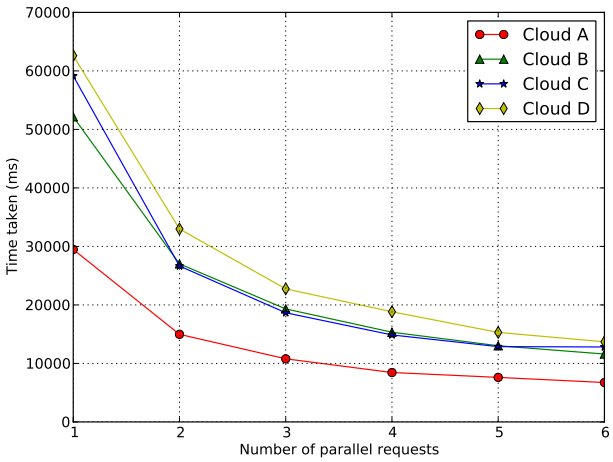

Fig. 4. Download time for individual cloud providers

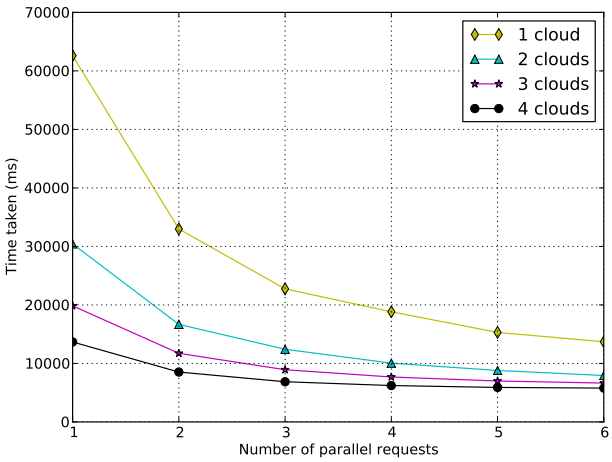

Fig. 5. Download time when using multiple cloud providers

to them as Cloud A, Cloud B, Cloud C and Cloud D. This assessment is independent to a certain degree from network coding. However, as detailed in Paragraph III-C3, network coding is needed to achieve these results consistently in a real-world environment, where the speed of the individual clouds varies.

An improvement proportional to $1/n$ can be observed in Figure 4 as the number of parallel downloads increases. This is as expected assuming that the providers do not artificially throttle the requests. The curve flattens around six, so there is next to no incentive to use more than six parallel downloads per provider. Another observation is that Cloud A is significantly faster then the other three regardless of the limitation on the number of parallel downloads. This shows that not all clouds have the same characteristics and explains the later gains of network coding that we see throughout the paper.

We continue by showing the benefit of increasing the number of clouds in a distributed storage system in terms of retrieval time. An important aspect is that the limitation on the number of parallel downloads is per provider, so using several of them means that the actual number of parallel connections in the system is multiplied by the number of providers. This is one of the reasons why employing more providers gives a clear advantage over the single cloud solution.

Figure 5 shows the measured gains when using more than one cloud service to retrieve files. Note, Cloud A is significantly faster than the other three clouds, the limitations on available storage space would still mean in practice that a multi–cloud approach is needed to achieve a certain download time given an amount of storage space and a required level of reliability. Subsection II-B gives a closed formula for calculating the optimal fraction of data to be distributed to each cloud under such heterogeneous conditions. This basically means that the faster clouds need to be able to provide proportionally more storage space. Figure 6 illustrates this by showing the time in milliseconds (ms) when each encoded packet was received and processed. In this example six parallel requests per provider were allowed and decoding finished at 5660ms. Cloud A has a significantly shorter round–trip time when retrieving the data compared to the other providers and

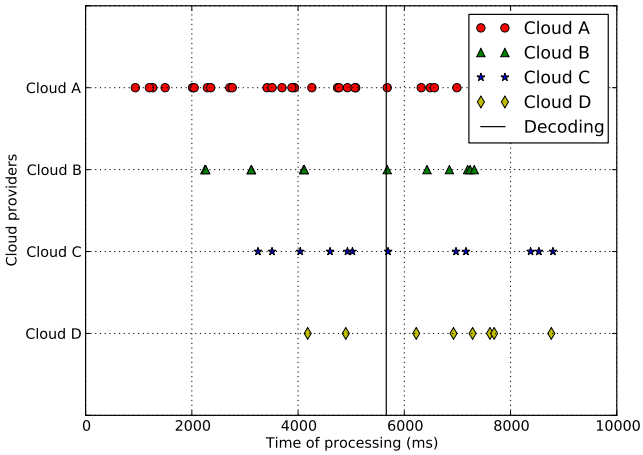

Fig. 6. Packet processing time

manages to download significantly more useful packages than the other providers. If less packets were stored on Cloud A, decoding would have been delayed. Also interesting is the differing characteristics of the individual clouds. Even though Cloud B and Cloud C generally manage to download the same amount of packages, the arrival pattern shows great differences, perhaps due to the method used to process the requests by the providers.

2) Number of redundant packets: Figure 6 also illustrates the disadvantage of using a higher number of parallel downloads. Therefore, distributed clouds have an advantage over single clouds, but they come with a price. At the time of decoding, there are $k-1$ packets in the download window for the provider that received the last useful packet and k for all others. These extra packets arrive after decoding is complete and therefore must be discarded. Generally, if we consider a set of N clouds with k parallel downloads per provider, the number of redundant packets is

$$n_{red} \leq N \cdot k - 1. \tag{10}$$

As shown in Figure 7, this upper bound is in fact achieved in practice in every case. This should be considered a worst–case

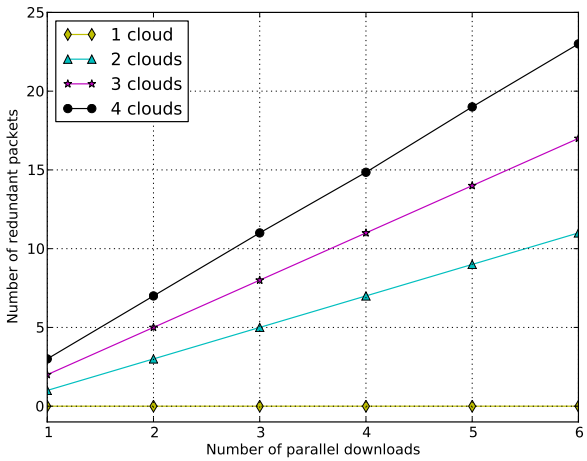

Fig. 7. Number of redundant packets

scenario, it happened because there was no limitation placed on the stored data, each cloud contained packets with data covering the entire file. If such a limitation was in place, the number of redundant packets would have been less, because once all the packets from a provider had been requested, the size of the download window for that provider would have decreased with each response.

To minimize the number of redundant packets necessitates either a smarter scheduling algorithm, which tries to predict how many packets will be received from each provider beforehand and makes the requests accordingly or a reduction in the number of parallel downloads. Considering the download times presented in Figure 5, a strong case can be made for using no more than two or three parallel downloads, as using more has a negligible effect on decreasing the download speed whilst having a great effect on the increase of redundantly downloaded packets. As given in Equation 10, redundancy grows linearly with the number of clouds as well.

3) Motivation for Network Coding: We examined the number of packets received from each provider as we have shown this should be a primary factor when determining the distribution of packets among the clouds. Providers that are able to supply packets faster should get a bigger cut to minimize the required download time, as discussed in Subsection II-B. We have found that there is a significant variance in the distribution of the number of useful packets retrieved from each provider as shown in Figure 8. As previously discussed in Paragraph III-C1, Cloud A retrieves most packages before the others. Even so, all four clouds still show a significant spread. The same is true when using two or three clouds, but in this case the differing distributions of the providers are even better visible. This is an indication that even during error free operation, the ratio of the download times of the providers changes, necessitating a distribution and retrieval schema that is able to adapt to the dynamically changing conditions. When considering a conventional non-encoded approach, it is imperative that the algorithm doing the distribution and retrieval take into account what data is stored where. The location of each packet in the original file should also be stored. It is easy to see the advantage of an approach that employs network coding,

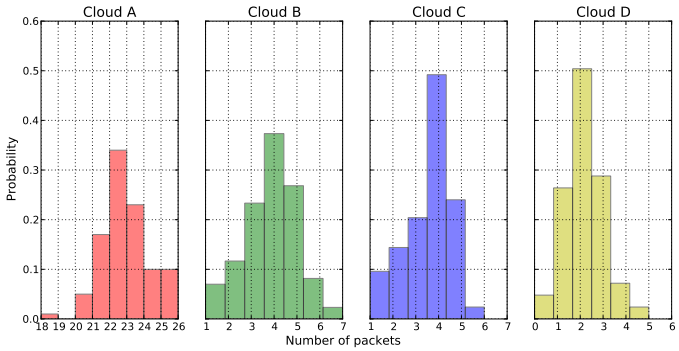

Fig. 8. Distribution of received packets when using four clouds

since there is no need for this bookkeeping. It is not important which packet is placed on which cloud, only the number of packets on each cloud. This simplicity ensures that as long as the distribution of the packets is proportional to the ratio of the download speeds of the providers, the time taken to download a file will be optimal.

The same applies when devising a scheduling schema for the packets. With a non–encoded approach care must be taken to not download the same packet from different providers. Furthermore, if a packet takes significantly longer to download than is expected, then perhaps this rule should no longer be applied and the packet should be downloaded from a different provider in order to keep overall download time as low as possible. This however leads to more redundant information traveling on the network and there is no guarantee that the original packet does not arrive in the meantime. It is also not obvious which provider should be used to make the extra request. Generally, download scheduling algorithms for the non–encoded approach can be devised which perform well in more or less static environments. However, it is more difficult to create one solutions which adapts well to changing conditions. We have used an algorithm that tries to place the emphasis on making the clouds race against each other by having them start at different points in the original file. For example for two clouds, the first starts from the beginning and goes forward, the second starts from the end and goes backward. For a larger number of clouds, multiple starting locations are defined. However, we have not dealt with packets taking longer to download than usual, as it is not clear what is the best approach, even whether new requests lower retrieval time or actually introduce an additional delay into the system. With a network coded approach there are no duplicate packets, each contains unique, useful data (if we assume all packets are linearly independent) and the order with which packets arrive is irrelevant. This enables the network coded approach to outperform a non–encoded one in most cases as shown in Figure 9 while being more robust and simple at the same time.

When employing a single cloud a replication based approach is slightly faster as it does not have a decoding overhead. However, once at least two clouds are used, in average the network coded approach achieves shorter download times.

4) Complexity: We have shown that employing network coding can achieve a significant gain in the required time taken to retrieve the original file. However, it is important to

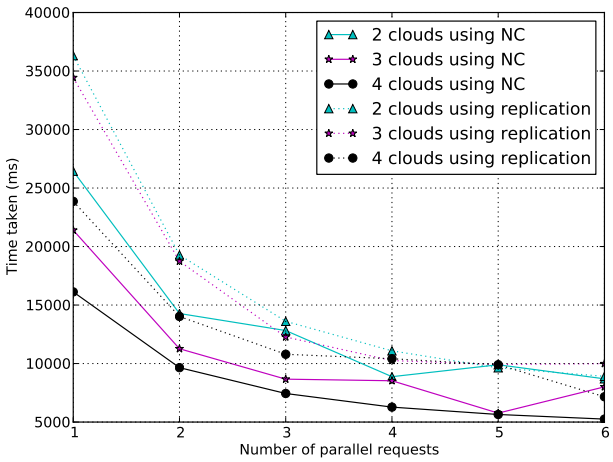

Fig. 9. Comparison of download times when employing network coding versus a replication based approach

note that decoding adds an extra time overhead. Measurements show that this extra time is around 657ms for the given configuration. This can be lowered by opting for a smaller field size or symbol size. For example, in the case of GF(2) the decoding overhead drops to less than 100ms, but the number of linearly dependent packages rises, which in turn delays the time when decoding is finished. Minimizing the decoding overhead and network usage is a trade-off that needs to be considered for any given scenario. Even though we have an extra delay in decoding, there is the benefit that we will download a minimum number of packets. Without network coding there is no decoding delay but a larger number of packets will be needed.

IV. User and Cloud Operator Benefits

From the user's point of view, the distributed approach increases data availability and data privacy. Furthermore in delay sensitive use cases, the user will experience faster reaction times of the cloud. A commercial aspect is that users have limited free storage by several cloud providers. The distributed approach allows those users to accumulate several *small* clouds into one large overlay cloud. Even commercial cloud users can benefit from the distributed cloud approach. Current cloud approaches try to *lock in* users by making it expensive to move data away from one cloud provider to another one. This makes it hard for users to change cloud providers. Using network coding allows to build up new cloud storage until the more expensive provider can be fully switched off.

The cloud operator has the possibility to move around data in the cloud more conveniently. Network coding, as no other coding technique nowadays, has the possibility to derive extra redundancy on the fly in a very efficient way. In case the data is distributed over several storage entities in a network coded fashion, each of the entities, or even a subset of the entities, can build up more redundancy if the system wants to do that, without contacting all other entities as required by end to end coding schemes. Such a flexible scheme enables the cloud providers to allocate the cloud space more efficiently, e.g. moving highly requested data locally closer to the requesting communication nodes as well as deleting information not needed.

V. Conclusion

In this paper we introduced a general framework for network coding enabled distributed storage over multiple commercial cloud solutions. Our implementation using commercial cloud solutions (Dropbox, Box, Skydrive, and Google Drive) showed that from a user's perspective the distributed cloud in combination with network coding has advantages in terms of costs, reliability, download speed and privacy. The paper shows that the distributed approach alone has benefits over a single cloud approach and that network coding improves performance even more. The extra overhead in order to achieve additional reliability is small and nearly vanishes if more parallel instances are available. In our test bed we have also shown the improvement in the download speed by a factor of six using well–known cloud solutions, operating over them as an overlay cloud. Beside the advantages for the user, the benefits for the network operator have been listed, such as the possibility to generate or remove extra redundancy at edge caches whenever content becomes more or less requested. The advantage of network coding enabled distributed clouds is that each cloud is used at its maximum speed even in dynamically changing environments. Additionally, in order to perform the generation or deletion of extra information, network coding is able to do so even if there is only access to a subset of the stored data.

Acknowledgment

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. 10-081621), 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] 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.

[2] 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.

[3] 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.

[4] 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

[5] A. Le and A. Markopoulou, "Nc-audit: Auditing for network coding storage," in *Int. Symp. on Network Coding (NetCod)*, 2012, pp. 155–160.

[6] Steinwurf ApS. (2012) Kodo Git repository on GitHub. [Online]. Available: http://github.com/steinwurf/kodo

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