Draft Submission to... **XX,** X–XX *Cambridge University Press, 2016*
doi:doi:10.1017/XXXX

The Fault Tolerant Distributed Data Caching for Mobile Devices using LDPC Codes

*

As the use of mobile data has increased tremendously and become common place, the overhead of base stations in communicating mobile data has reached enormous levels overlaoding the underlying infrastructure. In order to reduce this overhead, inter-cell communication between the devices of the same cell can be performed with the distributed data caching. Nevertheless, due to departure of old nodes and arrival of the new nodes, the data stored in the outgoing node may lead to permenant data loss in a distributed data caching context. In this paper, we propose and analyze the use of LDPC codes as a viable fault tolerance method. We compare the LDPC codes with the well known other techniques such as replication and Reed-Solomon coding within the context of distributed data caching.

Introduction

Device to device (D2D) communication can be used as an alternative way for retrieving data from a base station (**Wang and Golrezai**; Han and Ansari 2014). In this communication model, devices can communicate directly with each other without any base station intervention. D2D technology not only decreases the requirement of communication between the base station and nodes, but also reduces the access time for retrieving popular content. In this way, whenever a mobile device wants to retrieve content, it should first request it from the nearby devices through D2D communication. Only in the case of content unavailability, nodes request the content from the base station. On the other hand, if the content is available in the nearby devices, the content can be accessed without contacting the base station. The requester device can also cache the content for further requests coming from other nodes in the same cell at the expense of using some space in their local storage.

In the context of D2D communication, the data stored in the device can be considered lost whenever that device leaves the cell. One way to prevent this undesirable situation is the addition of controlled redundancy to the cached content. This redundancy can be obtained by using several methods such as replication (copy), MDS erasure codes, locally repairable codes, etc.

Several fault tolerant schemes for the distributed data caching have been considered in recent studies (Pääkkönen, Hollanti, and Tirkkonen 2015; Pedersen et al. 2016). The use of regenerating codes as a fault tolerance mechanism for distributed caching is proposed in (Pääkkönen, Hollanti, and Tirkkonen 2015). The immediate repair of the lost content is considered. The theoretical results indicate that the coded caching reduces the communication cost only if the popularity of the lost data is in a specific range (Pääkkönen, Hollanti, and Tirkkonen 2015).

It has been stated that using "lazy repair" method can decrease the cost of node repair compared to the case where the repairs are employed only by BS in (Pedersen et al. 2016). Further, they showed that MDS codes repair the content less costly than the other well-known techniques which are tailor-made for node repair process in distributed storage systems.

*

In this paper, we consider the use of Low Density Parity Check (LDPC) codes as a fault tolerance method for distributed data caching. LDPC codes are compared with other methods with various code rates and redundancy levels. LDPC codes are linear block codes and they are first proposed by Galleger in his Ph.D. thesis (Gallager 1963). The parity check matrix of these codes has a sparse structure. These codes are usually expressed by bipartite graphs consisting of two types of nodes check nodes and variable nodes. The variable nodes contain original data whereas the check nodes contain the sum of the selected variable nodes according to the corresponding row of the parity check matrix. These codes are essentially constructed to sustain reliable communication on a erroneous channel. Then, they have found new application areas such as a redundancy method for distributed storage systems in different studies (Plank and Thomason 2003; Gaidioz, Koblitz, and Santos 2007; Wei et al. 2014; Yongmei, Fengmin, and Cher 2015).

In erasure codes, k symbols are coded and extenden to n symbols. So, From a distributed storage perspective, the distinctive feature of LDPC codes is that increasing n and k while keeping the ratio constant does not affect the decoding complexity (Yongmei, Fengmin, and Cher 2015). In that respect, these codes are suitable for dynamic systems where the number of incoming and outgoing nodes are not predictable as in the cellular networks. Thus, by exploiting this feature, the number of n and k can be changed in accordance with network size. (Same way, how about fountain codes? They can also scale like that with their generator matrices?)

(It is true, but adding these codes' results will be exhausting in this phase, how about considering them in the next studies maybe on "the code parameter switching".)

In distributed storage systems, node repair becomes infeasible when there are insufficient number of symbols for the repair. Fortunately, in the cellular networks with the help of the base station node repair could still be possible even if there exist insufficient data within the operating cell. The node repair in LDPC code can be performed in multiple ways (Yongmei, Fengmin, and Cher 2015). The first approach is based on repairing symbols one by one while the other approach considers repairing multiple symbols all at the same time. In essence, both of the methods utilize the belief propagation decoding approach. The main obstacle of this approach is that the decoding process fails if any of the required symbols is not stored in available nodes. This obstacle can be prevented through the assitance of the base station by sacrificing the communication cost.

In this paper, we investigate the encoding and node repair cost of the distributed data caching system for mobile networks when LDPC codes are used as a redundancy scheme. Furthermore, we compare the performance of the two different node repair techniques of LDPC codes. The cost is defined regarding time; the number of symbols downloaded from nodes and base station. We find out that increasing the code rate reduces the success probability of the multiple symbol repair method whereas increases the communication cost. Moreover, LDPC codes are compared with Reed Solomon code and 2-way-replication techniques through Monte Carlo simulations.

LDPC Repair Process

In this study, different repair approaches are proposed for repair processes. Different from the classical distributed storge systems, available required symbols are retrieved from the available helper nodes in the cell, and the remaining required symbols are retrieved from the base station for both methods in these approaches. (Is this some differentiating scheme compared to these studies? In other words, do they not perform the repair operations the same way we do it here? Need to look at the paper in detail.)

Yes, we have different repair techniques since we have an alternative way of communication using BS. On the other hand, the use of recovery equations idea and the Equation (1) used in Multiple repairs are the same .

Multiple nodes may leave the cell in a δ period, so the symbols stored in the leaving nodes should be repaired. In the first approach, the lost symbols are repaired one by one (Wei et al. 2014). In the second approach, the lost symbols are repaired all at once (Yongmei, Fengmin, and Cher 2015).

As a strategy, we prefer the multiple repair method whenever there are more than one failure. If our attempt for multiple and simultaneous recovery fails, the method of single repair at a time is employed. In the following subsections, these methods are explained by emphasizing the differences due to different settings of the cellular networks.

Single Repair

This repair method is based on the recovery equations. The recovery equations are determined as follows. A row of the parity check matrix is selected where the column corresponding to the lost symbol value is one. Preferably, the row with the minimum row weight is chosen in order to use low bandwidth. Then, the symbols corresponding to ones in the row except the lost symbol are downloaded from the related nodes. Since the nodes store multiple symbols, the departure of a single node may lead to the loss of multiple symbols at the same time. Therefore, in this setting, the other symbols may also be lost in the recovery equation that is used to repair the lost symbol. Thus, in this study, the recovery equation is constructed according to using the recovery equation having either minimum term or the requiring minimum BS communication. In the single repair approach lost symbols are repaired one by one using the related recovery equations.

(But we assumed single node departure (loss). Do you mean that the departed node leads to only and only one symbol loss in a given recovery equation? and What if we cannot find any row with this constraint? - Use BS?)

Maybe the expression "Single Repair" leads to a misunderstanding. Nodes can store multiple symbols. No, in this setting multiple symbols are repaired using recovery equations e.g. $R_1 = S4 + S5 + S7$ for S_1, R_3 for S_9, etc. Assume that R_1 is used for repairing $S1$; however, S5 is a symbol is also lost because of some other nodes' departure.

We propose two different single repair algorithms. The first one uses a greedy approach while the other uses a randomized approach to provide lower BS communication. Initially, both algorithms attempt to repair lost symbols without the base station as much as possible. But, algorithms differ in the way of repairing lost symbols involving base station interaction.

In the first algorithm, when neither of the lost symbols is repaired using a recovery equation that do not required the base station communication, one of the lost symbols is repaired which requires the minimum base station contact. In this case, this symbol can be used as a helper symbol in the further repairs. So, if at time t, there is no repair without the base station intervention can be performed, the next repair may be realized without using the base station at time $t + 1$ using the previously repaired symbol as the helper node. Every lost symbol is first attempted to be repaired without using any base station communication.

(But How are we going to judiciously choose that lost symbol to make sure it will help others? Perhaps an algorithm?)

(I elaborate the operations in Algorithm 1 and also in below, the explanation of the Algorithm 1)

The first algorithm is given in Algorithm (1). It presents a method to repair the lost symbols one by one. This algorithm uses a set called $\mathcal{L}$ for storing the lost symbols. The algorithm starts with generating recovery equations using parity check matrix for all of the lost symbols then these recovery equations are represented by two sets called R_1 and R_2 according to whether an equation contains any lost symbol or not. Namely, R_1 stores the equation set that does not involve any BS communication while R_2 stores equations requiring BS communication. An equation in R_2 consists of the indices of the symbols contained in the equation. In this algorithm, R_2 contains at most one equation for any lost symbol. The sets R_1 and R_2 have an increasing order with regarding the number of symbols involved in the equations and the number of symbols required to be gathered from the base station respectively.

Moreover, the algorithm uses a function $\mathcal{F}(X)$ which is defined for mapping a recovery equation to the lost symbol that it repairs. The domain of this function is the set of recovery equations, and the codomain is the set of the lost symbols. The algorithm starts repairing symbols using R_1 set. In this phase, when a lost symbol is repaired it removed from lost symbols set $\mathcal{L}$ as well as equations in R_1 and R_2 where takes part as a term. These operations are repeated until no equation left in R_1. After this phase, the lost symbols are repaired using the equations in R_2. Here equations may have also lost symbols. These symbols are gathered from BS. The second phase of the algorithm consists of these operations. As far as, there is any symbol in $\mathcal{L}$, the next lowest BS communication required equation is used from the R_2 set. Update operations over the sets R_1 and R_2 is realized with preserving their increasing order property.

Actually, the order of using recovery equations affects the number of retrieving symbols from the base station communication. Algorithm (1) represents a greedy approach by ordering the equations respect to the number of lost symbols which they contain. However, this approach is not guaranteed the optimal solution for the minimum overall base station communication. The problem of finding the order of recovery equations which results the minimum base station communication is an NP-Hard problem. This problem is similar to the Travelling Salesperson Problem. The minimum cost path should be found as in the TSP, but here the edge weights change dynamically. The more detailed description of the problem can be found in Appendix A.

To further reduce the BS communication, we propose Algorithm (2) using a randomized approach. The first phases of the both algorithms are the same. The first phase of the Algorithm (1) and Algorithm (2) are the same. After this phase, in Algorithm (2), a new set called $\overset{\triangle}{R_2}$ is constructed. $\overset{\triangle}{R_2}$ contains ϕ equation tuples having low score values among set R_2. Each tuple is consisted of t equations. These tuples have costs according to their BS communication need. Generating of these tuples is conducted using a genetic algorithm. Tuples having two equations ara generated at first and then using these tuples, tuples having four equations are constructed, this process continues up to generating tuples having t equations.

Initially, R_2 contains the same equations as in the Algorithm (1). At the beginning of the iteration, R_2 is copied to χ. The tuples in $\overset{\triangle}{R_2}$ has an increasing order. The algorithm is continued as follows.

A random integer, $index$, is generated proportionally to the score of equations in R_2 from the interval $[1, |R_2| - t]$. Then, one tuple is selected from $\overset{\triangle}{R_2}$ with a probality that is inverse proportional to their scores. The scores of the tuples are updated on each iteration as $scores_{t_0}$+$R_2 - R_2 \setminus \{R_2[index], R_2[index+1], ..., R_2[index+t]\}$. On each iteration, equations $\{\chi[index], \chi[index+1], ..., \chi[index+t]\}$ is removed and a selected tuple $\overset{\triangle}{R_2}[index'...index'+t]$ takes their place. If the score of χ is smaller then the score of R_2, R_2 is updated with the content

Algorithm 1 Single Repair Algorithm V1

input $\mathcal{L}, \mathcal{R}_1, \mathcal{R}_2$

output $score$, C

function FirstPhase($\mathcal{L}$, $\mathcal{R}_1, \mathcal{R}_2$)
while $\mathcal{R}_1 \neq \emptyset$ **do**
    $R \leftarrow R_1[1]$
    repair node $\mathcal{F}(\mathcal{R})$ using R
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{F}(R)$
    for each $r \in \mathcal{R}_1 \wedge \mathcal{F}(\mathcal{R}) \in r$ **do**
        $r \leftarrow r \setminus \mathcal{F}(R)$
    end for
    for each $r \in \mathcal{R}_\in \wedge \mathcal{F}(\mathcal{R}) \in r$ **do**
        $r \leftarrow r \setminus \mathcal{F}(\mathcal{R})$
        if $r \cap \mathcal{L} = \emptyset$ **then**
            $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \setminus r$
            $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup r$
        end if
    end for
    $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \setminus R$
end while
function SecondPhase($\mathcal{L}$, $\mathcal{R}_2$)
while $\mathcal{L} \neq \emptyset$ **do**
    $R \leftarrow \mathcal{R}_2[1]$
    download $R \cap \mathcal{L}$ from BS
    $score = score + |R \cap \mathcal{L}|$
    $C = C \cup score$
    repair $\mathcal{F}(\mathcal{R})$
    for each $r \in \mathcal{R}_2$ **do**
        $r \leftarrow r \setminus \{\mathcal{F}(\mathcal{R}) \cup \{\mathcal{L} \cap R\}\}$
    end for
end while
return score =0

of χ. In every iteration the score of the χ is tried to be decreased. After maxIter iterations the first $|L|$ equations are used for repairing symbols in L.

Algorithm 2 Single Repair Algorithm V2

input $\mathcal{L}$, $\mathcal{R}_1, \mathcal{R}_2$, $\overset{\triangle}{R}_2, maxIter, t$

1: **function SingleRepairV2**($\mathcal{L}$, $\mathcal{R}_1, \mathcal{R}_2, t$)
2: $bestScore \leftarrow$ **FirstPhase**($\mathcal{L}$, $\mathcal{R}_1$, $\mathcal{R}_2$) ikinci aşamada çalıştırmadan maliyeti ne kedar hesaplıyacağız
3: construct $\overset{\triangle}{R}_2$ as containing best ϕ t-mers
4: **while** $it < maxIter$ **do**
5: $\chi \leftarrow R_2$
6: $index \leftarrow$ generate a biased random integer from the interval $[1, |R_2| - t]$ proportionally to the scores
7: $index' \leftarrow$ generate a biased random integer from the interval $[1, |\overset{\triangle}{R}_2|]$ inverse proportionally to the scores
8: $\chi[index...index + t] \leftarrow \overset{\triangle}{R}_2[index'...index' + t] \setminus \chi[index...index + t]$
9: $newScore \leftarrow score(\chi)$
10: **if** newScore<bestScore **then**
11: $bestScore \leftarrow newScore$
12: $R_2 \leftarrow \chi$
13: $\overset{\triangle}{R}_2 \leftarrow \overset{\triangle}{R}_2 \setminus \overset{\triangle}{R}_2[index']$
14: **end if**
15: $it \leftarrow it + 1$
16: **end while**
17: return R_2 =0

We may add/try two other algorithms. The third can be based on totally genetic algorithm i.e. $t = |\mathcal{L}|$, The fourth one, we may modify Algorithm 2 as an adaptive version of it as follows: t is started with a small value and in every iteration if it succeeds (if if condisiton is held in line 10), t is increased.

(I assume above expression is not complete? - Is this what has been proposed in Wei, 2014 or in Yongmei,2015?, or something we are proposing?)

(Like I said in the e-mail, initially, it was only AlgorithmV1. Now, both of the algorithms are added.)

Multiple Repair

In this repair method, many lost symbols can be repaired at once. This method is based on the following equation proposed by Yongmei, Fengmin, and Cher 2015:

$$\overline{d_{lost}} = h^{-1} g \overline{d_{required}} \tag{1}$$

In the Equaton 1, $\overline{d_{lost}}$ represents the vector of the lost symbols, $\overline{d_{required}}$ represents the symbols required from the helper nodes, g stands for the generator matrix of the helper symbols

and h is the sub matrix of the parity check matrix which contain one(s) in the diagonal element of the columns of the lost symbols and this matrix has to be invertible.

(I think we may need to put more information here.)

(Exactly, the degraded :) performance of multiple repair was detained me from completing the section. I was searching different ways to improve its communication cost. Maybe array LDPC codes will be helpful.)

Simulation Settings

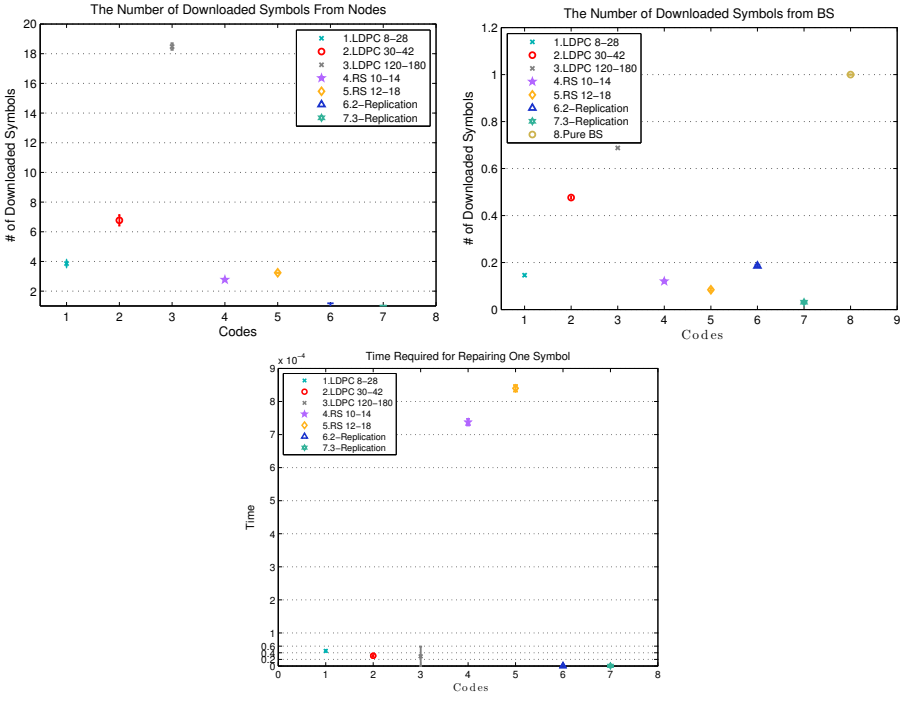

Figure 1: Simulation Results (What does it mean to download 0.2 symbols from the BS? Shouldn't it be an integer? Do you mean average? - Also should we include other modern codes? such as MSR, MBR, cooperative versions? LRCs?)

(For the comment in the caption - 0.2 is the value the average number of symbols downloaded from BS for repairing only one symbol. For example, we can get 8 symbols from the other nodes and 1 symbol from the BS for repairing one symbol. - Also should we include other modern codes? such as MSR, MBR, cooperative versions? LRCs? -Actually, it would be better. But, I guess it will be very time consuming. In case of adding these codes extends the scope of the paper. In this case this study may be turned into a journal paper, of course if you agree :)) Initially, a file of F bytes is considered to be cached among some storage nodes in the cell. The file is partitioned into stripes and each stripe includes k packets, called symbols of size 1 byte and these packets are encoded into n coded symbols such that the code rate is $\frac{k}{n}$. Then the encoded data is distributed into m nodes in the cell at an equal rate so that these m nodes are called as

storage nodes. This process is repeated for all stripes. All this encoded data of this raw F bytes is stored in the base station in the cell. There are also empty nodes that have no data in the cell. We assume that each new node arriving in the cell is also an empty node.

Each storage node stores exactly the same number of bytes with each other (Pedersen et al., 2016). In other words, no residual data is used in this system model.

Arrival-Departure Model

Nodes arrive into the cell at rate λ (arrival rate) according to a Poisson process with independent and identically distributed (i.i.d.) exponential random inter-arrival times given by the probability density function (pdf)

$$f_X(x) = \lambda e^{-\lambda x} \tag{2}$$

and similarly, nodes leave the cell at rate μ (departure rate) with i.i.d. exponential random inter-departure times given by the pdf

$$f_Y(y) = \mu e^{-\mu y} \tag{3}$$

(This also looked incomplete. I filled the empty spots a bit.)

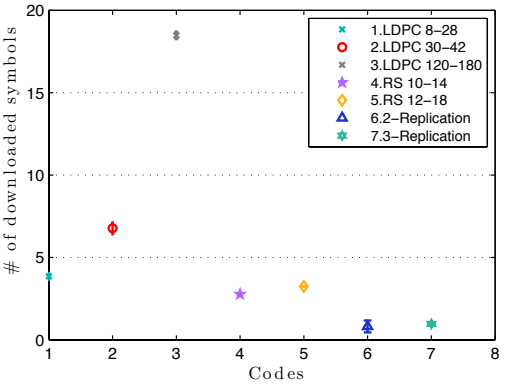

Figure 2: Number of Symbols Downloaded from Nodes for Repairing One Symbol

EXAMPLE SCENARIO

- $M/M/\infty$ model
- arrival rate
- departure rate
- lazy repair
- 30/42 (LDPC için)
  - $W_C = 4$

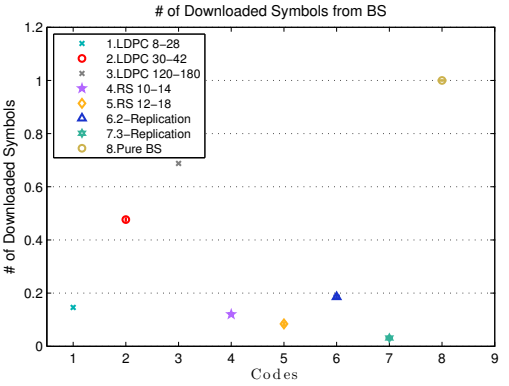

Figure 3: Number of symbols downloaded from Base Station for Repairing One Symbol

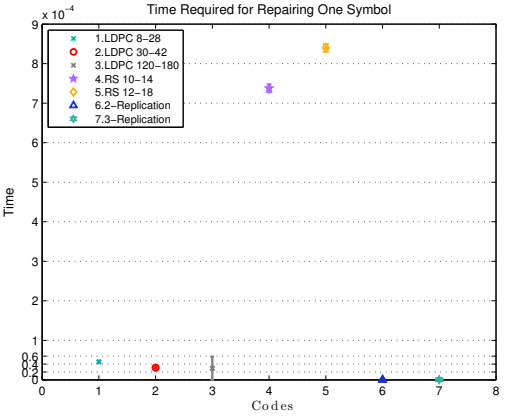

Figure 4: Elapsed Time for Repairing One Symbol

  - W_R = 14
  - n = 42 (low-density'i korumak için 3 katı aldım)
  - initial storage node sayısı = 7
  - toplam başlangıç node sayısı = 42
  - bu durumda H matrisi boyutu = 12x42
  - bu durumda G matrisi boyutu = 30x42
  - her stripe içinde node başına 6 sembol düşüyor

- 120/180 (LDPC için)
  - W_C = 20
  - W_R = 60
  - n = 180 (low-density'i korumak için 3 katı aldım)
  - initial storage node sayısı = 10

  - toplam başlangıç node sayısı = 50
  - bu durumda H matrisi boyutu = 60x180
  - bu durumda G matrisi boyutu = 120x180
  - her stripe içinde node başına 18 sembol düşüyor

- 8/28 (LDPC için)
  - $W_C = 4$
  - $W_R = 14$
  - $n = 28$ (low-density'i korumak için 3 katı aldım)
  - initial storage node sayısı = 7
  - toplam başlangıç node sayısı = 56
  - bu durumda H matrisi boyutu = 8 x 28
  - bu durumda G matrisi boyutu = 20x28
  - her stripe içinde node başına 4 sembol düşüyor

APPENDIX A.

Let us assume that we have 8 different recovery equations, $\{r_1, r_2, ..., r_8\}$, in total for repairing the lost symbols in $\mathcal{L}$. We can use any sequence for repairing symbols. But, the problem is finding sequence having the minimum sequence. This problem can be represented on a graph given in the Fig. (5). The equations can be represented as vertices and the scores can be represented in edge weights. Since no restriction exists on which equation to start with, a redundant node named as n is added which is adjacent to each equation. Now, the problem is turned into a dynamic travelling salesperson problem: To find a smallest cost path starting from and ending in node n. In this new problem, at the beginning the edge weights outgoing from node n is known. The other edge's weight can be changed if any node adjacent to the edge is travelled. In other words, the weight of the edge may be changed whenever a node adjacent to the edge is travelled. Or this problem can be seen as a hard decision decoding using minimum symbol without any ripple restriction. (dynamic weighted set cover) More generally, the problem is finding a minimum cost path on a set of recovery equations where edge costs are can be changed. Moreover, in this problem travelling at most L vertices is sufficient. Denote that after travelling at most L vertices, the weight of edges adjacent to unvisited vertices becomes zero.

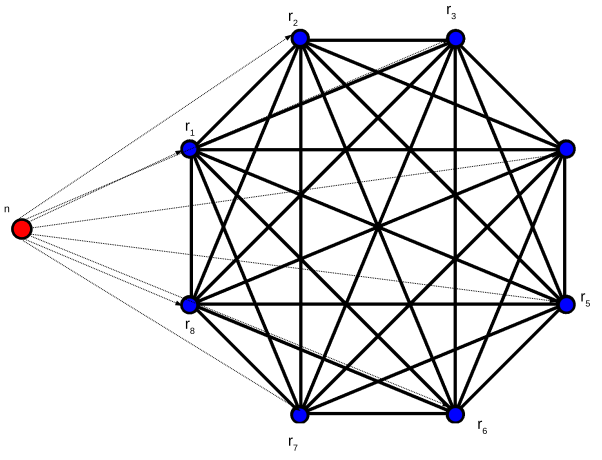

Figure 5: A representative graph.

References

Gaidioz, Benjamin, Birger Koblitz, and Nuno Santos. 2007. "Exploring High Performance Distributed File Storage Using LDPC Codes." *Parallel Comput.* (Amsterdam, The Netherlands, The Netherlands) 33, nos. 4-5 (May): 264–274. ISSN: 0167-8191. doi:10.1016/j.parco.2007.02.003. http://dx.doi.org/10.1016/j.parco.2007.02.003.

Gallager, Robert G. 1963. *Low-Density Parity-Check Codes.*

Han, T., and N. Ansari. 2014. "Offloading Mobile Traffic via Green Content Broker." *IEEE Internet of Things Journal* 1, no. 2 (April): 161–170. ISSN: 2327-4662. doi:10.1109/JIOT.2014.2316805.

Pääkkönen, Joonas, Camilla Hollanti, and Olav Tirkkonen. 2015. "Device-to-Device Data Storage with Regenerating Codes." In *Multiple Access Communications - 8th International Workshop, MACOM 2015, Helsinki, Finland, September 3-4, 2015, Proceedings,* 57–69. doi:10.1007/978-3-319-23440-3_5. https://doi.org/10.1007/978-3-319-23440-3%5C_5.

Pedersen, J., A. Graell i Amat, I. Andriyanova, and F. Brännström. 2016. "Distributed Storage in Mobile Wireless Networks With Device-to-Device Communication." *IEEE Transactions on Communications* 64, no. 11 (November): 4862–4878. ISSN: 0090-6778. doi:10.1109/TCOMM.2016.2605681.

Plank, J. S., and M. G. Thomason. 2003. *On the Practical Use of LDPC Erasure Codes for Distributed Storage Applications.* Technical report CS-03-510. University of Tennessee, September.

Wei, Y., Y. W. Foo, K. C. Lim, and F. Chen. 2014. "The Auto-configurable LDPC Codes for Distributed Storage." In *2014 IEEE 17th International Conference on Computational Science and Engineering,* 1332–1338. December. doi:10.1109/CSE.2014.254.

Yongmei, Wei, Chen Fengmin, and Lim Khai Cher. 2015. “Large LDPC Codes for Big Data Storage.” In *Proceedings of the ASE BigData & SocialInformatics 2015,* 1:1–1:6. ASE BD&SI ’15. Kaohsiung, Taiwan: ACM. ISBN: 978-1-4503-3735-9. doi:10.1145/2818869.2818881. http://doi.acm.org/10.1145/2818869.2818881.