LDPC Kodlarında Artık Veri Kullanımı
Residual Data Usage in LDPC Codes

Erdi Kaya
Ankara University
Computer Engineering
Ankara, Turkey
erdik@ankara.edu.tr

Massoud Pourmandi
Boğaziçi University
Electrical Electronics Engineering
Istanbul, Turkey
massoud.pourmandi@boun.edu.tr

Elif Haytaoglu
Pamukkale University
Computer Engineering
Denizli, Turkey
eacar@pau.edu.tr

Suayb S. Arslan
MEF University
Computer Engineering
Istanbul, Turkey
arslans@mef.edu.tr

2022 30th Signal Processing and Communications Applications Conference (SIU) | 978-1-6654-5092-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SIU55565.2022.9864789

***Özetçe* —Dağıtık sistemlerde ve dağıtık ön-bellekleme sistemlerinde, kodlanmış veri boyutu depolama düğümü sayısına tam olarak bölünemediğinde genel olarak fazladan veri depolama işlemi gerçekleştirilmektedir. Bu çalışmada, baz istasyonunun ve düğümlerin kodlanmış veriyi depoladığı bir ön-bellekleme sistemi için, fazladan veri eklenmeden kodlanmış sembollerin depolama düğümlerine dağıtılması konusu incelenmiştir. Silinti kodları olarak zaman açısından yüksek kodlama verimliliğine sahip LDPC kodları kullanılmış olup düğüm tamir zamanı açısından fazladan verinin kaydedildiği kodlanmış veri paylaştırımı yaklaşımı ile fazladan verinin kullanılmadığı (*artık veri paylaştırımı*) kodlanmış veri paylaştırımı yaklaşımlarının performansı karşılaştırılmıştır. Bu kapsamda düğüm tamiri zamanı ile toplamda depolanması gereken verinin miktarı ile ilgili sonuçlar elde edilmiştir.**

Anahtar Kelimeler—Dağıtık sistemler, önbellekleme, cihazlar arası iletişim, silinti kodları.

***Abstract*—In distributed storage systems/coded caching systems, padding operations should be performed when the encoded data cannot be divided by the number of storage nodes evenly. Thus, extra zero values are stored in one of the nodes to balance each node's storage content. In this study, distribution of data to storage nodes with no padding was investigated for distributed caching context in which a base station and devices both store the coded data. In other words, no redundancy (no-padding) is included into the encoded data. This approach is named as *residual data distribution*. LDPC codes are selected as the erasure code due to their low complexity encode/decode operations. Moreover, performance comparisons were conducted between using traditional data distribution approach (with padding) and using residual data (use of no-padding) (*standard*) in terms of repair time. In our work, the effect of no-padding data usage on the repair time and the ratios of storage savings have been also demonstrated.**

Keywords—distributed systems, caching, device-to-device communication, erasure codes.

I. INTRODUCTION

The device-to-device communication model can be exploited by keeping the data in the caches of the cellular devices and by communicating directly with each other when needed [1]. This form of communication, which is established directly without using a base station, is different from the traditional communication model and can be utilized to decrease overall base station communication. In other words, the data kept in the caches of the devices can be shared directly between the devices with no central authority. By using this communication model devices can download the desired content from nearby devices if exist, otherwise they can use base station as the last alternative. This approach constitutes the base of the coded caching systems.

In distributed data storage systems, inherently in coded caching systems, padding operations should be performed when the encoded data can not be divided evenly by the number of storage nodes. In this study, unlike other studies, residual data (data that cannot be evenly divided by the number of storage nodes) was investigated in distributed coded caching scenario without introducing any redundancy. In general, caching strategies can be divided into uncoded and coded caching. Erasure correcting codes are generally used in coded caching paradigm and these codes have the potential to reduce the amount of storage space required relative to that of the uncoded caching [2]–[4]. One of the options to fulfill the requirements of the coded caching is presented as low density parity check (LPDC) codes in [5]. Several studies considered LDPC codes for distributed systems as well [6]–[10]. Similarly, in this study, as for the erasure code, LDPC codes are used and performance comparisons were conducted between using traditional data distribution and residual data distribution approaches in terms of repair time/latency. Thus, the effect of residual data (no padding) distribution on repair time has been also demonstrated. According to the obtained results, the residual data distribution method can save storage space up to $\%5.78$, whereas it deteriorates the repair time as expected.

This paper is organized as follows: in Section II system model and data distribution methods are given, in Section III the simulation results are presented and the discussions related the advantages and disadvantages of these two data distribution methods are made.

II. SYSTEM MODEL AND METHOD

In the system model, it is assumed that there are a total of T nodes and a single base station serving a cellular network. It is also assumed that these nodes can communicate with each other through a device-to-device communication protocol. The simulation model, in which different residual data usages are tested, was created according to the $M/M/\infty$ queuing model. The probabilities of the nodes entering into and exiting out

978-1-6654-5092-8/22/$31.00 ©2022 IEEE

of the cell, i.e., the possibilities of entering and leaving the coverage area of the base station, are assumed to be *Poisson* as assumed in [2], [4].

In this system, a single file of size B bytes is partitioned into k blocks and these blocks are encoded into $n > k$ blocks all of which are equal size. The encoded data is distributed in the caches of s distinct nodes, where $s < T$. LDPC erasure codes are used for encoding the data where the base station is assumed to have a copy of the encoded data. In this study, similar to the previous studies [10], [11], we have considered a device-to-device communication protocol for the distributed caching system in which base stations are acting as helpers. Each storage node stores $t \leq \lceil \frac{n}{s} \rceil$ symbols, with $\lceil \frac{n}{s} \rceil < k$. If a data catching node leaves the cell, an empty node is selected from the cell instead. A lazy repair process as applied in [4] is adopted at certain time intervals for recovering these lost nodes.

The main motivation of this study is to investigate two symbol distribution approaches, namely standard symbol distribution and residual data distribution method on the aspect of repair time and storage usage. In the fist one, if the condition $s|n$ is not satisfied, one of the storage node stores extra zero valued data to equalize its encoded data size with that of the other storage nodes. On the other hand, in the second approach, no extra padding symbols are used, hence this approach requires less storage space in total.

In this paper, we use array LDPC erasure codes [12]. A node repair algorithm [10], based on a greedy approach is used to minimize the use of base stations in the selection and repair process of the relevant recovery equation. Using this approach, the symbols of the lost node are repaired. According to this method, recovery equations for the lost symbols are determined and one of them is selected.

A. LDPC Codes

In the $(n - k \times n)$ parity check matrix of an (n, k) LDPC code, the number of ones in rows and columns are expressed as w_r and w_c, respectively. This matrix consists of 0 and 1 values, and the number of 1's is usually much less than 0's. If w_c is a constant for each column, then $w_r = w_c(\frac{n}{k})$. An LDPC code is called regular if w_r and w_c are both constant. After generating parity check matrix, $(k \times n)$ generator matrix can be constructed through elementary row/column operations, which then can be used to encode the data.

The symbols in the recovery equation used to repair a lost symbol are called helper symbols. These helper symbols are cached in storage nodes within the cellular network. In addition, if more than one symbol is stored in the lost node, some of these helper symbols can be one of the other lost symbols in this node. If these helper symbols are located in other nodes, they are first downloaded from these nodes. If any one of the helper symbols in the recovery equation cannot be found in other local nodes, local repair ceases entirely and the lost symbol/s are downloaded directly from the base station instead.

In Fig. 1, an example scenario of the repair process of one of the nodes is shown in which the original message symbols are encoded based on a regular $(18, 9)$ LDPC code.

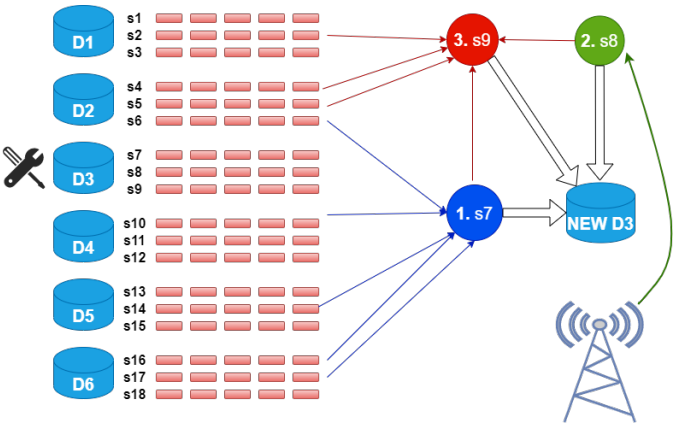

Fig. 1: Repair approach for regular (18,9) LDPC code.

In the parity check matrix of this code, the weights of the ones in the columns and rows are assumed to be $w_c = 3$ and $w_r = 6$, respectively. Moreover, in this scenario, six storage nodes are used in total. Hence, three encoded symbols are stored in each storage node. Now, let us consider the case that the node named D3 is lost and thus the repair process needs to be initiated for 3 encoded symbols stored in D3. We name the node NEW D3 as the "newcomer" and it is selected from the empty nodes in the cell. Let us assume the lost symbols to be $s7$, $s8$ and $s9$, respectively. Note that there are 3 candidate recovery equations due to the value of w_c for each lost symbol. While choosing the recovery equation, we attempt to minimize the use of the base station as in [10]. In other words, the helper symbols that appear in the recovery equations are downloaded directly from the other 5 local storage nodes without using the base station. Accordingly, one of the three candidate recovery equations is selected for each lost symbol. After selecting a recovery equation for each symbol, the algorithm given in [10] is applied. The recovery equations chosen for this repair scenario are shown in Eq. (1), Eq. (2), and Eq. (3).

$$s7 = s6 \oplus s10 \oplus s14 \oplus s16 \oplus s17 \quad (1)$$

$$s8 = s1 \oplus s9 \oplus s11 \oplus s13 \oplus s15 \quad (2)$$

$$s9 = s2 \oplus s4 \oplus s5 \oplus s7 \oplus s8 \quad (3)$$

It is seen that the number of helper symbols in each equation is equal and $w_r - 1 = 5$ due to the use of regular LDPC code. At this point, initially, the related lost symbols are repaired by using the recovery equations that do not require the use of a base station. After the completing the repair process of each lost symbol, all candidate recovery equations for the remaining lost symbols are re-organized by checking the base station use cases. First, the symbol $s7$ and then $s8$ are repaired by downloading symbols from the helper nodes.

The symbols $s7$ and $s8$ in Eq. (3) were lost symbols. But they can be used for this equation since these symbols were repaired in the previous process. By using this approach, the number of symbols downloaded from the base station is reduced. During this repair process, one symbol was downloaded from the base station while a total of 8 symbols was taken from other nodes.

Symbol Index	1st Iteration	2nd Iteration	3rd Iteration
1	Node 1	Node 1	Node 1
2	Node 1	Node 1	Node 1
3	Node 1	Node 1	Node 1
4	Node 2	Node 2	Node 2
5	Node 2	Node 2	Node 2
6	Node 2	Node 2	Node 2
7	Node 3	Node 3	Node 3
8	Node 3	Node 3	Node 3
9	Node 3	Node 3	Node 3
10	Node 4	Node 4	Node 4
11	Node 4	Node 4	Node 4
12	Node 4	Node 4	Node 4

Symbol Index	1st Iteration	2nd Iteration	3rd Iteration
1	Node 1	Node 1	Node 1
2	Node 1	Node 1	Node 1
3	Node 2	Node 2	Node 2
4	Node 2	Node 2	Node 2
5	Node 3	Node 3	Node 3
6	Node 3	Node 3	Node 3
7	Node 4	Node 4	Node 4
8	Node 4	Node 4	Node 4
9	Node 1	Node 1	Node 1
10	Node 2	Node 2	Node 2
11	Node 3	Node 3	Node 3

Fig. 2: Symbol distribution methods: Standard symbol distribution method (left), Symbol distribution method with residual data (right).

B. Standard Symbol Distribution Method

In the standard symbol distribution method, while the encoded symbols are distributed to the storage nodes, the objective is to keep an equal number of symbols in each one of these storage nodes. Thus, an equal number of symbols are stored in each storage node after the symbol distribution. For this purpose, if necessary, extra zero symbols may be stored. A scenario of a standard symbol distribution method is shown in Fig. 2 *(left)*. In this scenario, 11 encoded symbols are distributed to 4 storage nodes. Since it is desired to keep an equal number of encoded symbols in each node, 3 symbols are stored in any storage node. Accordingly, the 12th symbol is not actually an encoded symbol and is given a value of zero to ensure the desired balance. Despite the use of extra storage space, the main motivation of this method is to save time in repair operations. The iterations in the figure represent loop states related to encoding process i.e., the file is divided into partitions and each partition is encoded separately. More specifically during the encoding process, each partition is encoded with an (n, k) LDPC code, and resulting n symbols are distributed according to the mapping shown in Fig. 2 *(left)*. Equation (4) shows the range of symbol indices that symbol i held in storage node j can take. In other words, the symbol indices that storage node j can take are represented by this equation. n comes from the parameter of LDPC code, while the variable s refers to the number of storage nodes.

$$(j-1)\left\lceil \frac{n}{s} \right\rceil + 1 \leq i \leq (j-1)\left\lceil \frac{n}{s} \right\rceil + \left\lceil \frac{n}{s} \right\rceil \tag{4}$$

C. Symbol Distribution Method with Residual Data

The concept of no-padding or residual data is related to the storage of irregular number of symbols in the storage nodes. In other words, it is a method of distribution of encoded symbols to storage nodes. In Fig. 2 *(right)*, a sample scenario for the symbol distribution method with residual data distribution is presented. In this example, the first 8 symbols of the 11 encoded symbols are distributed in pairs to 4 storage nodes. The 9th, 10th and 11th symbol indexes are the symbol indexes where the residual data will be kept. The symbols kept in these symbol indexes are defined as residual data. Starting from the 9th symbol index, each symbol is sent to a node in turn from the first storage node. Symbols corresponding to the same symbol index in each cycle are stored in the same storage node. In this study, like in the scenario of Fig. 2 *(right)*, the symbol indexes of residual data are always matched with the same nodes in each loop. By this distribution based on residual data usage, storing extra symbols are avoided in the nodes. In this distribution method, it is aimed to save storage space instead of saving time in repair operations. Equation (5) shows the relationship between j. storage node and i. symbol that is non-residual, while equation (6) shows the relationship between j. storage node and i. symbol that is residual.

$$i \leq s\left\lfloor \frac{n}{s} \right\rfloor \Rightarrow j = \left\lfloor \frac{i}{\left\lfloor \frac{n}{s} \right\rfloor} \right\rfloor + \left(i \mod \left\lfloor \frac{n}{s} \right\rfloor \right) \tag{5}$$

$$i > s\left\lfloor \frac{n}{s} \right\rfloor \Rightarrow j = i - s\left\lfloor \frac{n}{s} \right\rfloor \tag{6}$$

III. Results

Initially, raw data file having the size 128KB is encoded according to the LDPC code design in [12] and then this encoded data is distributed to 30 storage nodes using both distribution approaches. Then the node repair operations are realized in time intervals according to $M/M/\infty$ queuing model.

Both standard and residual data distribution methods were tested for the codes (388,194), (424,318), (908,454) and (2056,1542) LDPC codes, respectively. The repair times for these approaches can be seen in Fig 3 and 4. The green lines in the figures point out the codes using the symbol distribution method with residual data distribution, while the orange lines represent the codes using the standard symbol distribution method. In these figures, r refers to the number of residual symbol indices for each LDPC code. According to the results, the repair times of codes with symbol distribution method based on residual data are longer than the repair times of codes based on the standard distribution method as expected. For (388,194) LDPC code, it was observed that the distribution method based on residual data has %64.04, %34.57 and %28.24 higher repair times per node than the standard distribution method at 0.1, 0.5 and 0.9 of Δ values that are repair intervals, respectively. For (424,318) LDPC code, the distribution method based on residual data has %30.57, %0.86 and %7.37 higher repair times per node than the standard distribution method at 0.1, 0.5 and 0.9 of Δ values, respectively. For (908,454) LDPC code, the distribution method based on residual data has %54.03, %12.53 and %12.15 higher repair times per node than the standard distribution method at 0.1, 0.5 and 0.9 of Δ values, respectively. Also, for (2056,1542) LDPC code, the distribution method based on residual data has %16.94, %8.81 and %8.69 higher repair times per node than the standard distribution method at 0.1, 0.5 and 0.9 of Δ values, respectively.

Apart from the repair time comparisons, the symbol distribution method with residual data is compared with the standard symbol distribution method in terms of data storage. In this

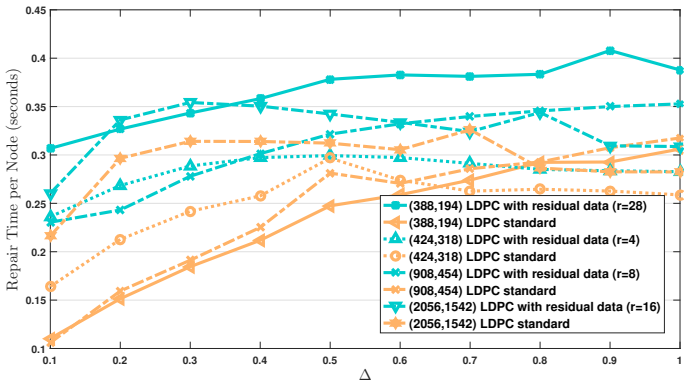

Fig. 3: Comparison of symbol distribution methods in terms of repair time of a single node.

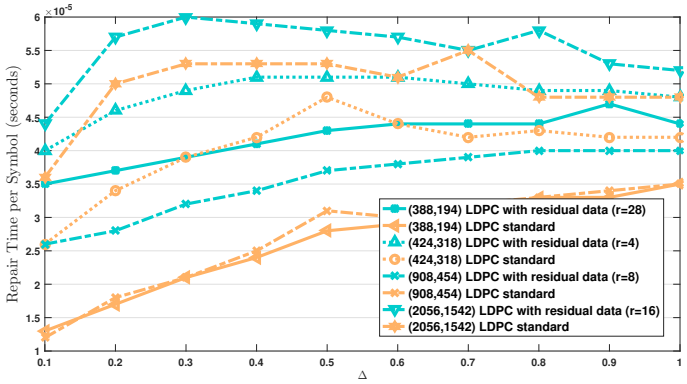

Fig. 4: Comparison of symbol distribution methods in terms of repair time of a single symbol.

comparison, the raw data file of 128 KB was used as before. After this file data was expanded according to the code rate of the relevant LDPC code, this extended (encoded) data was distributed to 30 storage nodes. Table 1 shows the amount of encoded symbols stored in the storage nodes according to the LDPC codes. These results are also presented in Fig. 5. As can be seen from the results, residual data usage saves almost %0.51 storage space for (388,194) LDPC code, %5.78 storage space for (424,318) LDPC code, %2.37 storage space for (908,454) LDPC code and %0.68 storage space for (2056,1542) LDPC code. It is seen that the use of residual symbols saves a little in terms of storage space.

LDPC Codes	Encoded Data (in bytes)
(388,194) LDPC code residual	262288
(388,194) LDPC code standard	263640
(424,318) LDPC code residual	175112
(424,318) LDPC code standard	185850
(908,454) LDPC code residual	262412
(908,454) LDPC code standard	268770
(2056,1542) LDPC code residual	176816
(2056,1542) LDPC code standard	178020

Table I: Comparison of encoded data in 30 storage nodes in terms of LDPC codes

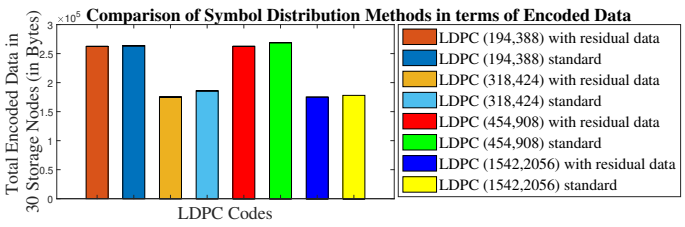

Fig. 5: Total number of symbols in bytes distributed across 30 storage nodes.

REFERENCES

[1] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1801–1819, 2014.

[2] J. Pääkkönen, C. Hollanti, and O. Tirkkonen, "Device-to-device data storage for mobile cellular systems," in *2013 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2013, pp. 671–676.

[3] J. Paakkonen, C. Hollanti, and O. Tirkkonen, "Device-to-device data storage with regenerating codes," in *Multiple Access Communications*. Cham: Springer International Publishing, 2015, pp. 57–69.

[4] J. Pedersen, A. G. i Amat, I. Andriyanova, and F. Brännström, "Distributed storage in mobile wireless networks with device-to-device communication," *IEEE Transactions on Communications*, vol. 64, no. 11, pp. 4862–4878, 2016.

[5] E. Haytaoglu, E. Kaya, and S. S. Arslan, "On the fault tolerant distributed data caching using ldpc codes in cellular networks," *arXiv preprint arXiv:2010.14781*, 2020.

[6] J. S. Plank and M. G. Thomason, "A practical analysis of low-density parity-check erasure codes for wide-area storage applications," in *International Conference on Dependable Systems and Networks, 2004*. IEEE, 2004, pp. 115–124.

[7] Y. Wei, Y. W. Foo, K. C. Lim, and F. Chen, "The auto-configurable ldpc codes for distributed storage," in *2014 IEEE 17th international conference on computational science and engineering*. IEEE, 2014, pp. 1332–1338.

[8] H. Park, D. Lee, and J. Moon, "Ldpc code design for distributed storage: Balancing repair bandwidth, reliability, and storage overhead," *IEEE Transactions on Communications*, vol. 66, no. 2, pp. 507–520, 2017.

[9] W. Yongmei and C. Fengmin, "Guided systematic random ldpc for distributed storage system," in *Proceedings of the 2017 International Conference on Information Technology*, 2017, pp. 355–359.

[10] E. Haytaoglu, E. Kaya, and S. S. Arslan, "Data repair-efficient fault tolerance for cellular networks using ldpc codes," *IEEE Transactions on Communications*, vol. 70, no. 1, pp. 19–31, 2022.

[11] E. Kaya, E. Haytaoglu, and S. S. Arslan, "Data repair in bs-assisted distributed data caching," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2020, pp. 1–4.

[12] E. Eleftheriou and S. Olcer, "Low-density parity-check codes for digital subscriber lines," in *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No. 02CH37333)*, vol. 3. IEEE, 2002, pp. 1752–1757.

ACKNOWLEDGEMENT

This study is supported by TUBITAK under grant no 119E235.