Consecutive Switch Codes

Sarit Buzaglo, *Member, IEEE*, Yuval Cassuto, *Senior Member, IEEE*, Paul H. Siegel, *Fellow, IEEE*, and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*— Switch codes, first proposed by Wang *et al.*, are codes that are designed to increase the parallelism of data writing and reading processes in network switches. A network switch is required to write n incoming packets and read k outgoing packets while using m memory banks, each able to write and read one packet per time unit. Each set of n packets written to the switch simultaneously is called a generation. The objective is to store the packets in the banks such that every request of k packets, which can belong to previous generations, can be handled by reading at most one packet from every bank. In this paper, we study a new type of switch codes that can simultaneously deliver large packet request and good coding rate. These attractive features are achieved by relaxing the request model to a natural sub-class we call *consecutive requests*. For this new request model, we define a new type of codes called *consecutive switch codes*. These codes are studied in both the computational and combinatorial models, corresponding to whether the data can be encoded or not. For binary codes, we also study an intermediate model in which a coded packet is formed by the XOR operations of at most two input packets. We present several code constructions and prove the optimality of one family of these codes by providing the corresponding lower bound. Finally, we introduce a construction of conventional switch codes, which improves upon the best known results for the case $n = k$.**

***Index Terms*— Batch codes, network switch, switch codes.**

I. Introduction

SWITCH codes were first suggested a few years ago by Wang *et al.* in [20] for networking applications and were further studied in [4]–[7], [19], and [21]. A network switch is a device used to connect between a computer network and external devices. The main task of the network switch is to process and forward packets from the input ports to their designated output ports. Upon arrival, the packets from the input ports are processed and stored in a switch fabric comprising multiple memory **banks**. In each time unit, called a **generation**, at most one packet can be written to or read from each bank. The main problem facing a network switch is that it needs to fulfill an arbitrary request for packets stored in its memory, subject to the limitation of physically reading at most one packet from each bank per time unit.

To address this problem, switch codes form a coding scheme in which incoming packets are encoded before their write to the memory, such that it is guaranteed that an arbitrary set of packets can be decoded while reading at most one physical packet from each bank. Since the packet requests are arbitrary, it is intuitively clear that each packet, after being encoded to the banks, should have multiple options to be read and decoded from the banks, in order to avoid contention between multiple requested packets stored in the same bank. Given the current trend of scaling switches by adding more parallel banks (other scaling methods have become more challenging and/or costly), switch codes are likely to become a useful tool to provide efficient rate scaling by overcoming bank contention between requested packets. For switch codes to succeed within the network-switch application, they should be made efficient in terms of both redundancy and various measures of complexity, which is the exact aim of this paper. Mathematically speaking, a switch code is required to satisfy the following property. Assume there are n input packets, k output packets, and m banks. In each generation the n input packets are encoded into m packets which are stored in the banks. Then, in each generation, the code can fulfill every request for k packets, which is specified by the write-generation number and the packet index in the generation for each of the k packets.

Switch codes are associated with the family of codes called **batch codes**. Batch codes were first studied in the previous decade by Ishai *et al.* [11] and recently in [13], [15], [16], [18], and [22]. A batch code encodes n information symbols into m buckets such that any request for k information symbols can be answered by reading at most one, and more generally at most t, symbols from each bucket. If the set of k requested symbols can have repetitions and each bucket stores a single symbol, then the batch code is called a **multi-set primitive batch code** and it is equivalent to a switch code of the same parameters. For completeness, we prove this relationship between batch codes and switch codes in Appendix A. (See also the discussion about this connection in [21].) Despite this equivalence, there are two distinctions

Manuscript received September 19, 2016; revised April 3, 2017; accepted July 3, 2017. Date of publication July 19, 2017; date of current version March 15, 2018. This work was supported by NSF under Grant 1405119. S. Buzaglo was supported in part by the Weizmann Institute of Science through the National Postdoctoral Award Program for Advancing Women in Science, in part by the Center for Memory and Recording Research, University of California at San Diego, and in part by the ISEF Foundation. Y. Cassuto was supported by the Israel Science Foundation. E. Yaakobi was supported in part by the Israel Science Foundation under Grant 1624/14 and the Binational Science Foundation under Grant 2015816. This paper was presented in part at the 2016 IEEE International Symposium on Information Theory.

S. Buzaglo is with the Center for Memory and Recording Research, University of California, San Diego, La Jolla, CA 92093-0401 USA (e-mail: sbuzaglo@ucsd.edu).

Y. Cassuto is with the Viterbi Electrical Engineering Department, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: ycassuto@ee.technion.ac.il).

P. H. Siegel is with the Electrical and Computer Engineering Department and the Center for Memory and Recording Research, University of California, San Diego, La Jolla, CA 92093-0407 USA (e-mail: psiegel@ucsd.edu).

E. Yaakobi was with the Center for Memory and Recording Research, University of California, San Diego, La Jolla, CA 92093-0407 USA. He is now with the Computer Science Department, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: yaakobi@cs.technion.ac.il).

Communicated by C. Xing, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2017.2728791

between switch codes and the more general batch codes. One distinction is that switch codes are commonly designed with $k = n$, which balances the output and input switching rates. Correspondingly, all the constructions we propose in this paper apply to this important case. Another distinction is that they introduce a time dimension, which inspires us to define and study a new type of switch codes called **consecutive switch codes**. This refinement of the model is shown herein to allow significant reduction in redundancy over the standard non-consecutive model.

In the original definition of batch and switch codes the packet requests are not constrained and can be taken from any previous generation. However, from a practical point of view it is reasonable to assume that packet requests in each generation are restricted to some ℓ previous consecutive generations. This motivates us to study consecutive switch codes, which follow this restriction on the packet requests. Another motivation to study consecutive switch codes is that this model also takes into account the number of packets that each bank can store. (A related and stronger type of codes, but closer to the original model, where for every $1 \leq r \leq n$, the rth input packet can be requested from at most ℓ generations, was studied in [5].) We study two main classes of these codes, namely, **combinatorial** and **computational** consecutive switch codes. In the combinatorial class, it is assumed that the packets stored in the banks are simply copies of the input packets (that is, they are not encoded), while in the computational class the packets can be encoded. We note that the combinatorial model of consecutive switch codes extends the corresponding one for batch codes which was extensively studied in the literature, see e.g [1]–[3], [17], as well as the combinatorial model of (non-consecutive) switch codes which was explored when switch codes were first proposed in [20]. In many scenarios in practice, the coding schemes are very restricted in the encoding and decoding complexities they can afford. One such scenario when the encoder and decoder can only use plain XOR operations was considered in [19] and [20]. In these papers the **degree** of a switch code was defined to be the maximum number of input packets that participate in the encoding of each parity packet (using only XOR operations) and limited degree switch codes were considered. We adapt the concept of the degree to consecutive switch codes and study such codes where the degree is two. Note that if the degree is limited to one then the consecutive switch code is combinatorial and if the degree is limited to n then the combinatorial switch code is computational (with no limitation on the degree). Thus, the limited degree case can be viewed as an intermediate model between the combinatorial and computational models.

The rest of the paper is organized as follows. In Section II, we formally define switch codes and batch codes and note the equivalence between switch codes and multi-set primitive batch codes. We also present in this section a construction of switch codes (and hence also of batch codes) for the case $n = k$, which improves upon the state-of-the-art results for this case. In Section III, we formally define consecutive switch codes. In Section IV, we give constructions and a bound for combinatorial consecutive switch codes and in Section V we give a construction of consecutive switch codes in which $\ell = 2$ and the degree is 2. In Section VI, we construct consecutive switch codes for the computational class. Finally, Section VII concludes the paper and lists several open problems.

II. Preliminaries

In this section we present some of the definitions and notation used throughout this paper. In particular, we formally define switch codes and describe their connection to batch codes [11].

For a positive integer n, denote by $[n]$ the set of n integers $\{1, 2, \ldots, n\}$. For two integers a, b, where $a < b$, denote by $[a, b]$ the set of $b - a + 1$ integers $\{a, a+1, \ldots, b\}$. A **multi-set** $\mathcal{M} = \{i_1, i_2, \ldots, i_k\}$ over $[n]$ of size k is a collection of k elements of $[n]$ with repetitions, i.e., an element can appear in $\mathcal{M}$ multiple times. The set of all natural numbers is denoted by $\mathbb{N}$ and a q element field is denoted by $\mathbb{F}_q$.

A **(ν, μ)-code**, $\mathcal{C}$, over $\mathbb{F}_q$, is a subset of $\mathbb{F}_q^\nu$ of size q^μ. An **encoder** for $\mathcal{C}$ is an injection from $\mathbb{F}_q^\mu$ to $\mathcal{C}$. Throughout this paper, we assume that a code $\mathcal{C}$ is equipped with an encoder, which will be denoted by $\mathcal{E}_\mathcal{C}$. If $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^\nu$ (over $\mathbb{F}_q$) then $\mathcal{C}$ is called **linear** and is referred to as a **$[\nu, \mu]$-code**.

For a string $\mathbf{x} \in \mathbb{F}_q^\mu$ and for a positive integer d, let $\mathcal{R}_d : \mathbb{F}_q^\mu \to \mathbb{F}_q^{d\mu}$ be the encoder of the d-repetition code, which encodes $\mathbf{x}$ to the concatenation of d copies of $\mathbf{x}$.

Definition 1: An ***$(n, k, m, t)_q$-switch code*** *is an infinite sequence $\{\mathcal{C}_T\}_{T \geq 1}$ of $(\nu = m, \mu = n)$-codes over $\mathbb{F}_q$ such that the following hold.*

1) *For every $T \geq 1$, a string $\mathbf{x}^{(T)} \in \mathbb{F}_q^n$ is encoded by $\mathcal{E}_{\mathcal{C}_T}$ to a string $\mathbf{c}^{(T)} \in \mathbb{F}_q^m$.*
2) *For every set of k different pairs*

$$I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [n] \times \mathbb{N},$$

there exists a set $J \subset [m] \times \mathbb{N}$, which depends upon only the set I, such that:

   i. *for every $r \in [k]$, the symbol $x_{i_r}^{(T_r)}$ can be recovered from $\{c_j^{(T_r)}\}_{(j,T_r) \in J}$.*
   ii. *for every $j \in [m]$, J contains at most t pairs of the form (j, T), for some $T \geq 1$.*

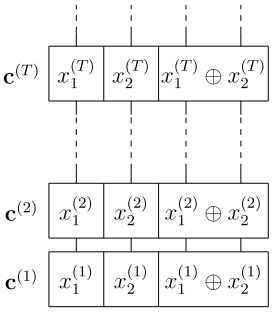
The set I is called the ***request set****, whereas the set J is called the* ***recovery set*** *for the request set I.*

Concretely, a switch code encodes strings of input packets into rows of a semi-infinite array and is able to recover any k packets from all the input strings by accessing at most t packets from each of the columns $\{c_j^{(T)}\}_{T \geq 1}$, $j \in [m]$. The **rate** of the switch code is defined by $R = n/m$.

Example 1: The following semi-infinite array forms an $(n = 2, k = 2, m = 3, t = 1)$-switch code.

If for example $I = \{(1, 1), (1, 2)\}$ then $J = \{(1, 1), (2, 2), (3, 2)\}$ is a recovery set for I, since $x_1^{(1)}$ can be recovered from $c_1^{(1)}$ and $x_1^{(2)}$ can be recovered from $c_2^{(2)}$ and $c_3^{(2)}$. Moreover, J consists of at most one element of the form (j, T), for each $j \in [3]$.

Remark 1: In our definition of switch codes we only required that the number of symbols read from each column is at most t, and all read symbols can be used to decode



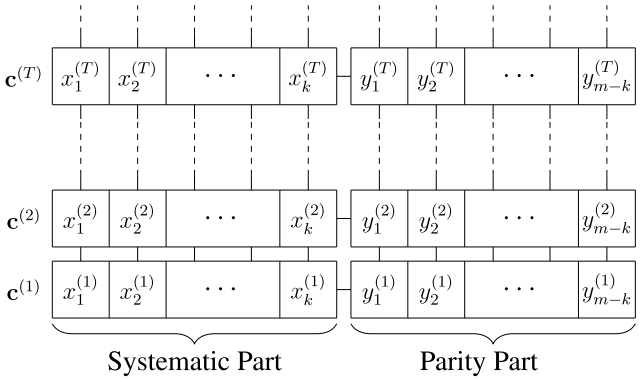


Fig. 1. Systematic $(k, m)_q$-switch code. For every k pairs (i_1, T_1), $(i_2, T_2), \ldots, (i_k, T_k)$, the symbols $x_{i_1}^{(T_1)}, x_{i_2}^{(T_2)}, \ldots, x_{i_k}^{(T_k)}$ can be recovered by accessing at most one entry from each column.

each of the requested symbols. An alternative definition, see e.g. [20], requires that each symbol has a disjoint recovery set of symbols (in the case of $t = 1$). However, these two definitions are equivalent since the requested symbols may belong to different rows, in which case a requested symbol $x_i^{(T)}$ can be recovered only by symbols that are read from the Tth row. Thus, the recovery sets of the different requested symbols are required to be disjoint as well. We chose the formulation as in Definition 1 since from the application point of view every output port can access any of the read symbols. We will later see that even though the distinction between these two definitions does not make a difference for switch codes, it does for consecutive switch codes.

By definition, a switch code is specified by an infinite sequence of codes and hence it might be very complicated to construct good codes, i.e., codes with high rate. Fortunately, as the next lemma states, it is enough to consider only switch codes that are obtained by repeating the same code for every generation. More precisely, for any set of parameters for which a switch code exists, there also exists a switch code of the same parameters, $\{\mathcal{C}_T\}_{T\geq 1}$, such that $\mathcal{C}_T = \mathcal{C}$, for all $T \geq 1$.

Lemma 1: If there exists an $(n, k, m, t)_q$-switch code $\{\mathcal{C}_T\}_{T\geq 1}$ then there exists an (m, n)-code $\mathcal{C}$ over $\mathbb{F}_q$ such that the infinite sequence of codes $\{\widetilde{\mathcal{C}}_T\}_{T\geq 1}$, where $\widetilde{\mathcal{C}}_T = \mathcal{C}$, for all $T \geq 1$, forms an $(n, k, m, t)_q$-switch code.

Proof: Since there are only finite number of (m, n)-codes over $\mathbb{F}_q$, there exists a code $\mathcal{C}$ that appears in the sequence $\{\mathcal{C}_T\}_{T\geq 1}$ an infinite number of times. The lemma now follows from the simple observation that any infinite subsequence of $\{\mathcal{C}_T\}_{T\geq 1}$ is again an $(n, k, m, t)_q$-switch code. ∎

By Lemma 1, we can restrict our discussion to switch codes that are formed by only one code $\mathcal{C}$. Henceforth, an (m, n)-code over $\mathbb{F}_q$ will be called an $(n, k, m, t)_q$-switch code if the infinite sequence of codes $\{\mathcal{C}_T = \mathcal{C}\}_{T\geq 1}$ is an $(n, k, m, t)_q$-switch code.

Two subsequences $\mathbf{u} = w_{i_1} w_{i_2} \ldots w_{i_r}$ and $\mathbf{v} = w_{j_1} w_{j_2} \ldots w_{j_s}$ of a string $\mathbf{w} \in \mathbb{F}_q^m$ are called **disjoint** if $\{i_1, i_2, \ldots, i_r\} \cap \{j_1, j_2, \ldots, j_s\} = \emptyset$. In [11], Ishai *et al.* proposed multi-set batch codes.

Definition 2: An ***$(n, N, k, m, t)_q$-multi-set batch code*** *encodes a string $\mathbf{x} \in \mathbb{F}_q^n$ into the concatenation of some m strings $\mathbf{y} = y_1 y_2 \ldots y_m$, $y_j \in \mathbb{F}_q^* \stackrel{\text{def}}{=} \cup_{v\in\mathbb{N}} \mathbb{F}_q^v$ for all $j \in [m]$, of total length N, such that for every multi-set $\mathcal{M} = \{i_1, i_2, \ldots, i_k\}$ over $[n]$ of size k, the k symbols $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ can be recovered from $\mathbf{y}$, where the following conditions hold.*

1) *For every $j \in [m]$, at most t symbols from each of the strings y_j are accessed.*
2) *For every $1 \leq r < s \leq k$, the two subsequences of $\mathbf{y} = y_1 y_2 \ldots y_m$ that are used to recover x_{i_r} and x_{i_s}, respectively, are disjoint.*
3) *The k position sets of the subsequences of $\mathbf{y}$ that are accessed depend only on $\mathcal{M}$.*

In [11] the authors defined the general concept of batch codes, which are by default not multi-set batch codes. However, we will only consider multi-set batch codes, and henceforth we refer to multi-set batch codes as **batch codes** for short. In [11] the authors also considered the concept of **primitive batch code**, where each of the m strings into which the input is encoded is of length one, i.e. $y_j \in \mathbb{F}_q$ for all $j \in [m]$, and hence $N = m$. Even though the following connection between batch codes and switch codes is somewhat known, we state it in the following lemma for the completeness of the results in the paper and provide the proof in Appendix A.

Lemma 2: For $k \leq n$, a code $\mathcal{C}$ is an $(n, N = m, k, m, t = 1)_q$-primitive batch code if and only if $\mathcal{C}$ is an $(n, k, m, t = 1)_q$-switch code.

In this paper we will consider only switch codes for which $n = k$ and $t = 1$ and we denote these codes by $(k, m)_q$-switch codes. This case was also studied in [11], [19] and [20], however most of the constructions in [11] (and also all the constructions in [15] and [18]) apply to cases in which k is much smaller than n. The case $n = k$ is motivated by the need to equate the switch write and read rates and $t = 1$ models a simple memory delivering one data packet per time unit. Note that in this case the recovery sets must be disjoint. Furthermore, we mostly consider systematic switch codes, i.e. we assume that for all $\mathbf{x} \in \mathbb{F}_q^k$, $\mathcal{E}_{\mathcal{C}}(\mathbf{x}) = \mathbf{xy}$, for some $\mathbf{y} \in \mathbb{F}_q^{m-k}$ (see Figure 1). In fact, all the constructions in this paper use linear codes, which always admit a systematic encoder [14]. An intriguing question is whether or not for all parameters for which switch codes exist, systematic switch codes also exist. Unfortunately, we do not have the answer to this question; however, we remark that our state-of-the-art construction of binary switch codes yields systematic switch codes.

A construction of $(k, m)_2$-switch codes, where $m = k^2/\log_2 k$ was given in [19]. This construction is optimal

in the setting in which each parity check bit is restricted to be the sum of at most $\log_2 k$ information bits. In [11], a construction of $(k, m)_2$-switch code with $m = k^{\log_2 3} = \Omega(k^{1.58})$ was presented. Our main result in this section is a construction of $(k, m)_2$-switch codes with $m \approx 2k^{1.5}$. Our construction significantly improves upon the results in [11] and [19] and to the best of our knowledge, it is the best known construction of a $(k, m)_2$-switch code and thus also of batch codes with the same parameters. The proof of this result appears in Appendix B.

Theorem 1: For every sufficiently large k, there exists a $(k, m)_2$-switch code with $m \approx 2k^{1.5}$.

III. Consecutive Switch Codes

Primitive batch codes and switch codes are equivalent, as Lemma 2 states, yet there is a significant conceptual difference between these two formulations. Unlike batch codes, switch codes introduce a time dimension which motivates us to define a variation of switch codes, which we refer to as **consecutive switch codes**. Consecutive switch codes are designed for a natural sub-class of the request set I in Definition 1. As for switch codes, these codes are capable of retrieving k information symbols, from different generations, by accessing at most t entries from each column. However, for ℓ-consecutive switch codes the k information symbols must belong to ℓ consecutive generations. One motivation for this variation of switch codes is that in practice two input strings that were encoded in a short time interval store correlated data, and therefore are likely to be of interest to the same user. Restricting the switch codes to this natural sub-class of requests allows us to increase the rate dramatically, and thus to design practical-rate codes that behave like switch codes for the more common queries of information symbols. Another motivation is that consecutive switch codes, as will be shown later, are also an adaptation of switch codes to the practical scenario in which the number of packets that can be stored in a bank is limited.

Definition 3: An $\boldsymbol{(n, k, m, t)_q}$*-$\boldsymbol{\ell}$-consecutive switch code is an infinite sequence of codes $\{\mathcal{C}_T\}_{T\geq 1}$ of $(v = m, \mu = n)$-codes over $\mathbb{F}_q$ such that the following hold.*

1) *For every $T \geq 1$, a string $\mathbf{x}^{(T)} \in \mathbb{F}_q^n$ is encoded by $\mathcal{E}_{\mathcal{C}_T}$ to a string $\mathbf{c}^{(T)} \in \mathbb{F}_q^m$.*
2) *For every set of k pairs*

$$I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [n] \times [T, T+\ell-1],$$

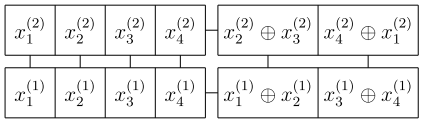
for some $T \geq 1$, there exists a set $J \subset [m] \times \mathbb{N}$ depending only on I, such that for every $r \in [k]$, the symbol $x_{i_r}^{(T_r)}$ can be recovered from $\{c_j^{(T)}\}_{(j,T)\in J}$ and for every $j \in [m]$, J contains at most t pairs of the form (j, T), for some $T \geq 1$.

Recall that for general switch codes, Lemma 1 states that instead of considering an infinite sequence of codes, it is enough to consider only one code. For ℓ-consecutive switch codes, the next lemma states that it is enough to consider only a sequence of ℓ codes. The proof of the lemma is trivial and thus omitted.

Lemma 3: If $\{\mathcal{C}_T\}_{T\geq 1}$ is an $(n, k, m, t)_q$-ℓ-consecutive switch code then $\{\widetilde{\mathcal{C}}_T\}_{T\geq 1}$, where $\widetilde{\mathcal{C}}_T = \mathcal{C}_u$ for $u \in [\ell]$ such that $T \equiv u \pmod{\ell}$, is also an $(n, k, m, t)_q$-ℓ-consecutive switch code.

By Lemma 3 we can restrict our discussion to switch codes that are formed by a sequence of ℓ codes, $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_\ell$, which are extended periodically. Henceforth, a sequence of ℓ codes, $\{\mathcal{C}_T\}_{T\in[\ell]}$, will be called an $(n, k, m, t)_q$-ℓ-consecutive switch code if the infinite sequence of codes that is formed by using $\{\mathcal{C}_T\}_{T\in[\ell]}$ periodically is an $(n, k, m, t)_q$-ℓ-consecutive switch code. From Lemma 3 we also conclude that ℓ-consecutive switch codes can be considered as an adaptation of switch codes to the more practical scenario in which the switch code is represented by a finite two-dimensional array with ℓ rows, rather than a semi-infinite one.

Example 2: The following $\ell \times m$ array forms an $(n = 4, k = 4, m = 6, t = 1)_2$-2-consecutive switch code.



If for example $I = \{(1, 1), (2, 1), (1, 2), (2, 2)\}$ then $J = \{(1, 1), (5, 1), (2, 2), (4, 2), (6, 2)\}$ is a recovery set for I, depending only on I, such that for all $j \in [6]$, (j, T) appears at most once at J. We remark that this code has degree two i.e., each parity-check bit is the XOR of at most two information bits, and it is a special case of the general construction given in Section V for 2-consecutive switch codes with degree two. Moreover, $\mathcal{C}_1 \neq \mathcal{C}_2$ and this code is optimal, i.e., there does not exist an $(n = 4, k = 4, m = 5, t = 1)_2$-2-consecutive code. Furthermore, there does not exist an $(n = 4, k = 4, m = 6, t = 1)_2$-2-consecutive switch code in which $\mathcal{C}_1 = \mathcal{C}_2$ and $\mathcal{C}_1$ is a linear code.

Note that in the last example we used the same symbols to recover both $x_1^{(1)}$ and $x_2^{(1)}$ and thus their recovery sets were not disjoint. Recall that in our definition of switch codes we only require to read at most one symbol from each column and thus it is possible to use the same read symbols to recover different requested symbols. If we were to use the other definition of switch codes [20], where the recovery sets have to be disjoint, the construction in the example would not have satisfied the requirements of a switch code.

An $(n, k, m, t)_q$-ℓ-consecutive switch code, $\{\mathcal{C}_T\}_{T\in[\ell]}$, is called **combinatorial** if there exists a matrix $F = (F_{T,j})_{T\in[\ell],\ j\in[m]}$ that takes values in $[n]$, such that for every $T \in [\ell]$, $\mathbf{c}^{(T)} = \mathcal{E}_{\mathcal{C}_T}(\mathbf{x}^{(T)}) = x_{F_{T,1}}^{(T)} x_{F_{T,2}}^{(T)} \ldots x_{F_{T,m}}^{(T)}$. The matrix F is called the **index matrix** of the switch code. Note that $\{\mathcal{C}_T\}_{T\in[\ell]}$ is completely determined by its index matrix F. Intuitively, a combinatorial ℓ-consecutive switch code does not use any coding to encode a string $\mathbf{x}$, only copies of the entries of $\mathbf{x}$ in some order. Hence, the alphabet size q does not play an important role in the combinatorial case and therefore we omit the subscript q from the notation of such codes and assume that the symbols are taken from some alphabet Σ. A consecutive switch code in which the symbols can be encoded, as opposed to only repeated, is called **computational**; in particular, a combinatorial switch code is by definition also a computational switch code. By default, a

$$F = \begin{pmatrix} 1 & 2 & 3 & 4 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 & 4 & 4 & 4 \end{pmatrix}$$

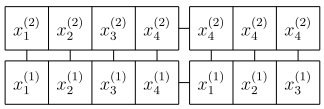


Fig. 2. Example of a combinatorial $(4, 7)$-2-consecutive switch code and its index matrix $F \in [4]^{2\times 7}$.

consecutive switch code is computational. For linear binary consecutive switch codes we define the **degree** to be the maximum number of input symbols that participate in the encoding of each parity symbol (using only XOR operations). In Section V we consider 2-consecutive switch codes in which the degree is two. These codes are in some sense the simplest computational consecutive switch codes that are not combinatorial.

As for general switch codes, we will consider only $(n = k, k, m, t = 1)_q$-ℓ-consecutive switch codes and we denote these codes by $(k, m)_q$-ℓ-consecutive switch codes (as mentioned above, in the combinatorial case we omit the subscript q). For two positive integers k and ℓ, let $\mathcal{A}(k, \ell)$ be the smallest integer m for which a combinatorial (k, m)-ℓ-consecutive switch code exists. A combinatorial (k, m)-ℓ-consecutive switch code is called **optimal** if $m = \mathcal{A}(k, \ell)$. The simplest combinatorial (k, m)-ℓ-consecutive switch code is the ***ℓ*-repetition switch code** in which every code $\mathcal{C}^{(T)}$ encodes a string $\mathbf{x} \in \Sigma^k$ into $\mathcal{R}_\ell(\mathbf{x})$ (the concatenation of ℓ copies of $\mathbf{x}$), and thus $\mathcal{A}(k, \ell) \leq \ell k$. In Section IV we present a construction of combinatorial consecutive switch codes for every $\ell \in [2, k]$. In particular, we show that $\mathcal{A}(k, 2) = 2k - 1$, while $\mathcal{A}(k, \ell)$ is much smaller than $k\ell$ for $\ell \geq 3$. Figure 2 shows an optimal combinatorial 2-consecutive switch code with parameters $(k, m) = (4, 7)$, along with its index matrix. Notice that the codes $\{\mathcal{C}^{(T)}\}_{T\in[\ell]}$ are depicted in an increasing order of the index T from bottom to top, whereas the row indexing of the index matrix F follows the conventional row indexing for matrices, i.e., row indices increase from top to bottom.

IV. Combinatorial Consecutive Switch Codes

In this section we construct combinatorial (k, m)-ℓ-consecutive switch codes, for every $\ell \in [2, k]$ and also show a lower bound on $\mathcal{A}(k, 2)$, which assures that the construction for $\ell = 2$ is optimal.

We start with the simplest case in which $\ell = 2$.

Construction 1: Let $F \in [k]^{2\times 2k-1}$ be defined by

$$F = \begin{pmatrix} 1 & 2 & \cdots & k & 1 & 2 & \cdots & k-1 \\ 1 & 2 & \cdots & k & k & k & \cdots & k \end{pmatrix}.$$

Theorem 2: Let $\{\mathcal{C}_T\}_{T\in\{1,2\}}$ be the combinatorial switch code whose index matrix is the matrix F from Construction 1. Then $\{\mathcal{C}_T\}_{T\in\{1,2\}}$ is a $(k, m = 2k - 1)$-2-consecutive switch code.

Proof: We have to show that for every set of k pairs $I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [k] \times \{1, 2\}$ there exist k distinct indices $j_1, j_2, \ldots, j_k$, such that $F_{T_r, j_r} = i_r$ for all $r \in [k]$.

The indices $j_1, j_2, \ldots, j_k$ are defined as follows. If $(i_r, T_r) = (i, 1)$ for some $i \in [k - 1]$ then $j_r = i + k$. If $(i_r, T_r) = (i, 2)$ for some $i \in [k - 1]$ then $j_r = i$. The column index k can be used for the recovery of one pair of the form (k, T). If both $(k, 1)$ and $(k, 2)$ were requested then there exists $i \in [k - 1]$ such that $(i, 1) \notin I$ and thus $(k, 2)$ can be recovered from $j = i + k$. ∎

Construction 1 provides us with 2-consecutive switch codes in which $m = 2k - 1$, i.e., m is only one less than the length of the trivial 2-repetition switch code. The next theorem states that the code from Construction 1 is optimal, namely $\mathcal{A}(k, 2) = 2k - 1$.

Theorem 3: If $\{\mathcal{C}_T\}_{T\in\{1,2\}}$ is a combinatorial (k, m)-2-consecutive switch code, then $m \geq 2k - 1$.

Proof: Let F be the index matrix of $\{\mathcal{C}_T\}_{T\in[2]}$ and let $G(L, R, E)$ be the bipartite graph whose left and right vertex sets are $L = [k]$ and $R = [k]$, respectively, and whose edge set E consists of all the edges of the form $e_j = (F_{1,j}, F_{2,j})$, $j \in [m]$. (E may contain parallel edges.) In particular, $|E| = m$. Since $\{\mathcal{C}_T\}_{T\in\{1,2\}}$ is a (k, m)-2-consecutive switch code, it follows that for every set of k pairs $I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [k] \times \{1, 2\}$ there exist k distinct indices $j_1, j_2, \ldots, j_k$ such that $F_{T_r, j_r} = i_r$, for all $r \in [k]$. This implies that for all $S_1 \subseteq L$ and $S_2 \subseteq R$, where $s = |S_1| + |S_2| \leq k$, there exist s distinct edges of the form (u, v), where $u \in S_1$ or $v \in S_2$.

Assume to the contrary that $m \leq 2k - 2$. We will show the existence of $S_1 \subseteq L$ and $S_2 \subseteq R$, where $s = |S_1| + |S_2| \leq k$, such that the number of edges (u, v) for which $u \in S_1$ or $v \in S_2$ is less than s and derive a contradiction.

Since $|E| = m < |L| + |R| - 1$, it follows that G contains $b \geq 2$ connected components, $G_1(L_1, R_1, E_1)$, $G_2(L_2, R_2, E_2), \ldots, G_b(L_b, R_b, E_b)$. We claim that at least two of these connected components are trees. Indeed, if none of these connected components is a tree, then $|E_i| \geq |L_i| + |R_i|$ for all $i \in [b]$ and hence $|E| \geq 2k$. If only one connected component is a tree then $|E| \geq 2k-1$. Assume without loss of generality that G_1 and G_2 are trees and that $|L_1| + |R_1| \leq |L_2| + |R_2|$. Let $S_1 = L_1$ and $S_2 = R_1$. Then $s = |S_1| + |S_2| \leq (|L| + |R|)/2 \leq k$. Notice that since G_1 is a connected component, it follows that for every edge $(u, v) \in E$, $u \in S_1$ if and only if $v \in S_2$, and there exist exactly $|E_1|$ edges that connect an element of S_1 with an element of S_2. Since G_1 is a tree, it follows that $|E_1| = |S_1| + |S_2| - 1 < s$, a contradiction. ∎

Given a matrix $F \in [k]^{\ell\times m}$ we define the **index graph** of F to be the bipartite graph $G_F(L, R, E)$, with left vertex set $L = [k]$, right vertex set $R = [m]$, and an edge set E that consists of all the pairs of the form $(i, j) \in L \times R$, such that $F_{T,j} = i$, for some $T \in [\ell]$. Intuitively, the set R corresponds to the columns of the matrix F, the set L corresponds to all possible entries of F, and an edge (i, j) indicates that i appears in the jth column of F. Note that E may contain parallel edges if some symbol appears more than once in some column of F. Furthermore, the graph G_F has the property that the degree of each vertex in R is exactly ℓ. An example of a matrix $F \in [6]^{3\times 4}$ and its index graph are given in Figure 3.

$$F = \begin{pmatrix} 3 & 4 & 5 & 6 \\ 4 & 3 & 6 & 5 \\ 1 & 2 & 1 & 2 \end{pmatrix}$$

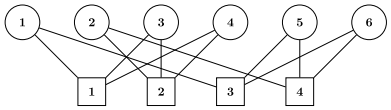


Fig. 3. Example of a matrix $F \in [6]^{3\times 4}$ and its index graph, where the vertex sets L and R are represented by the circles and squares, respectively. Note that this graph is also a $(6, 4, 3, 3)$-matching graph.

$$M(3,5) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \\ 3 & 4 & 5 & 1 & 2 \end{pmatrix}$$

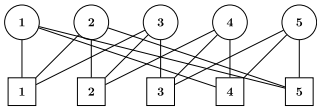


Fig. 4. The matrix $M(\omega, \delta)$ and its index graph $G_{\omega,\delta}$ for $\omega = 5$ and $\delta = 3$. Again, the circle nodes represent the left vertex set L, while the square nodes represent the right vertex set R.

Given a bipartite graph $G(L, R, E)$, for every $S \subseteq L$ we define $N(S) \subseteq R$ to be the set of all vertices in R that are connected by an edge to some vertex in S.

Definition 4: A bipartite graph $G(L, R, E)$ is called a ***(k, ν, ℓ, r)-matching graph*** *if the following hold.*

1) *Its left and right vertex sets are of sizes $|L| = k$ and $|R| = \nu$, respectively.*
2) *The degree of each vertex in R is ℓ.*
3) *If $S \subseteq L$ is of size $s \leq r$ then $|N(S)| \geq s$.*

By Hall's theorem [8], condition (3) is equivalent to the condition that for every $S \subseteq L$ of size $s \leq r$ there exists a **matching**, i.e., there exists s disjoint edges from S to R. Figure 3 illustrates a $(6, 4, 3, 3)$-matching graph. Matching graphs are, in a sense, a special type of **expander graphs** [9]. However, we are not aware of any result on expander graphs that fits this description of matching graphs. The study of bipartite expander graphs focuses on the setting in which the degree restriction in item (2) is either omitted or imposed on the vertex set L. Moreover, the neighborhood of $S \subseteq L$ is required to "expand" S, i.e. to be much larger than the set S, and not only to be at least of the same size as S, as required in item (3).

We will show how matching graphs can be useful to construct combinatorial ℓ-consecutive switch codes, but first we need one more definition. The **row cyclic shift mapping** $RS : [k]^{\ell\times m} \to [k]^{\ell\times m}$ is defined by $(RS(F))_{i,j} \stackrel{\text{def}}{=} F_{i-1,j}$, for $i \in [2, \ell]$, and $(RS(F))_{1,j} \stackrel{\text{def}}{=} F_{\ell,j}$, for all $j \in [m]$. Define $RS^0(F) \stackrel{\text{def}}{=} F$ and for $i \in [\ell - 1]$, define the ***i*th row cyclic shift** of a matrix F by

$$RS^i(F) \stackrel{\text{def}}{=} \underbrace{RS \circ RS \circ \cdots \circ RS}_{i \text{ times}}(F).$$

Construction 2: Let $D \in [k]^{\ell\times\nu}$ be a matrix whose index graph is a $(k, \nu, \ell, k/2)$-matching graph and let $m = k + (\ell - 1)\nu$. Define the matrix $F^{(SC)} \stackrel{\text{def}}{=} (F_1|F_2|\cdots|F_\ell) \in [k]^{\ell\times m}$, where $F_1 \in [k]^{\ell\times k}$ is the matrix

$$F_1 \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 2 & \cdots & k \\ 1 & 2 & \cdots & k \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2 & \cdots & k \end{pmatrix}$$

and for all $b \in [2, \ell]$, $F_b \stackrel{\text{def}}{=} RS^{b-2}(D)$.

Theorem 4: Let $\{\mathcal{C}_T\}_{T\in[\ell]}$ be the combinatorial switch code whose index matrix is the matrix $F^{(SC)}$ from Construction 2. Then $\{\mathcal{C}_T\}_{T\in[\ell]}$ is a (k, m)-ℓ-consecutive switch code.

Proof: We have to show that for every set of k pairs $I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [k] \times [\ell]$ there exist k distinct indices $j_1, j_2, \ldots, j_k$, such that $F^{(SC)}_{T_r, j_r} = i_r$. Note that $F^{(SC)}_{T,i} = (F_1)_{T,i} = i$, for all $(i, T) \in [k] \times [\ell]$. For all $r \in [k]$, if there is no $s \in [k] \setminus \{r\}$ such that $i_s = i_r$ then we set $j_r = i_r$. Hence, we can assume without loss of generality that for every $r \in [k]$, there exists $s \in [k] \setminus \{r\}$, such that $i_r = i_s$. Let $S = \{i \ : \ \exists r \in [k],\ i_r = i\}$. Then $|S| \leq k/2$. Since the index graph of the matrix D, $G_D(L, R, E)$, is a $(k, \nu, \ell, k/2)$-matching graph, it follows that there exists a matching for $S \subset L$ in the graph $G_D(L, R, E)$. This implies that there exist $|S|$ distinct integers $\hat{j}_1, \hat{j}_2, \ldots, \hat{j}_{|S|} \in [\nu]$ such that for all $1 \leq r \leq |S|$, i_r appears in the $\hat{j}_r$th column of D. Since, $F_b = RS^{b-2}(D)$, for all $2 \leq b \leq \ell$, it follows that i_r appears in $\ell - 1$ distinct rows of the columns of $F^{(SC)}$ indexed by $\hat{j}_r+k$, $\hat{j}_r+k+\nu, \ldots, \hat{j}_r+k+(\ell-2)\nu$. If for some $b \in [0, \ell-2]$, $F^{(SC)}_{T_r, \hat{j}_r+k+b\nu} = i_r$ then we set $j_r = \hat{j}_r + k + b\nu$. Otherwise, we set $j_r = i_r$ and we have that $F^{(SC)}_{T_r, j_r} = i_r$. In this case, for every $s \neq r$, if $i_s = i_r$ then there exists a unique $b \in [0, \ell - 2]$ such that $F^{(SC)}_{T_s, \hat{j}_s+k+b\nu} = i_s$. ∎

Notice that the last $k - 1$ columns of the matrix F from Construction 1 define a matrix whose index graph is a $(k, k - 1, 2, k/2)$-matching graph. Therefore, Construction 1 is a special case of Construction 2 and Theorem 4 is a generalization of Theorem 2. Next, an example of a 3-consecutive switch code is given.

Example 3: Let

$$D = \begin{pmatrix} 3 & 4 & 5 & 6 \\ 4 & 3 & 6 & 5 \\ 1 & 2 & 1 & 2 \end{pmatrix}$$

be the matrix from Figure 3, whose index graph is a $(6, 4, 3, 3)$-matching graph. Then

$$F^{(sc)} = \left(\begin{array}{cccccc|cccc|cccc} 1 & 2 & 3 & 4 & 5 & 6 & 3 & 4 & 5 & 6 & 1 & 2 & 1 & 2 \\ 1 & 2 & 3 & 4 & 5 & 6 & 4 & 3 & 6 & 5 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 1 & 2 & 1 & 2 & 4 & 3 & 6 & 5 \end{array}\right)$$

is the index matrix of a combinatorial $(6, 4)$-3-consecutive switch code.

In order to apply Theorem 4 we must construct a $(k, \nu, \ell, k/2)$-matching graph. To this end, for every pair of positive integers δ and ω, $\delta \leq \omega$, we define the bipartite graph $G_{\omega,\delta}(L, R, E)$ where $L = [\omega]$, $R = [\omega]$ and $E = \{(u, v) \in L \times R \ : \ \exists r \in [\delta],\ u \equiv v + r - 1 \pmod{\omega}\}$.

The graph $G_{\omega,\delta}$ is the index graph of the matrix $M(\omega, \delta) \in [\omega]^{\delta\times\omega}$ defined by

$$(M(\omega, \delta))_{i,j} \equiv i + j - 1 \pmod{\omega}.$$

Figure 4 shows the matrix $M(\omega, \delta)$ and the graph $G_{\omega,\delta}$ for $\omega = 5$ and $\delta = 3$.

For a bipartite graph $G(L, R, E)$ and for $X \subset R$ we denote by $G(L, R, E)\backslash X$ the subgraph of $G(L, R, E)$ that is obtained from G by removing from R all the vertices of X and all

the edges of the form $(u, v) \in L \times X$. The next lemma states a property of the graph $G_{\omega,\delta}$ that will be useful for our construction of a $(k, \nu, \ell, k/2)$-matching graph.

Lemma 4: Let $\delta \in [\omega]$ and consider the graph $G_{\omega,\delta}(L, R, E)$. If $X \subset R$ is of size $\delta - 1$ then the graph $G_{\omega,\delta} \setminus X$ is an $(\omega, \omega - \delta + 1, \delta, \omega - \delta + 1)$-matching graph.

Proof: We prove the lemma by induction on δ and ω. For the basis of induction we need to verify that the lemma holds for $\delta = 1$ and for every $\omega \geq 1$. This case is trivial, since X can only be the empty set and $G_{\omega,1}$ is an $(\omega, \omega, 1, \omega)$-matching graph.

For the induction hypothesis we assume that we can remove any $\delta - 2$ vertices from the right vertex set of $G_{\omega-1,\delta-1}$ and obtain an $(\omega - 1, \omega - \delta + 1, \delta - 1, \omega - \delta + 1)$-matching graph.

For the induction step, let $S \subset L$ and $X \subset R$ have sizes $\omega - \delta + 1$ and $\delta - 1$, respectively. By symmetry, we assume without loss of generality that $\omega \notin S$. Let $b = \max\{v : v \in X \setminus [\omega - \delta + 2, \omega - 1]\}$. Since $|X| = \delta - 1$ and $|[\omega - \delta + 2, \omega - 1]| = \delta - 2$, it follows that b is well defined. We claim that $G_{\omega,\delta} \setminus \{b\}$ (the subgraph of $G_{\omega,\delta}$ that is obtained by removing the vertex b from R) contains a subgraph, $\widehat{G}$, that is isomorphic to the graph $G_{\omega-1,\delta-1}$ by a relabeling of its right vertex set (and using the identity mapping on the left vertex set). If we can prove this claim then by the induction hypothesis $\widehat{G} \setminus (X \setminus \{b\})$ is an $(\omega - 1, \omega - \delta + 1, \delta - 1, \omega - \delta + 1)$-matching graph with left vertex set $[\omega - 1]$. Since $\omega \notin S$ and $|S| = \omega - \delta + 1$, it follows that there exists a matching for S in $\widehat{G} \setminus (X \setminus \{b\})$ and therefore also in $G_{\omega,\delta} \setminus X$, as desired.

Hence, to conclude the proof of the lemma we prove the existence of the subgraph $\widehat{G}$. We distinguish between two cases according to whether or not $b = \omega$.

Case 1: $b \neq \omega$. For the relabeling of the vertex set $R \setminus \{b\}$, let $\phi : [\omega] \setminus \{b\} \to [\omega - 1]$ be the bijection defined by

$$\phi(v) = \begin{cases} v + 1, & \text{if } v \in [b - 1] \\ v, & \text{if } v \in [b + 1, \omega - 1] \\ 1, & \text{if } v = \omega. \end{cases}$$

It is sufficient to show that the set of edges $\widetilde{E} = \{(u, v) \in [\omega - 1] \times [\omega] \setminus \{b\} : \exists r \in [\delta - 1],\ u \equiv \phi(v) + r - 1 \pmod{\omega - 1}\}$ is contained in the edge set of $G_{\omega,\delta} \setminus \{b\}$. Assume that $(u, v) \in \widetilde{E}$. If $v \in [b - 1]$, then since $b \leq \omega - \delta + 1$ it follows that $u = v + r$ for some $r \in [\delta - 1]$ and therefore $u \equiv v + r - 1 \pmod{\omega}$ for some $r \in [2, \delta]$. Hence, (u, v) is an edge of $G_{\omega,\delta} \setminus \{b\}$. If $v \in [b + 1, \omega - 1]$ or $v = \omega$ then either $u = v + r - 1$ or $u = v + r - \omega$, for some $r \in [\delta - 1]$, depending on whether or not $v + r - 1 < \omega$. In any case, it is an edge of $G_{\omega,\delta} \setminus \{b\}$.

Case 2: $b = \omega$. In this case we define the relabeling of the vertex set $R \setminus \{\omega\}$ to be the identity function. It is sufficient to show that the set of edges $\widetilde{E} = \{(u, v) \in [\omega - 1] \times [\omega - 1] : \exists r \in [\delta - 1],\ u \equiv v + r - 1 \pmod{\omega - 1}\}$ is contained in the edge set of $G_{\omega,\delta} \setminus \{\omega\}$. Assume that $(u, v) \in \widetilde{E}$. Then either $u = v + r - 1$ or $u = v + r - \omega$, for some $r \in [\delta - 1]$, depending on whether or not $v + r - 1 < \omega$. In any case, it is an edge of $G_{\omega,\delta} \setminus \{\omega\}$. ∎

Given a matrix $M = (M_{i,j}) \in [\omega]^{\delta \times \omega}$ and a positive integer α, define the matrix $M + \alpha \in [1 + \alpha, \omega + \alpha]^{\delta \times \omega}$, where $(M + \alpha)_{i,j} = M_{i,j} + \alpha$, for all $i \in [\delta]$ and $j \in [\omega]$.

Construction 3: Let $k = (\ell - 2) f (f + 1)$, for some positive integer f, $\omega = (\ell - 2) f$, and $\delta = \ell - 1$. Define the matrix $D \in [k]^{\ell \times (\ell-2) f^2}$ by

$$D \stackrel{\text{def}}{=} \left(\begin{array}{c|c|c|c} M_1 & M_2 & \cdots & M_f \\ \mathbf{x} & \mathbf{x} & \cdots & \mathbf{x} \end{array} \right),$$

where $\mathbf{x} = (k - \omega + 1, k - \omega + 2, \ldots, k)$ and for all $j \in [f]$, $M_j = M(\omega, \delta) + (j - 1)\omega \in [(j - 1)\omega + 1, j\omega]^{\delta \times \omega}$.

Theorem 5: The index graph of the matrix D from Construction 3 is a $(k, k - \omega, \ell, k/2)$-matching graph.

Proof: By the definition of the index graph of D, $G_D(L, R, E)$, we have that $L = [k]$, $R = [k - \omega]$, and every vertex $v \in R$ has degree ℓ. Since the index graph of M_j, $j \in [f]$, is equivalent to $G_{\omega,\delta}$, it follows that G_D contains f disjoint subgraphs $G_1(L_1, R_1, E_1), G_2(L_2, R_2, E_2), \ldots, G_f(L_f, R_f, E_f)$, $L_j = [(j - 1)\omega + 1, j\omega]$ and $R_j = [(j - 1)\omega + 1, j\omega]$, for all $j \in [f]$, where each subgraph is equivalent to $G_{\omega,\delta}$.

Let $S \subset L = [k]$ of size $|s| \leq k/2$. We need to show that $|N(S)| \geq s$. For all $j \in [f + 1]$, let $S_j = S \cap L_j$, $s_j = |S_j|$, and let $d_j = \min\{\omega - s_j, \ell - 2\}$. By Lemma 4, for all $j \in [f]$, we can choose any set of d_j vertices of R_j, X_j, and find a matching for S_j in $G_j \setminus X_j$. Therefore, by the definition of D, we have the flexibility to choose the sets X_j, so we will have a matching for the set $(\cup_{j=1}^{f} S_j) \cup B$, where B is any subset of $[k - \omega + 1, k]$ of size $\sum_{j=1}^{f} d_j$. In particular, if $s_{f+1} \leq \sum_{j=1}^{f} d_j$, then we have a matching for S. Otherwise, let $r = |\{j \in [f] : d_j = \omega - s_j < \ell - 2\}|$. Notice that since $s_{f+1} > \sum_{j=1}^{f} d_j$ we have that

$$\begin{aligned} s = \sum_{j=1}^{f+1} s_j &> \sum_{j=1}^{f} s_j + d_j = r\omega + \sum_{\substack{j \in [f]: \\ d_j = \ell - 2}} s_j + d_j \\ &\geq r\omega + (f - r)(\ell - 2) \end{aligned}$$

and hence

$$r\omega + (f - r)(\ell - 2) < \frac{k}{2} = \frac{\omega(f + 1)}{2},$$

which implies that $r < \frac{f-1}{2}$. Hence,

$$s_{f+1} > \sum_{j=1}^{f} d_j \geq (\ell - 2)(f - r) \geq \frac{(\ell - 2)(f + 1)}{2}.$$

By the definition of G_D, it follows that $|N(u_1)| = f$ and $N(u_1) \cap N(u_2) = \emptyset$, for all $u_1, u_2 \in [k - \omega + 1, k]$. Therefore,

$$\begin{aligned} |N(S)| &\geq |N(S_{f+1})| = s_{f+1} f \\ &> \frac{(\ell - 2)(f + 1) f}{2} = k/2 \geq s. \end{aligned}$$

∎

Combining Theorems 4 and 5 we conclude the following.

Corollary 1: If $k = (\ell - 2) f (f + 1)$, for some positive integer f, then

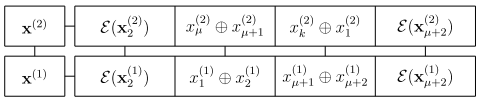
$$\mathcal{A}(k, \ell) \leq \ell k - (\ell - 1)(\ell - 2) f \approx \ell k - (\ell - 1)\sqrt{(\ell - 2)k}.$$

V. 2-Consecutive Switch Code of Degree 2

In this section we show a construction of a linear binary $(k, m = 2k-2)$-2-consecutive switch code of degree two, i.e., each parity symbol is formed by the XOR operations of at most two input symbols. This construction reduces the redundancy by one over the optimal combinatorial (degree-1) 2-consecutive switch code.

Construction 4: Given an even integer k, let $\mu = k/2$ and let $\mathcal{E} : \mathbb{F}_2^{\mu-1} \to \mathbb{F}_2^{\mu-2}$ be the encoder that maps $\mathbf{x} = (x_1, x_2, \ldots, x_{\mu-1}) \in \mathbb{F}_2^{\mu-1}$ to $\mathcal{E}(\mathbf{x}) = (\mathcal{E}(\mathbf{x})_1, \mathcal{E}(\mathbf{x})_2, \ldots, \mathcal{E}(\mathbf{x})_{\mu-2})$, where $\mathcal{E}(\mathbf{x})_i = x_i \oplus x_{i+1}$, for all $i \in [\mu-2]$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the two $[m = 2k-2, k]$-codes defined as follows.

- *For $T \in \{1, 2\}$, decompose the Tth input string $\mathbf{x}^{(T)}$ as $\mathbf{x}^{(T)} = (x_1^{(T)}, \mathbf{x}_2^{(T)}, x_{\mu+1}^{(T)}, \mathbf{x}_{\mu+2}^{(T)})$, where $\mathbf{x}_2^{(T)} = (x_2^{(T)}, x_3^{(T)}, \ldots, x_\mu^{(T)})$ and $\mathbf{x}_{\mu+2}^{(T)} = (x_{\mu+2}^{(T)}, x_{\mu+3}^{(T)}, \ldots, x_k^{(T)})$.*
- *The string $\mathbf{x}^{(T)}$ is systematically encoded to $\mathbf{c}^{(T)}$ as shown in the following array.*



Theorem 6: Let $\{\mathcal{C}_T\}_{T\in\{1,2\}}$ be the code specified by Construction 4 for some even k. Then $\{\mathcal{C}_T\}_{T\in\{1,2\}}$ is a $(k, m = 2k-2)$-2-consecutive switch code of degree two.

Proof: First, notice that, for all $T \in \{1, 2\}$, $\mathbf{x}_2^{(T)}, \mathbf{x}_{\mu+2}^{(T)} \in \mathbb{F}_2^{\mu-1}$ and hence $\mathcal{E}(\mathbf{x}_2^{(T)})$ and $\mathcal{E}(\mathbf{x}_{\mu+2}^{(T)})$ are correctly defined vectors of length $\mu-2$, given by $\mathcal{E}(\mathbf{x}_2^{(T)})_i = x_{i+1}^{(T)} \oplus x_{i+2}^{(T)}$ and $\mathcal{E}(\mathbf{x}_{\mu+2}^{(T)})_i = x_{\mu+i+1}^{(T)} \oplus x_{\mu+i+2}^{(T)}$, for all $i \in [\mu-2]$. Therefore, the code $\mathcal{C}^{(T)}$ is indeed of length $m = k+2(\mu-2)+2 = 2k-2$ and degree two.

Given a request set $I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [k] \times \{1, 2\}$ we need to show the existence of a recovery set for I. To this end we distinguish between 6 cases.

Case 1: There exist $r \in [3, \mu]$, $s \in [\mu+3, k]$, and $\widehat{T}_r, \widehat{T}_s \in \{1, 2\}$, such that $(r, \widehat{T}_r), (s, \widehat{T}_s) \notin I$. We assume that $\widehat{T}_r = \widehat{T}_s = 1$, however it will be clear from the proof that different choices of $\widehat{T}_r$ and $\widehat{T}_s$ can be handled similarly.

Since $I \subset \widehat{I} = ([k] \times \{1, 2\}) \setminus \{(r, 1), (s, 1)\}$, it suffices to show a recovery set for $\widehat{I}$. We will first show how to recover $[2, \mu] \times \{1, 2\} \setminus \{(r, 1)\}$ by reading systematically at most one of the symbols $x_j^{(1)}, x_j^{(2)}$, $j \in [2, \mu]$, and from $\mathcal{E}(\mathbf{x}_2^{(2)})$. In the first step, the symbol $x_r^{(2)}$ is read systematically from $\mathbf{x}^{(2)}$. We then continue in a sequential manner. For every $i \in [r+1, \mu]$, assume that $x_{i-1}^{(2)}$ was already recovered. Then the symbol $x_i^{(1)}$ is read systematically and the symbol $x_i^{(2)}$ is recovered from $x_{i-1}^{(2)}$ and $\mathcal{E}(\mathbf{x}_2^{(2)})_{i-2} = x_{i-1}^{(2)} \oplus x_i^{(2)}$. Similarly, for every $i \in [2, r-1]$, assume that $x_{i+1}^{(2)}$ was already recovered. Then the symbol $x_i^{(1)}$ is read systematically and the symbol $x_i^{(2)}$ is recovered from $x_{i+1}^{(2)}$ and $\mathcal{E}(\mathbf{x}_2^{(2)})_{i-1} = x_i^{(2)} \oplus x_{i+1}^{(2)}$.

From symmetry we have that $[\mu+2, k] \times \{1, 2\} \setminus \{(s, 1)\}$ can be recovered by reading systematically at most one of the symbols $x_j^{(1)}, x_j^{(2)}$, $j \in [\mu+2, k]$, and from $\mathcal{E}(\mathbf{x}_{\mu+2}^{(2)})$.

It remains to show how to recover $x_1^{(T)}$ and $x_{\mu+1}^{(T)}$, $T \in \{1, 2\}$. The symbols $x_1^{(2)}$ and $x_{\mu+1}^{(2)}$ are read systematically. Since we already recovered $x_2^{(1)}$ and $x_{\mu+2}^{(1)}$ (regardless of the values of $\widehat{T}_r$ and $\widehat{T}_s$), we can recover $x_1^{(1)}$ and $x_{\mu+1}^{(1)}$ from the parities $x_1^{(1)} \oplus x_2^{(1)}$ and $x_{\mu+1}^{(1)} \oplus x_{\mu+2}^{(1)}$, respectively.

Thus, we have a recovery set for $\widehat{I}$ in which at most one entry from each column of the array is read. Such a recovery set for $\widehat{I}$, where $k = 8$, is given in Example 4.

Case 2: $I = [1, \mu] \times \{1, 2\}$. In this case the symbol $x_1^{(1)}$ is read systematically and we use the parity symbol $x_k^{(2)} \oplus x_1^{(2)}$ and the systematic symbol $x_k^{(2)}$ to recover $x_1^{(2)}$. We then use the parity symbol $x_1^{(1)} \oplus x_2^{(1)}$ together with $x_1^{(1)}$ to recover $x_2^{(1)}$. As in case 1, for every $i \in [3, \mu]$, if $x_{i-1}^{(1)}$ was already recovered then the symbol $x_i^{(2)}$ can be read systematically and the symbol $x_i^{(1)}$ can be recovered from $x_{i-1}^{(1)}$ and $\mathcal{E}(\mathbf{x}_2^{(1)})_{i-2} = x_{i-1}^{(1)} \oplus x_i^{(1)}$.

Case 3: There exist two indices $r, s \in [k] \setminus [2, \mu]$ and $\widehat{T}_r, \widehat{T}_s \in \{1, 2\}$ such that $I = ([2, \mu] \times \{1, 2\}) \cup \{(r, \widehat{T}_r), (s, \widehat{T}_s)\}$. It is easy to see that the only non-trivial sub-case is when $(1, 2) \in I$. In all other cases, $x_2^{(1)}$ can be recovered by $x_1^{(1)}$ and $x_1^{(1)} \oplus x_2^{(1)}$, from which point the remaining symbols can be recovered in a sequential manner as in case 2. Now assuming $(1, 2) \in I$, if $(k, 1) \notin I$, then $x_1^{(2)}$ can be recovered from $x_k^{(2)}$ and $x_k^{(2)} \oplus x_1^{(2)}$ and $x_1^{(1)}$ is read systematically for the recovery of the remaining symbols in a sequential manner as in case 2. If $(k, 1) \in I$, then $x_k^{(1)}$ can be recovered by reading $x_{k-1}^{(1)}$ systematically and from $\mathcal{E}(\mathbf{x}_{\mu+2}^{(1)})_{\mu-2} = x_{k-1}^{(1)} \oplus x_k^{(1)}$, from which point the remaining symbols can be recovered as if $(k, 1) \notin I$.

Case 4: There exist two indices $r, s \in [2, k] \setminus [3, \mu]$ and $\widehat{T}_r, \widehat{T}_s \in \{1, 2\}$ such that $I = ([3, \mu] \times \{1, 2\}) \cup \{(1, 1), (1, 2), (r, \widehat{T}_r), (s, \widehat{T}_s)\}$. It is easy to see that the only non-trivial sub-case is when $(2, 2), (k, 1) \in I$. In all other cases one of the symbols $x_1^{(1)}$ or $x_1^{(2)}$ can be recovered by either $x_2^{(1)}$ and $x_1^{(1)} \oplus x_2^{(1)}$ or by $x_k^{(2)}$ and $x_k^{(2)} \oplus x_1^{(2)}$, after which the remaining requests can be recovered in a sequential manner as in case 2. Now assuming $(2, 2), (k, 1) \in I$, we can recover $x_k^{(1)}$ by reading the systematic symbol $x_{k-1}^{(1)}$ and the parity symbol $\mathcal{E}(\mathbf{x}_{\mu+2}^{(1)})_{\mu-2} = x_{k-1}^{(1)} \oplus x_k^{(1)}$ and then recover $x_1^{(2)}$ by reading the systematic symbol $x_k^{(2)}$ and parity symbol $x_k^{(2)} \oplus x_1^{(2)}$. The remaining requests in $([3, \mu] \times \{1, 2\}) \cup \{(2, 2)\}$ can now be recovered in both generations similarly to case 2.

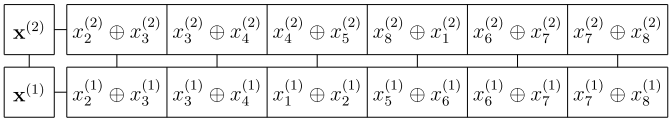
Case 5: All indices $[3, \mu]$ are requested from both generations and neither of index 1 or 2 is requested from both generations. This case is trivially easier than case 4.

Case 1 covers the scenario in which there exist $r \in [3, \mu]$ and $s \in [\mu+3, k]$ that were both requested from at most one generation. Cases 2–5 cover the scenario in which all indices $[3, \mu]$ are requested from both generations. The sixth case, where all indices from $[\mu+3, k]$ are requested from both generations, is symmetric to the union of cases 2–5. ∎

Example 4: The following array forms a binary $(k = 8, m = 14)$-2-consecutive switch code of degree 2.

If for example

$$I = \{(2, 1), (2, 2), (4, 1), (4, 2), (6, 1), (6, 2), (8, 1), (8, 2)\}$$



then we read $x_3^{(2)}$ from the systematic part and use the parity symbols $x_2^{(2)} \oplus x_3^{(2)}$ and $x_3^{(2)} \oplus x_4^{(2)}$ to recover $x_2^{(2)}$ and $x_4^{(2)}$, respectively. The symbols $x_2^{(1)}$ and $x_4^{(1)}$ are read systematically. Similarly, we can recover the symbols $x_6^{(1)}, x_6^{(2)}, x_8^{(1)}, x_8^{(2)}$ by reading $x_6^{(1)}, x_7^{(2)}, x_8^{(1)}$ systematically and from the parity symbols $x_6^{(2)} \oplus x_7^{(2)}$, $x_7^{(2)} \oplus x_8^{(2)}$. Thus, we recovered the set I. As shown in case 1 of the proof of Theorem 6, it is also possible to recover the symbols $x_1^{(1)}, x_1^{(2)}, x_5^{(1)}, x_5^{(2)}$ by reading $x_1^{(2)}$ and $x_5^{(2)}$ systematically and from the parity symbols $x_1^{(1)} \oplus x_2^{(1)}$ and $x_5^{(1)} \oplus x_6^{(1)}$.

VI. Computational Consecutive Switch Codes

In this section we study computational consecutive switch codes, i.e., consecutive switch codes that are not necessarily combinatorial, and show constructions of ℓ-consecutive switch codes for arbitrary k and ℓ.

Before we present our constructions, we review a few more fundamental concepts in coding theory. Let $\mathcal{C}$ be a (ν, μ)-code over $\mathbb{F}_q$. The **minimum distance** of the code $\mathcal{C}$ is defined by

$$d(\mathcal{C}) \stackrel{\text{def}}{=} \min\{d_H(\mathbf{u}, \mathbf{v}) \ : \ \mathbf{u}, \mathbf{v} \in \mathcal{C}, \ \mathbf{u} \neq \mathbf{v}\},$$

where $d_H(\mathbf{u}, \mathbf{v})$ is the **Hamming distance** between $\mathbf{u} = (u_1, u_2, \ldots, u_\nu)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_\nu)$ defined by

$$d_H(\mathbf{u}, \mathbf{v}) \stackrel{\text{def}}{=} |\{j \in [\nu] \ : \ u_j \neq v_j\}|.$$

It is well known that if the minimum distance of $\mathcal{C}$ is d then $\mathcal{C}$ can correct any $d-1$ erasures. More formally, given an encoder $\mathcal{E}_\mathcal{C} : \mathbb{F}_q^\mu \to \mathbb{F}_q^\nu$, there exists a decoder $\mathcal{D}_\mathcal{C} : (\mathbb{F}_q \cup \{?\})^\nu \to \mathbb{F}_q^\mu$ such that if $\mathbf{z} \in (\mathbb{F}_q \cup \{?\})^\nu$ is obtained from $\mathcal{E}_\mathcal{C}(\mathbf{x})$ by at most $d-1$ erasures, where an erasure is indicated by the symbol ?, then $\mathcal{D}_\mathcal{C}(\mathbf{z}) = \mathbf{x}$. Another interpretation of this property, which we will repeatedly use in the rest of the section, is that $\mathbf{x}$ is recoverable from any $\nu - d + 1$ entries of $\mathcal{E}(\mathbf{x})$. In this section we will use only linear codes and refer to a $[\nu, \mu]$-code with minimum distance d as a **$[\nu, \mu, d]$-code**.

Next, we present our constructions. Let us first start with the case of $\ell = 2$.

Construction 5: Let $\mathcal{C}$ be a $[k, \mu, d]$-code $\mathcal{C}$ over $\mathbb{F}_q$, where $d \geq \lfloor k/2 \rfloor + 1$ and let $\mathcal{E} : \mathbb{F}_q^\mu \to \mathbf{F}_q^k$ be a systematic encoder for $\mathcal{C}$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the two $[m = 2k - \mu, k]$-codes defined as follows.

- *For the first generation, $\mathbf{x}^{(1)} = (x_1^{(1)}, \ldots, x_k^{(1)}) \in \mathbb{F}^k$ is encoded to*

  $$\mathbf{c}^{(1)} = (x_1^{(1)}, \ldots, x_{k-\mu}^{(1)}, x_{k-\mu+1}^{(1)}, \ldots, x_k^{(1)}, y_1^{(1)}, \ldots, y_{k-\mu}^{(1)}),$$

  where

  $$\mathcal{E}(x_{k-\mu+1}^{(1)}, \ldots, x_k^{(1)}) = (x_{k-\mu+1}^{(1)}, \ldots, x_k^{(1)}, y_1^{(1)}, \ldots, y_{k-\mu}^{(1)}).$$

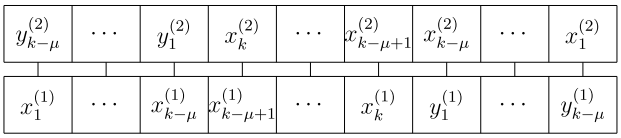


Fig. 5. Description of the $(k, m = 2k - \mu)_q$-2-consecutive switch code from Construction 5.

- *For the second generation, $\mathbf{x}^{(2)} = (x_1^{(2)}, \ldots, x_k^{(2)})$ is encoded to*

  $$\mathbf{c}^{(2)} = (y_{k-\mu}^{(2)}, \ldots, y_1^{(2)}, x_k^{(2)}, \ldots, x_{k-\mu+1}^{(2)}, x_{k-\mu}^{(2)}, \ldots, x_1^{(2)}),$$

  where

  $$\mathcal{E}(x_{k-\mu+1}^{(2)}, \ldots, x_k^{(2)}) = (x_{k-\mu+1}^{(2)}, \ldots, x_k^{(2)}, y_1^{(2)}, \ldots, y_{k-\mu}^{(2)}).$$

The resulting code sequence $\{\mathcal{C}_1, \mathcal{C}_2\}$ is illustrated in Figure 5.

Theorem 7: The code sequence $\{\mathcal{C}_1, \mathcal{C}_2\}$ from Construction 5 forms a $(k, m = 2k - \mu)_q$-2-consecutive switch code.

Proof: Since $\mathcal{C}$ is a $[k, \mu, d]$-code over $\mathbb{F}_q$, it follows that each $\mathbf{x} \in \mathbb{F}_q^\mu$ can be recovered from any $\lceil k/2 \rceil$ entries of $\mathcal{E}(\mathbf{x})$.

We next prove that $\{\mathcal{C}_1, \mathcal{C}_2\}$ satisfies the requirement of a $(k, 2k - \mu)_q$-2-consecutive switch code. Given a request set $I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [k] \times \{1, 2\}$, let k_1 and k_2 be the number of elements in I of the form $(i, 1)$ and $(i, 2)$, respectively, $i \in [k]$. In particular, $k_1 + k_2 = k$. Assume without loss of generality that $k_1 \leq k_2$. The k_1 symbols from the first generation are read systematically from the first k entries of $\mathbf{c}^{(1)}$. Hence, no symbol is read from the last $k - \mu$ columns of the first (bottom) row of the array (see Figure 5). As for the requested symbols from the second generation, every symbol from the first $k - \mu$ entries of $\mathbf{x}^{(2)}$ is read systematically from the last $k - \mu$ entries of $\mathbf{c}^{(2)}$, since no symbol was read from these columns for the first generation. In order to read (if necessary) the symbols from the last μ entries of $\mathbf{x}^{(2)}$ it suffices to read any $\lceil k/2 \rceil$ entries of $\mathcal{E}(x_{k-\mu+1}^{(2)}, \ldots, x_k^{(2)})$, which is stored in the first k columns of the array. This is possible, since the number of available columns (from which no symbol was read) is $k - k_1 \geq \lceil k/2 \rceil$. ∎

Corollary 2: There exists a $(k, m = 1.5k)_q$-2-consecutive switch code over a field of size $q + 1 \geq k$ and a binary $(k, m)_2$-2-consecutive switch code, where $m = 2k - \lfloor \log_2 k \rfloor$.

Proof: This corollary follows immediately from Theorem 7. One simply has to note that for $q > k$, there exists a $[k, \mu = \lceil k/2 \rceil, d = \lfloor k/2 \rfloor + 1]$-code $\mathcal{C}$ over $\mathbf{F}_q$, e.g., a Reed-Solomon code (if $k = q$ or $k = q + 1$, one might use extended or doubly-extended Reed-Solomon codes), and that there exists a binary $[k = 2^s - 1, \mu = s, d = \lfloor k/2 \rfloor + 1]$-code, e.g., a shortened Reed-Muller code with s variables and total degree one. (See [14] for more information on Reed-Solomon and Reed-Muller codes.) ∎

The next Theorem provides a lower bound on the length of a systematic 2-consecutive switch code.

Theorem 8: If $\{\mathcal{C}^{(T)}\}_{T \in \{1,2\}}$ is a systematic $(k, m)_q$-2-consecutive switch code, then $m \geq 1.5k$.

Proof: If the first $k/2$ entries from each of the strings $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are requested then at most $k/2$ requested entries can be recovered from the systematic part. Hence, in order to

recover the remaining symbols, the length of the parity part must be at least $k/2$. ∎

Notice that although the 2-consecutive switch code from Construction 5 is not systematic, if $q+1 \geq k$, a systematic $(k, 1.5k)_q$-2-consecutive switch code can be constructed by taking $\mathcal{C}^{(1)} = \mathcal{C}^{(2)} = \mathcal{C}$, where $\mathcal{C}$ is a systematic MDS code of length $1.5k$ and dimension k. Hence, in the systematic case the lower bound on m from Theorem 8 is tight.

We can generalize Construction 5 for an arbitrary number of rows.

Construction 6: For $\ell \geq 2$, $0 < \alpha$, integers $\mu = \alpha k$, $\nu = (\ell-1)(1-\alpha)k + \alpha k$, and $q > \nu$, let $\mathcal{C}$ be a $[\nu, \mu, d = \nu - \mu + 1]_q$-code over $\mathbb{F}_q$ (e.g., a Reed-Solomon code) and let $\mathcal{E} : \mathbb{F}_q^{\mu} \to \mathbb{F}_q^{\nu}$ be a systematic encoder for $\mathcal{C}$. Let $m = \ell(1-\alpha)k + \alpha k$ and let $\{\mathcal{C}_T\}_{T\in[\ell]}$ be a sequence of $[m, k]$-codes defined as follows.

- *For $T \in [\ell]$, the Tth generation input string $\mathbf{x}^{(T)}$ is partitioned into two parts, $\mathbf{x}^{(T)} = (\mathbf{x}_1^{(T)}, \mathbf{x}_2^{(T)})$, where $\mathbf{x}_1^{(T)} \in \mathbb{F}^{(1-\alpha)k}$ consists of the first $(1-\alpha)k$ entries of $\mathbf{x}^{(T)}$ and $\mathbf{x}_2^{(T)} \in \mathbb{F}^{\alpha k}$ consists of the last αk entries of $\mathbf{x}^{(T)}$.*
- *The string $\mathbf{x}^{(T)}$ is encoded to $\mathbf{c}^{(T)}$ which consists of two parts which we refer to as the left and right parts of $\mathbf{c}^{(T)}$. The left part consists of the first $\ell(1-\alpha)k$ entries, where we treat these entries as ℓ blocks, each of length $(1-\alpha)k$. The right part consists of the last αk entries.*
- *The first part of the input string, i.e. $\mathbf{x}_1^{(T)}$, is stored systematically in the $(1-\alpha)k$ entries of the Tth block of the left part of $\mathbf{c}^{(T)}$.*
- *The second part of the input string, i.e. $\mathbf{x}_2^{(T)}$, is stored systematically in the αk entries of the right part of $\mathbf{c}^{(T)}$.*
- *Let $\mathbf{y}^{(T)} \in \mathbb{F}^{\nu-\mu}$ be the string for which $\mathcal{E}(\mathbf{x}_2^{(T)}) = (\mathbf{x}_2^{(T)}, \mathbf{y}^{(T)})$. The string $\mathbf{y}^{(T)}$ is stored in the remaining blocks of $\mathbf{c}^{(T)}$ (with no specific order). Note that this step is possible since the number of available entries is*

$$(\ell-1)(1-\alpha)k = \nu - \mu.$$

Fig. 6 shows the structure of $\{\mathcal{C}_T\}_{T\in[\ell]}$.

Theorem 9: For $\alpha = \frac{(\ell-1)^2}{\ell(2\ell-3)}$, the sequence of codes $\{\mathcal{C}_T\}_{T\in[\ell]}$ from Construction 6 is a $(k, m)_q$-switch code with

$$m = \ell(1-\alpha)k + \alpha k = \left(\frac{\ell}{2} + \frac{3}{4} - \frac{1-3\ell/4}{2\ell(\ell-3/2)}\right)k.$$

Proof: Since $\mathcal{C}$ is a $[\nu, \mu = \alpha k, d = \nu - \mu + 1]$-code over $\mathbb{F}_q$, it follows that each $\mathbf{x} \in \mathbb{F}_q^{\mu}$ can be recovered from any $\mu = \alpha k$ entries of $\mathcal{E}(\mathbf{x})$.

We next prove that $\{\mathcal{C}_T\}_{T\in[\ell]}$ from Construction 6 satisfies the requirement of a $(k, m = \ell(1-\alpha)k + \alpha k)_q$-$\ell$-consecutive switch code. Given a request set $I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [k] \times [\ell]$, let k_T, $T \in [\ell]$, be the number of elements in I of the form (i, T), $i \in [k]$. In particular, $\sum_{T=1}^{\ell} k_T = k$. Assume for now that $k_1 \leq k_2 \leq \cdots \leq k_\ell$, while it will be clear from the proof that different orders can be handled similarly. The requested symbols are recovered by the following steps.

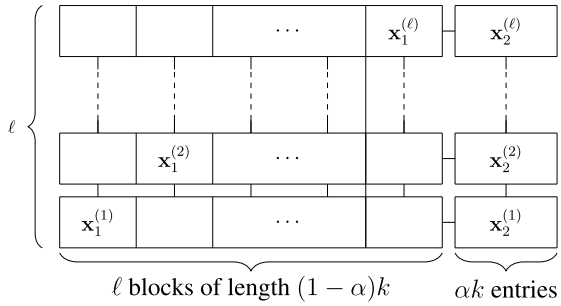


Fig. 6. Description of the $(k, m = \ell(1-\alpha)k + \alpha k)_q$-$\ell$-consecutive switch code from Construction 6. For all $T \in [\ell]$, the empty blocks in the Tth row store $\mathbf{y}^{(i)}$, where $\mathcal{E}(\mathbf{x}_2^{(i)}) = (\mathbf{x}_2^{(i)}, \mathbf{y}^{(i)})$.

1) The requested symbols from the first generation are read systematically from the k_1 entries where they are stored in the left and right parts of $\mathbf{c}^{(1)}$.
2) The requested symbols from the second generation are read in two steps. All requested symbols from $\mathbf{x}_1^{(2)}$ are read systematically from the second block in the left part of $\mathbf{c}^{(2)}$. If symbols from $\mathbf{x}_2^{(2)}$ are requested and cannot be read systematically, then we read any αk entries from $\mathbf{c}^{(2)}$ which belong to either the first block of its left part or its right part. For the success of this step we need to require that
$$(1-\alpha)k + \alpha k - k_1 \geq \alpha k$$
and since $k_1 \leq k/\ell$, it is enough to require that
$$\frac{\ell-1}{\ell} \geq \alpha = \frac{(\ell-1)^2}{\ell(2\ell-3)},$$
which holds for $\ell \geq 2$. Note that the number of columns used in this part is at most $k_2 + \alpha k$.
3) The requested symbols from the Tth generation, $T \in [3, \ell]$, are also read in two steps. All requested symbols from $\mathbf{x}_1^{(T)}$ are read systematically from the Tth block in the left part of $\mathbf{c}^{(T)}$. If symbols from $\mathbf{x}_2^{(T)}$ are requested, which again cannot be read systematically, then we read any αk symbols from available entries in $\mathbf{c}^{(T)}$, which belong to either blocks $1, 2, \ldots, T-1$ from its left part or its right part. For the success of this step we need to require that
$$(T-1)(1-\alpha)k + \alpha k - (k_1 + \cdots + k_{T-1}) - (T-2)\alpha k \geq \alpha k$$
and since $k_1 + k_2 + \cdots + k_{T-1} \leq (T-1)k/\ell$, it is enough to require that
$$\frac{(T-1)(\ell-1)}{\ell} \geq (2T-3)\alpha = (2T-3)\frac{(\ell-1)^2}{\ell(2\ell-3)},$$
which holds for all $T \in [3, \ell]$.

∎

Note that, for $\ell = 2$, we get from Theorem 9 the same result as in Corollary 2 of $m = 1.5k$. Indeed, Constructions 5 and 6 coincide for $\ell = 2$ up to changing the order of blocks in the array. In general, according to Theorem 9, we get a reduction of almost a half of the number of columns over a

trivial construction, which uses ℓk columns, or Construction 2, which uses about $(1-o(1))\ell k$ when ℓ is fixed. For $\ell = 3$, we get that the number of columns is $\frac{19}{9}k \approx 2.11k$. We show in the next construction how to improve this result to only $2k$ columns.

Construction 7: For an integer $\mu = k/3$ and for $q+1 \geq k$, let $\mathcal{C}$ be a $[k, \mu, d = 2k/3+1]$-code over $\mathbb{F}_q$ (e.g., a doubly extended Reed-Solomon code) and let $\mathcal{E} : \mathbb{F}_q^{k/3} \to \mathbb{F}_q^k$ be a systematic encoder for $\mathcal{C}$ such that $\mathcal{E}(\mathbf{x}) = (\mathbf{x}, \mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x}))$, for $\mathcal{E}_1, \mathcal{E}_2 : \mathbb{F}_q^{k/3} \to \mathbb{F}_q^{k/3}$. Let $\widehat{\mathcal{E}}_2 : \mathbb{F}_q^{2k/3} \to \mathbb{F}^{k/3}$ be defined by $\widehat{\mathcal{E}}_2(\mathbf{u}, \mathbf{v}) = \mathcal{E}_2(\mathbf{u}) + \mathcal{E}_2(\mathbf{v})$, for all $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^{k/3}$. Define a sequence of three $[m = 2k, k]$-codes $\{\mathcal{C}_T\}_{T\in[3]}$, by the following steps.

- *For $T \in [3]$, the Tth input string $\mathbf{x}^{(T)}$ is partitioned into three parts of length $k/3$, $\mathbf{x}^{(T)} = (\mathbf{x}_1^{(T)}, \mathbf{x}_2^{(T)}, \mathbf{x}_3^{(T)})$.*
- *The string $\mathbf{x}^{(T)}$ is encoded to $\mathbf{c}^{(T)}$ which consists of six blocks of length $k/3$.*
- *The structure of the codewords $\mathbf{c}^{(T)}$, $T \in [3]$ is shown in the following array.*

$\mathbf{x}_3^{(3)}$	$\mathcal{E}_1(\mathbf{x}_3^{(3)})$	$\widehat{\mathcal{E}}_2(\mathbf{x}_1^{(3)}, \mathbf{x}_3^{(3)})$	$\mathcal{E}_1(\mathbf{x}_1^{(3)})$	$\mathbf{x}_1^{(3)}$	$\mathbf{x}_2^{(3)}$
$\widehat{\mathcal{E}}_2(\mathbf{x}_1^{(2)}, \mathbf{x}_3^{(2)})$	$\mathcal{E}_1(\mathbf{x}_1^{(2)})$	$\mathbf{x}_1^{(2)}$	$\mathbf{x}_2^{(2)}$	$\mathbf{x}_3^{(2)}$	$\mathcal{E}_1(\mathbf{x}_3^{(2)})$
$\mathbf{x}_1^{(1)}$	$\mathbf{x}_2^{(1)}$	$\mathbf{x}_3^{(1)}$	$\mathcal{E}_1(\mathbf{x}_3^{(1)})$	$\widehat{\mathcal{E}}_2(\mathbf{x}_1^{(1)}, \mathbf{x}_3^{(1)})$	$\mathcal{E}_1(\mathbf{x}_1^{(1)})$

Theorem 10: The sequence of codes $\{\mathcal{C}_T\}_{T\in[3]}$ from Construction 7 forms a $(k, m = 2k)_q$-3-consecutive switch code.

Proof: For all $T \in [3]$, assume that k_T symbols are requested from the input string $\mathbf{x}^{(T)}$, where $k_1 + k_2 + k_3 = k$, and assume without loss of generality that $k_1 \leq k_2 \leq k_3$. Furthermore, let $k_{T,1}$, $k_{T,2}$, and $k_{T,3}$ be the number of symbols requested from the strings $\mathbf{x}_1^{(T)}, \mathbf{x}_2^{(T)}$, and $\mathbf{x}_3^{(T)}$, respectively. The requested symbols are recovered according to the following steps.

1) For $T = 1$, the k_1 requested symbols of $\mathbf{x}^{(1)}$ are read systematically from the k_1 entries in which they are stored in the first, second, and third block of $\mathbf{c}^{(1)}$.
2) For $T = 2$, the $k_{2,2}+k_{2,3}$ requested symbols of $\mathbf{x}_2^{(2)}$ and $\mathbf{x}_3^{(2)}$ are read systematically from the fourth and fifth blocks of $\mathbf{c}^{(2)}$. Since $\mathcal{C}$ is a $[k, k/3, 2k/3+1]$ code, it follows that any $k/3$ entries from the string $(\mathbf{x}_1^{(2)}, \mathcal{E}_1(\mathbf{x}_1))$ are sufficient to recover $\mathbf{x}_1^{(2)}$. We read
$$A = \min\left\{\frac{k}{3} - k_{1,3}, \frac{2k}{3} - k_{2,2} - k_{2,3} - k_{1,3}\right\}$$
entries from the third block of $\mathbf{c}^{(2)}$ and $k/3 - A$ entries from the second block of $\mathbf{c}^{(2)}$. To show this is possible, we need to show that $0 \leq A \leq k/3 - k_{1,3}$ and $k/3 - A \leq k/3 - k_{1,2}$. Clearly, $A \leq k/3 - k_{1,3}$. From $0 \leq k_{1,3} \leq \frac{k}{3}$ and $0 \leq k_{2,2} + k_{2,3} + k_{1.3} \leq \frac{2k}{3}$ we have that $A \geq 0$. If $A = k/3 - k_{1,3}$ then $k/3 - A = k_{1,3} \leq k/3 - k_{1,2}$. If $A = 2k/3 - k_{2,2} - k_{2,3} - k_{1,3}$ then $k/3 - A = k_{2,2} + k_{2,3} + k_{1,3} - k/3$ and since $k_{1,2} + k_{1,3} + k_{2,2} + k_{2,3} \leq 2k/3$, it again follows that $k/3 - A \leq k/3 - k_{1,2}$. This proves the correctness of this step.
3) For $T = 3$, the $k_{3,2}$ requested symbols of $\mathbf{x}_2^{(3)}$ are read systematically from the sixth block of $\mathbf{c}^{(3)}$. In order to recover the string $\mathbf{x}_3^{(3)}$, read $\frac{k}{3}$ entries from the first and second blocks, which are sufficient to successfully recover the string $\mathbf{x}_3^{(3)}$. It is possible to read $\frac{k}{3}$ entries from the first two blocks since the number of entries already read from these blocks is $k_{1,1} + k_{1,2} + \frac{k}{3} - A$, and
$$\begin{aligned} &k_{1,1} + k_{1,2} + \frac{k}{3} - A \\ &= k_{1,1} + k_{1,2} + \frac{k}{3} - \min\left\{\frac{k}{3} - k_{1,3}, \frac{2k}{3} - k_{2,2} - k_{2,3} - k_{1,3}\right\} \\ &= k_{1,1} + k_{1,2} + \frac{k}{3} + \max\left\{k_{1,3} - \frac{k}{3}, k_{2,2} + k_{2,3} + k_{1,3} - \frac{2k}{3}\right\} \\ &= \max\left\{k_1, k_{2,2} + k_{2,3} + k_1 - \frac{k}{3}\right\} \leq \frac{k}{3}. \end{aligned}$$
In order to recover the string $\mathbf{x}_1^{(3)}$, read $k/3$ entries from blocks three, four, and five of $\mathbf{c}^{(3)}$. First note that, since $\mathbf{x}_3^{(3)}$ was already recovered, the value of $\mathcal{E}_2(\mathbf{x}_3^{(3)})$ is known and thus we can assume that the string stored in the third block is $\mathcal{E}_2(\mathbf{x}_1^{(3)})$. Hence, reading $k/3$ entries from these three blocks is sufficient to decode the message $\mathbf{x}_1^{(3)}$. It is possible to read this number of entries since the number of entries read so far from these three blocks is
$$\begin{aligned} &k_{1,3} + A + k_{2,2} + k_{2,3} \\ &\quad \leq k_{1,3} + \frac{2k}{3} - k_{2,2} - k_{2,3} - k_{1,3} + k_{2,2} + k_{2,3} = \frac{2k}{3}. \end{aligned}$$
■

VII. Conclusion and Open Problems

In this paper the concept of ℓ-consecutive switch codes, in which the request sets are restricted to ℓ consecutive generations, was studied. This natural variation of switch codes allows better coding rates than those achieved by conventional switch codes and yet admits a large collection of common request sets. Consecutive switch codes were studied mainly for the combinatorial case and the computational case, but also when the degree is limited by two. For the combinatorial case, we constructed (k, m)-ℓ-consecutive switch codes for every $\ell \geq 2$, where $m \approx \ell k - (\ell-1)\sqrt{(\ell-2)k}$, and found a tight lower bound on m when $\ell = 2$. When the degree is limited by two, we showed a construction that reduces the redundancy by one over the optimal combinatorial 2-consecutive switch code. For the computational case, constructions of a $(k, m = 1.5k)$-2-consecutive switch code and a $(k, m = 2k)$-3-consecutive switch code were given, along with a construction of a (k, m)-ℓ-consecutive switch code, for every $\ell \geq 2$, in which $m \approx \frac{1}{2}\ell k$ when ℓ is large. We also studied conventional switch codes and presented the best known construction of binary (k, m)-switch codes. Table I provides a summary of all the constructions in the paper. For constructions of consecutive switch codes the table shows the value of ℓ and for constructions of computational switch codes the table shows the size of the field, q.

TABLE I
SUMMARY OF CONSTRUCTIONS

Const-raction	Syste-matic	Consec-utive	m	Combina-torial
Const. 1	Yes	$\ell = 2$	$2k-1$	Yes
Const. 2	Yes	$\ell \geq 2$	$\approx \ell k - \ell\sqrt{\ell k}$	Yes
Const. 4	Yes	$\ell = 2$	$2k-2$	Yes, degree 2
Const. 5	No	$\ell = 2$	$1.5k$ $2k - \log_2 k$	$q \geq k-1$ $q = 2$
Const. 6	No	$\ell \geq 2$	$\approx \ell k/2$	$q \gtrsim (\ell-1)k/2$
Const. 7	No	$\ell = 3$	$2k$	$q \geq k-1$
Const. 8	Yes	No	$\approx 2k^{1.5}$	$q = 2$

While the results in the paper advance the study of switch codes, there are still several interesting problems which are left open. Some of them are summarized as follows.

1) Find lower bounds on the length m of binary (k, m)-switch codes.
2) Improve the constructions and find bounds for combinatorial consecutive switch codes for arbitrary ℓ (not necessarily fixed).
3) Find constructions and bounds for linear binary ℓ-consecutive switch codes of limited degree.
4) Improve the constructions and find bounds for computational consecutive switch codes. Of particular interest is to determine whether the construction of $(k, m = 1.5k)_q$-2-consecutive switch codes is optimal in the non-systematic case, either by finding a suitable lower bound on the length or by demonstrating an improved construction.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for insightful comments which contributed to the discussion in the paper. In particular, the authors thank one of the reviewers for contributing Theorem 8, which provides a lower bound on the length of systematic 2-consecutive switch codes.

E. Yaakobi is thankful for the hospitality provided by the Center for Memory and Recording Research during his stay.

APPENDIX A
PROOF OF LEMMA 2

In this appendix we prove Lemma 2 which states that for $k \leq n$, a code $\mathcal{C}$ is an $(n, N = m, k, m, t = 1)_q$-primitive batch code if and only if it is an $(n, k, m, t = 1)_q$-switch code.

Proof of Lemma 2: Assume that $\mathcal{C}$ is an $(n, N = m, k, m, t = 1)_q$-primitive batch code. Given a set of k pairs $I = \{(i_1, T_1), (i_2, T_2), \ldots, (i_k, T_k)\} \subset [k] \times \mathbb{N}$, let $\mathcal{M}$ be the multi-set of size k, $\{i_1, i_2, \ldots, i_k\}$. Since $k \leq n$, it follows that there exists $\mathbf{z} \in \mathbb{F}_q^n$ such that $z_{i_r} = x_{i_r}^{(T_r)}$, for all $r \in [k]$. Since $\mathcal{C}$ is an $(n, N = m, k, m, t = 1)_q$-primitive batch code, it follows that there exist k pairwise disjoint subsequences of $\mathcal{E}_{\mathcal{C}}(\mathbf{z}) = \mathbf{y}, \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$, such that z_{i_r} can be recovered from $\mathbf{u}_r$, for all $r \in [k]$. Moreover, the position sets which specify these subsequences in $\mathbf{y}$ depend only on $\mathcal{M}$ and not on the entries of $\mathbf{z}$. This implies that if we denote by J_r the positions of $\mathbf{u}_r$ in $\mathbf{y}$ then, for all $r \in [k]$, $x_{i_r}^{(T_r)}$ can be recovered from the subsequence of $\mathbf{c}^{(T)} = \mathcal{E}_{\mathcal{C}}(\mathbf{x}^{(T)})$ that is specified by J_r. Hence, the set $J = \{(j, T_r) \ : \ r \in [k], \ j \in J_r\}$ is a recovery set for I that depends only on I, such that for all $j \in [m]$, $|\{T \ : \ (j, T) \in J\}| \leq 1$. Thus, $\mathcal{C}$ is an $(n, k, m, t = 1)_q$-switch code.

Conversely, assume that $\mathcal{C}$ is an $(n, k, m, t = 1)_q$-switch code. Given a multi-set $\mathcal{M} = \{i_1, i_2, \ldots, i_k\}$ and $\mathbf{z} \in \mathbb{F}_q^n$, let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)} \in \mathbb{F}_q^n$ be such that $x_{i_r}^{(r)} = z_{i_r}$, for all $r \in [k]$. Since $\mathcal{C}$ is an $(n, k, m, t = 1)_q$-switch code, it follows that there exists a recovery set for $I = \{(i_1, 1), (i_2, 2), \ldots, (i_k, k)\}$, $J \subset [m] \times \mathbb{N}$, such that for every $r \in [k]$, the symbol $x_{i_r}^{(r)}$ can be recovered from $\{c_j^{(T)}\}_{(j,T) \in J}$ and for every $j \in [m]$, J consists of at most one element of the form (j, T). Moreover, J depends only on I and since the same code is used for every generation, it depends only on $\mathcal{M}$. Hence, $z_{i_1}, z_{i_2}, \ldots, z_{i_k}$ can be recovered from disjoint subsequences of $\mathcal{E}_{\mathcal{C}}(\mathbf{z})$, whose positions are specified by the disjoint sets $J_r = \{j \ : \ (j, r) \in J\}$, $r \in [k]$, which depend only on $\mathcal{M}$. Thus, $\mathcal{C}$ is an $(n, N = m, k, m, t = 1)_q$-primitive batch code. □

APPENDIX B
PROOF OF THEOREM 1

In this appendix we prove Theorem 1, which states that for all sufficiently large k, there exists a $(k, m)_2$-switch code with $m \approx 2k^{1.5}$.

We achieve this result by using the family of **one-step majority logic decodable code** [12, pp. 273–275]. The connection between this class of codes and distributed storage was first observed in [10]. We show how such codes can be used to construct $(k, m)_2$-switch codes in general and then we apply this method to a specific type of one-step majority logic decodable code.

A (ν, μ)-code $\mathcal{C}$ over $\mathbb{F}_q$ is called a **(ν, μ)-one-step majority logic decodable code with availability b** if for every $\mathbf{x} \in \mathbb{F}_q^{\mu}$, and for all $i \in [\nu]$, there exist b disjoint subsequences of $\mathcal{E}_{\mathcal{C}}(\mathbf{x})$ that can each recover the symbol x_i. Moreover, the positions of these subsequences in $\mathcal{E}_{\mathcal{C}}(\mathbf{x})$ depend only on i.

Construction 8: Let $b \in [k]$ and let $\mathcal{C}(b)$ be a (ν, k)-one-step majority logic decodable code over $\mathbb{F}_q$ with availability b. Define the (m, k)-code , $\mathcal{C}$, over $\mathbb{F}_q$, where $m = bk + \lfloor k/b \rfloor \cdot \nu$, as follows. For every $\mathbf{x} \in \mathbb{F}_q^k$,

$$\mathcal{E}_{\mathcal{C}}(\mathbf{x}) = \mathcal{R}_b(\mathbf{x})\mathcal{R}_{\lfloor k/b \rfloor}(\mathcal{E}_{\mathcal{C}(b)}(\mathbf{x})),$$

i.e., $\mathcal{E}_{\mathcal{C}}(\mathbf{x})$ is the concatenation of b copies of $\mathbf{x}$ followed by $\lfloor k/b \rfloor$ copies of $\mathcal{E}_{\mathcal{C}(b)}(\mathbf{x})$.

Theorem 11: The code $\mathcal{C}$ from Construction 8 is a $(k, m)_q$-switch code or equivalently $\mathcal{C}$ is an $(n = k, N = m, k, m, 1)_q$-primitive batch code.

Proof: Let $\mathbf{z} = \mathcal{E}_{\mathcal{C}(b)}(\mathbf{x})$ and let $\mathbf{c} = \mathcal{E}_{\mathcal{C}}(\mathbf{x}) = \mathcal{R}_b(\mathbf{x})\mathcal{R}_{\lfloor k/b \rfloor}(\mathbf{z})$. We have to show that for any multi-set of k indices $\mathcal{M} = \{i_1, i_2, \ldots, i_k\}$, the information symbols $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ can be recovered from $\mathbf{c}$, where for every $r, s \in [k]$, $r < s$, the two subsequences of $\mathbf{c}$ that are used for the recovery of x_{i_r} and x_{i_s}, respectively, are disjoint.

For every $i \in [k]$, let r_i be the number of appearances of i in the multi-set $\mathcal{M}$. Note that if $i \notin \mathcal{M}$ then $r_i = 0$ and hence

$\sum_{i=1}^{k} r_i = k$. For every $i \in [k]$, let $m_i = \min\{b, r_i\}$. Since $m_i \leq b$, we can recover m_i copies of x_i directly from $\mathcal{R}_b(\mathbf{x})$. For $i \in [k]$ such that $r_i > m_i = b$, we recover the remaining $r_i - b$ copies of x_i from the $\lfloor k/b \rfloor$ copies of $\mathbf{z}$ in $\mathbf{c}$. Since $\mathcal{C}(b)$ is a one-step majority logic decodable code, it follows that there exist b disjoint subsequences of $\mathbf{z}$ that can each recover x_i. By using at most $\left\lceil \frac{r_i - b}{b} \right\rceil \leq \frac{r_i}{b}$ copies of each of the b subsequences in $\mathcal{R}_{\lfloor k/b \rfloor}(\mathbf{z})$ we can recover $r_i - b$ copies of $\mathbf{x}_i$.

To complete the proof we have to show that we have enough copies of $\mathbf{z}$ in $\mathcal{R}_{\lfloor k/b \rfloor}(\mathbf{z})$ to recover $r_i - b$ copies of x_i, for every $i \in [k]$ for which $m_i = b < r_i$. For every $j \in [\nu]$ we use at most $\frac{r_i}{b}$ copies of z_j to recover x_i. Hence, we use at most

$$\left\lfloor \sum_{i=1}^{k} \frac{r_i}{b} \right\rfloor \leq \left\lfloor \frac{k}{b} \right\rfloor$$

copies of z_j and indeed $\mathcal{R}_{\lfloor k/b \rfloor}(\mathbf{z})$ consists of enough copies of $\mathbf{z}$. ∎

Note that, for a given k, Construction 8 provides the smallest value of m when the availability b is approximately $\sqrt{k}$. One such code is a binary **cyclic difference-set code**. The proof of the following lemma can be found in [12, p. 293].

Lemma 5: The binary cyclic $(\nu = 2^{2r} + 2^r + 1, \mu = 2^{2r} + 2^r - 3^r)$*-difference-set code is a* (ν, μ)*-one-step majority logic decodable code with availability* $b = 2^r + 1 \approx \sqrt{\mu}$.

We are now in a position to prove Theorem 1.

Proof of Theorem 1: Let r be the smallest integer for which $k \leq \mu = 2^{2r} + 2^r - 3^r$ and let $\widehat{\mathcal{C}}$ be the binary cyclic $(\nu = 2^{2r} + 2^r + 1, \mu)$-difference-set code. By Lemma 5 we have that $\widehat{\mathcal{C}}$ is a (ν, μ)-one-step majority logic decodable code with availability $\widehat{b} = 2^r + 1 \approx \sqrt{\mu}$. By shortening the code $\widehat{\mathcal{C}}$ we obtain a $(\nu - \mu + k, k)$-one-step majority logic decodable code, $\widetilde{\mathcal{C}}$, with availability b, where b can take any value in $[\widehat{b}]$. In particular, since $\widehat{b} \approx \sqrt{\mu}$ and $\mu \geq k$, it follows that we may choose b to be approximately $\sqrt{k}$. By Theorem 11, the code $\mathcal{C}$ that is obtained from Construction 8 by setting $\mathcal{C}(b)$ to be the code $\widetilde{\mathcal{C}}$ with $b \approx \sqrt{k}$ is a $(k, m)_2$-switch code, where

$$m = bk + \left\lfloor \frac{k}{b} \right\rfloor (\nu - \mu + k).$$

Since r is the smallest integer for which $k \leq 2^{2r} + 2^r - 3^r$, it follows that $k > 2^{2r-2} + 2^{r-1} - 3^{r-1}$, and therefore $\nu - \mu + k = 3^r + 1 + k \approx k$. Thus, m is approximately $2k^{1.5}$. □

REFERENCES

[1] S. Bhattacharya, S. Ruj, and B. Roy, "Combinatorial batch codes: A lower bound and optimal constructions," *Adv. Math. Commun.*, vol. 6, no. 2, pp. 165–174, 2012.

[2] R. A. Brualdi, K. P. Kiernan, S. A. Meyer, and M. W. Schroeder, "Combinatorial batch codes and transversal matroids," *Adv. Math. Commun.*, vol. 4, no. 3, pp. 419–431, 2010.

[3] C. Bujtás and Z. Tuza, "Relaxations of Hall's condition: Optimal batch codes with multiple queries," *Appl. Anal. Discrete Math*, vol. 6, no. 1, pp. 72–81, 2012.

[4] S. Buzaglo, E. Yaakobi, Y. Cassuto, and P. H. Siegel, "Consecutive switch codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 660–664.

[5] Y. M. Chee, F. Gao, S. T. H. Teo, and H. Zhang, "Combinatorial systematic switch codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 241–245.

[6] R. Cohen and Y. Cassuto, "Algorithms and throughput analysis for MDS-coded switches," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 656–660.

[7] R. Cohen and Y. Cassuto, "Placement and read algorithms for high throughput in coded network switches," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 250–254.

[8] R. Diestel, *Graph Theory*, 2nd ed. New York, NY, USA: Springer, 2000, ch. 2.

[9] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc.*, vol. 43, pp. 439–561, Aug. 2006.

[10] P. Huang, E. Yaakobi, H. Uchikawa, and P. H. Siegel, "Binary linear locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 11, pp. 6268–6283, Nov. 2016.

[11] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proc. 36th Annu. ACM Symp. Theory Comput.*, vol. 36. 2004, pp. 262–271.

[12] S. Lin and D. J. Costello, *Error Control Coding*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[13] H. Lipmaa and V. Skachek, "Linear batch codes," in *Coding Theory and Applications* (CIM Series in Mathematical Sciences), vol. 3. Cham, Springer, 2015, pp. 245–253.

[14] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[15] A. S. Rawat, Z. Song, A. G. Dimakis, and A. Gál, "Batch codes through dense graphs without short cycles," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 1477–1481.

[16] A. S. Rawat, Z. Song, A. G. Dimakis, and A. Gál, "Batch codes through dense graphs without short cycles," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1592–1604, Apr. 2016.

[17] N. Silberstein and A. Gál, "Optimal combinatorial batch codes based on block designs," *Des., Codes Cryptograph.*, vol. 72, no. 2, pp. 409–424, Feb. 2016.

[18] A. Vardy and E. Yaakobi, "Constructions of batch codes with near-optimal redundancy," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 1197–1201.

[19] Z. Wang, H. M. Kiah, and Y. Cassuto, "Optimal binary switch codes with small query size," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 636–640.

[20] Z. Wang, O. Shaked, Y. Cassuto, and J. Bruck, "Codes for network switches," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Turkey, Jul. 2013, pp. 1057–1061.

[21] Z. Wang, H. M. Kiah, Y. Cassuto, and J. Bruck, "Switch codes: Codes for fully parallel reconstruction," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2061–2075, Apr. 2017.

[22] H. Zhang and V. Skachek, "Bounds for batch codes with restricted query size," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 1192–1196.

Sarit Buzaglo (S'14–M'15) was born in Israel in 1983. She received the B.Sc. and M.Sc. degrees from the Department of Mathematics, Technion–Israel Institute of Technology, Haifa, Israel, in 2007 and 2010, respectively. In 2014, she received her Ph.D. degree from the Department of Computer Science, Tecnion-Israel Institute of Technology. She is currently a postdoctoral researcher in the Center for Memory and Recording Research at University of California, San Diego. She is also an awardee of the Weizmann Institute of Science - National Postdoctoral Award Program for Advancing Women in Science, in 2014. Her research interests include coding theory, algebraic error-correction coding, coding for advanced storage devices and systems, and combinatorics.

Yuval Cassuto (S'02–M'08–SM'14) is a faculty member at the Andrew and Erna Viterbi Department of Electrical Engineering, Technion-Israel Institute of Technology. His research interests lie at the intersection of the theoretical information sciences and the engineering of practical computing and storage systems. During 2010-2011 he has been a Scientist at EPFL, the Swiss Federal Institute of Technology in Lausanne. From 2008 to 2010 he was a Research Staff Member at Hitachi Global Storage Technologies, San Jose Research Center. From 2000 to 2002, he was with Qualcomm, Israel R&D Center, where he worked on modeling, design and analysis in wireless communications. He received the B.Sc. degree in Electrical Engineering, summa cum laude, from the Technion, Israel Institute of Technology, in 2001, and the M.S. and Ph.D. degrees in Electrical Engineering from the California Institute of Technology, in 2004 and 2008, respectively. Dr. Cassuto has won the 2010 Best Student Paper Award in data storage from the IEEE Communications Society, as well as the 2001 Texas Instruments DSP and Analog Challenge $100,000 prize.

Paul H. Siegel (M'82–SM'90–F'97) received the S.B. and Ph.D. degrees in mathematics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1975 and 1979, respectively. He held a ChaimWeizmann Postdoctoral Fellowship with the Courant Institute, New York University, New York, NY, USA. He was with the IBM Research Division, San Jose, CA, USA, from 1980 to 1995. He joined the faculty at the University of California, San Diego, CA, USA, in 1995, where he is currently a Professor of electrical and computer engineering with the Jacobs School of Engineering. He is affiliated with the Center for Memory and Recording Research where he holds an Endowed Chair and served as Director from 2000 to 2011. His research interests include information theory and communications, particularly coding and modulation techniques, with applications to digital data storage and transmission. He is a Member of the National Academy of Engineering. He was a Member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2009 to 2014. He was the 2015 Padovani Lecturer of the IEEE Information Theory Society. He was a recipient of the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He was the co-recipient of the 1992 IEEE Information Theory Society Paper Award and the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award. He served as a Co-Guest Editor of the 1991 Special Issue on Coding for Storage Devices of the IEEE TRANSACTIONS ON INFORMATION THEORY. He served as an Associate Editor of Coding Techniques of the IEEE TRANSACTIONS ON INFORMATION THEORY from 1992 to 1995, and as Editor-in-Chief from 2001 to 2004. He was also a Co-Guest Editor of the 2001 two-part issue on The Turbo Principle: From Theory to Practice and the 2016 issue on Recent Advances in Capacity Approaching Codes of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.

Eitan Yaakobi (S'07–M'12–SM'17) is an Assistant Professor at the Computer Science Department at the Technion Israel Institute of Technology. He received the B.A. degrees in computer science and mathematics, and the M.Sc. degree in computer science from the Technion–Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011. Between 2011-2013, he was a postdoctoral researcher in the department of Electrical Engineering at the California Institute of Technology. His research interests include information and coding theory with applications to nonvolatile memories, associative memories, data storage and retrieval, and voting theory. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010-2011.