

Codes for Network Switches

Zhiying Wang*, Omer Shaked[†], Yuval Cassuto[†], and Jehoshua Bruck*

*Electrical Engineering Department, California Institute of Technology, Pasadena, CA 91125, USA

[†]Electrical Engineering Department, Technion – Israel Institute of Technology, Technion City, Haifa 32000, Israel
zhiying@caltech.edu, shakedomer@gmail.com, ycassuto@ee.technion.ac.il, bruck@caltech.edu

Abstract—A network switch routes data packets between its multiple input and output ports. Packets from input ports are stored upon arrival in a switch fabric comprising multiple memory banks. This can result in memory contention when distinct output ports request packets from the same memory bank, resulting in a degraded switching bandwidth. To solve this problem, we propose to add redundant memory banks for storing the incoming packets. The problem we address is how to minimize the number of redundant memory banks given some guaranteed contention resolution capability. We present constructions of new switch memory architectures based on different coding techniques. The codes allow decreasing the redundancy by 1/2 or 2/3, depending on the request specifications, compared to non-coding solutions.

I. INTRODUCTION

Multi-port switches are commonly used as data processing and routing devices in computer networks. With the growth of the amount of computing devices, network links, and data transmission, switches are facing the challenge of serving growing rates of packet transmissions on an increasing number of ports. In this paper, we focus on the switch memory sub-system used to store packets between arrival from input ports to departure to output ports. Assume for now that the switch memory sub-system has the same number of input and output ports, say R ports. At a time slot, each input port writes one data packet into the memory, and each output port reads one data packet from the memory. The R packets written in the same time slot are called a *generation*, and they share the same row in subsequent diagrams. The total bandwidth of the memory sub-system should be R bits per second for reading, and same for writing. Figure 1 (a) is an example with $R = 2$ ports. As the number of ports R grows, as well as the rate of each port, the number of memory banks is also increased to parallelize data writing and reading on slower, and more affordable, memory devices. In the paper we assume to have multiple memory banks, each operating at a speed of one unit write and one unit read per time slot. Figure 1 (b) is an example with $R = 2$ ports and 2 banks. (A, B) , (C, D) , (E, F) are three generations. So far it appears that R memory banks are sufficient. For example, at the current slot we write in 2 units (E, F) (at top) and read out 2 units (A, B) (from bottom), as specified by the switch rate requirements.

However, the destinations of the packets may not be the same as their memory bank index. For example, if the output request is (A, C) , then due to the memory speed limitation, this request cannot be served timely. Hence we need to add redundant memory banks such that an arbitrary output request

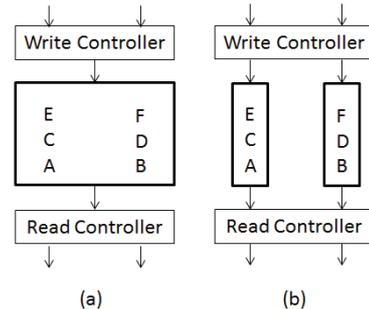


Figure 1. A switch with $R = 2$ ports. (a) Single memory writes and reads 4 packets per time slot. (b) 2 memory banks writes and reads 1 packet per time slot per bank. Each capital letter is a packet.

can be served. A naive solution is to replicate the original information R times and use R^2 banks. Since no more than R packets are required from the same input port, we can always serve any request of R units in one time slot. See Figure 2 (a) for an example. However, the question is raised whether the same flexibility can be attained with lower redundancy using coding techniques? Figure 2 (b) answers this question to the affirmative for $R = 2$. By adding the parity memory, any request of size 2 can be served such that every memory reads at most one packet. For example, if the output request is (A, C) , then we can read $A, D, C + D$ from the three memories. We define a *switch code* for R ports and n memory banks as a way to represent the input packets, such that any request of R output packets can be obtained by reading at most one packet from each memory bank. Such a code is in principle not confined to encode over a single generation, but multi-generation codes add load on the memory through recoding, or require to manage a separate packet cache. Therefore, in this paper we focus on encoding functions within a generation.

Intuitively, a switch code requires that each symbol (or packet) in the codeword can be recovered by *multiple disjoint* subsets of symbols. In this sense, they are similar to locally decodable codes [6] where every symbol can be probabilistically recovered by a small number of symbols in the presence of errors. Another similar notion is locally repairable codes [3], where a symbol can be recovered from some (fixed set of) other symbols. In addition, self-repairing codes [5] are erasure codes such that a symbol can be repaired from some (fixed number of) other fragments given the number of erasures. All of these notions have the common requirement of local reconstruction, however codes for switches are required

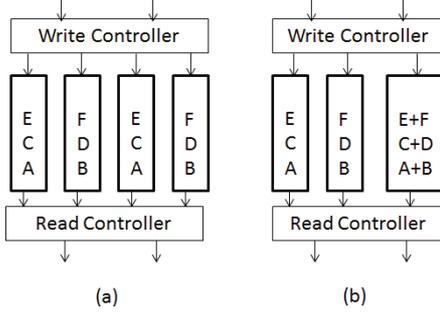


Figure 2. Codes for a switch with $R = 2$ ports. (a) Replication. (b) The code with the parity. The last memory computes XOR of the two packets from each generation.

to have as many disjoint reconstructions as the number of information symbols.

The first result of this paper is a code construction using parities of two symbols. This construction uses $R(R-1)/2$ redundant memories, and can serve any request of R output symbols, which amounts to half the redundancy of a non-coding scheme. The second result focuses on one-burst requests, i.e., multiple requested packets from one input port, and no more than one request for the others. It is practical to consider this special type of request because a single burst can be used to empty the longest request queue, and consequently shorten the worst-case delay. We generalize the code construction such that the parities are XOR of more than two symbols, and obtain a better code rate. In particular, we construct codes based on block designs. When there are three symbols for each parity, the code has $R(R-1)/3$ redundant memories.

We will set up the switch code problem formally in Section II. Then we construct codes of pairs of parities in Section III, and codes based on block designs in Section IV. Finally, Section V concludes the paper.

II. PROBLEM SETTINGS

Notations: For simplicity, we denote by “+” binary XOR. The set of integers $\{i, i+1, \dots, j\}$ is denoted by $[i, j]$ for $i \leq j$ and $\{1, 2, \dots, i\}$ is written as $[i]$ for integer $i \geq 1$.

We will use a bit (also considered as a symbol or a packet) to represent an entry in a memory bank. Suppose there are R_1 input bits and R_2 requested bits. That is, the number of input and output ports is R_1, R_2 , respectively. In this paper we assume they are the same $R_1 = R_2 = R$ unless stated otherwise. And let n be the total number of memory banks, $n \geq R$. We say $n - R$ is the amount of *redundancies*. For the i -th generation, let $U_i = (u_{i,0}, u_{i,1}, \dots, u_{i,R-1})$ be the R -bit input, and $X_i = (x_{i,0}, x_{i,1}, \dots, x_{i,n-1})$ the n -bit written vector. The encoding of a switch code is a function f from the input bits to the written bits:

$$f : \{U_i\}_{i \geq 0} \mapsto \{X_i\}_{i \geq 0}.$$

Notice that the function can be computed across generations. But as mentioned, we concentrate on functions within a

generation. A function can be different from one generation to another. However, there are only a finite number of functions from R input bits to n written bits. If we consider large enough memory banks, we can always find R generations with the same encoding function. Therefore, w.l.o.g., we restrict the encoding function to be identical in every generation:

$$f : U \mapsto X,$$

where $U = (u_0, u_1, \dots, u_{R-1})$ and $X = (x_0, x_1, \dots, x_{n-1})$ are input and written bits of one generation, $u_i, x_j \in \mathbb{F}_2$ for all $i \in [0, R-1], j \in [0, n-1]$. We call the subscript i of u_i an information or systematic *element*.

The output *request* is $R = R_2$ arbitrary information bits from the previous generations. A *solution* to the request is a way to compute these R bits from no more than one bit from each memory. If we have a solution for bits from R different generations, then since the encoding is identical for all generations, obviously we have a solution for bits from less than R generations. Therefore we can limit our attention to the worst case of bits from R generations, and the order of the bits and generation indices are not of importance. A *request vector* is an ordered R -tuple of indices, $L = (l_0, l_1, \dots, l_{R-1})$, where the i -th output bit is an element $l_i \in [0, R-1]$, and $l_i \leq l_j$ for all $0 \leq i \leq j \leq R-1$. For example, if $R = 5$ then $L = (0, 2, 2, 2, 3)$ requires one bit from elements 0, 3 and three bits from element 2. A solution is a partition of the memory banks $[0, n-1]$ into R parts, together with a set of R decoding functions. Denote the decoding partition by $M = ((m_{0,0}, \dots, m_{0,t_0}), (m_{1,0}, \dots, m_{1,t_1}), \dots, (m_{R-1,0}, \dots, m_{R-1,t_{R-1}}))$, and the decoding functions by g_i , $i \in [0, R-1]$,

$$u_i = g_i(x_{m_{i,0}}, x_{m_{i,1}}, \dots, x_{m_{i,t_i}}).$$

Since M is a partition, each memory bank is read from no more than once, therefore satisfies the constraint of bandwidth. If a memory bank is not used, we put it in an arbitrary part i , and the decoding function does not use it. An (n, R) *switch code* is the encoding and decoding functions that satisfy the above definitions.

Example 1 Consider the code in Figure 2 (c). The encoding function is $(x_0, x_1, x_2) = (u_0, u_1, u_0 + u_1)$. If the output is (A, D) then the request vector is $(0, 1)$ and the decoding partition is $((0, 2), (1))$. The decoding functions are $u_0 = x_0 + 0 \cdot x_2$, $u_1 = x_1$. It can be seen that the partition is not unique since we can put memory 2 in either of the two parts. If the output is (A, C) on the other hand, the request vector is $(0, 0)$, the decoding partition is $((0), (1, 2))$ and the decoding functions are $u_0 = x_0$, $u_0 = x_1 + x_2$.

Theorem 1 Without coding, or only through replications, the number of memories is at least $n = R_1 R_2$ given R_1 input ports and R_2 output ports.

Proof: We assumed that the encoding function is identical in every generation. Now every information bit should be duplicated R_2 times to serve a request for the same information element R_2 times. ■

In this paper, we will show that through coding, we can decrease the redundancy by a half or two thirds, depending on the parameters. We will focus on *systematic codes* where the information bits are stored in the first R memories, and the rest are parity bits.

One can think of different variations of switch codes, and they can be applied in different queueing scenarios. A special kind of request is *one-burst request*, where only one integer repeats in the vector L . That is, $l_0 < l_1 < \dots < l_i = l_{i+1} = \dots = l_j < l_{j+1} < \dots < l_{R-1}$ for some $0 \leq i \leq j \leq R-1$. The burst length, or the number of repetitions is $j-i+1$, and l_k , $k \notin [i, j]$, are called *singletons*. Another type is *limited-repetition request*, namely each integer repeats at most a certain number of times. A third generalization is *consecutive-generation request*, where the requested bits come from up to t consecutive generations, $t \leq R$. In this case we will use t vectors to represent requests from each generation, and the sum of the vector lengths is R .

III. CODES WITH PAIR (DEGREE 2) PARITIES

In this section we construct switch codes with $n = (R_1 + 1)\lceil R_2/2 \rceil$. First we solve for the case of $R = R_1 = R_2$ and then for the general case. In addition, we construct codes for consecutive-generation requests.

Construction 1 Let $R = R_1 = R_2$. The code is constructed as the information bits and XOR of all pairs of information bits. Figure 2 is an example of this code for $R = 2$. When $R = 3$, the code is $X = (u_0, u_1, u_2, u_0 + u_1, u_0 + u_2, u_1 + u_2)$.

Theorem 2 The $(n = R(R+1)/2, R)$ switch code in Construction 1 can solve any request.

Proof: We solve every information bit u_i by either read it directly or by XOR of u_i and $u_i + u_j$, for some $j \neq i, j \in [0, R-1]$. Each of the R systematic elements will be used to obtain one information bit. For two different systematic elements i, j , the corresponding parities (if used) are also different. Otherwise this parity is simply XOR of x_i, x_j and we can read them directly. Thus all memories used are disjoint. ■

Systematic codes whose parities are computed from pairs can be represented by an undirected graph \mathcal{G} . Here information elements are vertices and parity of pairs are edges. The number of edges indicates the redundancy. We call such codes *systematic parity-pair* codes. The above construction corresponds to the complete graph K_R .

The following is a construction for different R_1 and R_2 values. When $R = R_1 = R_2$ is even, the redundancy is the same as Construction 1.

Construction 2 First consider the $(R_1 + 1, R_1)$ parity code $X = (u_0, \dots, u_{R_1-1}, x_{R_1})$ with $x_{R_1} = \sum_{i=0}^{R_1-1} u_i$. Any request of size 2 can be solved in X : one bit can be read directly, the other bit can be solved through all the rest elements, or read directly. Second duplicate $\lceil R_2/2 \rceil$ times the code X . Thus any request of size R_2 can be solved and the number of memories is $n = (R_1 + 1)\lceil R_2/2 \rceil$.

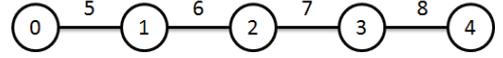


Figure 3. Graphic representation of a $(9,5)$ code. It solves all 2-consecutive-generation requests.

Next we consider requests from t consecutive generations, $t \leq R$. Clearly such requests are also repetition-limited because every information element will be requested no more than t times.

Lemma 3 A lower bound for the number of redundancies of a systematic pair-parity code is $n - R \geq \lceil R(t-1)/2 \rceil$, when a request is restricted to t consecutive generations.

Proof: A necessary condition for the existence of a solution for t requests from the same element is this element being contained in at least R memory banks. Or the degree of each vertex in \mathcal{G} should be at least $t-1$. ■

We will show that this bound is not tight by the following theorem.

Theorem 4 Construct the $(2R-1, R)$ switch code as

$$x_i = \begin{cases} u_i, & i \in [0, R-1] \\ u_{i-R} + u_{i-R+1}, & i \in [R, 2R-2] \end{cases}$$

Then this code solves any 2-consecutive-generation request, and has optimal redundancy for a systematic pair-parity code.

Proof: This construction can be associated with a graph \mathcal{G} that is a line connecting all elements. If one (or more) elements receive 2 requests, there is at least one element that receives no requests. We will use this element as a starting point of the second generation, and use the edges to traverse the entire graph. The bits of the first generation will be read directly.

To show optimality of redundancy we assume the redundancy is less than $R-1$. But by Lemma 3 we know every vertex has degree at least one. So the graph becomes at least two disjoint lines. Consider the shortest line which has length no more than $R/2$. It is not possible to solve the request that outputs of all the elements on this line twice. Hence we get a contradiction. ■

Figure 3 shows an example of this code. The labels of the vertices and edges are indices of the memory bank. Suppose the request is $(0, 2)$ from the first generation and $(2, 3, 4)$ from the second. Then it can be solved by reading 0, 2 of the first generation, and 1, 6, 7, 8 of the second.

IV. CODES BASED ON BLOCK DESIGNS

In this section we generalize the above section and construct switch codes by XORING triples of information bits. We first derive a lower bound on the redundant bits with respect to the degree of parity bits (number of bits XORed). Then construct codes with $n = R(R+2)/3$ based on block designs that solve one-burst requests. Notice that Theorem 1 still applies for one-burst requests, so our codes reduce the redundancy to $1/3$ compared to non-coding schemes.

Consider a bipartite graph with variable vertices (information bits) and check vertices (parity bits). If an information bit is added to a parity bit, there is an edge between them. Let d_j be the degree of the j -th encoded bit, namely x_j XORs d_j information bits, $j \in [0, n-1]$. Similar to Lemma 3 every element should be contained in at least R memories. Hence

$$\sum_{j=0}^{n-1} d_j \geq R^2.$$

Moreover assume constant degree parities and denote by $d = d_R = \dots = d_{n-1}$ the degree. Then $R + (n-R)d \geq R^2$ and we obtain a lower bound on the redundancy, as stated below.

Lemma 5 *A systematic switch code with constant degree parities satisfies*

$$n - R \geq R(R - 1)/d.$$

This lemma shows that a systematic code may have less redundancies with larger parity degree. Next we will construct codes with degree $d = 3$ and redundancy $n - R = R(R - 1)/3$.

In Construction 1, an output bit u_i is solved by either itself or XOR of a parity bit $u_i + u_j$ and an information bit u_j . Similarly, when the parity has degree 3, we will solve a bit u_i by either itself or XOR of two parity bits $u_i + u_j + u_k$, $u_j + u_k + u_l$, and an information bit u_l . Associate each parity bit with the block (or subset) of information elements it involves. In this section, the notations of the binary sum and the block will both represent a parity bit. For example, the bits $u_i + u_j + u_k$, $u_j + u_k + u_l$ are associate with $\{i, j, k\}$, $\{j, k, l\}$, respectively, then we see an intersection of size two between these two blocks. In other words, the pair $\{j, k\}$ appears in both of these blocks. We say the bit i can be solve by

$$\{l\}, \{i, j, k\}, \{j, k, l\}. \quad (1)$$

We call these three bits as systematic and parity *helpers* of the output element i .

A *balanced incomplete block design* is a system with b blocks of size k and a total of R elements. Moreover, it requires that every element repeats r times and every subset of size t appears exactly λ times. A *triple system* with $\lambda = 2$ is a balanced incomplete block design such that each block contains 3 elements out of a total of R elements, and every pair of elements appear exactly twice in the blocks. Since the code is systematic and a request might require R bits from the same element, every element repeats $R - 1$ times in the blocks. By counting we know there should be a total of $R(R - 1)/3$ parity or redundant bits. A switch code does not necessarily corresponds to a block design but we will show that this approach gives us some nice constructions.

Example 2 *Consider the following triple system with $R = 6$ elements and $R(R - 1)/3 = 10$ parity blocks: $\{0, 1, 2\}, \{0, 2, 3\}, \{0, 1, 4\}, \{1, 2, 5\}, \{0, 3, 5\}, \{2, 3, 4\}, \{0, 4, 5\}, \{1, 4, 3\}, \{1, 5, 3\}, \{2, 5, 4\}$. Every element repeats $R - 1 = 5$ times. Let a switch code contain the 6 systematic*

$$\begin{array}{ll} (2r + 1, 5m - r, 5m + r + 1) & 0 \leq r \leq m - 1 \\ (2m - 2r - 2, 2m + r + 1, 4m - r - 1) & 0 \leq r \leq m - 2 \\ (2m - 2r - 1, 2m + r, 4m - r - 1) & 0 \leq r \leq m - 1 \\ (2r + 2, 5m - r - 1, 5m + r + 1) & 0 \leq r \leq m - 2 \\ (2m, 3m, 5m) & \\ (4m) & \text{short;twice} \end{array}$$

Figure 4. Difference triples when $R = 12m$.

bits and 10 parity bits. Suppose the request is $(0, 0, 0, 0, 0, 0)$, then we can solve it in the following way:

$$\{0\} \quad (2)$$

$$\{1\}, \{0, 3, 5\}, \{1, 5, 3\}$$

$$\{2\}, \{0, 4, 5\}, \{2, 5, 4\} \quad (3)$$

$$\{3\}, \{0, 1, 4\}, \{1, 4, 3\} \quad (4)$$

$$\{4\}, \{0, 2, 3\}, \{2, 3, 4\}$$

$$\{5\}, \{0, 1, 2\}, \{1, 2, 5\}$$

We can see that by using one bit from all of the memories, we are able to output bit 0 six times. Similarly, one can check that it is possible to output any bit six times. And any one-burst request can be solved by using solutions excluding the singletons. For example, for $(0, 0, 0, 1, 4, 5)$ we can use equations (2)(3)(4) and singletons $\{1\}, \{4\}, \{5\}$. Therefore, this code serves any one-burst request.

The above example shows that by using triples instead of pairs as parities, we decrease the number of redundancies from 15 to 10 when $R = 6$. We will induce three different kinds of switch codes from triple systems that solves different kinds of requests.

Cyclic Construction.

A triple system admitting the cyclic group of automorphisms is called *cyclic*. In other words, if $\{i_1, i_2, i_3\}$ is a block in the system, then so is $\{i_1 + j, i_2 + j, i_3 + j\}$ for any $j \in [0, R - 1]$. In this construction all additions are computed modulo R . A triple (d_1, d_2, d_3) is a *difference triple* if $d_i \leq R/2$ and $d_1 + d_2 + d_3 \equiv 0 \pmod R$ or $d_1 + d_2 \equiv d_3 \pmod R$. Once we have this difference triple, we can write down blocks in the form $\{i, i + d_1, i + d_1 + d_2\}$ for all $i \in [0, R - 1]$. A special case is when $d_1 = d_2 = d_3 = R/3$ and R divisible by 3, then we write the difference triple as a short vector (d_1) .

Let D be a multiset that contains the integers from 1 to $\lceil R/2 \rceil - 1$ each twice, and $R/2$ once when R is even. A cyclic triple system is equivalent to a partition of D into difference triples [1]. The following lemma is also due to [1].

Lemma 6 *If $R \equiv 0 \pmod{12}$, there is a cyclic triple system. The difference triples are given in Figure 4. We assume $R = 12m$ for some integer m .*

For example, if $R = 12$ then the first difference triple is $(1, 5, 6)$, and all triples $\{i, i + 1, i + 6\}$, $i \in [0, 11]$ belong to the system.

Theorem 7 *The triple system constructed by Lemma 6 can serve a one-burst request with burst length 10 except for 9 choices of singletons.*

Proof: We will show that there are 10 ways to solve the 0-th bit. Since the code is cyclic, this suffices for all other bursts. Besides the element $\{0\}$ itself, we show 9 other solutions in Figure 5. One can check that all the blocks listed are from this code and the helpers are disjoint when $m \geq 2$. Thus we can serve the burst if the singletons in the request do not include any element in the set $A = \{1, 2m, 2m+1, 5m+1, 6m, -(5m+1), -(2m+1), -(2m), -1\}$. ■

In fact the solutions are not unique but the systematic helpers can only be in the set A due to the structure of the difference triples. Hence constant 10 is the longest burst solvable even though R grows linearly with m . An observation of Figure 5 is that only values in the form of $km - 1, km, km + 1$, $0 \leq k \leq 11$ are included in the systematic and parity helpers. Notice that the code is cyclic, so if two bursts are separated by at least $3 \pmod m$, the corresponding helpers will be disjoint. As a result, if we have several bursts of elements $l_1, l_2, \dots, l_t \in [0, 12m - 1]$, with burst lengths no more than 10 and

$$l_i - l_j \geq 3 \pmod m \quad (5)$$

for any $i \neq j \in [t]$, then we are able to solve the request. For example, if $m = 9, R = 108$, we can serve three bursts of length 10: bursts of elements $0, 21, 60$, which equal to $0, 3, 6 \pmod m$. In general we have the following result.

Theorem 8 *The cyclic construction can serve $\lfloor m/3 \rfloor$ bursts of length 10 that satisfy (5).*

In addition to serving multiple bursts, this construction also has the advantage of being cyclic and symmetric for all input and output ports.

Top-down Construction.

The main idea of the top-down construction is to break blocks of size 4 into triples. Let us start with an example. Consider the block $\{1, 2, 3, 4\}$ of size 4 and its subsets of size 3:

$$\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}. \quad (6)$$

Take any two triples out of these four, we get two parities that look exactly the same as those in (1). In other words, the subsets of block of size 4 are good candidates to construct our triple system. To output element i , we can use j as the systematic helper, for any $j \neq i$. And the corresponding parities are

$$\{1, 2, 3, 4\} \setminus \{j\} \quad (7)$$

$$\{1, 2, 3, 4\} \setminus \{i\} \quad (8)$$

It should be noted that no matter which j we use, (8) is always a parity helper. In other words, among the three elements j , $j \neq i$, we can only use one of them as a systematic helper.

Suppose we have a balanced incomplete block design D_1 with R elements, $R(R-1)/12$ blocks of size 4. Moreover every element repeats $(R-1)/3$ times and every pair appears once. Such systems are also called quadruple systems. For simplicity, assume $(R-1)/3$ is an integer. Now construct another block design D by changing every block in D_1 into four blocks of size 3. For example, if $\{1, 2, 3, 4\}$ is a block in

D_1 , then triples in (6) all belong to D . It is easy to check that D is a triple system with $R(R-1)/3$ blocks, $R-1$ repeats, and every pair appears exactly twice.

Example 3 *Take D_1 as the block design with $R = 25$ elements and 50 blocks of size 4 (see [2]). The design is formed by dicyclic solution in two families: $\{a_1, a_2, b_1, e_5\}, \{a_1, a_3, c_1, d_4\}$. All the blocks are generated by cyclic shift on the letters and then on the subscripts. Then we label the elements from 0 to 24 and break blocks down to 200 triples.*

We have an output-helper graph like Figure 6. In this graph, every vertex is a systematic element. A directed edge (i, j) means that j is a possible systematic helper of i . Since every pair (i, j) appears in D_1 , every output i can use any systematic helper $j \neq i$. So the graph is actually complete. We group three helpers (like $1, 5, 24$) to emphasize that only one can be used. We call every group a *helper group* or a *group* for a given output element. Once we choose a systematic helper from a group, the edge corresponds to two parity helpers. For example, if output is 0 and helper is 1, then the edge between them means parities $\{1, 5, 24\}, \{0, 5, 24\}$.

The following theorem states that if we have a quadruple system D_1 then we have a switch code serving one-burst requests, where the size of burst is a third of ports.

Theorem 9 *The top-down construction serves any one-burst request with burst length $(R-1)/3 + 1$.*

Proof: Let us check requests with one burst of length $(R-1)/3 + 1$. We will use Example 3 to illustrate the idea. Given a request vector $(0, \dots, 0, a_9, \dots, a_{24})$ where $a_0 = 0$ is repeated $(R-1)/3 + 1 = 9$ times, and the rest are singletons. Besides $\{0\}$ itself, choose one systematic helper h_1, \dots, h_8 from each group to solve 0 as follows.

- Initialize a set T as the elements not requested $\{a_1, \dots, a_8\}$.
- For all $i \in [8]$, if there exists a member of T in group i , choose it as h_i and remove it from T .
- For the rest groups, choose arbitrarily an element from the group as h_i . And use one element (w.l.o.g. assume element a_i) from T as a systematic helper for h_i , because h_i is a requested singleton and $a_i \neq h_i$.

See Figure 7 for an example.

Besides disjoint systematic helpers, we also need disjoint parity helpers. First consider edges (h_i, a_i) and (h_j, a_j) for some $i \neq j \in [8]$ such that these edges exist. Note that by the above decoding algorithm h_i, h_j, a_i, a_j are all distinct. The corresponding parities are

$$\{h_i, x, y\}, \{x, y, a_i\}$$

$$\{h_j, z, w\}, \{z, w, a_j\}$$

for some x, y, z, w . They are sub-blocks of $A = \{h_i, a_i, x, y\}$ and $B = \{h_j, a_j, z, w\}$ in the quadruple system D_1 . Since D_1 contains each pair only once, there exist identical parities on these edges only if $A = B$, or $(x, y) = (h_j, a_j), (z, w) = (h_i, a_i)$. But even in this case, the parity helpers are disjoint. Second consider edges $(0, h_i)$ and $(0, h_j)$ for all $i \neq j \in [8]$. Since they belong to different blocks in D_1 , they are disjoint.

$\{1\}$	$\{-1, 0, 5m\}$	$\{-1, 1, 5m\}$
$\{2m\}$	$\{-4m + 1, -2m - 1, 0\}$	$\{-4m + 1, -2m - 1, 2m\}$
$\{2m + 1\}$	$\{0, 1, 5m + 1\}$	$\{1, 2m + 1, 5m + 1\}$
$\{5m + 1\}$	$\{-5m, -3m, 0\}$	$\{5m + 1, -5m, -3m\}$
$\{6m\}$	$\{-2m + 1, 0, 4m + 1\}$	$\{4m + 1, 6m, -2m + 1\}$
$\{-5m - 1\}$	$\{0, 2m - 1, 4m - 1\}$	$\{2m - 1, 4m - 1, -5m - 1\}$
$\{-2m - 1\}$	$\{-5m - 1, -5m, 0\}$	$\{-5m - 1, -5m, -2m - 1\}$
$\{-2m\}$	$\{-5m - 1, -5m + 1, 0\}$	$\{-5m - 1, -5m + 1, -2m\}$
$\{-1\}$	$\{-2m + 1, 0, 2m\}$	$\{-2m + 1, -1, 2m\}$

Figure 5. 9 solutions of a burst of element 0 for the cyclic construction. Each row is a solution. The first column shows the systematic helpers, and the rest are parity helpers.

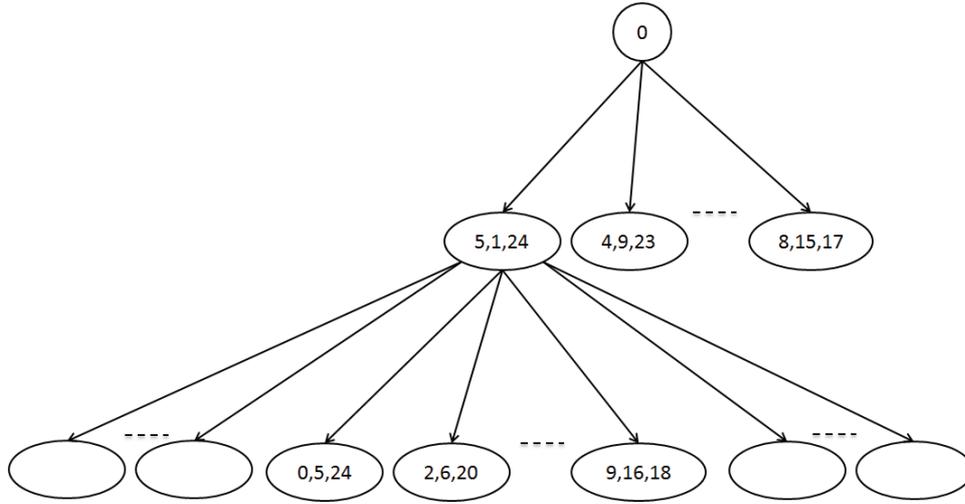


Figure 6. Output and systematic helpers based on 50 blocks of size 4 and $R = 25$ elements. Every output can have any other element as its systematic helper. Every output has 8 groups of helpers.

Third consider edges $(0, h_i)$ and (h_j, a_j) for some $i \neq j \in [8]$ such that these edges exist. Again $0, h_i, h_j, a_j$ are all distinct and therefore the parities are disjoint. At last consider $(0, h_i)$ and (h_i, a_i) for $i \in [8]$ such that these edges exist. But notice that a_i does not belong to the i -th group. So these two edges belong two different blocks in D_1 and are disjoint. So we can output any one burst of length $(R - 1)/3 + 1$ plus singletons.

Suppose the burst length is smaller than $(R - 1)/3 + 1$. Suppose a_i is also requested, $i \in [8]$. If a_i is a helper of 0, do not use it to solve 0 but only read a_i . If a_i is a helper of h_i , do not use h_i to solve 0 but only read a_i, h_i . ■

This construction can be generalized to parities that are XOR of more than three elements. For example, if one has a block design of block size k , then by breaking it down to blocks of size $k - 1$, one may use one systematic helper and two parity helpers to output a bit. Meanwhile since the parity has higher degree, we may expect to get smaller redundancy.

Linear Construction.

The idea is to have a simple way to decide the pair j, k in (1) given the output i and the systematic helper l . Then try to see weather we can use each triple as many times as possible, i.e., three times for different outputs. One of the simplest mapping from (i, l) to (j, k) would be a linear function. The

resulting design has the same parameters as the previous two constructions, namely $(R - 1)/3$ triples and every pair appears exactly twice.

To emphasize that j, k are functions of i, l , we rename $f = j, g = k$. We are going to define the pair f, g by the following linear mapping:

$$\begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} i \\ l \end{bmatrix}.$$

In other words, $f = f(i, l), g = g(i, l)$ are functions of i, l . When we omit the arguments, we write f, g to denote $f(i, l), g(i, l)$, respectively. Since every pair should appear in the design, we would like the mapping to be bijection. So

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

should be invertible. In fact, due to the symmetry of the roles of i and l , if we switch i and l , the resulting pair should be g, f . In other words, $a = d, b = c$.

Thus we have triples

$$\{i, f, g\}, \{l, f, g\} \quad (9)$$

as blocks. We would like to use each triple 3 times for different values of i, l . Therefore, we

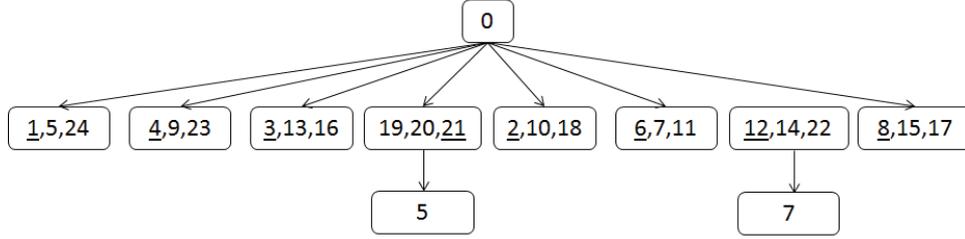


Figure 7. Output a burst of length 9. Assume $(0, \dots, 0, 9, 10, \dots, 24)$ is the request and $T = \{1, 2, \dots, 8\}$ are unrequested elements. Use the underlined $(h_1, h_2, \dots, h_8) = (1, 4, \dots, 8)$ as systematic helpers for the burst of 0. If a requested singleton is not underlined, simply read it. Otherwise use a_i as the helper of h_i , e.g., the singleton 21 has the systematic helper 5.

block index	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	0	2	0	3	0	1	0	2	0	1	3	1	
2	2	4	4	4	4	1	3	2	3	1	5	5	2	
6	6	5	5	6	6	3	4	3	5	5	6	6	4	

Figure 8. A linear construction with $R = 7$ and 14 blocks. Each column is a block. Notice that every triple can be used three times as a parity helper. For example, block 1,2 can help solve element 0; block 1,5 can help solve 2; block 1,9 can help solve element 6.

would like $\{f, f(f, l'), g(f, l')\}, \{g, f(g, l''), g(g, l'')\}, \{g, f(i', g), g(i', g)\}, \{f, f(i'', f), g(i'', f)\}$ to be identical to one of the blocks in (9). Here i', i'', l', l'' are some other elements. There are quite a few possibilities for such constraints. We were able to find a consistent solution to the following:

$$(i, f, g) = (g(i', g), f(i', g), g) = (f(i'', f), f, g(i'', f)),$$

$$(l, f, g) = (f(f, l'), f, g(f, l')) = (g(g, l''), f(g, l''), g).$$

Here all the triples are ordered. The values of a, b, c, d should guarantee that there exist solutions of i', i'', l', l'' given arbitrary i, l . The solutions to the above equations are:

$$a = d = \frac{1}{2} + \frac{\sqrt{-3}}{6}, b = c = \frac{1}{2} - \frac{\sqrt{-3}}{6},$$

or exchange a, d with b, c , which is equivalent. It can be checked that this ensures the matrix A to be invertible. We will choose the number of elements $R > 3$ to be a prime number such that $-3 \bmod R$ is a perfect square.

We note here that by the formula for the Legendre symbol [4] which indicates whether -3 is a perfect square, this condition is equivalent to R being a prime and $(R-1)/3$ being an integer. Specifically, we can show that -3 being a square mod R is equivalent to R being a square mod 3.

For example, let $R = 7$, then

$$A = \begin{bmatrix} 2 & 6 \\ 6 & 2 \end{bmatrix}$$

and one check that it actually gives a triple system of 14 blocks with twice very pair in Figure 8. For another example, when $R = 19$,

$$A = \begin{bmatrix} 17 & 3 \\ 3 & 17 \end{bmatrix}$$

and we get 114 blocks.

Theorem 10 *The linear construction serves any one-burst request and has $R(R-1)/3$ redundancies, when $R > 3$ is a prime and $-3 \bmod R$ is a perfect square.*

Proof: First we check the number of redundancies. By construction, every output-helper pair i, l gives us two triples but every triple is used for three such pairs. All together there are $\binom{R}{2} \times 2/3 = R(R-1)/3$ triples or redundancies.

Second we show any one-burst is solvable. We for now consider the worst case when the request is R times of the same bit i . By the construction of the triples, one can use systematic helper $\{l\}$ and parity helpers as in (9). Since $b \neq 0$, we know f is a bijection between l and $f(i, l)$ for the fixed i . Therefore the parities $\{i, f(i, l), g(i, l)\}$ are disjoint and $\{l, f(i, l), g(i, l)\}$ are disjoint for all $l \neq i, l \in [0, R-1]$. Moreover, suppose there is a common parity $\{i, f(i, l), g(i, l)\} = \{l', f(i, l'), g(i, l')\}$ for some other helper $l' \neq l, i$, then the only possibilities are $i = g(i, l')$ or $i = f(i, l')$. However, both result in $l' = i$ and we get a contradiction. As a result, we can use each of the systematic elements as a helper, and disjoint parity helpers at the same time. Now consider the burst length being less than R . If j is also requested, $j \neq i$, simply do not use j as a helper for i , and read j directly. ■

This construction has the advantage of the least constraint on the burst length. With any of the above constructions one can build switch codes that serve burst requests. As mentioned, if we have the ability to solve burst requests, we can lower the worst-case delay in the memories.

V. CONCLUSIONS

In this paper we proposed the switch code problem, that is, how to use coding techniques to solve unbalanced requests in memories of network switches. We constructed pair-parity codes with $R(R-1)/2$ redundancies and those with $R(R-1)/3$ redundancies using different ideas of block designs. Better understanding on block designs can result in

good switch codes. At the same time, good switch codes will be good combinatorial designs with certain parameters.

The subject of switch code still requires more studies. For example a lower bound on redundancy without degree constraint is unknown, and a network-coding type of argument may lead to interesting results. It is also our future work to study constructions with more than three elements in a parity, and also codes with different degrees of the parities. Finally, coding across generations may lead to even smaller redundancy.

REFERENCES

- [1] C. J. Colbourn and A. Rosa, *Triple Systems*. Oxford University Press, 1999.
- [2] R. Fisher and F. Yates, *Statistical tables for biological, agricultural, and medical research*. Longman Group United Kingdom, June 1995.
- [3] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *Information Theory, IEEE Transactions on*, vol. 58, no. 11, pp. 6925–6934, nov. 2012.
- [4] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers (Fifth edition)*. Oxford University Press, 1980.
- [5] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1215–1223.
- [6] S. Yekhanin, "Locally decodable codes," *Computer Science–Theory and Applications*, pp. 289–290, 2011.