Combinatorial Systematic Switch Codes

Yeow Meng Chee*, Fei Gao†, Samuel Tien Ho Teo* and Hui Zhang*
*School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
†Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore

***Abstract*—Multiport switches are commonly used as data processing and routing devices in computer networks. 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 lead to memory contention when distinct output ports request packets from the same memory bank, resulting in a degraded switching bandwidth. To solve this problem, switch codes are introduced by Wang et al. [1] as a tradeoff between redundancy and service.**

Using techniques from combinatorial design theory, we improve their result on switch codes serving any one-burst request to a denser set of parameters. New constructions for switch codes serving repetition limited request and consecutive-generation request are also given.

I. Introduction

A network switch is a computer networking device that connects devices together in a computer network. By using a form of packet switching, a multiport network switch can process and forward data packets from input ports to designated output ports. They manage the flow of data across a network by only transmitting a received message to the device for which the message was intended. A subsystem of a network switch, called switch memory, is used to temporarily cache data packets between their arrival from input ports and departure at output ports. With the rapid growth of the number of nodes in computer networks and the capacity of data flows, the memory bandwidth of network switches becomes one of the bottlenecks for high speed data transmission in the whole computer network. To address this problem, we can either stick to a single memory bank and try to improve its data exchange speed, or we can use multiple slower and low-price memory banks to parallelize data writing and reading processes. The latter approach disadvantaged in situations where only a few banks are handling most of the requests. For example, the requests on output ports connected to web servers or data centers are significantly more frequent than those connected to personal computers. Due to the slower speed of each memory bank, these hot requests cannot be served efficiently. Wang et al. [1] characterized the trade-off between the workload of memory banks and the amount of stored information, and proposed coding techniques to eliminate memory contention.

Suppose the switch memory subsystem has R_{in} input ports and R_{out} output ports. At every time slot, each input port writes one data packet into the memory, and each output port reads (at most) one data packet from the memory. The packets written in the same time slot are called a *generation*. The total bandwidth of the memory subsystem is R_{in} packets per time slot for writing, and R_{out} for reading. Let n denote the number of memory banks in the subsystem to store the data packets. We are interested in the size of n, given $R_{\mathsf{in}}, R_{\mathsf{out}}$, such that any R_{out} requested packets can be served within the speed limitation.

As an example, let $R_{\mathsf{in}} = R_{\mathsf{out}} = 2$, and consider two generations (A, B), (C, D) that are written to the memory subsystem, where A and C are in the same bank while B and D are in the other (see Fig. 1(a)). If the required outputs are A and B, then two memory banks suffice to serve this request within only one time slot. On the other hand, if the output requests are A and C instead of A and B, then due to memory speed limitation, this request cannot be served within one time slot. To resolve this problem, we can append redundant memory banks to make arbitrary output request on two symbols be served in time. One naive solution is to replicate every received packet for R_{out} copies as in Fig. 1(b). Then we can always serve any request of R_{out} packets in one time slot.

The same flexibility can be attained, however, with lower redundancy, by employing coding techniques when storing packets in memory banks. For example, we can output A, C in one time slot with $n = 3$ memory banks. In each generation, we store the original packets in two banks and their binary XOR in the third, that is, storing the two generations $(A, B, A + B)$ and $(C, D, C + D)$ (as in Fig. 1(c)). Here we assume the data packets are represented in binary format and follow the convention to denote binary XOR operator by "+". Now, we can serve the same request by reading $A, D, C + D$ from the three banks in one time slot and recover C as the binary sum of D and $C + D$.

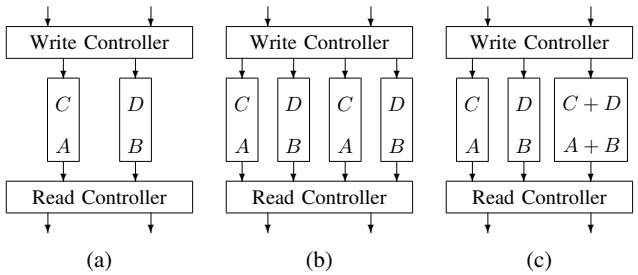

Fig. 1: Codes for a network switch with $R_{\mathsf{in}} = R_{\mathsf{out}} = 2$ ports. (a) Direct input/output. (b) Replication. (c) Coding with parity.

Formalizing the ideas behind this example, Wang et al. [1] introduced *switch codes* operating on n memory banks to serve requests on R_{in} input ports and R_{out} output ports. For every generation, an input vector of R_{in} symbols is encoded to a codeword of n symbols, and stored at the corresponding n memory banks. The code should be able to recover requests

of R_{out} symbols by reading no more than one symbol from each memory bank. The goal is to reduce n given R_{in} and R_{out}.

In this paper, we give combinatorial constructions of switch codes, improving on previous results of Wang et al. [1].

II. Problem Formulation

The set of integers $\{i, i+1, \ldots, j\}$ is denoted by $[i, j]$ and the finite field of q elements, q a prime power, is denoted by $\mathbb{F}_q$. Data packets and written packets are assumed to be over the same alphabet Σ. We call each entry in the memory banks a *bit*, *symbol*, or *packet* interchangeably. The number of input ports, output ports, and memory banks are denoted by R_{in}, R_{out}, and n, respectively. Note that $n \geq R_{\mathsf{in}}, R_{\mathsf{out}}$ and we call $n - R_{\mathsf{in}}$ the *redundancy* of the switch.

The *encoding* of a switch code is an injective function f from generations of input data packets to written bits in memory banks,

$$\begin{aligned} f : \left\{\Sigma^{R_{\mathsf{in}}}\right\}_{i\geq 0} &\longrightarrow \left\{\Sigma^{n}\right\}_{i\geq 0} \\ \{U_i\}_{i\geq 0} &\longmapsto \{X_i\}_{i\geq 0}, \end{aligned}$$

where $U_i = (u_{i,0}, u_{i,1}, \ldots, u_{i,R_{\mathsf{in}}-1})$ are the R_{in} input bits and $X_i = (x_{i,0}, x_{i,1}, \ldots, x_{i,n-1})$ are the n written bits, with the subscript i indexing generations. As noted by Wang et al. [1], even though the encoding functions can be computed across generations and/or acting differently for distinct generations, we can investigate the ones that are identical in all generations without loss of generality. In this case, we have

$$\begin{aligned} f : \Sigma^{R_{\mathsf{in}}} &\longrightarrow \Sigma^{n} \\ U &\longmapsto X, \end{aligned} \tag{1}$$

where $U = (u_0, u_1, \ldots, u_{R_{\mathsf{in}}-1})$ and $X = (x_0, x_1, \ldots, x_{n-1})$ are input and written bits of *one* generation. We call the subscript j of u_j an *information* or *systematic element*.

Let u_{i_0,l_0}, u_{i_1,l_1}, $\ldots$, $u_{i_{R_{\mathsf{out}}-1},l_{R_{\mathsf{out}}-1}}$ be arbitrary R_{out} request bits from previous generations. We call the multiset $L = (l_0, l_1, \ldots, l_{R_{\mathsf{out}}-1})$ the corresponding *request vector*. Assuming that these R_{out} request bits are from $t \leq R_{\mathsf{out}}$ different generations, we denote the elements in the request vector corresponding to the request bits from the i-th generation by L_i, $0 \leq i \leq t-1$. We call $L_0, L_1, \ldots, L_{t-1}$ the *request sets* and we have $|L_i| \geq 1$ and $\sum_{i=0}^{t-1} |L_i| = R_{\mathsf{out}}$. Notice that since we are using the same encoding function for every generation, the subscriptions are just for clarity and have no meaning regarding the actual order of these generations.

Example 1. Suppose we have a request of four output bits from three different generations: one generation is requested to output its 0-th and 1-st bits and other two generations are requested to output their 2-nd bits, that is, the request bits are $\{u_{0,0}, u_{0,1}, u_{1,2}, u_{2,2}\}$. Then the corresponding request vector is $L = \{0, 1, 2, 2\}$ and the request sets are $L_0 = \{0, 1\}$, $L_1 = \{2\}$, $L_2 = \{2\}$.

A *solution* to a request of R_{out} bits is a way to recover these bits by retrieving no more than one bit from each memory bank. More explicitly, the solution to a request from t generations consists of a set $\mathcal{P} = \{P_0, P_1, \ldots, P_{t-1}\}$ composed of t mutually disjoint subsets of $[0, n-1]$, together with a bijective map ϕ from $\mathcal{L} = \{L_i : i \in [0, t-1]\}$ to $\mathcal{P}$:

$$\begin{aligned} \phi : \mathcal{L} &\longrightarrow \mathcal{P} \\ L_i &\longmapsto \phi(L_i), \end{aligned} \tag{2}$$

and t decoding functions g_i, $0 \leq i \leq t-1$, such that the request bits are correctly recovered, that is,

$$g_i : \{x_j : j \in \phi(L_i)\} \longmapsto \{u_k : k \in L_i\}. \tag{3}$$

Definition 1. A *t-consecutive-generation switch code*, or more precisely an $(n, R_{\mathsf{in}}, R_{\mathsf{out}}; t)$ *switch code*, is a pair of encoding and decoding functions that satisfies the conditions (1)–(3).

Since the encoding is identical for all generations, if we have a solution for bits from R_{out} generations, then we obviously have a solution for bits with the same request vector for less than R_{out} generations. Hence, we can limit our attention to the worst case where the order of the bits and generation indices are not of importance. An $(n, R_{\mathsf{in}}, R_{\mathsf{out}})$ switch code is a pair of encoding and decoding functions that does not require information on the generations of request bits. If $R = R_{\mathsf{in}} = R_{\mathsf{out}}$, an $(n, R_{\mathsf{in}}, R_{\mathsf{out}})$ is referred to as an (n, R) switch code.

One can consider variations of switch codes where different request scenarios apply. The t-consecutive-generation switch codes are applicable when the data packets expire after a fixed time slot and only the newest t generations are of interest. Another class concerns *limited-repetition* request, namely each integer in the request vector repeats at most a certain number of times. This class of switch codes are applicable when the workload is balanced.

One special variation of switch codes recover requests with only one element in the request vector for multiple times (called a *burst*), and the others at most once. In practice, a burst can be used to clean up the longest request queue among the memory banks and improve the wort-case delay.

Definition 2. A one-burst request has only one integer repeats in the request vector $L = (l_0, l_1, \ldots, l_{R_{\mathsf{out}}-1})$, that is, $l_0 < l_1 < \cdots < l_i = l_{i+1} = \cdots = l_j < l_{j+1} < \cdots < l_{R_{\mathsf{out}}-1}$ for some $0 \leq i \leq j \leq R_{\mathsf{out}} - 1$. The number of repetition $j - i + 1$ is called the *burst length*, and l_k, $k \notin [i, j]$, are called *singletons*.

We are interested in minimizing n when $R_{\mathsf{in}}, R_{\mathsf{out}}$ and the alphabet Σ are fixed. In this paper, we only consider the binary case, that is, when $\Sigma = \mathbb{F}_2$, and the encoding and decoding functions are all linear. *Systematic codes* where the input bits are stored directly into the first R_{in} of a total of n memory banks, and the rest are parity bits would be our main focus.

Wang et al. [1] pointed out that without coding, or only through replications, the number of memory banks is at least $n = R_{\mathsf{in}} R_{\mathsf{out}}$, given R_{in} input ports and R_{out} output ports. They also construct switch codes with $n = (R_{\mathsf{in}} + 1)\lceil R_{\mathsf{out}}/2 \rceil$ memory banks solving any request of size R_{out}.

In this paper, we continue the investigation on constructing variations of systematic switch codes satisfying different

requests types. We construct switch codes solving any t consecutive-generation request with redundancies $(R_{\mathsf{in}}-1)(t-1)$ provided $t-1 \leq \lfloor R_{\mathsf{in}}/2 \rfloor$ and $t-1 \leq R_{\mathsf{in}} - \lfloor R_{\mathsf{out}}/t \rfloor$. Wang et al. [1] established a class of switch codes serving any one burst request with redundancies $R(R-1)/3$ whenever R is an odd prime and -3 is a perfect square in $\mathbb{F}_R$. We improve on this result by showing that such switch codes exist for all sufficiently large $R \equiv 0, 1 \bmod 6$. A class of switch codes for limited-repetition with redundancies $O(R^{3/2})$ is also given.

III. Parity-Pair Switch Codes for Consecutive-Generation Requests

In this section, we discuss bounds for systematic parity-pair switch codes serving any t consecutive-generation requests.

A *systematic parity-pair* switch code is a systematic code whose parities are computed from pairs of information bits. Such a code can be represented by an undirected graph $G = (V, E)$, called its *parity-pair graph*. Here the information elements are vertices and parity of pairs are edges. The number of vertices and edges indicates the number of input ports R_{in} and the redundancy $n - R_{\mathsf{in}}$, respectively. For example, the encoding $(u_0, u_1, u_2) \mapsto (u_0, u_1, u_2, u_0 + u_1, u_1 + u_2)$ can be represented by the graph $G = (V, E)$ with vertices $V = \{0, 1, 2\}$ and edges $E = \{\{0, 1\}, \{1, 2\}\}$.

There are only two ways to recover a request on an information element $i \in [0, n-1]$. The first is to directly read the corresponding systematic bit u_i. The second is to compute u_i from some parity nodes containing it with help of other systematic bits or other parity nodes. Therefore, a necessary condition for the existence of a solution for t requests of an element is that this element being contained in at least $t-1$ parity nodes. In the settings of parity-pair codes, this means that the degree of each vertex in the parity-pair graph G should be at least $t-1$. The following lower bound is established by Wang et al. [1].

Proposition 1 (Wang et al. [1])**.** *A lower bound for the redundancy of a systematic pair-parity code is* $n - R_{\mathsf{in}} \geq \left\lceil \frac{R_{\mathsf{in}}(t-1)}{2} \right\rceil$*, when a request is restricted to* t *consecutive generations.*

Wang et al. [1] proved that the lower bound in Proposition 1 is not tight by showing that an optimal systematic pair-parity $(2R-1, R)$ switch code for two consecutive-generation requests has redundancy $R-1$. Here, we show more generally that the lower bound in Proposition 1 cannot be achieved when $2 \leq t \leq \lfloor R_{\mathsf{out}}/2 \rfloor$. We need the following proposition.

Proposition 2. *Suppose* $2 \leq t \leq \lfloor R_{\mathsf{out}}/2 \rfloor$*. Let* G*, with* R_{in} *vertices, be the parity-pair graph of a systematic parity-pair code. If there exist two adjacent vertices, each having degree at most* $t-1$*, then* G *cannot recover a request of* t *consecutive generations.*

Proof. Let i and j be two adjacent vertices in G, each with degree at most $t-1$. We make a request on information elements i and j for t consecutive generations, where $t \leq \lfloor R_{\mathsf{out}}/2 \rfloor$. To recover i for t consecutive generations, every edge of incident on i is accessed once and i itself is accessed once. This includes the edge $\{i, j\}$, which cannot be used to recover j. Hence j can only be recovered at most $t-1$ times. This is a contradiction. □

Corollary 1. *When* $2 \leq t \leq \lfloor R_{\mathsf{out}}/2 \rfloor$*, the lower bound of Proposition 1 cannot be achieved.*

Proof. The lower bound in Proposition 1 is achieved only if every vertex in the parity-pair graph G has degree $t-1$, or one vertex in G has degree t and all others have degree $t-1$. In either case, there exists a pair of adjacent vertices, each with degree $t-1$. This leads to a contradiction via Proposition 2. □

We now construct switch codes serving any t consecutive-generation request, generalising a construction method of Wang et al. [1, Theorem 4]. The construction is based on edge-disjoint hamiltonian paths in a complete graph. We require the following classical result.

Proposition 3 (see, for example, [2, Chapter 2.3])**.** *A complete graph on* n *vertices contains* $\lfloor n/2 \rfloor$ *edge-disjoint hamiltonian paths.*

Theorem 1. *There is an* $(n, R_{\mathsf{in}}, R_{\mathsf{out}}; t)$ *switch code with redundancy* $(R_{\mathsf{in}}-1)(t-1)$*, provided* $t-1 \leq \lfloor R_{\mathsf{in}}/2 \rfloor$ *and* $t-1 \leq R_{\mathsf{in}} - \lfloor R_{\mathsf{out}}/t \rfloor$*.*

Proof. Let G be a subgraph of the complete graph on R_{in} vertices, formed by the edges of $t-1$ edge-disjoint hamiltonian paths, where $t-1 \leq \lfloor R_{\mathsf{in}}/2 \rfloor$. Such a G exists by Proposition 3. Note that G has R_{in} vertices and $(t-1)(R_{\mathsf{in}}-1)$ edges. We claim that G is the parity-pair graph of the required switch code serving any R_{out} request bits of t consecutive generations.

Let the $t-1$ edge-disjoint hamiltonian paths in G be $P_1, P_2, \ldots, P_{t-1}$, and let the request sets of the t consecutive generations be $L_0, L_1, \ldots, L_{t-1}$. Without loss of generality, assume that L_0 has the smallest size among the L_i's, so that $|L_0| \leq \lfloor R_{\mathsf{out}}/t \rfloor \leq R_{\mathsf{in}} - (t-1)$. Therefore we can choose $t-1$ distinct elements $h_1, h_2, \ldots, h_{t-1}$ from $[0, R_{\mathsf{in}}-1] \setminus L_0$. These $t-1$ elements are used to help in the recovery of request bits.

To serve the requests in L_0, we directly read the corresponding systematic elements in written bits. To recover the requested bits in L_i, for $1 \leq i \leq t-1$, we use h_i and edges in the hamiltonian path P_i as follows. Suppose $(e_1, e_2, \ldots, e_{R_{\mathsf{in}}-1})$ is the list of edges of P_i such that e_j is incident with e_{j+1}, for $1 \leq j \leq R_{\mathsf{in}} - 2$. Then there exists j such that $e_j = \{h_i, a\}$. We can then recover a by taking the sum of h_i and the parity $h_i + a$. Once we know the elements in the parity-pair e_j, we can then recover the elements in the parity-pair represented by the two edges immediately preceding and succeeding e_j. For example, the edge succeeding e_j is $\{a, b\}$ for some b. We can recover b by taking the sum of a (which is already known) and the parity $a+b$. By induction, we can recover all the elements along the path P_i. □

v	Blocks
6	$\{0,1,2\}\{0,2,3\}\{0,1,4\}\{1,2,5\}\{0,3,5\}\{2,3,4\}\{0,4,5\}$ $\{1,3,4\}\{1,3,5\}\{2,4,5\}$
7	$\{0,1,2\}\{0,1,3\}\{0,2,4\}\{0,3,5\}\{0,4,6\}\{0,5,6\}\{1,2,6\}$ $\{1,3,4\}\{1,4,5\}\{1,5,6\}\{2,3,5\}\{2,3,6\}\{2,4,5\}\{2,4,6\}$

TABLE I: Small $(v,\{3\},2)^\star$-designs.

The $(n, R_{\mathsf{in}}, R_{\mathsf{out}}; t)$ switch code of Theorem 1 is optimal in the case $R_{\mathsf{in}} = R_{\mathsf{out}}$ and $t = 2$ [1]. But it remains open if it is optimal for general parameters.

IV. Switch Codes for One-Burst Requests

In this section, we consider switch codes serving any one burst request with $R_{\mathsf{in}} = R_{\mathsf{out}} = R$ input and output ports. Wang et al. [1] used a linear construction to construct switch codes serving any one-burst request with redundancy $R(R-1)/3$, where $R > 3$ is a prime number such that -3 is a perfect square in $\mathbb{F}_R$. In this section, we utilize techniques from combinatorial design theory to construct switch codes with redundancy $R(R-1)/3$ serving any one-burst request, for a denser set of parameters. Before diving into details, let us first introduce some definitions.

Generalizing the graph representation in the last section, we use set systems to simplify the discussion on systematic switch codes. A *set system* is a pair $(X, \mathcal{B})$, where X is a finite set of *points* and $\mathcal{B}$ is a collection of subsets of X. The elements of $\mathcal{B}$ are called *blocks*. A set system is said to be *simple* if $\mathcal{B}$ contains no repeated blocks.

We can associate with every binary systematic switch code a set system $(X, \mathcal{B})$ by identifying each systematic bit with a point in X and each parity bit with a block in $\mathcal{B}$ containing the information elements involved in the parity bit. For example, let $U = (u_0, u_1, \ldots, u_{R_{\mathsf{in}}-1})$ be the R_{in} input bits and $(X, \mathcal{B})$ be a set system with $X = [0, R_{\mathsf{in}} - 1]$. We define the encoding function associated with $(X, \mathcal{B})$ as follows. The written bits have length $n = R_{\mathsf{in}} + |\mathcal{B}|$ and consist of two parts: systematic bits $x_i = u_i$ for $i \in [0, R_{\mathsf{in}} - 1]$ and parity bits $x_B = \sum_{j \in B} u_j$ for $B \in \mathcal{B}$. Hence every set system $(X, \mathcal{B})$ represents the encoding function of a systematic switch code with $R_{\mathsf{in}} = |X|$ input bits and $n - R_{\mathsf{in}} = |\mathcal{B}|$ redundant parity bits.

Let K be a set of integers. A (v, K, λ)-design is a set system $(X, \mathcal{B})$ with $|X| = v$ and $|B| \in K$ for all $B \in \mathcal{B}$, such that each 2-subset of X appears in exactly λ blocks. We define a $(v, \{3\}, 2)^\star$-*design* to be a simple $(v, \{3\}, 2)$-design $(X, \mathcal{B})$ satisfying the following conditions:

- (C1) $|\mathcal{B}_a \cap \mathcal{B}_b| = 1$ for all distinct $a, b \in X$, where $\mathcal{B}_a = \{\{x, y\} : \{a, x, y\} \in \mathcal{B}\}$.
- (C2) There do not exist three blocks of the form $\{a, b, c\}$, $\{a, b, d\}$, $\{b, c, d\}$ in $\mathcal{B}$.

Examples of $(v, \{3\}, 2)^\star$-designs for $v \in \{6, 7\}$ are presented in Table I.

Theorem 2. *A $(v, \{3\}, 2)^\star$-design gives a solution for any one-burst request of $R_{in} = R_{out} = v$ bits.*

Proof. Assume the request vector is $(0, 0, \ldots, 0, s_1, \ldots, s_{v-t})$ with burst length of 0 being t and $v - t$ other singletons, $1 \le t \le v$. To recover one copy of 0 and all the singletons, we directly read and output the corresponding systematic bits. Let the unrequested elements be $h_1, h_2, \ldots, h_{t-1}$, we will use them to recover the other $t - 1$ copies of 0.

For each $1 \le i \le t-1$, by condition (C1), there is a unique pair $\{a_i, b_i\}$ in both $\mathcal{B}_0$ and $\mathcal{B}_{h_i}$. Denote the block $\{0, a_i, b_i\}$ by B_i^0 and $\{h_i, a_i, b_i\}$ by B_i^1. To recover the request element 0 (or equivalently the input bit u_0), we compute the sum of the systematic element h_i and the two parity bits corresponding to blocks B_i^0 and B_i^1, that is, $u_0 = x_{h_i} + x_{B_i^0} + x_{B_i^1} = u_{h_i} + (u_0 + u_{a_i} + u_{b_i}) + (u_{a_i} + u_{b_i} + u_{h_i})$.

To ensure the decoding procedures read at most one bit from each memory bank, we need to verity that the $2(t-1)$ blocks B_i^0, B_i^1 are all distinct, $1 \le i \le t-1$. Since $0 \in B_i^0$ and $0 \notin B_j^1$, we have $B_i^0 \ne B_j^1$ for any $1 \le i, j \le t-1$. If there exist $1 \le i < j \le t-1$ such that $B_i^0 = B_j^0$, then the pair $\{a_i, b_i\} = \{a_j, b_j\}$ are contained in three blocks $B_i^0 = B_j^0$, B_i^1 and B_j^1 of $\mathcal{B}$. This contradicts the definition of the design. If $B_i^1 = B_j^1$ for some $1 \le i < j \le t-1$, then the three blocks B_i^0, B_j^0 and $B_i^1 = B_j^1$ violate condition (C2). □

We now present a recursive construction for $(v, \{3\}, 2)^\star$-designs. Some auxiliary designs are required. A *group divisible design*, denoted (K, λ)-GDD, is a triple $(X, \mathcal{G}, \mathcal{B})$ such that

- (i) $(X, \mathcal{B})$ is a set system and $|B| \in K$ for $B \in \mathcal{B}$,
- (ii) $\mathcal{G}$ is a partition of X into parts called *groups*,
- (iii) any pair of points contained in a group does not appear in any block of $\mathcal{B}$, and
- (iv) any other pair of points appears in exactly λ blocks of $\mathcal{B}$.

The *type* of the GDD is the multiset $\{|G| : G \in \mathcal{G}\}$ and is denoted by an "exponential" notation: $1^{i_1}2^{i_2}3^{i_3}\cdots$ denotes i_m occurrences of m. Note that a (v, K, λ)-design is a special (K, λ)-GDD with each group containing only one point.

We define a $(\{3\}, \lambda)^\star$-GDD to be a simple $(\{3\}, \lambda)$-GDD satisfying conditions (C1′), (C2′) and (C2), where (C1′) and (C2′) are the following:

- (C1′) For all distinct i, j such that $\{i, j\} \in G$ for some $G \in \mathcal{G}$, we have $|\mathcal{B}_i \cap \mathcal{B}_j| = 0$.
- (C2′) For all distinct i, j such that $\{i, j\} \not\subseteq G$ for any $G \in \mathcal{G}$, $|\mathcal{B}_i \cap \mathcal{B}_j| = 1$.

We adapt well known methods such as Wilson's Fundamental Construction and Filling In Groups Construction (see, for example, [3, Section IV.2.1]) for the construction of $(v, \{3\}, 2)^\star$-designs and $(\{3\}, \lambda)^\star$-GDDs.

Theorem 3. *(Fundamental Construction) Let $(X, \mathcal{G}, \mathcal{B})$ be a* (master) *$(K, 1)$-GDD, and $\omega : X \to \mathbb{Z}_{\ge 0}$ be a weight function. For any subset $S \subseteq X$, let $\widehat{S} = \cup_{a \in S}(\{a\} \times \mathbb{Z}_{\omega(a)})$. Suppose that for each $B \in \mathcal{B}$, there exists an* (ingredient) *$(\{3\}, 2)^\star$-GDD $(\widehat{B}, \{\widehat{\{x\}} : x \in B\}, \mathcal{C}_B)$ of type $\{\omega(x) : x \in B\}$. Then $(\widehat{X}, \{\widehat{G} : G \in \mathcal{G}\}, \cup_{B \in \mathcal{B}} \mathcal{C}_B)$ is a $(\{3\}, 2)^\star$-GDD of type $\{\sum_{x \in G} \omega(x) : G \in \mathcal{G}\}$.*

t	$\mathcal{A}_t$
6	$\{0,13,27\}\{0,3,28\}\{0,23,28\}\{0,7,17\}\{0,20,35\}$ $\{0,22,26\}\{0,27,34\}\{0,21,32\}\{0,16,17\}\{0,31,34\}$
7	$\{0,6,19\}\{0,26,31\}\{0,8,12\}\{0,18,23\}\{0,1,4\}\{0,9,24\}$ $\{0,17,27\}\{0,6,31\}\{0,8,41\}\{0,3,16\}\{0,2,12\}\{0,2,22\}$
8	$\{0,12,35\}\{0,6,44\}\{0,5,34\}\{0,7,44\}\{0,11,25\}$ $\{0,12,45\}\{0,13,30\}\{0,19,47\}\{0,7,9\}\{0,5,27\}$ $\{0,10,30\}\{0,15,42\}\{0,9,26\}\{0,2,47\}$
9	$\{0,4,14\}\{0,8,34\}\{0,12,46\}\{0,11,17\}\{0,1,30\}\{0,2,50\}$ $\{0,39,44\}\{0,32,37\}\{0,11,12\}\{0,23,38\}\{0,40,47\}$ $\{0,16,29\}\{0,28,51\}\{0,19,41\}\{0,21,24\}\{0,19,52\}$

TABLE II: Base blocks for small $(\{3\},2)^\star$-GDDs.

Theorem 4. *(Filling In Groups) Let $s \in \{0,1\}$. Suppose $(X,\mathcal{G},\mathcal{B})$ is a $(\{3\},2)^\star$-GDD. Let S be a set of size s disjoint from X, and $X' = X \cup S$. Suppose further that for each $G \in \mathcal{G}$, there exists a $(|G|+s,\{3\},2)^\star$-design $(G \cup S, \mathcal{B}_G)$. Then $(X \cup S, \mathcal{B} \cup (\cup_{G\in\mathcal{G}}\mathcal{B}_G))$ is an $(|X|+s,\{3\},2)^\star$-design.*

Lemma 1. *There exists a $(\{3\},2)^\star$-GDD of type 6^t for $t \in [6,9]$.*

Proof. For $t \in [6,9]$, let $X_t = \mathbb{Z}_{6t}$, $\mathcal{G}_t = \{\{0,t,2t,3t,4t,5t\}+i : 0 \le i \le t-1\}$ and $\mathcal{B}_t = \{B+i : B \in \mathcal{A}_t, i \in \mathbb{Z}_{6t}\}$, where $\mathcal{A}_t$ is as listed in Table II. Then $(X_t,\mathcal{G}_t,\mathcal{B}_t)$ is the desired $(\{3\},2)^\star$-GDD of type 6^t. □

Proposition 4. *There exists a $(v,\{3\},2)^\star$-design for all $v \equiv 0,1 \bmod 6$, $v \ge 1116$.*

Proof. Let $K = [6,9]$. There exists a $(t,K,1)$-design for any $t \ge 186$ by [4], [5]. Taking $(t,K,1)$-designs as master design and $(\{3\},2)^\star$-GDDs of type 6^k for $k \in K$ as ingredient designs, we apply Theorem 3 and obtain $(\{3\},2)^\star$-GDDs of type 6^t for all $t \ge 186$. Since we have $(v,\{3\},2)^\star$-design for $v \in \{6,7\}$, Theorem 4 implies that we have $(v,\{3\},2)^\star$-design for all $v \equiv 0,1 \bmod 6$, $v \ge 1116$. □

Combining Theorem 2 and Proposition 4, we have:

Corollary 2. *For any $R \equiv 0,1 \bmod 6$, $R \ge 1116$, there exists an (n,R)-switch code with redundancy $n-R = R(R-1)/3$ that serves any one-burst request.*

V. Switch Codes for Limited-Repetition Requests

In this section, we consider a special case of limited-repetition requests. More specifically, we study the requests $\mathbb{L}$ consisting of all request vectors L with at most s integers $m_0, m_1, \ldots, m_{s-1}$ appearing more than once, for some fixed integer k and $0 \le s \le (R-1)/(k-1)$. We further require each m_i to appear at most $(R-1)/(k-1)-i+1$ times in L.

Theorem 5. *If there exists an $(R,\{k\},1)$-design, then there exists an (n,R)-switch code with redundancy $\frac{R(R-1)(k+1)}{k(k-1)}$ that can serve any request vectors in $\mathbb{L}$.*

Proof. Let $(X,\mathcal{B})$ be an $(R,\{k\},1)$-design. For each block $B \in \mathcal{B}$, let $\mathcal{A}_B = \{A \subseteq B : |A| = k-1\}$. We claim that the switch code $\mathcal{C}$, using the parity checks $\mathcal{A} = \mathcal{B} \cup (\cup_{B\in\mathcal{B}}\mathcal{A}_B)$, can serve any request in $\mathbb{L}$. Clearly, the redundancy of $\mathcal{C}$ is $|\mathcal{A}| = |\mathcal{B}|(k+1) = \frac{R(R-1)(k+1)}{k(k-1)}$.

For a request $L \in \mathbb{L}$, we recover all the distinct elements in L once by directly reading the systematic nodes. Let $\tau_L = \{m_0, m_1, \ldots, m_{s-1}\}$ be the integers in L appearing more than once. For each element $m \in \tau_L$, there are $\frac{R-1}{k-1}$ blocks in $\mathcal{B}$ containing it. So we can recover m_0 for additional $(R-1)/(k-1)$ times by the sum of two parity bits storing B and $B \setminus \{m_0\}$ for each block $B \in \mathcal{B}$ containing m_0. For each $1 \le i \le s-1$, there exists exactly one block containing both m_i and m_j for $0 \le j \le i-1$, and it has already been used to recover the element m_j. Therefore, at most $(R-1)/(k-1)-i$ blocks containing m_i but no m_j, $j < i$, can be used to recover m_i. Hence $\mathcal{C}$ serves any requests from $\mathbb{L}$. □

Example 2. Let $X = \mathbb{Z}_{13}$ and $\mathcal{B} = \{B_i = \{i, i+1, i+4, i+6\} : i \in \mathbb{Z}_{13}\}$. Then $(X,\mathcal{B})$ is a $(13,4,1)$-design. Take an arbitrary 3-subset from X, e.g., $\tau_L = \{0,1,2\}$. We show a solution for the request vector $L = (0,0,0,0,0,1,1,1,1,2,2,2,5)$.

For all the distinct elements $\{0,1,2,5\}$ in L, we can directly return the corresponding systematic nodes. For $0 \in \tau_L$, we can further resolve it four times with parity nodes containing 0 and their reductions, that is, B_i and $B_i \setminus \{0\}$ for $i = 0$, 7, 9 and 12. Then for $1 \in \tau_L$, we can resolve it with parity nodes containing 1 but not 0 and their reductions, that is, B_i and $B_i \setminus \{1\}$ for $i = 1$, 8 and 10. Finally, for $2 \in \tau_L$, we can resolve it with parity nodes containing 2 but not 0 or 1, that is, B_i and $B_i \setminus \{2\}$ for $i = 2$ and 11.

Example 3. Let q be a prime power. By Theorem 5, with the finite projective plane of order q, which is a $(q^2+q+1, q+1, 1)$-design, we get a switch code of q^2+q+1 input and output ports with redundancy $(q^2+q+1)(q+2)$. With a the finite affine plane of order q, which is a $(q^2, q, 1)$-design, we get a switch code of q^2 input and output ports with redundancy $(q+1)^2 q$. They both serve all requests with $s \le q+1$ elements requested at most $q-s+3$ times and other bits requested at most once. The asymptotic redundancies of both codes are $O(R^{3/2})$.

VI. Conclusion

In this paper, we give new combinatorial constructions of switch codes serving consecutive-generation requests, one burst requests and limited-repetition requests with lower redundancies.

References

[1] Z. Wang, O. Shaked, Y. Cassuto, and J. Bruck, "Codes for network switches," in *ISIT 2013 – Proceedings of the 2013 IEEE International Symposium on Information Theory*, 2013, pp. 1057–1061.

[2] N. Hartsfield and G. Ringel, *Pearls in graph theory*. Academic Press, Inc., Boston, MA, 1994.

[3] C. J. Colbourn and J. H. Dinitz, Eds., *Handbook of combinatorial designs*, 2nd ed. Chapman & Hall/CRC, Boca Raton, FL, 2007.

[4] A. C. H. Ling, X. Zhu, C. J. Colbourn, and R. C. Mullin, "Pairwise balanced designs with consecutive block sizes," *Des. Codes Cryptogr.*, vol. 10, no. 2, pp. 203–222, 1997.

[5] C. J. Colbourn, E. R. Lamken, A. C. H. Ling, and W. H. Mills, "The existence of Kirkman squares—doubly resolvable $(\nu,3,1)$-BIBDs," *Des. Codes Cryptogr.*, vol. 26, no. 1-3, pp. 169–196, 2002.