

Algorithms and Throughput Analysis for MDS-Coded Switches

Rami Cohen and Yuval Cassuto

Department of Electrical Engineering

Technion - Israel Institute of Technology

Technion City, Haifa 3200003, Israel

Email: rc@tx.technion.ac.il, ycassuto@ee.technion.ac.il

Abstract—Network switches and routers need to serve packet writes and reads at rates that challenge the most advanced memory technologies. As a result, scaling the switching rates is commonly done by parallelizing the packet I/Os using multiple memory units. For improved read rates, packets can be coded with an $[n,k]$ MDS code, thus giving more flexibility at read time to achieve higher utilization of the memory units. In the paper, we study the usage of $[n,k]$ MDS codes in a switching environment. In particular, we study the algorithmic problem of maximizing the instantaneous read rate given a set of packet requests and the current layout of the coded packets in memory. The most interesting results from practical standpoint show how the complexity of reaching optimal read rate depends strongly on the writing policy of the coded packets.

I. INTRODUCTION

Ever increasing demand for network bandwidth pressures switch and router vendors to scale their products at a fast pace. The most crucial component for throughput scaling is the memory sub-system that comprises the switching fabric. As it becomes extremely difficult or prohibitively costly to scale the read and write rates of memory units (MUs), an alternative solution of choice is to deploy multiple such units in parallel. Memory contention in switches occurs when multiple packets requested for read at a given instant happen to reside in the same memory unit. Because every memory unit can deliver a single chunk of data per time instant, this contention will result in a loss of switching throughput (while at the same time a different memory unit will be idle). Our ability to avoid such contention by clever packet placement is limited by the fact that the reading schedule of packets is not known upon arrival of the packets to the switch.

This issue has driven switch vendors to seek methods to reduce memory contention and thus increase the switching throughput. One particularly promising technique is to introduce *redundancy* to the packet-write path, such that upon read the switch controller will enjoy greater flexibility to read the requested packets from memory units not contended by other read requests. This redundancy is introduced in the form of *coding*, whereby additional coded chunks are calculated from the incoming packet and written along with it in the switch memory. The simplest scheme of coding applied to packets is *replication*, where the additional chunks written with the packet are simply copies of the same packet. The advantage of this scheme is in its simplicity, requiring only trivial encoding and decoding. A more advanced packet-coding scheme uses $[n,k]$ maximum distance separable (MDS) codes [1]. This coding scheme takes an input of k packet chunks and encodes them into a codeword of n chunks ($k \leq n$), where any

k chunks taken from the n code chunks can be used for the recovery of the original k packet chunks. This maximal flexibility in retrieving the packet makes MDS codes very well suited to use in switch memories. Latency comparison between replication and MDS codes was pioneered by Huang et al. [2]. It was shown that for $k = 2$, the average latency for serving a packet decreases significantly when a certain scheduling model is used. This analysis was later extended by Shah et al. in [3], [4], where bounds on latency performance under multiple scheduling policies were investigated.

In this paper, we provide a model of a coded switch that considers the number of memory units in use and the code parameters. Then we put our focus on maximizing and analyzing the *throughput* of the switch. The notion of throughput we pursue here is the active MUs serving packets out of the MUs in the switch. Increasing the number of packets in the system (i.e., the load) gives more choice to the reader, and is thus expected to improve the throughput. For the probabilistic analysis of throughput we use a static distribution on the requested-packets' locations (governed by the write policy), and assume that the switch observes this static distribution in steady state. The problem of achieving maximal throughput is formulated in two equivalent ways as problems in set theory and in graph theory. These formulations allow us later to obtain insights, algorithms, and bounds for the problem.

This paper is structured as follows. In Section II, we provide complexity analysis of the throughput maximization problem, where in Section III we discuss suboptimal algorithms and bounds on the optimal solution. We then provide a structured version of the problem in Section IV and show that it admits a polynomial-time solution. Finally, conclusions are given in Section V.

II. PROBLEM FORMULATION AND COMPLEXITY

Consider a scenario in which multiple packets are stored in MUs, to be later forwarded by a switch. Let us assume that each packet consists of k chunks, which are MDS-encoded into n chunks. The n chunks are then stored in n MUs out of N available ones ($k \leq n \leq N$), where overlapping is allowed (i.e., two or more packets may share one or more MUs). Out of many packets currently stored in the switch memory, a request arrives for L packets, with the objective to read as many out of these packets in a single time instant. Recall that k chunks out of the n encoded ones are sufficient for recovering a packet. We know in which MUs these L packets are stored, and wish to find methods for reading as many packets as possible simultaneously, with the constraint that

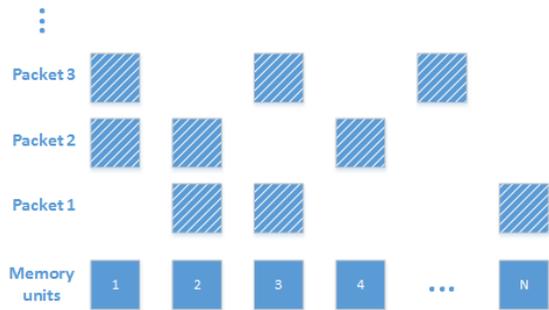


Fig. 1: Illustration of nkMTP. The patterned rectangles represent encoded data chunks ($n = 3$).

each MU can be accessed *only once* in the reading process. Let us denote by L^* the maximal number of packets that can be read. We consider the following notion of throughput as a performance measure.

Definition 1: (Throughput)

The throughput ρ of the system is defined as:

$$\rho = \frac{L^*k}{N}. \quad (1)$$

We name the problem of maximizing the throughput ρ as the $[n, k]$ -maximal throughput problem, or nkMTP. An instance of the problem is illustrated in Figure 1, where data chunks of multiple packets are shown on top of MUs storing them.

The motivation for using MDS codes in this setting rather than simple replication is demonstrated in Figure 2. Here we compare the use of a $[4, 2]$ MDS code for encoding 2 chunks of a packet to 2 uses of a $[2, 1]$ repetition code (i.e., 2-way replication), one use for each chunk. To read a packet when the repetition code is used, the reader needs to read one chunk from a specific subset of 2 MUs out of the 4 MUs storing the packet, and an additional chunk from the other subset of 2 MUs. On the other hand, when a $[4, 2]$ MDS code is used, the reader can recover the packet by reading two chunks from *any* 2 MUs out of the 4 MUs storing the packet chunks. The resulting curves in Figure 2 show that on average the $[4, 2]$ MDS code allows reading more simultaneous packets than the $[2, 1]$ repetition code. Note that both schemes write 4 chunks per packet, and hence this advantage comes without increasing the write load.

Evidently, the way packets are stored dictates how they should be read for optimal results. In particular, each write/read scheme combination may result in different number of packets that can be read. In this work, we will consider two write schemes that are easy to implement, and their optimally matched read schemes. For analysis purposes, we provide in the rest of this section two equivalent formulations of nkMTP. Consider the N available MUs as the elements of the set $S = \{1, 2, \dots, N\}$. Each packet $i = 1, 2, \dots, L$ is stored in MUs indexed by a subset S_i of S , where $|S_i| = n$ and the subsets may overlap. The set theory formulation of nkMTP is as follows.

Problem 1: (nkMTP, set theory formulation)

Input: Set $S = \{1, 2, \dots, N\}$ and L subsets of S , $S_i \subseteq S$, such that $|S_i| = n$.

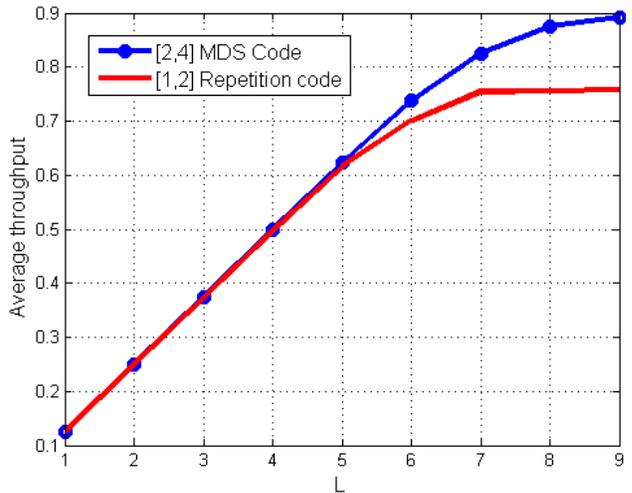


Fig. 2: Comparison between a $[4, 2]$ MDS code and two uses of the $[2, 1]$ repetition code ($N = 16$).

Output: Subsets $S'_i \subseteq S_i$, $|S'_i| = k$, $S'_i \cap S'_j = \emptyset$ ($i \neq j$), such that the number of subsets is maximal.

Example 1:

$N = 5, L = 3, n = 3$. The packets are stored in the MUs indexed by the sets $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 4, 5\}$, $S_3 = \{3, 4, 5\}$. If $k = n = 3$, we have that $L^* = 1$ and the recovered packet can be either 1, 2 or 3 since $S_i \cap S_j \neq \emptyset$ for all $i, j = 1, 2, 3$. If $k = 2$, a possible solution is $S'_1 = \{1, 2\}$, $S'_2 = \{4, 5\}$ with $L^* = 2$. Note that no more than 2 packets can be read in this case, since $L^* \leq \lfloor N/k \rfloor = 2$. Finally, if $k = 1$ all the packets can be read, and one possible solution is $S'_1 = \{1\}$, $S'_2 = \{2\}$, $S'_3 = \{3\}$.

In addition to the set theory formulation, nkMTP can be formulated equivalently on a graph. Consider a *bipartite* graph $G = (V_G, E_G)$, where V_G denotes the vertices of G and E_G denotes the edges of G . In addition, let us denote by $\deg(v)$ the *degree* of a vertex $v \in V_G$. Since G is bipartite, V_G can be partitioned into two *disjoint* sets of vertices, let us denote them by X_G and Y_G . Thinking of X_G as packets, and of Y_G as MUs, vertex i in X_G will be connected to vertex j in Y_G if one of the encoded chunks of packet i is stored in MU j . The resulting graph has the following properties: $\deg(x) = n$ ($\forall x \in X_G$), $|X_G| = L$, $|Y_G| = N$, and $|E_G| = nL$. nkMTP can be now formulated as follows.

Problem 2: (nkMTP, graph theory formulation)

Input: Graph representation G of nkMTP.

Output: Subsets $X_H \subseteq X_G, Y_H \subseteq Y_G, E_H \subseteq E_G$ with the properties $\deg(x) = k, \deg(y) \in \{0, 1\}$ ($\forall x \in X_H, y \in Y_H$), such that $|X_H|$ is maximal.

For instance, Example 1 is represented as the graph in Figure 3. The output of Problem 2 is essentially a *subgraph* H of G , providing a solution to nkMTP with throughput $\rho = |X_H|k/N$.

A. Complexity

In this subsection, we analyze the computational complexity of nkMTP. Clearly, a simple approach for solving nkMTP is to consider all possible assignments of MUs to packets, and to choose the assignment leading to the maximal number of packets that can be read. However, this approach is clearly

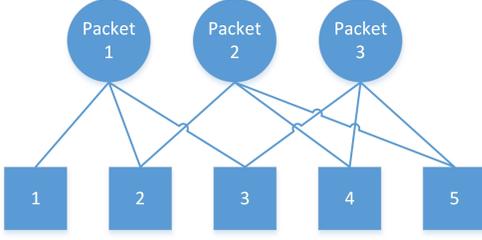


Fig. 3: nkMTP from Example 1 formulated on a graph.

inefficient since its complexity scales exponentially in L . In fact, nkMTP can be solved in polynomial time if $k = 1, n \geq 1$ (i.e., each packet consists of one chunk and the repetition code is used) or $k = n = 2$. On the other hand, nkMTP is NP-hard for $3 \leq k \leq n$.

Theorem 1: For $k = 1, n \geq 1$ or $k = n = 2$, nkMTP is solvable in polynomial time.

Proof When $k = 1, n \geq 1$, nkMTP is equivalent to finding a subgraph H of G (a graph representation of nkMTP) that is a *maximum bipartite matching* [5], i.e., containing the largest number of *matched* pairs (x, y) , $x \in X_G, y \in Y_G$, such that each pair is connected by an edge and the edges are pair-wise non-adjacent. When $k = n = 2$, consider the N MUs as the vertices of a (uni-partite) graph, where an edge in this graph connects two MUs shared by the same packet. A *maximum matching* in this graph will provide the largest number of disjoint pairs of MUs, each pair serving a packet, corresponding to an optimal solution of the nkMTP instance. Efficient algorithms are known for finding maximum matching in both cases [5]. ■

Theorem 2: nkMTP is NP-hard for $3 \leq k \leq n$.

To prove Theorem 2, we *reduce* the l -set packing (l -SP) problem [6], known to be NP-hard, to nkMTP. In l -SP, there are L sets, each of size l , and the problem is to find the maximal number of pair-wise disjoint sets. The details of the reduction are provided in [7]. The consequence of the hardness result of Theorem 2 is that no efficient algorithms are expected to be found for solving nkMTP when $3 \leq k \leq n$. However, in the next sections we provide algorithmic and analytic results that help solving the nkMTP problem in practical settings. We will also see variants of nkMTP for which we do find polynomial-time algorithms.

III. PROBABILISTIC ANALYSIS AND BOUNDS

The fact that nkMTP turns out to be NP-hard for interesting coding parameters is important theoretically, but should not discourage one from seeking high-throughput coded switching. In this section we provide tools that will help find good coded-switching solutions in a practical setup.

A. Lower bounding the maximal solution

In this sub-section, we provide a lower bound on the number of packets that can be read, for a given nkMTP instance. Consider the following randomized algorithm, applied to a graph formulation of nkMTP (Problem 2):

Algorithm 1:

- 1) Calculate the degree of each MU.

- 2) Assign each MU independently at random to one of its connected packets with probability $1/d$, where d is the MU degree.
- 3) Return the set of packets having at least k connected MUs.

The packets in the set returned by Algorithm 1 can be read, since at least k chunks are available for each packet. We now turn to calculate the expected size of the set returned by Algorithm 1, which will be shown to serve as a lower bound on L^* . Denote by l_i an indicator random variable that equals 1 if packet i was assigned at least k MUs, and equals 0 otherwise. In addition, define auxiliary indicator random variables $l_{i,j}$, which equals 1 if MU j from the set of MUs connected to packet i (i.e., $j \in S_i$, where S_i are defined in Section II) ends up connected to packet i . Thus:

$$\Pr(l_i = 1) = \Pr\left(\sum_{j \in S_i} l_{i,j} \geq k\right). \quad (2)$$

Denote by $d_{i,j}$ the degree of MU j connected to packet i . $l_{i,j}$ are Bernoulli random variables whose success probabilities are $1/d_{i,j}$. These random variables are independent for the same i and different j , but not identically distributed (since the MU degrees may vary). The distribution of these random variables is called *Poisson binomial distribution* [8]. A closed-form expression for the right-hand side of Equation (2), denoted $Q(i)$, is obtained using [8]:

$$Q(i) = 1 - \sum_{s=0}^{n-1} \left\{ \left[\sum_{t=0}^{k-1} e^{-j2\pi st/(n+1)} \right] \cdot \prod_{j \in S_i} \left[(1/d_{i,j}) \cdot e^{j2\pi s/(n+1)} + (1 - 1/d_{i,j}) \right] \right\} / (n+1). \quad (3)$$

We are now ready to provide a lower bound on L^* .

Theorem 3: For a given nkMTP instance, the number of packets that can be read is lower-bounded as follows:

$$L^* \geq \sum_{i=1}^L Q(i). \quad (4)$$

Proof Denote by I the cardinality of the set returned by Algorithm 1. I is a random variable and its expected value (over realizations of sets provided by Algorithm 1) is:

$$E[I] = E\left[\sum_{i=1}^L l_i\right] = \sum_{i=1}^L E[l_i] = \sum_{i=1}^L \Pr(l_i = 1) = \sum_{i=1}^L Q(i). \quad (5)$$

Since $E[I]$ is an expected value, there must exist a valid solution to nkMTP with cardinality at least $E[I]$. The existence of such set leads to the lower bound (5). ■

Example 2:

Consider the nkMTP instance of Figure 3. The MU degrees are $d_{1,1} = 1, d_{1,2} = 2, d_{1,3} = 2$ and so on. The lower bounds on L^* , obtained using (4), are 2.75, 1.75 and 0.5 for $k = 1, k = 2$ and $k = 3$, respectively, where the corresponding L^* values are 3, 2 and 1.

B. Expected performance of nkMTP ensembles

In this sub-section, we analyze the nkMTP in a random setting, where we consider random *ensembles* of nkMTP instances with fixed parameters k, n, N and L . Assuming a graph formulation (Problem 2), an instance taken from an ensemble consists of a graph G with L packets and N MUs, where the n chunks of each packet are stored independently and uniformly at random at n MUs. For each ensemble, we would like to estimate the probability of maximum throughput, i.e. the existence of a solution in form of a subgraph H such that $X_H = X_G$. The method is to identify a condition for the existence of such a solution, derived from the following extension of Hall's theorem.

Theorem 4: (Extended Hall's Theorem [9]) Consider a bipartite graph $G = (X_G, Y_G, E_G)$. Then, G satisfies $\deg(x) = k$ and $\deg(y) \in \{0, 1\}$ for all $x \in X_G$ and all $y \in Y_G$, if and only if for every subset W of X_G ,

$$|T(W)| \geq k|W|, \quad (6)$$

where $T(W)$ is the set of vertices (in Y_G) adjacent to the vertices in W .

That is, the extended Hall's theorem provides a necessary and sufficient condition to determine whether MUs in Y_G can be assigned to packets in X_G , such that the degree of each vertex in X_G will be k . This is equivalent to the existence of a maximum throughput solution in G (i.e., each packet can be served). In the sequel we say that *Hall's condition* holds for a given subset W of X_G if the condition (6) of Theorem 4 holds.

Denote by $u_m = u_m(|W|, n; N)$ the probability that the union of $|W|$ sets, each containing n elements taken independently and uniformly at random from the set $\{1, 2, \dots, N\}$, results in a set of size m ($m = 1, 2, \dots, N$). An expression for u_m was derived in a previous work by us [10], based on a Markov model that is an extension of the balls-and-bins model [11]. For a given subset W of X_G , denote by P_W the probability that the number of neighbours of W is greater than or equal to $k|W|$ (meaning that Hall's condition holds for W). Since each vertex in W is of degree n , P_W is equivalent to the probability that the random union of $|W|$ sets of size n (the number of MUs that store each packet) results in a set of size that is greater than or equal to $k|W|$. Thus,

$$P_W = \sum_{m=k|W|}^N u_m(|W|; n, N). \quad (7)$$

Note that P_W from (7) is independent of the particular choice of W . Instead, it depends on the *size* of W , i.e., $P_W = P_{|W|}$.

Calculating the probability that all subsets $W \subseteq X_G$ satisfy Hall's condition is a difficult task, due to dependencies between the subsets. However, a *necessary* condition for an instance to contain a maximum throughput solution is that the set X_G satisfies Hall's condition, which happens with probability $P_{|X_G|}$. Therefore, an upper bound on the probability that an instance contains a maximum throughput solution is simply $P_{|X_G|}$. As $P_{|X_G|}$ is obtained by raising a Markov matrix to power $|X_G|$, its calculation is efficient. In Figure 4, we first calculate the probability that a random graph G contains a maximum throughput solution, by averaging

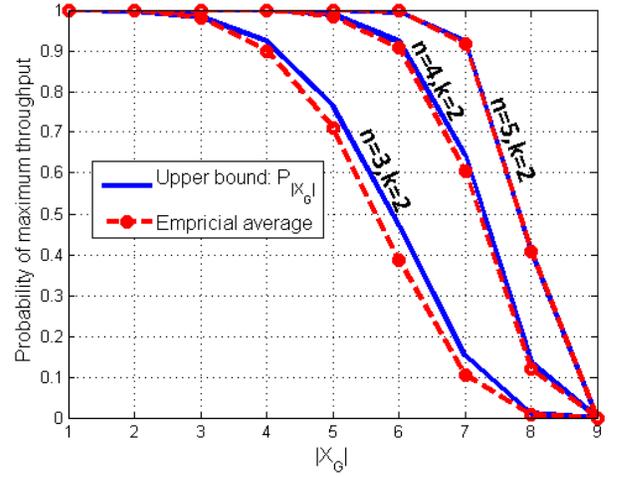


Fig. 4: The probability that all $|X_G|$ packets of a random nkMTP instance can be served, as a function of k and n ($N = 16$).

over the optimal solution sizes of 10,000 instances for $k = 2$ and varying values of n . We then compare this empirical average to the upper bound obtained using $P_{|X_G|}$. The results show that the upper bound provided by $P_{|X_G|}$ is tight and captures the behaviour of the empirical average. The bound is tighter as n becomes larger compared to k , meaning that in such cases the probability that Hall's condition is satisfied for all X_G subsets is dominated by the probability that the set X_G satisfies Hall's condition. As expected, better average performance is achieved as n increases (i.e., when redundancy increases). It is also demonstrated that $P_{|X_G|} = 0$ for $|X_G| > \lfloor N/k \rfloor$, since the maximal number of packets that can be served is upper bounded by $\lfloor N/k \rfloor$.

Hall's condition provides a convenient way for estimating expected performance of an ensemble, when concentrating on maximum throughput solution. This way, instead of estimating the probability of maximum throughput empirically by averaging over the optimal solution sizes of many nkMTP instances (whose solution is hard in general, see Section II-A), we may resort to calculate $P_{|X_G|}$ instead. This is especially useful for large values of n and L , for which a direct solution of nkMTP (e.g., by considering all possible assignments of MUs to packets), may be prohibitively complex. In addition, we obtained an efficient way for choosing parameters k and n and load L such that good performance is expected.

IV. POLYNOMIAL-TIME SOLUTION FOR A STRUCTURED VARIANT OF NKMTP

Motivated by the hardness of the nkMTP problem proved above, we now turn to consider variants of the problem that can be solved efficiently. Our approach to make the problem easier to solve is by changing the packet writing policy upon packet arrival. We show in this section that, by a moderate restriction on the MUs chosen to store the n chunks of a packet, we turn the maximal-throughput read problem to a tractable one.

To the nkMTP problem discussed above we add the constraint that the n encoded chunks of each packet are stored in n *consecutive* MUs. We will refer to this scheme as

CnkMTP, for which we show that polynomial-time solution exists. We first sort the sets S_i in a non-decreasing order of the maximal index of the MUs they contain, and w.l.o.g. we consider instances of CnkMTP in which the packets are sorted accordingly. We begin with an empty set of packets, denoted Λ , and an empty set of MUs, denoted Ω . The following algorithm solves CnkMTP for $k \leq n$.

Algorithm 2:

- 1) Initialize $S'_i = S_i$ for $i = 1, \dots, L$.
- 2) Set $i := 1$.
- 3) If $|S'_i| \geq k$, add i to Λ , and add the k lowest elements of S'_i to Ω .
- 4) Remove all the elements added to Ω from all the sets S'_j , for $j > i$.
- 5) Set $i := i + 1$. If $i > L$, stop. Otherwise, go to step 3.

Theorem 5: The set of packets Λ and their corresponding MUs in Ω found by Algorithm 2 are an optimal solution to CnkMTP.

Proof The proof starts by observing that w.l.o.g. the k MUs assigned to a read packet are consecutive. If not, we can always exchange MUs between packet j and packets $j' > j$ to make the assignment consecutive.

Now we prove that if $|S'_i| \geq k$ in step 3, then i appears in the optimal solution. We prove by induction on i . Assume all packets $1, \dots, i-1$ can be chosen optimally according to Algorithm 2. Then we show that the i -th packet can be chosen in the same way. We assume by contradiction that $|S'_i| \geq k$ and there is no optimal solution that contains packet i . Then we look at the smallest $j > i$ for which packet j appears in the optimal solution. Then from the fact that its k assigned MUs are consecutive it is possible to shift the assignment to the first MU index in S'_i , and replace j by i in the optimal solution without affecting the selection of any $j' > j$. This is a contradiction. ■

The operations required in Algorithm 2 are simple shifting, in addition to the sorting of the packets. Hence its running time is clearly polynomial. We also consider a special case of CnkMTP, in which the N MUs are partitioned into N/n blocks of size n (assuming that n is a divisor of N), where the encoded chunks of each packet are restricted to one of those blocks. We term this scheme as CnkMTP on blocks. A comparison between nkMTP and CnkMTP is given in Figure 5 for $k = 3, n = 4$ and $N = 16$. This figure reveals a very interesting tradeoff between throughput and computational complexity. Adding structure to the write policy in CnkMTP results in some loss of throughput compared with the unrestricted nkMTP. However, this throughput can be attained efficiently, where for nkMTP it is intractable to reach the throughput efficiently for large problem instances. It is also shown that CnkMTP gives better throughput compared with an even more structured writing policy, CnkMTP on blocks.

V. CONCLUSION

In this paper, we analyzed the fundamental limits of using MDS codes in a switching environment. We proved that in its most general form, the problem of obtaining maximum throughput for a set of requested packets is a hard problem. Therefore, we provided bounds and algorithmic tools to aid

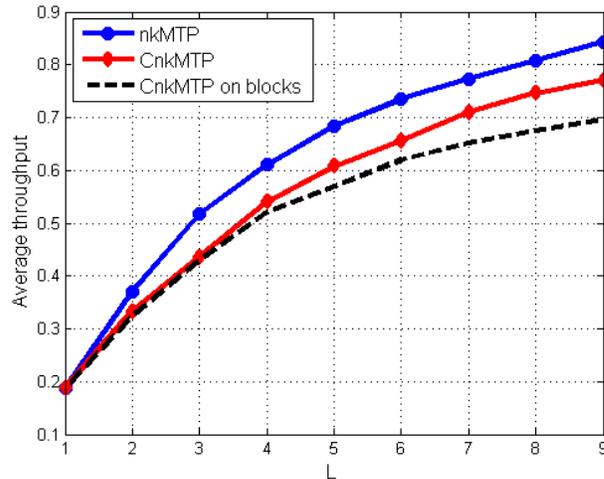


Fig. 5: Scheme comparison.

its solution in practice. By a simple modification of the writing policy used by the switch, we have shown how the problem can be solved efficiently. Our work leaves many interesting problems for future research, most immediately how to tailor the switch and code parameters to match real-life network workloads. From a practical point of view, we currently investigate the performance of certain MDS codes in a switching environment, taking encoding/decoding overhead into account.

VI. ACKNOWLEDGEMENT

The authors would like to thank A. Nekrasov and S. Shulga for their contribution to performance evaluation simulations. This work was partly conducted under a joint ISF-UGC grant. In addition, it was supported by a Marie Curie CIG grant and by the Israel Ministry of Science and Technology.

REFERENCES

- [1] W. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.
- [2] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, "Codes can reduce queueing delay in data centers," *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 2766–2770, July 2012.
- [3] N. Shah, K. Lee, and K. Ramchandran, "When do redundant requests reduce latency?" in *2013 51st Annual Allerton Conference on Communication, Control, and Computing*, Oct 2013, pp. 731–738.
- [4] —, "The MDS queue: Analysing the latency performance of erasure codes," in *2014 IEEE International Symposium on Information Theory (ISIT)*, June 2014, pp. 861–865.
- [5] A. Bondy and U. Murty, *Graph theory*. Springer, 2008.
- [6] E. Hazan, S. Safra, and O. Schwartz, "On the complexity of approximating k -set packing," *Comput. Complex.*, vol. 15, no. 1, pp. 20–39, May 2006.
- [7] R. Cohen and Y. Cassuto, "Algorithms and throughput analysis for MDS-coded switches," *arXiv:1504.04803 [cs.IT]*, April 2015.
- [8] M. Fernandez and S. Williams, "Closed-form expression for the poisson-binomial probability density function," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 2, pp. 803–817, April 2010.
- [9] M. Viderman, "LP decoding of codes with expansion parameter above $2/3$," *Information Processing Letters*, vol. 113, no. 7, pp. 225 – 228, 2013.
- [10] R. Cohen and Y. Cassuto, "LDPC codes for partial-erasure channels in multi-level memories," *2014 IEEE International Symposium on Information Theory (ISIT)*, pp. 2087–2091, July 2014.
- [11] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.