Latency Analysis for Distributed Coded Storage Systems

Ajay Badita, *Student Member, IEEE*, Parimal Parag, *Member, IEEE*, and Jean-Francois Chamberland, *Senior Member, IEEE*

***Abstract*— Modern communication and computation systems often consist of large networks of unreliable nodes. Still, it is well known that such systems can provide aggregate reliability via redundancy. While duplication may increase the load on a system, it can lead to significant performance improvement when combined with the judicious management of extra system resources. Prime examples of this abstract paradigm include multi-path routing across communication networks, content access from multiple caches in delivery networks, and master/slave computations on compute clusters. Several recent contributions in the area establish bounds on the performance of redundant systems, characterizing the latency-redundancy tradeoff under specific load profiles. Following a similar line of research, this paper introduces new analytical bounds and approximation techniques for the latency-redundancy tradeoff for a range of system loads and a class of symmetric redundancy schemes, under the assumption of Poisson arrivals, exponential service-rates, and fork-join scheduling policy. The proposed approach can be employed to efficiently approximate the latency distribution of a queueing system at equilibrium. Various metrics can subsequently be derived for this system, including the mean and variance of the sojourn time, and the tail decay rate of the stationary distribution. This paper also establishes the stability region in terms of arrival rates for redundant systems with certain symmetries. Finally, it offers selection guidelines for design parameters to provide latency guarantees based on the proposed approximations. Findings are substantiated by numerical results.**

***Index Terms*— Data storage systems, forward error correction, content distribution networks, distributed information systems, Markov processes, equilibrium distribution, queueing analysis.**

I. Introduction

THE operation of the Internet and the satisfaction of its users increasingly depend on the rapid and reliable availability of files and other data. Terabytes of digital traffic travel over network subcomponents every second, predominantly consisting of multimedia content. A large portion of this data is hosted on content delivery networks and distributed storage systems. These systems keep information redundantly at multiple servers to ensure reliability and availability, despite the fact that individual components are failure prone. Forward error correction has emerged as a fundamental means to enhance information storage and improve content dissemination across networks. For instance, preserving data over a collection of agents using maximum distance separable (MDS) coding can increase the reliability of a storage system, while limiting the amount of redundancy needed to achieve a prescribed performance level [1]. Potential applications for such a technology range from large data centers and cloud storage, to peer-to-peer systems with ubiquitous access. Recent years have seen intense investigation of several aspects of persistent storage and advanced coding for distributed systems. These topics include efficient erasure correcting codes [2], [3], recovery schemes [4], repair traffic requirements [5], [6], and related notions of security [7]–[9].

Advances in low complexity codes for distributed storage with low overhead for repair and regeneration have also fueled an interest in understanding the tradeoff between redundancy and access time [10]–[21]. This contribution is aligned with these latter considerations, and further explores the interplay between storage coding and request completion times. In this context, it is pertinent to note that content access time in distributed storage is related to the computation time of master/slave tasks in compute clusters with job replication [13]–[18]. Since a job completion queue is mathematically equivalent to a request queue for content delivery, we restrict our discussion of the problem to access delay in distributed storage systems with the understanding that the tools developed herein apply to both settings.

Fundamentally, data management in distributed storage involves a few key elements. First, the original file is partitioned into blocks and subsequently encoded into n pieces. These pieces are stored at different locations. When the file is requested, the user must gather at least k pieces to recover its content and reconstruct the original file. Requirements on which k blocks can be employed to recover the file depend on the coding strategy adopted by the system. Two encoding schemes have become emblematic cases to study the latency-redundancy tradeoff. One approach is based on file fragmentation and duplication, a version of *block repetition coding*. Under this scheme, a request is fulfilled by collecting one information piece of every possible kind. The original

Manuscript received September 6, 2017; revised January 15, 2019; accepted March 24, 2019. Date of publication April 11, 2019; date of current version July 12, 2019. This work was supported in part by the National Science Foundation (NSF) under Grant CCF-1619085, and in part by the Science and Engineering Research Board (SERB) under Grant DSTO-1677. This paper was presented in part at 2017 INFOCOM.

A. Badita and P. Parag are with the Department of Electrical Communication Engineering, Indian Institute of Science, Bengaluru 560012, India (e-mail: parimal@iisc.ac.in).

J.-F. Chamberland is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843-3128 USA.

Communicated by J. Kliewer, Associate Editor for Coding Techniques.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2019.2909868

file is subsequently reconstructed by appending the various pieces. More elaborate coded abstraction relies on the fact that, over large fields, it is straightforward to create packets that are less restrictive for reconstruction. As such, the original file can be recovered from any combination of k blocks via a standard decoding process. We refer to this latter scheme as *MDS coding*.

The greater flexibility afforded by file encoding can improve the content recovery process. However, this distributed structure must be paired with a download strategy that dictates which servers should be contacted by a particular user. This process can be centralized [21], or it can be accomplished using redundant requests with cancellation [14], [22]. One subtlety associated with preemption stems from the option to discard redundant requests when k jobs are initiated or when k jobs are completed. Irrespective of this distinction, a file request exits the system whenever all the necessary chunks have been delivered.

Under Poisson arrival and exponential service assumption, the request queue for content stored over a distributed system can be modeled by a continuous-time Markov process with a countable state space; yet such processes often prove difficult to analyze. Consequently, a candidate approach to understand these systems is to find models that stochastically dominate the evolution of the actual Markov chains and are easier to characterize. For example, Shah *et al.* [20], Lee *et al.* [21] consider a scheme for distributed storage where a file request is fulfilled by contacting k servers. They then propose tractable quasi-birth-death (QBD) processes that can bound the performance of the request queue using matrix-geometric methods [23]–[25]. Following this approach, one can numerically evaluate key attributes of these dominating QBD processes, thereby establishing performance bounds on the original system and enabling the comparison of alternate implementations. Likewise, Joshi *et al.* [13]–[15] propose fork-join queues that bound the performance of a content request queue for redundant requests with cancellation. Implicitly, they are analyzing an MDS coding scheme where any size-k subset of servers can finish the job. This methodology is employed to provide an upper bound on mean completion time.

A simpler alternative to finding queue distributions and mean delay is finding the mean completion time. Mean download time is studied under *i.i.d.* exponential service time assumption for (n, k) MDS code for $k = 2$ in [26], and for availability code under Poisson arrivals in [27]. An $M/G/1$ approximation for mean download time is proposed for specific value of $k = 2$ in [26], whereas an upper bound for mean download time is proposed for high arrival rates along with a characterization for low arrival rates in [27].

Different scheduling and replication policies for general service times are considered in [28]. Dynamic scheduling and replications for file dependent service time is considered in [29], and bounds and approximations are proposed for the response time of specific policies. Cancellation delay for replicated task is considered in [30] and latency optimal dynamic scheduling policies are proposed for two server systems with Poisson arrivals and exponential service.

A. Main Contributions

In this article, we revisit the distributed storage framework where file requests are fulfilled by a sub-collection of servers, each possessing one data block. We introduce a general framework for symmetric codes and we focus on the two representative systems described above, block repetition and MDS coding. We adopt the replication-cancellation case, where redundant requests are discarded upon service completion.

Our main contribution is threefold. First, we introduce a new analytical approach to study content access delay in distributed storage systems. This approach is based on tandem queues with server pooling, and it applies to various coding scenarios including block repetition and MDS coding. Under this viewpoint, the service rate of the tandem queues are coupled and state-dependent. Second, we leverage this abstraction to establish novel upper and lower bounds on the latency performance of a distributed storage system with redundant requests. This is accomplished by defining analytically tractable tandem queues that dominate the performance of the original tandem queue from above and below. This alternate queueing model naturally leads to approximations for the operation of distributed storage systems. The performance of the dominating queues and the approximate systems are characterized completely, leading to closed-form expressions for sojourn time in the corresponding Markov chains. These results are especially pertinent for intricate coding schemes with a large number of servers where existing numerical techniques can become inadequate. Third, we find the stability regions for these tandem queues using Lyapunov techniques.

The remainder of this article is organized as follows. We introduce the coding model in Section II. Building on storage codes, we describe the queueing model for requests in Section III. Taking into consideration the arrival and departure processes, we derive Markov models for the ensuing Markov processes in Sections IV and V. Our theoretical findings, along with a detailed analysis, are presented in Section VI. Numerical results are reported in Section VII. Finally, we conclude with a brief discussion of our findings and we outline possible avenues of future research in the last section.

II. Coding Model

Going along with existing literature, we study the operation of a distributed storage system by focusing on a single file m. We assume that this file can be partitioned into k pieces $m_1, m_2, \ldots, m_k$. We view these entries as elements in a large finite field $\mathbb{F}_q$, writing

$$m = (m_1, m_2, \ldots, m_k) \in \mathbb{F}_q^k.$$

Let $C = [n, k, d]_q$ denote a linear code that maps a k-length message $m \in \mathbb{F}_q^k$ to an n-length codeword $x \in \mathbb{F}_q^n$, and with a minimum Hamming distance d between any two codewords. We can represent this encoded message by

$$x = C(m) = (C_1(m), C_2(m), \ldots, C_n(m)).$$

For simplicity, we take n/k to be an integer. The collection of codewords corresponding to all possible messages forms a

codebook, which we denote by

$$\mathcal{C} = \{C(m) : m \in \mathbb{F}_q^k\}.$$

Once encoded, message m is stored redundantly at n servers by assigning every coded component to a specific location. More precisely, symbol $x_j = C_j(m)$ is stored on server j. At a specific time instant during the data downloading process, a particular user may have acquired blocks corresponding to a subset $T \subseteq [n] \triangleq \{1, \ldots, n\}$ of available servers. We refer to the hosts that have provided the downloaded blocks as the *observed servers*. We can write the projection of codeword $x \in \mathbb{F}_q^n$ onto the subset of observed servers T as

$$x_T = (x_i : i \in T).$$

A k-element subset $S \subseteq [n]$ is called an *information set* [31], if the restriction $m \mapsto (C_j(m) : j \in S)$ forms a bijection, i.e.,

$$\mathcal{C}_S \triangleq \left\{(C_j(m) : j \in S) : m \in \mathbb{F}_q^k\right\} = \mathbb{F}_q^k.$$

In words, a set of observed servers is an information set whenever the intended user is able to recover unambiguously all k pieces of the original message m, with no extraneous blocks of downloaded information. We denote the collection of information sets for a code C as

$$\mathcal{I}(C) \triangleq \left\{S \subseteq [n] : |S| = k, \mathcal{C}_S = \mathbb{F}_q^k\right\}.$$

When the set T of observed servers is not an information set, we can define the notion of *useful servers*,

$$\begin{aligned} M(T) &= \{i \notin T : i \in S \text{ for some } S \in \mathcal{I}(C)\} \\ &= \bigcup_{S \in \mathcal{I}(C)} S \setminus T. \end{aligned}$$

This is the collection of servers that can provide a useful piece of information to the user, given the information acquired thus far. From this definition, it follows that

$$M(T) \subseteq [n] \setminus T \qquad \text{and} \qquad T \subseteq [n] \setminus M(T).$$

As an elementary example, suppose that an incoming request arrives in the queue with no data blocks. Then, all available servers are useful to this particular request at that time and $M(\emptyset) = [n]$.

The analysis framework developed in this article applies to all symmetric linear codes for which the cardinality of every set of useful servers $M(T)$ depends solely on the cardinality of the information subset T. For such codes, we can introduce a shorthand notation for the number of useful servers,

$$N_{|T|} \triangleq |M(T)|. \tag{1}$$

This condition implied by (1) may seem abstruse; however, we note that the two emblematic coding paradigms used in distributed storage, block repetition and MDS coding, possess this property. For convenience, we label instances of these classes of codes as $C^{\text{rep}} = [n, k, d^{\text{rep}}]$ and $C^{\text{mds}} = [n, k, d^{\text{mds}}]$.

For a block repetition code, each distinct piece m_i is stored at n/k servers. We denote the set of servers with message i as

$$\mathcal{S}_i = \{j \in [n] : x_j = m_i\}.$$

By assumption $|\mathcal{S}_i| = n/k$, and $\{\mathcal{S}_i : i \in [k]\}$ partitions the set of servers $[n]$. The minimum Hamming distance d^{rep} between two codewords for replication is n/k. Furthermore, we can decode the message m whenever we observe the data blocks associated with a subset of servers $S \subseteq [n]$ such that $|S| = k$ and $|S \cap \mathcal{S}_i| = 1$ for every $i \in [k]$. We can therefore write the collection of information sets for block repetition coding as

$$\mathcal{I}(C^{\text{rep}}) = \{S \subseteq [n] : |S| = k, |S \cap \mathcal{S}_i| = 1 \text{ for } i \in [k]\}.$$

We observe that there are $(n/k)^k$ elements in the collection $\mathcal{I}(C^{\text{rep}})$. For this coding scheme, once a request receives a message piece m_i, no other server in $\mathcal{S}_i$ can serve it anymore. Hence, the set of useful servers for any request with partial codeword x_T can be written as

$$M(T) = \bigcup_{i \notin T} \mathcal{S}_i \qquad \text{and} \qquad T \subseteq [n] \setminus M(T).$$

Since $\{\mathcal{S}_i : i \in [k]\}$ partitions the set of all servers $[n]$ and $|\mathcal{S}_i| = n/k$ for each $i \in [k]$, the number of useful servers is $N_{|T|} = (\max\{k - |T|, 0\})\, n/k$ for any information subset T.

MDS codes have optimal minimum Hamming distance, $d^{\text{mds}} = n - k + 1$. Under this scheme, each server $j \in [n]$ stores a symbol $x_j = C_j(m)$ such that any k distinct symbols suffice for the reconstruction of message m. Thus, in this case, we have the collection of information sets

$$\mathcal{I}(C^{\text{mds}}) = \{S \subseteq [n] : |S| = k\}.$$

Consequently, there are $\binom{n}{k}$ distinct elements in $\mathcal{I}(C^{\text{mds}})$. Since $\binom{n}{k} > (\frac{n}{k})^k$, MDS codes offer a greater number of decodable subsets compared to block repetition codes and hence they should perform better. Notice that, under this scheme, every remaining server can offer a useful block. For instance, once served by a server j, a request can be served by any other servers in $[n] \setminus \{j\}$. The set of useful servers for a request with partial codeword x_T is given by

$$M(T) = [n] \setminus T \qquad \text{and} \qquad T = [n] \setminus M(T)$$

for any $T \subseteq S \in \mathcal{I}(C^{\text{mds}})$. Also, the number of useful servers admits the simple form $N_{|T|} = n - |T|$ for any information subset T.

For illustrative purposes, we examine the operation of fragmentation and replication against that of MDS coding. We focus on the scenario where $k = 2$ and $n = 4$; this rudimentary setting offers an easy comparison between our upcoming numerical results and those published in [14] and [21]. For the $(4, 2)$ case, message m consists of two parts A and B.

In block repetition coding, the components A and B are each stored at two locations. A request is then completed when the user is successfully served by one cache containing block A and another cache holding block B. We can write the generator and parity-check matrices for the $(4, 2)$ replication code as

$$G^{\text{rep}} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad H^{\text{rep}} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

Contrastingly, in MDS coding, a file is partitioned into two pieces and independent linear combinations of these blocks are generated. This strategy necessitates that packets be long

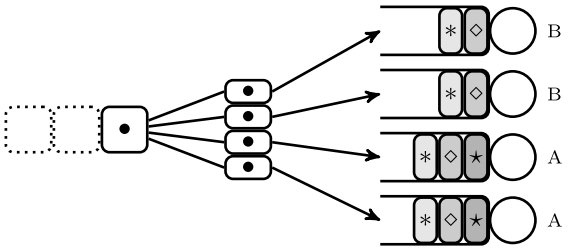

Fig. 1. This diagram depicts a distributed fork-join network with four caches. Two servers are storing symbol A, and the other two servers host symbol B. Under this divide and replicate paradigm, a request must obtain piece A and piece B to reconstruct the original media object.

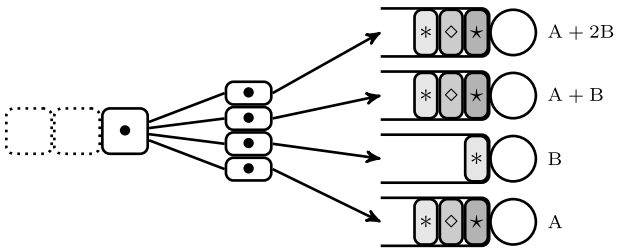

Fig. 2. In a more elaborate fork-join coded system, the various caches store independent messages. The original media object can therefore be recovered by decoding the content of any two distinct data blocks. Once this is achieved, the corresponding requests are dropped from the remaining queues.

enough to support encoding in a high-order field, a requirement easily met in practice. Every encoded message is then stored on a single server. For instance, when four caches are present, candidate messages could be A, B, A+B, A+2B. The original media object can then be recovered by successfully contacting any two servers, a slightly more flexible stipulation than before. The generator and parity-check matrices for a $(4, 2)$ MDS code appear below,

$$G^{\text{mds}} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \quad H^{\text{mds}} = \begin{bmatrix} -1 & -1 & 1 & 0 \\ -1 & -2 & 0 & 1 \end{bmatrix}.$$

The operations of these two alternate implementations for systems with four servers are depicted in Fig. 1 and Fig. 2, respectively.

III. Queueing Model

A single file is fragmented into k parts, and encoded to n symbols using a symmetric coding described in Section III-A. Each of the n encoded symbol is stored uniquely over one of the n caches. We assume Poisson arrivals of the requests for the file download. Requests for file m arrive at a central location, and they are placed in a queue of infinite buffer size. These requests are subsequently served by available n caches, each hosting a block of information. We assume that every user demands the whole message m, which can be recovered by downloading data components from all caches corresponding to an information set. That is, we assume an (n, k) fork-join scheduling where k is the size of the information set. We assume instantaneous decoding of the file from the information set of coded symbols, and an instantaneous cancellation of request at other servers, once file m can be decoded by the request. Service distribution at each cache is assumed to be independent, identical, and memoryless. These assumptions simplify the analysis and provide optimistic coding gains under our idealized conditions.

A. Symmetric Coded Storage

We assume there is a single file m in the system, divided into k fragments $(m_1, \ldots, m_k)$ and encoded into n symbols $(C_1(m), \ldots, C_n(m))$. Each cache $j \in [n]$ stores the distinct symbol $C_j(m)$. We assume a *symmetric coding scheme* such that after downloads from a subset $T \subseteq [m]$ servers, the number of useful servers $|M(T)|$ depends only on the cardinality of the set of observed servers T. A large majority of the existing literature studies the performance of MDS [15], [19], [21], [26] or replication codes [22], [26], [28]–[30], due to the favorable structural properties amenable to analysis. It turns out the cardinality of useful servers is a fundamental property that governs the system performance, and we can consequently generalize our study to symmetric codes. Performance analysis of general storage codes remains an open question. Many symmetric codes involve a non-trivial decoding at the receiver from the encoded pieces in an information set. In this work, we make the idealized assumption of negligible decoding latency. This approximation works well for the case when communication is expensive and processing is cheap. That is, when the decoding latency is much smaller than the access latency caused by limited communication resources.

B. Poisson Arrivals

We assume that the times between consecutive arrivals form a sequence of independent and exponentially distributed random variables, each with mean $1/\lambda$. That is, arrivals are assumed to be an instance of a Poisson process. This assumption is motivated by simplicity of analysis, and is a popular assumption in the performance analysis literature [26], [30].

C. Memoryless Service

The service time at a particular cache is an exponential random variable with rate μ, and it is independent of other requests and caches. The waiting times between consecutive service opportunities at a specific location form a Poisson process. These processes are independent from one cache to another. This assumption renders analysis tractable and it has been adopted by several previous works in this area [20]–[22], [26], [27], [30]. It has been shown in the literature [18], [32], [33] that shifted exponential distribution can well model the empirical service time distribution in data centers, and heavy tailed distributions can model the service time in large server farms [18], [34]–[36]. Nevertheless, we adopt exponential service distribution for mathematical convenience and tractability, and derive insights from this study to provide system design guidelines for general service distributions.

As a consequence of our modeling assumptions, when a user is served in parallel by multiple caches, the waiting time until it acquires its next block becomes the minimum among the exponential service times of all its assigned caches.

Since the service times across the storage system are exponentially distributed with rate μ, the waiting time for the parallel processing of a same user at ℓ locations is exponential with rate $\ell\mu$.

D. Service Constraints

Every server can supply one data block, which can ultimately be combined with other pieces to recover the original file m. For any $[n, k, d]$ linear coding strategy, a request that has already acquired data blocks $T \subseteq [n]$, can only be served by the set of useful servers $M(T)$. For instance, in block repetition coding, when a user already possesses blocks m_1 and m_2, then it can only be served by caches that offer different blocks, i.e.,

$$M(\{1,2\}) = \{i \in [n] : x_i \notin \{m_1, m_2\}\}.$$

On the other hand, a user can still be served by caches labeled $\{3, 4, \ldots, n\}$ after acquiring information blocks x_1 and x_2 under MDS coding. That is,

$$M(\{1,2\}) = [n] \setminus \{1,2\}.$$

E. Scheduling and Queue Management

As mentioned before, we investigate symmetric coding policies, such as block repetition and MDS coding. We refer to the task of matching requests to servers and managing queues as scheduling. Throughout, we consider an (n, k) *fork-join scheduling* policy [14]. This policy is attractive both from the analytical and practical perspective. On the practical side, it is a simple policy where the requests experience the shortest waiting and service time [28], and it is work-conserving in the sense that a useful server is never idle. This policy has desirable analytical properties such as stationarity that induces the Markov property on the queueing system.

In the fork step of this policy, every server has an associated individual queue, and each incoming request joins all n queues through replication. In the join step, a request leaves the system when it successfully gathers blocks associated with an information set and can therefore reconstruct the original file m. Furthermore, pending block queries are instantly dropped from useless queues when their parent request gathers new information blocks. Specifically, when a request acquires message subset T, all the servers in set $[n] \setminus M(T)$ abandon block queries associated with the underlying request. Under MDS coding, a request leaves the system when it gets served by any k servers; whereas a request leaves the system when it gets served by k caches with distinct blocks for block repetition coding. In all cases, block queries are served on a first-come-first-served (FCFS) basis at every location.

Coordination for this infrastructure is established through a central agent. When a server finishes processing a block query, it removes this block from its queue and notifies the central coordinating agent. It then starts processing the next block query in its queue. As notifications arrive at the central agent, servers entering the set of useless servers for this request are instructed to discard the associated block queries. When a request gathers k pieces from an information set, all its remaining block queries are dropped and the request exits the central location. This model is equivalent to the one studied in [14]–[16]. An alternative scheduling policy would limit the processing of concurrent block queries to a maximum of k locations with linearly independent content. This alternate scheduling model forms a basis for the findings reported in [20] and [21].

Cancellation delays in dropping the remaining $(n - k)$ queries for the request, when the request gathers k pieces from an information set, have been studied in [28] and [30]. It is shown to adversely impact the latency and throughput, since the amount of work done by workers processing the cancellation request is wasted. In this work, we have assumed zero cancellation overhead as a first order approximation to explore optimistic gains afforded by the coding.

F. Sample Path of Queue Evolution

As an example, consider a sample path for the evolution of queued requests under $(4, 2)$ MDS coding with message $m = (\mathrm{A}, \mathrm{B})$. We assume that the system is initially empty. When the first request arrives, it joins the queues at all four locations. Then, two additional requests arrive before the first service opportunity materializes. At this stage, the four queues each have three requests, which they process using an FCFS policy. Suppose that the first block successfully downloaded by request 1 is A, and that it is provided by server 1. At this stage, request 1 exits the queue of server 1, yet it continues to be served on all remaining servers. Server 1 starts serving the second request. Next, request 1 obtains A+B from server 3. Then, request 1 is immediately dropped from server 2 and server 4, as advised by the central agent. Having acquired an information set, user 1 can decode the original file m and it exits the system. This process continues in a similar manner, with new requests arriving in the system and data block being downloaded from the four caches.

IV. Continuous Time Markov Chain

In this section, we characterize Markov processes related to this queueing system. Recall that requests leave the system as soon as their gathered symbols form an information set $S \in \mathcal{I}(C)$. We denote all possible subsets of size less than k by

$$\mathcal{P}_k(n) = \{S \subseteq [n] : |S| < k\}.$$

For any work-conserving scheduling policy the set of messages gathered by any partially served request at any time t is an element of $\mathcal{P}_k(n)$. We denote the subset of blocks acquired by request i at time t by $S_i(t)$. The number of requests in the system at time t is denoted by $r(t)$. When the system is empty, its state is denoted by e. Otherwise, the state of the system at any time t is

$$\mathcal{S}(t) = (S_i(t) \in \mathcal{P}_k(n) : i \in [r(t)]). \tag{2}$$

Proposition 1. *For Poisson arrivals, exponential service times, and a Markov scheduling policy, the stochastic process* $\{\mathcal{S}(t) : t \geq 0\}$ *possesses the Markov property.*

Proof: Given Poisson arrivals and the exponential service model, there is at most a single event in an infinitesimal time interval. Suppose that there are $r > 0$ requests in the system at time t. Then, we can denote the state of the system by $\mathcal{S}(t) = (S_1, \ldots, S_r)$. An arrival leads to an increase in the number of requests in the system, driving the state to $(S_1, \ldots, S_r, \emptyset)$; this happens at rate λ. A service completion leads to a unit increase in the message sets of one of the requests; this occurs with rate

$$\left| \bigcup_{i=1}^{r} M(S_i) \right| \mu.$$

Upon request i receiving data block j, its message sets S_i changes to $S_i \cup \{j\}$ where $j \in M(S_i)$. The probability of this event depends on the scheduling policy. Since the server selection for requests is a function of the current state, this probability is independent of time and depends only on the current state. Hence, the system features the Markov property, as desired. ∎

Theorem 2. *For a distributed storage system with a symmetric code and fork-join queues with FCFS service, the set of useful servers $\{M(S_i(t)) : i \in [r(t)]\}$ is totally ordered by set inclusion. In addition, the size of message subsets $\{|S_i(t)| : i \in r(t)\}$ is non-decreasing in i.*

Proof: See Appendix. ∎

A. Transition Rates

Leveraging the structure of the system outlined in Theorem 2, we can succinctly describe the transition rates associated with this continuous-time Markov chain. State transitions due to arrivals occur at rate

$$Q\left(\mathcal{S}, (\mathcal{S}, \emptyset)\right) = \lambda.$$

Next, we consider request completions that do not lead to departures. Specifically, a service completion from server j to request i, where $j \in M(S_i) \setminus M(S_{i-1})$ and $|S_i| < k-1$, takes place at rate

$$Q(\mathcal{S}, (S_1, \ldots, S_{i-1}, S_i \cup \{j\}, S_{i+1} \ldots, S_r)) = \mu.$$

We note that the number of servers assigned to request i is $|M(S_i)| - |M(S_{i-1})|$. Thus, the aggregate rate at which this request is receiving service can be expressed as $(|M(S_i)| - |M(S_{i-1})|)\,\mu$. States where $|S_1| = k-1$ form a special case; in this situation, the head request acquires its last data block and departs from the queue at rate

$$Q((S_1, \ldots, S_r), (S_2, \ldots, S_r)) = |M(S_1)|\mu.$$

Collectively, these rates completely characterize the generator matrix for Markov chain $\mathcal{S}(t)$. As always, the diagonal elements of the generator matrix are defined such that the rows of the matrix sum to zero. The symmetry in the coding schemes suggests that this chain may be amenable to a reduction in state space for performance analysis. We explore this possibility below.

B. Reduction of State Space

For symmetric codes, we gather that the number of useful servers and, hence, the rates at which a particular request is being served within the Markov chain depend solely on the cardinalities of the message subsets.

Theorem 3. *Let $\mathcal{S}(t)$ be the state of the queueing system at time t, as defined in equation* (2)*. The number of fragments downloaded by the ith request at time t is denoted by $L_i(t) = |S_i(t)|$ for each $i \in r(t)$. Under the assumptions in Theorem 2, the process $L(t) = (L_i(t) : i \in [r(t)])$ is a continuous time Markov chain.*

Proof: Since all the servers are independent and they each have exponential service times with rate μ, the ensuing transition rates can be viewed as functions of $L(t)$. In other words, keeping track of the number of data blocks accumulated by each request, rather than recording the specific labels of the downloaded blocks for every request, is enough to trace the lengths of all the queues. Moreover, $L(t)$ inherits the Markov property from $\mathcal{S}(t)$ because the original process is Markov and transition rates in $L(t)$ are completely determined by its own state. ∎

Let $\ell = (\ell_1, \ldots, \ell_r)$ denote a possible state for $L(t)$. By Theorem 2, we already know that $\ell_1 \geq \ell_2 \geq \cdots \geq \ell_r$. As before, there are three types of transitions. First, a new packet may arrive; this event takes place at rate $Q(\ell, (\ell, 0)) = \lambda$. Second, a request can get an additional data block without leaving the system. This is only possible when $\ell_i < k-1$, and it occurs at rate

$$Q(\ell, (\ell_1, \ldots, \ell_i + 1, \ldots, \ell_r)) = \left(N_{\ell_i} - N_{\ell_{i-1}}\right)\mu,$$

where N_{ℓ_i} is the notation for the number of useful servers to request i introduced in (1), and $N_{\ell_0} = 0$. The last possibility corresponds to the head request obtaining its last block and exiting the system. This event only happens when $\ell_1 = k-1$ and, when admissible, it comes about with rate $Q(\ell, (\ell_2, \ldots, \ell_r)) = N_{\ell_1}\mu$. Together, these rates determine the generator matrix for reduced Markov chain $L(t)$.

V. Transformation of State Space

The reduced state space for process $L(t)$ defined in Theorem 3 yields admissible states that form sequences of non-increasing integers, with

$$k-1 \geqslant \ell_1 \geqslant \ell_2 \geqslant \cdots \geqslant \ell_r \geqslant 0.$$

Still, the number of elements in these sequences can be arbitrarily large. The structure identified above invites one further simplification with a convenient fixed-length representation for the states. The underlying idea is to employ a bijection between ℓ and a fixed-length vector by counting the occurrences of every integer between zero and $k-1$ in ℓ. This is achieved by defining a vector $y \in \mathbb{N}_0^k$ where individual component y_i is equal to the number of requests that have gathered exactly i data blocks. Moreover, the total number of requests in the system is $r = y_0 + \cdots + y_{k-1}$. Mathematically, we get

$$y_i = \left|\{i' \in [r] : \ell_{i'} = i\}\right|.$$

This bijective transformation of the original process to a k-length vector immediately produces a new Markov chain; this is formalized in Corollary 4.

Corollary 4. *For a distributed storage system with a symmetric code and fork-join queues with FCFS service, the random process*

$$Y(t) = (Y_0(t), Y_1(t), \ldots, Y_{k-1}(t)),$$

where $Y_i(t)$ denotes the number of requests that have gathered exactly i data blocks at time t, forms a continuous-time Markov chain.

Proof: From Proposition 1 and Theorem 3, we know that process $\mathcal{S}(t)$ and $L(t)$ are continuous-time Markov chains. By construction, $Y_i(t)$ denotes the number of requests that have accumulated i data blocks at time t. We can write the component of $Y(t)$ as

$$\begin{aligned} Y_i(t) &= |\{i' \in [r(t)] : |S_{i'}(t)| = i\}| \\ &= |\{i' \in [r(t)] : L_{i'}(t) = i\}|. \end{aligned}$$

Since there exists a bijective transformation between $L(t)$ and $Y(t)$, we must conclude that $Y(t)$ is also a continuous-time Markov chain. ∎

While straightforward, Corollary 4 is important because it reveals the form of the Markov chain we employ to assess the performance of competing implementations. Once established, this concise representation is easy to track and it captures the essential behavior of the queueing system. An interesting attribute of the reduced Markov chain is that its state space becomes agnostic to the identities of the information blocks, as long as the coding scheme possesses the symmetric property.

The transition rates of $Y(t)$ are inherited from $L(t)$. As an intermediate step in identifying these rates, we introduce a shorthand notation for possible next states and admissible transitions. To express transition rates in a compact form, it is important to keep track of which components in y are greater than zero; we define an appropriate index set as follows,

$$I(y) = \{i \in \{0, 1, \ldots, k-1\} : y_i > 0\}.$$

Also, let $e_i \in \mathbb{N}_0^k$ represent the standard k-dimensional unit vector along coordinate i. Suppose that the request queue is in state $y = (y_0, \ldots, y_{k-1}) \in \mathbb{N}_0^k$. The arrival of a new request in the queue leads to a transition to state $y + e_0$. The delivery of a data block to a request with $i - 1 \in I(y)$ pieces of information produces a transition to $y + e_i - e_{i-1}$. Likewise, a request receiving its last piece of information is only possible when $y_{k-1} > 0$, and it triggers a jump to state $y - e_{k-1}$. In summary, under arrivals and service completion events, we can track the evolution of the system in functional form with

$$a_i(y) \triangleq \begin{cases} y + e_0, & i = 0, \\ y - e_{i-1} + e_i, & i \in [k-1], \\ y - e_{k-1}, & i = k. \end{cases}$$

Again, we emphasize that a transition of type $a_i(\cdot)$ can only take place when the number of requests with $i - 1$ pieces is greater than zero, i.e., $i - 1 \in I(y)$. Otherwise, the underlying service opportunity trickles down to the next available request with fewer information blocks.

To capture the cascading effect in expressing transition rates, we need to keep track of server assignment. This can be accomplished by registering the next non-empty level up in y. We denote the smallest index above i for which the number of partially fulfilled request is non zero by

$$u_i(y) \triangleq k \wedge \min\{u > i : y_u > 0\}, \tag{3}$$

where $\wedge$ is the minimum of the two arguments.

Using the notation developed so far, we can systematically describe the non-zero off-diagonal entries of the generator matrix associated with continuous-time Markov process $Y(t)$. As before, new requests enter the queue at rate $Q(y, a_0(y)) = \lambda$. The number of servers assigned to level i and, hence, the rate at which its head request is being served can be described iteratively. We denote this rate $Q(y, a_i(y))$ for $i \in [k]$ by $\mu_{i-1}(y)$. When y_{k-1} is not zero, the number of servers dedicated to delivering a last piece to the head request is equal to N_{k-1}. Hence,

$$\mu_{k-1}(y) = Q(y, a_k(y)) = N_{k-1}\mu \mathbf{1}_{\{y_{k-1}>0\}}.$$

Whenever a coordinate becomes empty, the corresponding servers start focusing on requests with fewer pieces of information. This leads to a trickle down effect. Mathematically, we get

$$\begin{aligned} \mu_{i-1}(y) = Q(y, a_i(y)) &= \left(N_{i-1} - N_{u_{i-1}}\right) \mu \mathbf{1}_{\{y_{i-1}>0\}} \\ &= \mathbf{1}_{\{y_{i-1}>0\}} \sum_{j=i-1}^{u_{i-1}-1} (N_j - N_{j+1})\mu, \end{aligned} \tag{4}$$

where u_i is the next level up defined in (3).

When all n servers are busy, the aggregate rate of service is $n\mu$. However, under the fork-join FCFS scheduling policy, some servers may be idling. In particular, when a server is not useful to any active request, then it cannot provide meaningful content and the corresponding work opportunity is lost. Conceptually, the cascading effect reduces the probability of having idling servers by dynamically reassigning processing power to requests with fewer pieces.

An empowering means to view this system is to think of the number of requests y as customers who desire k different types of service in sequence. Under this abstraction, y_i denotes the number of customers waiting for service i. Once, a customer obtains service i, it proceeds to the queue for service $i + 1$. Under this alternate viewpoint, this multi-dimensional Markov chain forms a sequence of k queues in tandem. In this setting, the function $a_i(y)$ indicates the arrival of a customer in queue i due to a departure from queue $i - 1$.

That is, $Y(t) = (Y_0(t), \ldots, Y_{k-1}(t))$ is a sequence of k queues in tandem driven by Poisson arrivals with rate λ. The base service rate at queue i is given by $\gamma_i = (N_i - N_{i+1})\mu$. Yet, the servers are coupled in the following sense. When one of the queues is empty, its associated server pools its resources with the first non-empty queue preceding it. The operation of such a sequence of queues is depicted in Fig. 3 for two queues.

Since the exact identities of the messages obtained by the requests are irrelevant for performance evaluation, the reduced

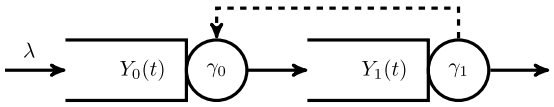

Fig. 3. This block diagram displays two queues in tandem with Poisson arrival rate λ and service rates γ_0 and γ_1, respectively. The length of queue i is denoted by $Y_i(t)$ for $i \in \{0, 1\}$. The dashed line from server 1 to server 0 symbolizes the fact that server 1 pools its resources with server 0 when queue 1 is empty.

Markov chain $Y(t)$ is well-suited for analysis. We observe that the state space of the continuous-time Markov process corresponding to any symmetric code remains unchanged. However, their rate matrices are different. This notion is reinforced below by studying our two archetypal coding schemes.

A. Transition Rates for Repetition and MDS Codes

For requests with $i \in \{0, \ldots, k-1\}$ information symbols, the respective number of useful servers for block repetition and MDS codes are equal to

$$N_i^{\text{rep}} = (k-i)\frac{n}{k}, \qquad N_i^{\text{mds}} = (n-i).$$

As discussed above, the base service rate at queue $i-1$ is given by $\gamma_{i-1} = (N_{i-1} - N_i)\,\mu$. As such, the base rates for block repetition coding are given by

$$\gamma_{i-1}^{\text{rep}} = \frac{n}{k}\mu, \quad i \in [k], \tag{5}$$

whereas the base rates for MDS coding can be expressed as

$$\gamma_{i-1}^{\text{mds}} = \begin{cases} \mu, & i \in [k-1], \\ (n-k+1)\mu, & i = k. \end{cases} \tag{6}$$

Clearly, the base rates differ significantly and they may result in distinct queueing behaviors. The effective transition rates change dynamically, as they depend on y through components equal to zero. These latter rates are obtained by substituting the expressions above into (4).

VI. Performance Analysis

In general, a network of coupled queues may be very difficult to analyze. Nevertheless, we can take advantage of the fact that $Y(t)$ represents a sequence of k queues in tandem. We leverage this property to derive several key results. First, we find the maximum arrival rate λ for which the Markov process $Y(t)$ remains positive recurrent in Section VI-A. In particular, we show that block repetition and MDS coding with fork-join FCFS scheduling share a same stability region. Second, we define uncoupled tandem queues that bound the performance of the coupled queue system represented by $Y(t)$ in Section VI-B. Third, we propose a decoupled tandem queue that approximates the behavior of Markov process $Y(t)$ for symmetric codes in Section VI-C. We use this decoupling strategy to compute approximations for mean delays and average queue lengths for both block repetition and MDS coding. We also explore scaling behavior for these two coding schemes as the number of servers n becomes large, while the code rate k/n is kept constant. Finally in Section VI-D, we show that among the class of symmetric codes, the approximate mean delay minimizing code is MDS.

A. Stability Regions

We refer to the set of arrival rates for which the process $Y(t)$ is positive recurrent as the *stability region*. We use the Foster-Lyapunov drift conditions to identify the stability region for a distributed coded system under FCFS fork-join scheduling employing arbitrary $[n, k, d]$ symmetric code. Let $\phi : \mathbb{N}_0^k \to \mathbb{N}_0^k$ be a bijection such that

$$\phi(y) = (y_0, y_0 + y_1, \ldots, y_0 + \ldots + y_{k-1}).$$

We define the following quadratic function $V : \mathbb{N}_0^k \mapsto \mathbb{R}_+$ as a potential or Lyapunov function for process $Y(t)$ in terms of the bijection ϕ,

$$V(y) = \frac{1}{2}\,\|\phi(y)\|_2^2 = \frac{1}{2}\sum_{j=0}^{k-1}\phi_j(y)^2, \qquad y \in \mathbb{N}_0^k.$$

When $Y(t) = y$, admissible transitions are to states $a_i(y)$ where $i-1 \in I(y)$ or $i = 0$. For these admissible transitions, we express their impact on function $\phi(\cdot)$ as follows,

$$\phi(a_i(y)) = \begin{cases} \phi(y) + \sum_{i=0}^{k-1} e_i, & i = 0, \\ \phi(y) - e_{i-1}, & i \in [k]. \end{cases}$$

The corresponding rates of transition from state y to state $a_i(y)$ are given by λ for $i = 0$ and $\mu_{i-1}(y)$ for $i - 1 \in I(y)$. For Markov process $Y(t)$ with generator matrix Q, the mean rate of change of function $V(Y(t))$, termed the mean drift rate is defined as $Q_V(y) = \sum_{y'} Q(y, y')[V(y') - V(y)]$. For the given Markov process $Y(t)$, the mean drift rate can be computed as

$$Q_V(y) = \sum_{i=0}^{k} Q(y, a_i(y))(V(a_i(y)) - V(y)).$$

For each level i, the individual potential difference between a next state and the current state is given by

$$V(a_i(y)) - V(y) = \begin{cases} \sum_{j=0}^{k-1}\left(\phi_i(y) + \frac{1}{2}\right), & i = 0, \\ -\left(\phi_{i-1}(y) - \frac{1}{2}\right), & i \in [k]. \end{cases}$$

Theorem 5. *For a distributed storage system with a symmetric code and fork-join queues with FCFS service, the stability region is equal to*

$$\lambda < \min\left\{\frac{\Gamma_i}{k-i} : i \in \{0, \ldots, k-1\}\right\},$$

where $\Gamma_i \triangleq \sum_{j=i}^{k-1}\gamma_j$ *is the useful service rate for level* i.

Proof: The mean drift rate of this Lyapunov function $V(\cdot)$ for continuous-time Markov process $Y(t)$ can be computed in terms of the arrival rate λ and the service

rates $(\mu_{i-1}(y) : i \in [k])$ as

$$\begin{aligned} Q_V(y) &= \lambda \sum_{i=0}^{k-1} \left(\phi_i(y) + \frac{1}{2}\right) - \sum_{i=0}^{k-1} \left(\phi_i(y) - \frac{1}{2}\right) \mu_i(y) \\ &= \frac{k\lambda}{2} + \frac{1}{2} \sum_{i=0}^{k-1} \mu_i(y) + \sum_{i=0}^{k-1} \phi_i(y)(\lambda - \mu_i(y)) \\ &= \frac{k\lambda}{2} + \frac{1}{2} \sum_{i=0}^{k-1} \mu_i(y) + R(y) \\ &\leq \frac{1}{2}(k\lambda + \Gamma_0) + R(y). \end{aligned}$$

The first term on the right-hand-side of the inequality is a constant. Furthermore, for each $i \in I(y)$, we have

$$\sum_{j=i}^{k-1} \mu_j(y) = \sum_{j=i}^{k-1} \gamma_j = \Gamma_i. \tag{7}$$

Recall that $\phi_i(y) = \sum_{j=0}^{i} y_j$ and, hence, we can get a new expression for $R(y)$ by interchanging the order of these summations,

$$\begin{aligned} R(y) &= \sum_{i=0}^{k-1} \phi_i(y)(\lambda - \mu_i(y)) = \sum_{i=0}^{k-1} \sum_{j=0}^{i} y_j(\lambda - \mu_i(y)) \\ &= \sum_{j=0}^{k-1} y_j \sum_{i=j}^{k-1} (\lambda - \mu_i(y)) = \sum_{j=0}^{k-1} y_j \left((k-j)\lambda - \Gamma_j\right). \end{aligned}$$

We emphasize that (7) only holds for $j \in I(y)$. Yet, when $j \notin I(y)$, $y_j = 0$ and the corresponding summand above becomes zero. This validates the use of Γ_j in every summand on the right-hand-side of the last equality.

If $\lambda < \Gamma_j/(k-j)$, then $R(y)$ is negative for any non-zero state $y \in \mathbb{N}_0^k$, with increasing magnitude as $\|y\|$ gets larger. As such, the mean drift rate is strictly negative and bounded away from zero for all but finitely many states $y \in \mathbb{N}_0^k$. Moreover, for these latter states, the mean drift rate $Q_V(y)$ is finite. Conversely, at the jth stage of the tandem queue, the average arrival rate is λ and requests need to be processed by the $k-j$ remaining queues. The total pooled service rate available is Γ_j. Therefore, any arrival rate λ that exceeds $\Gamma_j/(k-j)$ will make the queue unstable. Altogether, these results completely determine the stability region. ∎

We can apply this theorem to find stability regions for block repetition and MDS coding in a straightforward manner. This is accomplished in the two corollaries below.

Corollary 6. *For a distributed coded storage system operating under block repetition coding and fork-join queues with FCFS homogeneous service at rate μ, the stability region is*

$$\lambda \in \left[0, \frac{n\mu}{k}\right). \tag{8}$$

Proof: The base rates for block repetition coding already appeared in (5) with $\gamma_{i-1}^{\text{rep}} = \mu n/k$ for $i \in [k]$. This yields $\Gamma_i^{\text{rep}} = (k-i)\mu n/k$ and, as a consequence,

$$\frac{\Gamma_i^{\text{rep}}}{k-i} = \frac{n}{k}\mu.$$

The region of (8) immediately follows by Theorem 5. ∎

Corollary 7. *For a distributed coded storage system operating under MDS coding and fork-join queues with FCFS homogeneous service at rate μ, the stability region is*

$$\lambda \in \left[0, \frac{n\mu}{k}\right). \tag{9}$$

Proof: In this case, the base rates are given in (6). We can then calculate Γ_i^{mds} as follows,

$$\Gamma_i^{\text{mds}} = \sum_{j=i}^{k-1} \gamma_j^{\text{mds}} = (n-k+1)\mu + \sum_{j=i}^{k-2} \mu = (n-i)\mu.$$

For $i \in \{0, \ldots, k-1\}$, this leads to tight inequality

$$\frac{\Gamma_i^{\text{mds}}}{k-i} = \frac{n-i}{k-i}\mu \geq \frac{n}{k}\mu.$$

This establishes the stability region of (9) by Theorem 5. ∎

B. Bounding Techniques

In this section, we introduce two stochastic processes to bound the performance of distributed coded systems, in the sample-path sense. Specifically, we create uncoupled tandem queues that dominate the evolution of the tandem queue with dynamic service pooling. It is important to note that a series of queues in tandem with Poisson external arrivals and independent exponential service rates is a well-studied object. Although the departure from one queue becomes the arrival to the next one, the queue distributions are statistically independent [37]. In the following lemma, we provide a simple uniform bound for the service rate at each queue, independent of the state of the requests at other queues.

Corollary 8. *The transition rate $Q(y, a_i(y))$ is bounded below and above by*

$$\gamma_{i-1} \leq \sum_{j=i-1}^{u_{i-1}(y)-1} \gamma_j \leq \sum_{j=i-1}^{k-1} \gamma_j = \Gamma_{i-1}$$

for every $i-1 \in I(y)$. Equality always holds for $i = k$.

Proof: This is immediate from the definition of $u_{i-1}(y)$ because $i \leq u_{i-1} \leq k$. ∎

Since the service rate Γ_i uniformly upper bounds the dynamic service rate received by queue i in our original coupled tandem queue, we get the following lower bound on performance.

Lemma 9 (Lower Bound)**.** *Consider a continuous-time Markov chain $\underline{X}(t) \in \mathbb{N}_0^k$ with non-zero transition rates defined by*

$$Q(y, a_i(y)) = \begin{cases} \lambda, & i = 0 \\ \Gamma_{i-1}\mathbf{1}_{\{y_{i-1}>0\}}, & i \in [k]. \end{cases}$$

For all arrival rates λ such that the request queue Markov chain $Y(t) \in \mathbb{N}_0^k$ is positive recurrent, the tandem queue $\underline{X}(t)$ is sample path-wise less congested than $Y(t)$. Furthermore, the equilibrium distribution of $\underline{X}(t)$ is given by

$$\underline{\pi}(y) = \prod_{i=0}^{k-1} \left(1 - \frac{\lambda}{\Gamma_{k-1}}\right) \left(\frac{\lambda}{\Gamma_{k-1}}\right)^{y_i}$$

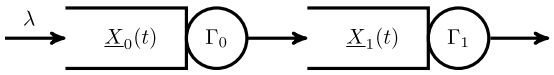

Fig. 4. This block diagram displays two queues in tandem with external arrival rate λ and service rates Γ_0 and Γ_1, respectively. The number of requests in $\underline{X}(t)$ is path-wise smaller than the number of requests in $Y(t)$ at every time t.

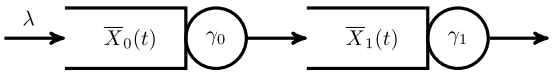

Fig. 5. This block diagram illustrates two queues in tandem with external arrival rate λ and service rates γ_0 and γ_1, respectively. The number of requests in $\overline{X}(t)$ is path-wise larger than the number of requests in $Y(t)$ at every time t.

and the mean sojourn time is equal to

$$\underline{W} = \sum_{i=0}^{k-1} \frac{1}{\Gamma_i - \lambda}.$$

Proof: Stochastic dominance originates from the fact that Markov process $\underline{X}(t)$ is derived from the process $Y(t)$ by adding extra service capacity for requests with i pieces. To find the equilibrium distribution of the process $\underline{X}(t)$, we recognize that Markov process $\underline{X}(t)$ is a sequence of tandem queues. Then, queue i has Poisson arrivals at rate λ and independent random service times distributed exponentially with mean $1/\Gamma_i$ [37]. Therefore, the joint distribution of $\underline{X}(t)$ is the product of the marginal distribution of each queue $\underline{X}_i(t)$. ■

This tandem queue is depicted in Fig. 4.

Similarly, we can find an upper bound on the queue sizes in the actual system by looking at a series of uncoupled queues with exponential service rates γ_i at queue i.

Lemma 10 (Upper Bound)**.** *Consider a continuous-time Markov chain $\overline{X}(t) \in \mathbb{N}_0^k$ with non-zero transition rates defined by*

$$Q(y, a_i(y)) = \begin{cases} \lambda, & i = 0 \\ \gamma_{i-1}\mathbf{1}_{\{y_{i-1}>0\}}, & i \in [k]. \end{cases}$$

For all arrival rates $\lambda < \min_{i\in[k]} \gamma_{i-1}$, the request queue Markov chain $Y(t) \in \mathbb{N}_0^k$ is positive recurrent and the tandem queue $\overline{X}(t)$ is sample path-wise more congested than $Y(t)$. Furthermore, the equilibrium distribution of $\overline{X}(t)$ is given by

$$\overline{\pi}(y) = \prod_{i=0}^{k-1} \left(1 - \frac{\lambda}{\gamma_i}\right)\left(\frac{\lambda}{\gamma_i}\right)^{y_i}$$

and the mean sojourn time is equal to

$$\overline{W} = \sum_{i=0}^{k-1} \frac{1}{\gamma_i - \lambda}.$$

Proof: Again, stochastic dominance emerges from the fact that the Markov process $\overline{X}(t)$ is obtained from $Y(t)$ by preventing resource pooling and letting servers idle when their respective queues are empty. In contrast, in the original system, available servers can transfer their content to requests at lower levels with fewer data blocks. The equilibrium distribution associated with process $\overline{X}(t)$ is dictated by its structure as a series of tandem queues [37]. Queue i experiences Poisson arrivals at rate λ, and it features random service times distributed exponentially with mean $1/\gamma_i$. Moreover, the joint distribution of $\overline{X}(t)$ is the product of the marginal distribution of each queue $\overline{X}_i(t)$. ■

This tandem queue is depicted in Fig. 5.

C. Approximation

Although the upper bound in Lemma 10 is reasonably good for block repetition codes, it becomes looser for MDS codes, especially when the arrival rate λ approaches μ. This is due to the fact that MDS coding heavily prioritizes requests with more pieces, thereby relying extensively on resource pooling to drain the system. Since this is not captured adequately by the decoupled system, the bounds becomes imprecise. Based on these bounds, it may appear that MDS codes do not perform as well as block repetition codes because of unequal resource allocation. In reality, the MDS coding scheme works better as its flexibility takes advantage of dynamic resource pooling. This performance improvement can be attributed to the fact that the extra service available at higher levels can easily trickle down when the queues are empty. In contrast, the equal partition of resources among all the levels imposed in block repetition coding is less suited to this cascading effect. Unfortunately, this idea is not captured adequately in the two bounding techniques. Nevertheless, these bounding techniques and the cascading effect naturally invite the following approximation.

We wish to approximate the original queueing system via a series of uncoupled queues in tandem, which we denote by

$$\tilde{X}(t) = \left(\tilde{X}_0(t), \ldots, \tilde{X}_{k-1}(t)\right).$$

As before, we take the external arrival process to be Poisson with rate λ. The servers feature independent exponential service times with rates $\tilde{\gamma}_i$ where $i \in \{0, \ldots, k-1\}$. We employ $\tilde{\pi}$ to denote the equilibrium distribution of this approximate system. Under this approximation, each queue i has a Poisson arrival of rate λ, and independent marginal distribution $\tilde{\pi}_i$.

Values for base rates $(\tilde{\gamma}_{i-1} : i \in [k])$ and marginal distributions $(\tilde{\pi}_{i-1} : i \in [k])$ are determined recursively. The underlying idea is to compensate for the cascading effect by shifting the service rates of servers at lower levels in the system. We start with exponential service rate $\tilde{\gamma}_{k-1} = \gamma_{k-1}$ because the last queue never gets additional resources. The probability distribution of this single queue can be computed using the single queue approximation of Poisson arrival rate λ and exponential service rate $\tilde{\gamma}_{k-1}$. When this queue is empty, which occurs with probability $\tilde{\pi}_{k-1}(0)$, its processing power is diverted to level $k-2$. The average processing rate at level $k-2$ then becomes $\tilde{\gamma}_{k-2} = \gamma_{k-2} + \tilde{\gamma}_{k-1}\tilde{\pi}_{k-1}(0)$. In turn, marginal distribution $\tilde{\pi}_{k-2}$ is obtained via the single queue approximation with Poisson arrivals at rate λ and approximate exponential service rate $\tilde{\gamma}_{k-2}$. Continuing this procedure, taking queues to be independent from one another, the average service rate associated with queue $i-1$ can be

approximated by applying the recursion

$$\tilde{\gamma}_{i-1} = \begin{cases} \gamma_{k-1}, & i = k \\ \gamma_{i-1} + \tilde{\gamma}_i \tilde{\pi}_i(0), & i \in [k-1]. \end{cases}$$

At every step, marginal distribution $\tilde{\pi}_i$ and the probability of queue i being empty is obtained through the single queue approximation with arrival rate λ and exponential service rate $\tilde{\gamma}_i$.

For this approximate system, the service rate $\tilde{\gamma}_{i-1}$ assigned to queue $i-1$ corresponds to its base service rate γ_{i-1} plus the rates inherited from its upstream neighbors when they are empty. We note that the approximated rates can also be unraveled; for $i \in [k-1]$,

$$\begin{aligned}\tilde{\gamma}_{i-1} = {} & \gamma_{i-1}\tilde{\pi}(y_i > 0) + (\gamma_{i-1} + \gamma_i)\tilde{\pi}(y_i = 0, y_{i+1} > 0) \\ & + \cdots + (\gamma_{i-1} + \cdots + \gamma_{k-1})\tilde{\pi}(y_i = 0, \ldots, y_{k-1} = 0).\end{aligned}$$

Once defined, Markov chain $\tilde{X}(t)$ can be characterized fully, and its stationary distribution admits a simpler form. These results appear in Theorem 11.

Theorem 11 (Approximation)**.** *Consider a continuous-time Markov chain $\tilde{X}(t) \in \mathbb{N}_0^k$ with non-zero transition rates defined by*

$$Q(y, a_i(y)) = \begin{cases} \lambda, & i = 0 \\ \tilde{\gamma}_{i-1}\mathbf{1}_{\{y_{i-1}>0\}}, & i \in [k]. \end{cases}$$

For all arrival rates $\lambda < \min_{i\in[k]} \tilde{\gamma}_{i-1}$, the Markov process $\tilde{X}(t)$ is positive recurrent, and the equilibrium distribution for this Markov process is given by

$$\tilde{\pi}(y) = \prod_{i=0}^{k-1} \left(1 - \frac{\lambda}{\tilde{\gamma}_i}\right)\left(\frac{\lambda}{\tilde{\gamma}_i}\right)^{y_i}.$$

The mean sojourn time of a request is equal to

$$\tilde{W} = \sum_{i=0}^{k-1} \frac{1}{\tilde{\gamma}_i - \lambda} = \sum_{i=0}^{k-1} \frac{1}{\Gamma_i - (k-i)\lambda}.$$

Proof: We observe that the Markov process $\tilde{X}(t)$ is a series of uncoupled queues in tandem. We know that the joint distribution of the process $\tilde{X}(t)$ is the product of the marginal distribution of each queue $\tilde{X}_i(t)$. Every queue i is subject to a Poisson arrival process at rate λ and exponential service with rate $\tilde{\gamma}_i$. Thus, for $\lambda < \min_{i\in[k]} \tilde{\gamma}_{i-1}$ such that the process $\tilde{X}(t)$ is positive recurrent, the marginal distribution $\tilde{\pi}_i$ of queue i is

$$\tilde{\pi}_i(y_i) = \left(1 - \frac{\lambda}{\tilde{\gamma}_i}\right)\left(\frac{\lambda}{\tilde{\gamma}_i}\right)^{y_i}.$$

We note that

$$\tilde{\gamma}_i \tilde{\pi}_i(0) = \tilde{\gamma}_i \left(1 - \frac{\lambda}{\tilde{\gamma}_i}\right) = \tilde{\gamma}_i - \lambda.$$

Hence, we can inductively compute the total service rate to queue i, leading to close-form expression

$$\tilde{\gamma}_{i-1} = \begin{cases} \Gamma_{i-1} - (k-i)\lambda, & i \in [k-1] \\ \Gamma_{k-1}, & i = k, \end{cases}$$

where $\Gamma_i = \sum_{j=i}^{k-1} \gamma_j$, as defined in Section VI. The statement of the theorem follows. ∎

Below, we derive expressions for the mean sojourn times associated with the approximate system $\tilde{X}(t)$ under block repetition and MDS coding. These expressions are especially useful when k and n are large.

Remark 12 (Mean Sojourn Time for Approximate System Under Block Repetition Coding). The mean sojourn time of a request in $\tilde{X}^{\text{rep}}(t)$ under block repetition coding is equal to

$$\tilde{W}^{\text{rep}} = \sum_{i=0}^{k-1} \frac{1}{(k-i)(\frac{n}{k}\mu - \lambda)}.$$

This result is obtained by computing Γ_i^{rep} using the base rates for block repetition coding found in (5), and then substituting the resulting expressions into the equation for mean sojourn time given in Theorem 11. From the integral bounds on the Harmonic sum, we know that $\log(k+1) \leqslant \sum_{t=1}^{k} \frac{1}{t} \leqslant \log k + 1$, and we can write

$$\frac{\frac{k}{n\mu}\log(k+1)}{(1 - \frac{k\lambda}{n\mu})} \leqslant \tilde{W}^{\text{rep}} \leqslant \frac{\frac{k}{n\mu}(\log k + 1)}{(1 - \frac{k\lambda}{n\mu})}.$$

Remark 13 (Mean Sojourn Time for Approximate System Under MDS Coding). The mean sojourn time of a request in $X^{\text{mds}}(t)$ under MDS Coding is given by

$$\tilde{W}^{\text{mds}} = \sum_{i=0}^{k-1} \frac{1}{(k-i)\left(\mu\frac{n-i}{k-i} - \lambda\right)}.$$

This argument is similar to that of Remark 12. First, use (6) to find expressions for Γ_i^{mds}. Then, substitute these expressions into the equation for mean sojourn time in Theorem 11. From the integral bounds on the sum $\int_1^{k+1} \frac{dx}{\mu(n-k)+(\mu-\lambda)x} \leqslant \sum_{t=1}^{k} \frac{1}{\mu(n-k)+(\mu-\lambda)t} \leqslant \int_0^k \frac{dx}{\mu(n-k)+(\mu-\lambda)x}$, we can write

$$\frac{1}{\mu-\lambda}\log\left(\frac{1 - \frac{(k+1)\lambda}{(n+1)\mu}}{1 - \frac{\lambda+k\mu}{(n+1)\mu}}\right) \leqslant \tilde{W}^{\text{mds}} \leqslant \frac{1}{\mu-\lambda}\log\left(\frac{1 - \frac{k\lambda}{n\mu}}{1 - \frac{k}{n}}\right).$$

If we keep the code-rate $\frac{k}{n}$ and the system load $\frac{k\lambda}{n\mu}$ fixed, then we infer from these results, that the waiting time for block repetition coding increases logarithmically with the number of servers n, whereas that of the MDS coded system remains essentially constant.

D. Delay Minimizing Coding Scheme

We are interested in finding the optimal coding scheme that minimizes the mean file download time for Poisson arrivals of requests, with *i.i.d.* memoryless service at each cache, and fork-join FCFS scheduling. In general this question remains open, however we can answer this for the class of $[n, k, d]$ symmetric coding schemes exploiting the tight approximations for mean sojourn times developed in Section VI-C.

Assume that data blocks are delivered at each server according to an exponential process with rate μ, independent of one another. Then, the request queue for such a system reduces to a tandem queue with resource pooling via cascading.

Every queue in the reduced model has a dedicated base service rate of γ_i, and the rate vector $\gamma \in \mathbb{R}_+^k$ belongs to the set

$$\mathcal{A} \triangleq \left\{ \gamma \in \mathbb{R}_+^k : \gamma_j \geq \mu, \sum_{j=0}^{k-1} \gamma_j = n\mu, \lambda(k-i) < \sum_{j=i}^{k-1} \gamma_j \right\}.$$

These three conditions correspond to the service available to individual queues in the reduced model is at least equal to one dedicated server, the total available service rate being equal to the aggregate resources of all the servers combined, and the allocation fulfilling the stability condition for arrival rate λ. Equivalently, we can write this collection as

$$\mathcal{A} = \left\{ \gamma \in \mathbb{R}_+^k : \Gamma_0 = n\mu, \ \lambda(k-i) < \Gamma_i \leq (n-i)\mu \right\}.$$

Under these constraints, we are interested in identifying the best allocation of service among the queues in the tandem sequence; that is, the allocation that minimizes the approximation for the mean sojourn time of a request. Not so surprisingly, MDS coding is optimal in that sense.

Theorem 14 (Delay minimizing code)**.** *The MDS coding scheme minimizes the approximate mean sojourn time for a fork-join queueing system with identical exponential servers among all symmetric codes.*

Proof: Since $\Gamma_0 = n\mu$ is fixed, the minimizer of the approximate mean sojourn time in $\tilde{X}(t)$ is given by

$$\gamma^* = \arg\min \left\{ \sum_{i=1}^{k-1} \frac{1}{\Gamma_i - (k-i)\lambda} : \gamma \in \mathcal{A} \right\}.$$

The objective function is a decreasing function of free variables $\{\Gamma_i : i \in [k-1]\}$, and hence the minimum is achieved when $\Gamma_i^* = (n-i)\mu$. This allocation corresponds to base rates

$$\gamma_{i-1}^* = \begin{cases} \mu, & i \in [k-1], \\ (n-k+1)\mu, & i = k, \end{cases}$$

which match the service rates associated with MDS coding, as seen in (6). ■

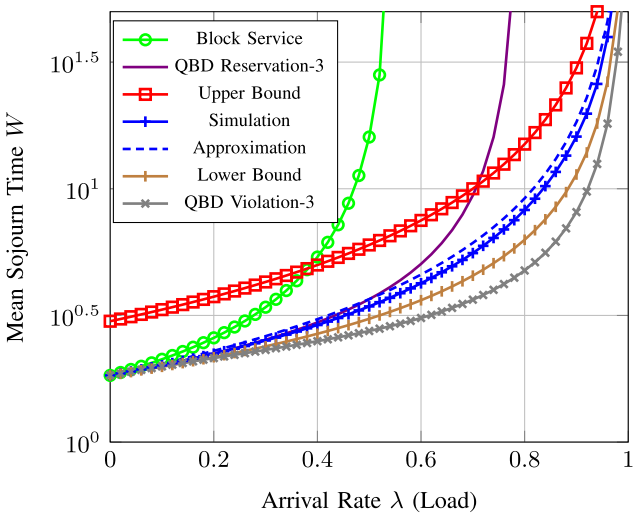

Fig. 6. This graph shows mean sojourn time for a $(9, 3)$ block repetition scheme as a function of arrival rate. It also displays the closed-form approximation, along with several upper and lower bounds.

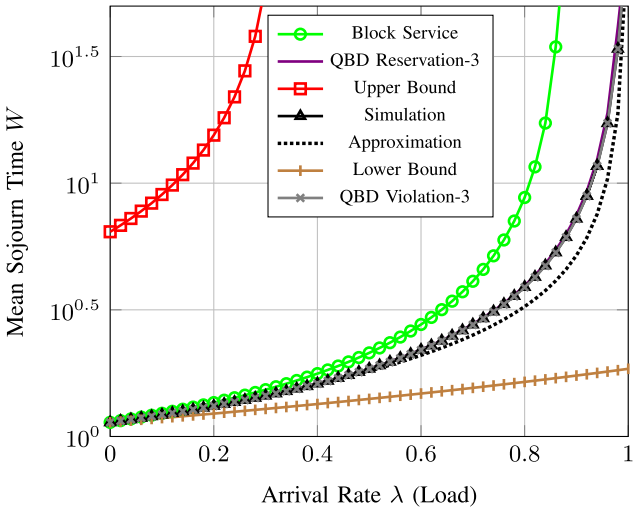

Fig. 7. This graph shows mean sojourn time for a $(9, 3)$ MDS scheme as a function of arrival rate. It also displays the closed-form approximation, along with several upper and lower bounds.

VII. Numerical Studies

System parameters for our numerical study are selected, partly, to enable a comparison with previously published results [14], [21]. We choose the random service time at every cache to be distributed exponentially with rate $\mu = k/n$, and these processes are independent from one another. We emphasize that the service rate is proportional to the number of pieces, and inversely proportional to the number of caches. This normalization step ensures that the system load is equal to the arrival rate λ, and the stability region is $\lambda < 1$ for block repetition and MDS codes. Further, it enables the easy comparison of systems with different design parameters. In a sense, the numerator k accounts for the size of data downloads. If a file is partitioned into k blocks, then individual caches should be able to serve blocks at k times the rate of file downloads. Similarly, the denominator n balances processing power. For a given performance budget, one should be able to choose between having one server with a nominal rate or n servers each with $1/n$ times the nominal rate. Altogether, this yields a normalized block service rate of k/n.

We adopt a $(9, 3)$ linear code for nine servers and three message pieces. One of our objectives in this work is to come up with analytical bounds and approximations to be able to quantify the latency gains obtained by various distributed storage codes. Figure 6 shows that the analytical bounds for block repetition coding are uniformly good over all stable system loads. This is due to the symmetry of service available to all partially fulfilled requests. Contrastingly, we see in Fig. 7 that the upper bound is not so good for MDS coding, and the lower bound also becomes loose as the load increases. The approximations, on the other hand, work very well for both coding schemes.

Figure 8 compares the mean sojourn times of a request with respect to the system load, for the two systems under consideration: Block repetition and MDS coding. As anticipated,

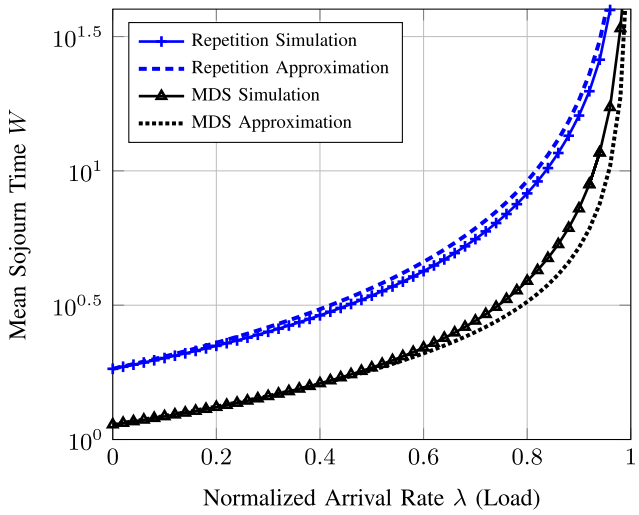

Fig. 8. This graph provides a comparison of the $(9, 3)$ MDS and $(9, 3)$ block repetition schemes. Due to its greater flexibility, the MDS coding scheme performs uniformly better, showcasing significant gains. This trend is accurately captured by the proposed closed-form approximations, which are computationally tractable, even for large systems.

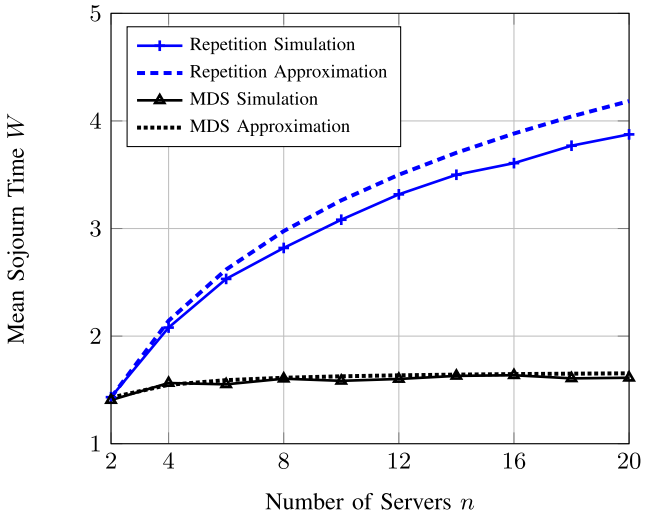

Fig. 9. This graph displays the mean sojourn times of a request for MDS and block repetition coding as the number of servers n increases. Throughout, the code rate is kept constant at $k/n = 0.5$ and the arrival rate is set to $\lambda = 0.3$. MDS coding is more suitable for scaled systems than block repetition coding.

the more sophisticated MDS coded system outperforms the implementation with block repetition coding. Also, we note that the approximated mean sojourn times, being quite close to true performance, reveal the potential gains associated with coded systems. This is especially important for systems with intricate coding schemes and multiple servers because, in such situations, the numerical QBD-based bounds adapted from [20], [21] become hard to evaluate whereas the approximations remain computationally tractable.

To further explore the behavior of distributed coded systems, we consider the case where the number of servers n increases while the code rate $k/n = 0.5$ is maintained. The rate of the Poisson arrival process is set to $\lambda = 0.3$ throughout. The mean sojourn times of a request for block repetition and MDS coding as functions of n appear in Fig. 9. Mean delay for MDS coding is essentially constant as the number of servers n grows; in contrast, it raises significantly for block repetition coding. Thus, MDS coding is more amenable to scaling than block repetition coding. This is very much aligned with the conclusions we draw from Remark 12 and Remark 13.

Finally, we explore performance as a function of message length or code rate. In this case, we fix the number of servers at $n = 24$, and we use a Poisson arrival rate of $\lambda = 0.45$. We vary the message length k for both block repetition and MDS codes. Figure 10 displays mean sojourn times for a request as functions of message length. We observe that the mean delay increases monotonically with reduced redundancy for block repetition coding. However, it features a unique minimum for MDS coding. Furthermore, MDS performance is robust to redundancy reduction, clearly outperforming the block repetition scheme for most rates. Altogether, MDS coding can be utilized for efficient storage without significantly impacting latency.

There are no tight closed-form approximations or bounds available for (n, k) fork-join queue for general independent

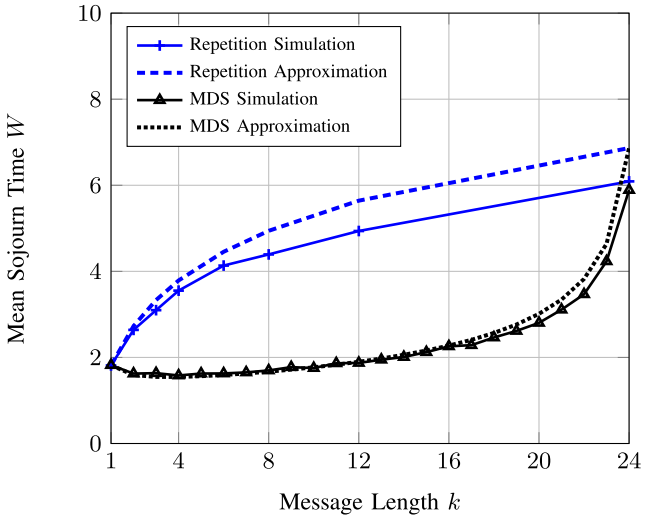

Fig. 10. For a fixed arrival rate $\lambda = 0.45$ and number of servers $n = 24$, this graph plots mean sojourn times for MDS and block repetition coding as message length k increases. Again, MDS codes perform significantly better than block repetition codes.

service times and Poisson arrival of requests, and is an area of future interest. However, we study the mean sojourn time for the general service times numerically and compare the qualitative behavior to the exponential service case. Towards this end, we selected two non memoryless service distributions: (i) shifted-exponential service with shift $c = 0.5$ and exponential rate μ such that the mean service time at each server is $c + \frac{1}{\mu} = n/k$, and (ii) Pareto distribution with shift $x_m = 0.6n/k$ and shape α such that the mean service time $\frac{x_m \alpha}{(\alpha-1)} = n/k$. These choices maintain the mean service time at each server identical to the case of exponential service.

We have plotted the mean sojourn time for $(9, 3)$ MDS and block repetition coded system for *i.i.d.* shifted-exponential service times and Pareto service times with increasing arrival rates in Fig. 11 and Fig. 12, respectively. As expected the

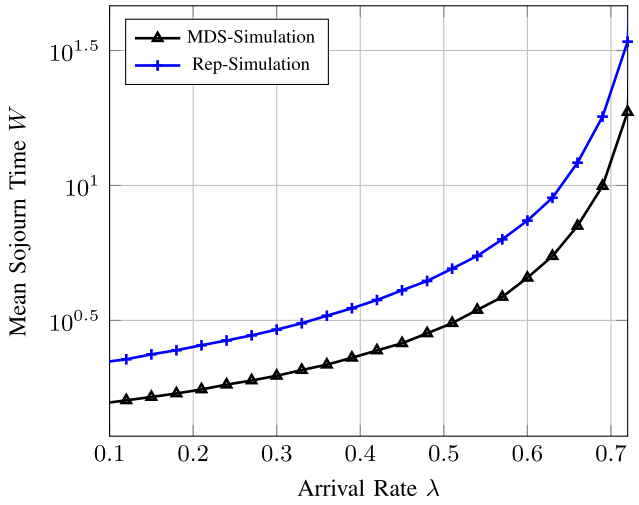

Fig. 11. This graph displays the mean sojourn times of a request for $(9, 3)$ MDS and $(9, 3)$ block repetition coding with shifted exponential service distribution (with shift c and rate μ) as the arrival rate λ increases. Throughout, the service mean is kept constant at $c + \frac{1}{\mu} = \frac{n}{k}$ and the shift is set to $c = 0.5$.

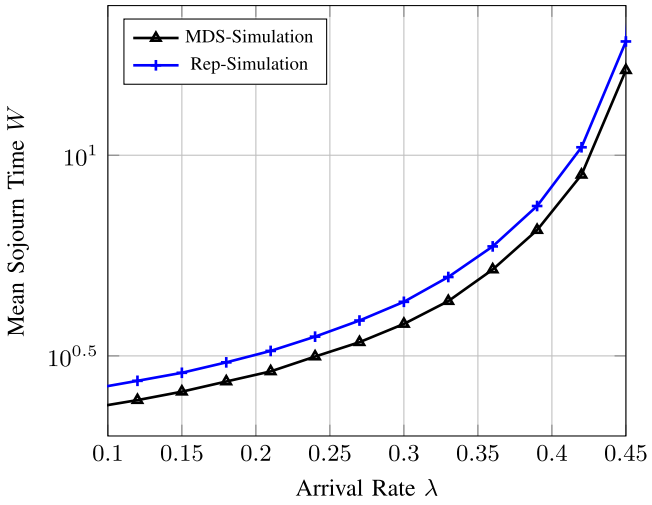

Fig. 12. This graph displays the mean sojourn times of a request for $(9, 3)$ MDS and $(9, 3)$ block repetition coding with Pareto service distribution (with shift x_m and shape α) as the arrival rate λ increases. Throughout, the service mean is kept constant at $\frac{x_m \alpha}{(\alpha-1)} = \frac{n}{k}$ and the shift is set to $x_m = 0.6\frac{n}{k}$

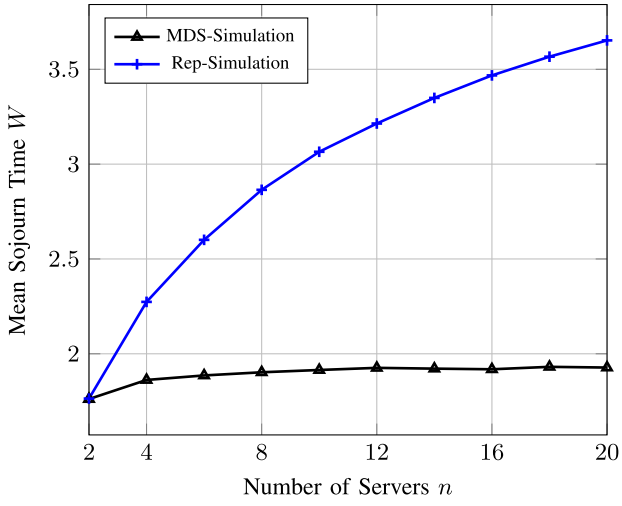

Fig. 13. This graph displays the mean sojourn times of a request for MDS and block repetition coding with shifted exponential service distribution as the number of servers n increases. Throughout, the code-rate is kept constant at $n/k = 2$, the shift is fixed as $c = 0.5$, and the arrival rate is set to $\lambda = 0.3$.

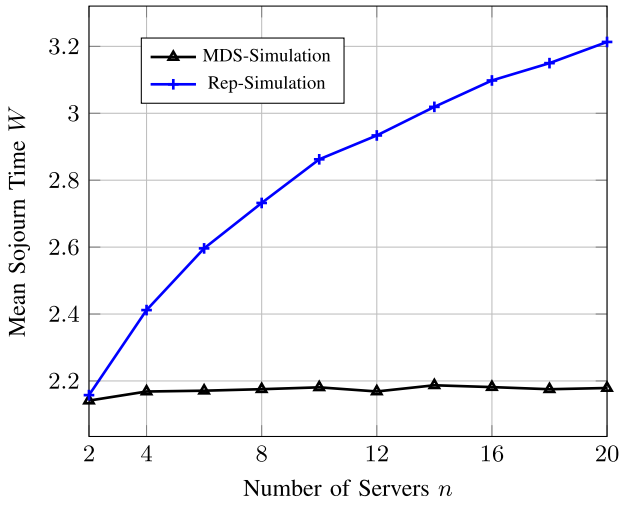

Fig. 14. This graph displays the mean sojourn times of a request for MDS and block repetition coding with Pareto service distribution as the number of servers n increases. Throughout, the code-rate is kept constant at $\frac{n}{k} = 2$, the shift is fixed as $x_m = 0.6\frac{n}{k}$, and the arrival rate is set to $\lambda = 0.3$.

mean sojourn time increases as the arrival rate increases, and MDS coded system outperforms block repetition coded system. Numerical studies suggest that supported arrival rate region for stability is strictly less than unity for both shifted-exponential and Pareto service.

Mean sojourn time for MDS and block repetition coded system for shifted-exponential and Pareto service times with increasing number of servers n is plotted in Fig. 13 and Fig. 14, respectively. For both the cases, the code rate $k/n = 0.5$ and the arrival rate $\lambda = 0.3$ is maintained. The qualitative behavior of mean sojourn time for MDS and block repetition coded system for these two non-memoryless services remain similar to the memoryless service case. We observe that as the number of servers n increase, the mean sojourn time for MDS coded system remains unchanged, while it increases logarithmically in n for the block repetition coded system.

VIII. Conclusion

In this work, we propose an analytical framework to study the latency redundancy tradeoff for distributed coded systems with FCFS fork-join scheduling. The focus is largely on symmetric codes and homogeneous servers. This novel framework enables a complete characterization of the Markov process that governs the evolution of this distributed system. We also propose two stochastically dominating processes that bound the performance of the actual system. These dominating processes provide uniform upper and lower bounds on the mean sojourn time of a request in the queue, for all system loads and all symmetric coding schemes. These bounds are good for block repetition coding, though they are not so tight for MDS coding, especially at higher loads. Together, these bounds naturally give rise to a stochastic approximation for the behavior of distributed coded systems. This approximation

seems very accurate for a range of system loads. Using this approximation, it is possible to make quantitative statements about the delay performance of the distributed coded systems for various system parameters and coding schemes. For instance, this approximation technique adequately captures the mean delay performance gains for MDS coding over block repetition coding. We note that MDS codes minimize the approximate mean delay among all symmetric codes.

We list the key insights derived for a symmetric coded system with Poisson arrival of requests for k fragments, encoded and stored on n *i.i.d.* and memoryless servers, and (n, k) fork-join scheduling.

1) This system is equivalent to a system of k pooled tandem queues.
2) At each stage $i \in \{0, \ldots, k-1\}$, there is an effective arrival rate of λ when the system is stable, since the output rate from a queue is equal to the input rate.
3) We denote the number of useful servers for request with i pieces as N_i. Then stage $k-1$ has N_{k-1} dedicated parallel servers.
4) At each stage $i < k-1$, there are $N_i - N_{i+1}$ dedicated servers, plus all the available servers from stage $i+1$ if it has an empty queue.

For exponential service, it turns out that the system is well approximated by considering aggregate arrival rate of $(k-i)\lambda$ at each stage i corresponding to the required number of pieces $(k-i)$, and aggregate service rate $(n-i)\mu$ corresponding to the parallel number of useful servers.

There exist several potential avenues for future research. Many such opportunities entail relaxing some of the assumptions made in this article including Poisson arrivals, homogeneous exponential service, symmetric coding, and independence among caches. Additional possibilities stem from incorporating new developments in distributed storage such as locally repairable codes or multi-file scenarios. Finally, looking at large deviations for distributed coded systems is also an interesting option for future work.

APPENDIX
PROOF OF THEOREM 2

Recall that $\mathcal{S}(t)$ is a continuous-time Markov process with a discrete state space. It suffices to make sure that these properties are maintained during state transitions in the jump chain of $\mathcal{S}(t)$ to show that the theorem holds at any time t. As such, we prove this result using mathematical induction at jump instants, focusing on the following key properties:

1) $|S_1(t)| \geqslant |S_2(t)| \geqslant \cdots \geqslant |S_{r(t)}(t)|$,
2) $M(S_1(t)) \subseteq M(S_2(t)) \subseteq \cdots \subseteq M(S_{r(t)}(t))$.

As a preliminary step in establishing the theorem, we show that, if properties 1 and 2 hold and there are two requests with message subsets of the same size, then the corresponding sets of useful servers are identical. Assume $|S_i(t)| = |S_{i+1}(t)|$ for some $i \in \{1, \ldots, r(t)-1\}$. It follows that $|M(S_i)(t)| = |M(S_{i+1})(t)|$ from code symmetry. Combining this statement with the set relation implied by property 2, i.e., $M(S_i(t)) \subseteq M(S_{i+1}(t))$, we gather that, being finite, these two sets of useful servers must be identical.

We are ready to proceed with the induction argument. The base case for this system is established at the onset when the system is empty. When there are no requests in the system, the state $\mathcal{S} = e$ and properties 1 and 2 are vacuously true. Furthermore, at the next instant in the jump chain, when a request arrives, the state changes to $(\emptyset)$. Again, the desired properties continue to hold trivially, with $r = 1, S_1 = \emptyset, M(S_1) = [n]$.

At this point, we turn to the inductive step for $r > 0$ requests. Suppose that the current state is $\mathcal{S} = (S_1, \ldots, S_r)$, and assume the two aforementioned properties hold for this particular state. We wish to show that these properties continue to hold for the next state in the jump chain. There are three distinct types of possible transitions to consider. First, if the jump instant corresponds to a new arrival, then the next state is given by

$$\mathcal{S}' = (S_1, \ldots, S_r, \emptyset).$$

Thus, it is evident that the two properties are maintained for this type of transitions.

As a second case, consider the situation when a piece is delivered by server j, but this action does not lead to the completion of a request. This piece will be received by request i where

$$j \in M(S_i) \setminus M(S_{i-1}). \tag{10}$$

Note that there is at most one such request by property 2. Moreover, this property necessarily implies that $M(S_{i-1})$ is a strict subset of $M(S_i)$ and $|S_i| < |S_{i-1}|$. In this case, the next system state can be expressed as

$$\mathcal{S}' = (S_1, \ldots, S_i \cup \{j\}, \ldots, S_r).$$

Property 1 necessarily holds because $|S_i| + 1 \leqslant |S_{i-1}|$. Since $j \notin M(S_{i-1})$, the set of useful servers for the partially fulfilled requests with information subsets $S_{i-1} \cup \{j\}$ and S_{i-1} are identical. Further, from the definition of useful servers, we have $M(T \cup U) = M(T) \cap M(U)$. These two facts together imply

$$M(S_{i-1}) = M(S_{i-1} \cup \{j\}) \subseteq M(S_i \cup \{j\}).$$

This then leads to property 2 for next state $\mathcal{S}'$.

The third type of transitions occurs when the request at the head of the queue has $k-1$ pieces and it acquires a useful piece, leading to its departure from the system. In this case, the state changes to

$$\mathcal{S}' = (S_2, \ldots, S_r).$$

The two properties are maintained through this type of transitions because the necessary pairwise conditions associated with properties 1 and 2 are inherited from the previous state, albeit with a shift in labeling.

Since the three types of transitions discussed above account for all possible transitions in the jump chain, we conclude that the inductive step is true. Consequently, properties 1 and 2 are valid at any time t for distributed storage systems with symmetric codes and fork-join queues with FCFS service, as claimed.

ACKNOWLEDGEMENT

The authors are grateful to Kannan Ramchandran, Salim El Rouayheb, Emina Soljanin, Rajesh Sundaresan, and Gauri Joshi for valuable discussions and shared insights.

REFERENCES

[1] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[2] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 569–584, Sep. 2006.

[3] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[4] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2809–2816, Jun. 2006.

[5] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Jan. 2010, pp. 1–5.

[6] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. IEEE Allerton Conf. Commun., Control, Comput.*, Sep./Oct. 2010, pp. 1510–1517.

[7] S. Jaggi *et al.*, "Resilient network coding in the presence of Byzantine adversaries," *IEEE Trans. Inf. Theory*, vol. 54, no. 6, pp. 2596–2603, Jun. 2008.

[8] D. Wang, D. Silva, and F. R. Kschischang, "Robust network coding in the presence of untrusted nodes," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4532–4538, Sep. 2010.

[9] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[10] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, "Codes can reduce queueing delay in data centers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2012, pp. 2766–2770.

[11] R. Rojas-Cessa, L. Cai, and T. Kijkanjanarat, "Scheduling memory access on a distributed cloud storage network," in *Proc. Wireless Opt. Commun. Conf. (WOCC)*, Apr. 2012, pp. 71–76.

[12] S. Chen *et al.*, "When queueing meets coding: Optimal-latency data retrieving scheme in storage clouds," in *Proc. IEEE Conf. Comput. Commun.*, Apr./May 2014, pp. 1042–1050.

[13] G. Joshi, Y. Liu, and E. Soljanin, "Coding for fast content download," in *Proc. Allerton Conf. Commun., Control, Comput.*, Oct. 2012, pp. 326–333.

[14] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 989–997, May 2014.

[15] G. Joshi, E. Soljanin, and G. W. Wornell, "Queues with redundancy: Latency-cost analysis," *SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 2, pp. 54–56, Sep. 2015.

[16] G. Joshi, E. Soljanin, and G. Wornell, "Efficient replication of queued tasks for latency reduction in cloud systems," in *Proc. Allerton Conf. Commun., Control, Comput.*, Sep. 2015, pp. 107–114.

[17] D. Wang, G. Joshi, and G. Wornell, "Efficient task replication for fast response times in parallel computation," in *Proc. Int. Conf. Meas. Modeling Comput. Syst. (SIGMETRICS)*. New York, NY, USA, 2014, pp. 599–600.

[18] D. Wang, G. Joshi, and G. Wornell, "Using straggler replication to reduce latency in large-scale parallel computing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 3, pp. 7–11, 2015.

[19] B. Li, A. Ramamoorthy, and R. Srikant, "Mean-field-analysis of coding versus replication in cloud storage systems," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[20] N. B. Shah, K. Lee, and K. Ramchandran, "When do redundant requests reduce latency?" *IEEE Trans. Commun.*, vol. 64, no. 2, pp. 715–722, Feb. 2016.

[21] K. Lee, N. B. Shah, L. Huang, and K. Ramchandran, "The MDS queue: Analysing the latency performance of erasure codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 2822–2842, May 2017.

[22] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, E. Hyytiä, and A. Scheller-Wolf, "Queueing with redundant requests: Exact analysis," *Queueing Syst.*, vol. 83, nos. 3–4, pp. 227–259, 2016.

[23] M. F. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications* (Probability: Pure and Applied). Boca Raton, FL, USA: CRC Press, 1989.

[24] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modeling* (ASA-SIAM Series on Statistics and Applied Probability). Philadelphia, PA, USA: SIAM, 1987.

[25] R. V. Evans, "Geometric distribution in some two-dimensional queuing systems," *Oper. Res.*, vol. 15, no. 5, pp. 830–846, Sep./Oct. 1967.

[26] M. F. Aktaş and E. Soljanin, "Heuristics for analyzing download time in MDS coded storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1929–1933.

[27] S. Kadhe, E. Soljanin, and A. Sprintson, "Analyzing the download time of availability codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1467–1471.

[28] Y. Sun, C. E. Koksal, and N. B. Shroff. (2016). "On delay-optimal scheduling in queueing systems with replications." [Online]. Available: https://arxiv.org/abs/1603.07322

[29] K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, and B. Van Houdt, "A better model for job redundancy: Decoupling server slowdown and job size," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3353–3367, Dec. 2017.

[30] K. Lee, R. Pedarsani, and K. Ramchandran, "On scheduling redundant requests with cancellation overheads," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1279–1290, Apr. 2017.

[31] P. Gopalan, G. Hu, S. Kopparty, S. Saraf, C. Wang, and S. Yekhanin, "Maximally recoverable codes for grid-like topologies," in *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA)*, Philadelphia, PA, USA, 2017, pp. 2092–2108.

[32] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2900–2904.

[33] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[34] E. Casalicchio and S. Tucci, "Static and dynamic scheduling algorithms for scalable Web server farm," in *Proc. 9th Euromicro Workshop Parallel Distrib. Process.*, Feb. 2001, pp. 369–376.

[35] R. C. Burns, R. M. Rees, and D. D. E. Long, "Efficient data distribution in a Web server farm," *IEEE Internet Comput.*, vol. 5, no. 4, pp. 56–65, Jul./Aug. 2001.

[36] M. Harchol-Balter, A. Scheller-Wolf, and A. R. Young, "Surprising results on task assignment in server farms with high-variability workloads," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 287–298, Jun. 2009.

[37] F. P. Kelly, *Reversibility and Stochastic Networks*. New York, NY, USA: Cambridge Univ. Press, 2011.

Ajay Badita (S'19) is a PhD candidate in the ECE department at Indian Institute of Science since Fall 2017. He received M.Tech. degree from NIT Rourkela in 2015 and B.Tech. degree from JNTU Kakinada in 2011, both in electronics and communication engineering. His research interests include delay-sensitive communication, compute, and storage in distributed systems.

Parimal Parag (S'04–M'11) is an assistant professor in the ECE department at Indian Institute of Science. Prior to that, he was a senior system engineer (R&D) at Assia Inc. in Redwood City (2011-2014). He received a Ph.D. degree from the Texas A&M University in 2011, the M.Tech. and B.Tech. degrees in 2004 from IIT Madras, all in electrical engineering. He was a co-author of the 2018 IEEE ISIT student best paper, and a recipient of the 2017 early career award from the Science and Engineering Research Board.

Jean-Francois Chamberland (S'98–M'04–SM'09) is a Professor in the Department of Electrical and Computer Engineering at Texas A&M University. He received a Ph.D. degree from the University of Illinois at Urbana-Champaign. His research interests are in the areas of computing, information, and inference. He has been the recipient of an IEEE Young Author Best Paper Award from the IEEE Signal Processing Society, and a Faculty Early Career Development (CAREER) Award from the National Science Foundation.