Latency Analysis for Distributed Storage

Parimal Parag*, Archana Bura*
*Department of Electrical Communication Engineering
Indian Institute of Science, Bengaluru, KA 560012, India
{parimal, archanabura}@ece.iisc.ernet.in

Jean-Francois Chamberland†
†Department of Electrical and Computer Engineering
Texas A&M University, College Station, TX 77843-3128
chmbrlnd@tamu.edu

***Abstract*—Modern communication and computation systems consist of large networks of unreliable nodes. Yet, it is well known that such systems can provide aggregate reliability via information redundancy, duplicating paths, or replicating computations. While redundancy may increase the load on a system, it can also lead to major performance improvements through the judicious management of additional system resources. Two important examples of this abstract paradigm are content access from multiple caches in content delivery networks and master/slave computations on compute clusters. Many recent articles in the area have proposed bounds on the latency performance of redundant systems, characterizing the latency-redundancy tradeoff under specific load profiles. Following a similar line of research, this article introduces new analytical bounds and approximation techniques for the latency-redundancy tradeoff for a range of system loads and two popular redundancy schemes. The proposed framework allows for approximating the equilibrium latency distribution, from which various metrics can be derived including mean, variance, and the tail decay of stationary distributions.**

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

I. Introduction

The data infrastructures that support our digitally connected world must face the incessant demands of its billions of users. This results in tens of terabytes of digital traffic traveling over network subcomponents every second. Multimedia content is the predominant source of Internet traffic, and a large portion of this data is hosted on content delivery networks. These networks store content redundantly at multiple servers to ensure reliability and availability, despite the fact that individual servers are failure prone.

Forward error correction and, more specifically, maximum distance separable (MDS) codes have emerged as fundamental means to enhance information storage and improve content dissemination across networks. For instance, preserving data over a collection of agents using MDS coding can increase the reliability of a storage system, while limiting the amount of redundancy necessary 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 witnessed 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 [5], [6] and security [7], [8].

Beyond reliability, latency is becoming a prime design criterion for distributed systems. This emerging viewpoint on overall performance has spurred several initiatives aimed at better understanding the impact of coding on latency performance of distributed storage systems [9]–[12]. In this context, it is pertinent to note that the content access time in distributed storage is related to the computation time of master/slave tasks in compute clusters with job replication [13]–[17]. Since a request queue for job completion is mathematically equivalent to a request queue for content access, we restrict our treatment of the problem to the latter scenario and, in particular, access delay in distributed storage systems.

We consider two competing implementations for our study of the latency-redundancy tradeoff. The first system is based on file fragmentation and duplication, a version of *repetition* coding. Under this scheme, a request is fulfilled by collecting one information piece of every available kind. The original file is subsequently reconstructed by appending the various pieces. The second type of implementation we are interested in is a more elaborate coded abstraction where a request is fulfilled by gathering any k out of n pieces. Every combination of k blocks is assumed to be linearly independent, and the original file can therefore be recovered via a standard decoding process. We will refer to this scheme as *MDS* coding. In this work, we show that both the above schemes can be studied under the same analytical framework of tandem queues with server pooling.

A. Literature Survey

Recent advances in low complexity codes for distributed storage that have low overhead for repair and regeneration bandwidth [1], [4]–[6], [18] have fueled an interest in understanding the delay-redundancy tradeoff [9]–[17], [19], [20]. In distributed storage systems, incoming requests initiate the acquisition of several chunks of files stored on multiple servers. These incoming requests are queued until all the necessary chunks are delivered. The request queue for multiple chunks of a file in a distributed storage system can be modeled by a multi-dimensional Markov chain, yet such chains often remain hard to analyze. Consequently, a main approach to understand these systems is to find systems that stochastically

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. CCF-1619085 and by the Defence Research and Development Organization (DRDO), Government of India under Grant No. DRDO-0654. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF or the DRDO.

978-1-5090-5336-0/17/$31.00 ©2017 IEEE

dominate this multi-dimensional Markov chain, and are easier to characterize. Authors in [10]–[12] find quasi birth-death processes that can bound the performance of a request queue modeled as a multi-dimensional Markov chain. Using matrix-geometric methods, one can numerically evaluate performance attributes of these quasi birth-death processes, thereby finding numerical bounds on the performance of the actual queue.

Similarly, in a compute cluster, a parallelizable job request can be divided into smaller jobs that can be processed concurrently at multiple servers. Jobs are queued until each of the sub-tasks are completed. This queue can also be modeled as a multi-dimensional Markov chain. In fact, the resulting abstraction is equivalent to the request queue Markov chain for a distributed storage system. Authors in [13]–[15] propose fork-join queues that bound the performance of the job queue. Implicitly, the authors are considering an MDS coding scheme where any equal sized subset of servers can finish the job. This methodology is employed to provide an upper bound on the mean time for the job completion.

B. Main Contributions

In this article, we offer an alternative analytical approach to study content access delay in distributed storage systems for repetition and MDS codes. We provide a framework where the model structure is independent of the coding scheme in that the coding scheme solely affects the model specific parameters. Hence, we can use the same analysis for replication and MDS coding. We emphasize that this framework can equivalently characterize job completion time when an incoming job is subdivided into sub-tasks and sent to a constrained set of servers.

We show that the request queue at a distributed storage system can be regarded as a series of queues in tandem, pooling their server resources. Under this viewpoint, the service of tandem queues is coupled and state-dependent. We find two simpler, analytically tractable tandem queues that dominate the performance of the original tandem queue from above and below. We completely characterize these dominating queues, providing upper and lower bounds on latency performance of a distributed storage system with redundant requests. The performance criteria we examine, are derived from the equilibrium distribution on the number of queued requests. We also find bounds and approximations for the waiting time of a typical request in the queue.

Our results answer open questions mentioned in [10]. For example, we provide analytical characterization of latency-redundancy tradeoff for a range of system loads. In addition, our proposed framework allows for answering other open questions such as quantification of redundancy-latency tradeoff for different coding schemes, and it allows for bounds on latency metrics more general than mean, such as variance and the distribution tail.

II. Coding Model

We assume that a single content m consists of k pieces $(m_1, m_2, \dots, m_k) \in \mathbb{F}_q^k$ for a sufficiently large finite field $\mathbb{F}_q$. These messages are stored redundantly at n servers. We assume linear codes for erasure channels, with generator matrix $G \in \mathbb{F}_q^{k \times n}$ and parity check matrix $H \in \mathbb{F}_q^{(n-k) \times n}$, such that $GH^T = 0$. We will refer to these codes as (n, k) linear codes.

For each message $m \in \mathbb{F}_q^k$, we have a codeword $x = mG$. We assume that a server $j \in [n]$ stores the information symbol x_j. It can be shown that the message m can be decoded by observing information symbols at a subset C of all servers, where the columns of the parity check matrix H corresponding to the unobserved servers are linearly independent. We assume that the parity check matrix H has full rank of $(n - k)$, and hence the minimum size of a set C of servers required to decode a message m is k. As mentioned above, we consider two competing cases, repetition and MDS coding. For simplicity, we take n/k to be an integer.

A. Repetition Coding

Under replication, each distinct piece m_i is stored at n/k servers. We denote the set of servers with message i as

$$C_i = \{j \in [n] : x_j = m_i\}.$$

Then, $|C_i| = n/k$ and $\{C_i : i \in [k]\}$ partitions the set of servers $[n]$. Furthermore, we can decode message m as soon as we observe the information pieces stored at a subset of servers $C \subset [n]$ such that $|C| = k$ and $|C \cap C_i| = 1$ for every $i \in [k]$. We emphasize that there are $(n/k)^k$ such subsets.

B. MDS Coding

In the case of MDS coding, each server $j \in [n]$ stores an independent symbol x_j, where $x = mG$. That is, the set of servers with information symbol x_j is $C_j = \{j\}$. For MDS codes, any $(n - k)$ columns of the parity check matrix H are linearly independent. Hence, observing information symbols stored at any size-k subset C of servers suffices to reconstruct message m. There are $\binom{n}{k}$ such subsets. Since $\binom{n}{k} > (\frac{n}{k})^k$, it is clear that MDS coding offer a much larger number of decodable subsets when compared to repetition coding. Consequently, MDS coding should perform better.

C. Example

For illustration, we compare the performance of fragmentation and duplication against that of an MDS coded system. We consider the simple scenario where $k = 2$, $n = 4$ and message m consists of two parts A and B. We call this example a $(4, 2)$ linear code with message $m = (\mathrm{A}, \mathrm{B})$. In the repetition case, the original file is split into two components, labeled A and B, and the messages are stored on two distinct servers. In this scenario, a request is completed when the user is successfully served by a cache containing A and another one holding B. On the other hand, in the MDS coding paradigm, a file is partitioned into two pieces and independent linear combinations of these blocks are created. This latter strategy necessitates that packets be long 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

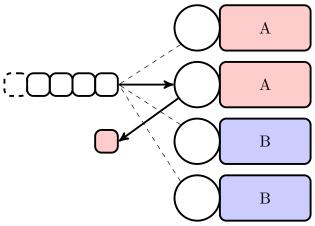

Figure 1. This figure depicts a distribution network with four caches. Two servers are storing message A, and the other two servers host message B. Under this divide and duplicate paradigm, a user must obtain piece A and piece B to reconstruct the original media object.

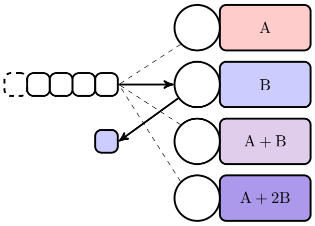

Figure 2. In a more elaborate coded system, the various caches store independent messages. The original media object can therefore be recovered by decoding the content of any two distinct data blocks.

instance, when four caches are present, candidate messages could be A, B, $\mathrm{A}+\mathrm{B}$, $\mathrm{A}+2\mathrm{B}$. The original media object can then be recovered by successfully contacting any two servers, a slightly more flexible stipulation than before. The operations of these alternate infrastructures for systems with four servers are depicted in Figure 1 and Figure 2, respectively.

III. Queueing Model

Requests for the stored object are queued at a central location with arbitrarily large waiting room. These requests are subsequently served by the servers that have the necessary content. Again, consider the $(4, 2)$ linear coding case with message $m = (\mathrm{A}, \mathrm{B})$. In the replication coding case, if a request already has message A, then it can only be served by the two servers that have message B. Contrastingly, in the MDS coding case, if a request has information symbol A, then it can be served by any of the remaining three servers with symbols B, $\mathrm{A}+\mathrm{B}$, or $\mathrm{A}+2\mathrm{B}$.

We assume that each incoming request needs all k different pieces of the content, and the inter-arrival time between the requests is exponentially distributed with mean $1/\lambda$. Each server supplies one piece of information, which can ultimately be combined with the other pieces to recover the desired file. We adopt a *push* model for service. Under this framework, each server works for a random amount of time and then instantaneously serves the head waiting request that needs its content. We call this server waiting time as the *service time* or *waiting time* by the server for each data block. We assume that service times at each server form a sequence of independent and identically distributed exponential random variables, each with rate $\mu = k/n$. Notice that we have scaled the service rate proportional to the number of pieces. This is motivated by the fact that, if the mean time to serve a k length file is n, then the mean time to serve a unit length file should be n/k. This random process is independent of the service times of other servers.

This service model is different from the usual *pull* service model. In a pull service model, a request is tied to a particular server during its random service time. Contrastingly, in the push model, a request can be served by any of the competing servers that finish waiting first. This is equivalent to all the servers serving the request in parallel, and as soon as one of them finishes, the rest of them drop serving this request. As such, if a request can be served by n available servers, the service time for this request is the minimum among the service times of the n servers. Although the pull model is more common in the literature, the push model is amenable to mathematical analysis, as evinced below.

A. Scheduling Model

We assume a work conserving policy under our push service model. That is, when a server becomes available after its random waiting time, it can choose to serve any of the requests that do not have its content. Upon taking this decision, the server will instantaneously transfer its content to a device that needs it, if there exists such a request. The selection of requests among all those that need the content from the server is referred to as scheduling in the present context.

Consider one sample path for the evolution of the request queue with $(4, 2)$ linear coding and message $m = (\mathrm{A}, \mathrm{B})$. We assume that the system is initially empty, and four requests arrive before any of the servers become available. Suppose that the first available server has content A. All four waiting requests have no piece; this server can then select any of the waiting requests for an instant transfer of content A. The first scheduling decision therefore consists in selecting one destination among those four requests to transmit A.

Once the server selects one of the waiting requests and transfers piece A, there remain three requests with no piece and one partially fulfilled request with piece A. Suppose that the next server becomes available before any further arrival, and this server possesses content B. This content is useful to both types of requests; the single request with piece A, and the three requests with no piece. Thus, the second and more important scheduling decision is the request selection among those with different set of information symbols.

Scheduling policies greatly affect system performance. In this article, we assume that the scheduler employs a shortest expected remaining processing time policy with preemption [21]. All the requests with an identical number of data blocks have the same expected remaining processing time. Among all such requests, a server selects the request that arrived first. However, among all the requests with different subsets of information symbols, the request with the largest number of information symbols has the shortest expected remaining processing time and is therefore given priority.

The authors in [20] show that this scheduling policy minimizes mean latency among all on-line scheduling algorithms.

As we will see, this scheduling policy greatly simplifies our analysis of the system, making the evolution of the underlying Markov process tractable.

B. Reduction of State Space

We can label requests according to the information they have already accumulated. Let $Y_S(t)$ denote the number of requests with subset $S \subset x$ of information symbols. Then, under Poisson arrivals and the exponential push service model, the collection

$$\tilde{Y}(t) = \{Y_S(t) \in \mathbb{N}_0 : S \subset x = \{x_1, \ldots, x_n\}\}$$

forms a Markov chain. For repetition coding, there are k distinct information symbols corresponding to the disjoint segments of message m. Thus, the state of the request queue for replication coding becomes

$$\tilde{Y}(t) = \{Y_S(t) \in \mathbb{N}_0 : S \subset m = \{m_1, \ldots, m_k\}, |S| < k\}.$$

The dimension of this Markov chain is $2^k - 1$, and we note that it grows exponentially in the number of pieces.

Contrastingly, there are n independent information symbols for the MDS coding scheme, with symbol x_j stored at server $j \in [n]$. The state of the request queue for MDS coding with a general scheduling policy is

$$\tilde{Y}(t) = \{Y_S(t) \in \mathbb{N}_0 : S \subset x, |S| < k\}.$$

This request queue has a Markov chain of dimension $\sum_{j=0}^{k-1} \binom{n}{j}$. Since $\binom{k}{j} \leq \binom{n}{j}$, we note that the dimension of the request queue for MDS coding can be significantly larger than the corresponding value for repetition coding. Yet, under priority scheduling and homogeneous servers, the dimension of the request queue reduces significantly for these two coding schemes. To formalize this result, we must first introduce a notation to keep track of the number of information labels for which $Y_S(t)$ is non-zero.

Definition 1. The collection of information labels associated with active requests at time t is defined as

$$\mathcal{S}(t) = \{S \subset x : Y_S(t) > 0\}.$$

In words, set $\mathcal{S}(t)$ represents the collection of information subsets accumulated by active requests at time t. We will show that, under a priority scheduling policy, the collection $\mathcal{S}(t)$ maintains a particular structure that reduces the effective dimension of the Markov chain.

Lemma 2. *For a linear code, under priority scheduling and the push service model, the collection of information subsets $\mathcal{S}(t)$ is totally ordered, where the order is defined in terms of set inclusion.*

Proof: We assume that the queueing system starts devoid of requests at time zero. We wish to prove Lemma 2 using mathematical induction on certain events. First, we note that the arrival events in this continuous-time Markov chain translate into an increase of $Y_\emptyset$, thereby trivially maintaining set ordering. Thus, we only need to consider the time instants corresponding to the expiration of waiting times at the servers. We identify such time-instants chronologically by t_p.

Suppose that the inductive hypothesis is true at time t_p, i.e., after exactly p service events. Specifically, the collection of information subsets $\mathcal{S}(t_p)$ is totally ordered by set inclusion. At the next service instant t_{p+1}, a symbol x_j becomes available. We can assume that there exists $S \in \mathcal{S}(t)$ such that $x_j \notin S$ without loss of generality; otherwise the state of the system will remain unchanged. Since $\mathcal{S}(t_p)$ is a chain, we can find the maximal set T such that

$$T = \max\{S \in \mathcal{S}(t_p) : x_j \notin S\}.$$

By the definition of the priority schedule, the head of the line request with the set of messages T gets x_j. It follows that,

$$Y_S(t_{p+1}) - Y_S(t_p) = \begin{cases} 0, & S \in \mathcal{S}(t_p) \setminus \{T, T \cup \{x_j\}\} \\ -1, & S = T \\ 1, & S = T \cup \{x_j\}. \end{cases}$$

It remains to show that $\mathcal{S}(t_{p+1})$ is too a chain. This is immediate when $T \cup \{x_j\}$ already belongs to $\mathcal{S}(t_p)$. Therefore, it suffices to consider the case when $Y_{T\cup\{x_j\}}(t_p) = 0$. In this case, all the subsets of T are necessarily subsets of $T \cup \{x_j\}$. Furthermore, all the sets including T must also include $T \cup \{x_j\}$ because T was the maximal set without x_j in $\mathcal{S}(t_p)$. Thus, set ordering ensues and the lemma holds. ■

Corollary 3. *For a linear code, under priority schedule and push service,*

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

is a Markov chain where $Y_i(t)$ denotes the number of requests that have exactly i information symbols at time t.

Proof: Recall that $Y_S(t)$ is a continuous-time Markov chain for any S. From Lemma 2, we know that, at any given time, there cannot be two distinct information subsets of equal size among active requests. Using the chain property, we can uniquely relabel existing subsets by their cardinality. Recognizing this structure and the symmetric nature of the system, we deduce that $Y(t) = \{Y_{i-1}(t) : i \in [k]\}$ is a Markov chain, where $Y_i(t) = Y_S(t)$ whenever $Y_S(t) > 0$ and $|S| = i$. ■

In some sense, Markov chain $\tilde{Y}(t)$ contains more information than its close relative $Y(t)$. The former keeps track of the information pieces collected by partially fulfilled requests, whereas the latter only records the number of pieces possessed by partially fulfilled requests. Yet, in view of Corollary 3, the more compact version retains the Markov property and it is much simpler to analyze. This step is crucial in our impending derivations. One advantage of the compact representation is the fact that the dimension of this reduced Markov chain $Y(t)$ is k, which grows linearly in the number of pieces. An example may help clarify the discussion.

For illustration purposes, we return to the $(4, 2)$ linear coding schemes with message $m = (\mathrm{A}, \mathrm{B})$. Suppose that we start with an empty system using replication coding. We consider the case where there are two arrivals in the system before a server becomes available. This event results in the number of requests with zero pieces increasing to two, with no partially fulfilled requests. When a server with content A becomes available, it transfers its content to the older request, resulting in one request with content A and one request with no accumulated content. Suppose that a second server with content B becomes available next, before any other arrival. Under priority scheduling, content B is transferred to the request that already possesses A and, thence, this request leaves the system. From this example, we gather that, regardless of the order of the server availability and arrivals, all partially fulfilled requests would have the same piece of information. That is, under no circumstance would we see partially fulfilled requests with content A and other partially fulfilled requests with content B concurrently. Similarly, for the $(4, 2)$ MDS coding scheme, we can argue that all partially fulfilled requests would have exactly one of the four possible contents A, B, A + B, or A + 2B.

Since the exact identity of the messages obtained by the requests is irrelevant for the performance analysis of the symmetric system, we focus on compact Markov chain $Y(t)$ as it is much simpler to analyze. We observe that the state space of the continuous-time Markov chains corresponding to repetition coding and MDS coding are identical. However, their rate matrices differs, as seen below.

C. State Transitions

Let $e_i \in \mathbb{N}_0^k$ denote the k-dimensional unit vector, with a one in the ith location. Consider a request queue state $Y(t) = y = (y_0, y_1, \ldots, y_{k-1}) \in \mathbb{N}_0^k$ at some instant t, where y_i denotes the number of requests with i information symbols. In general, two types of events can occur, namely the arrival of a request or the emergence of a service event. Under Poisson arrivals and exponential service, at most one such event can occur within an infinitesimal amount of time.

Upon arrival into the system, a request has no information symbols. Hence, a new request always increases $Y_\emptyset(t)$; that is, the state y transitions to state $e_0(y) \triangleq y + e_0$. For $i < k$, the transfer of a new content by an available server to a request with $i - 1$ pieces leads to a unit decrease in the number of requests with $i - 1$ pieces, and a corresponding unit increase in the number of requests with i pieces. This leads to a state transition from state y to state $e_i(y) \triangleq y - e_{i-1} + e_i$. Finally, a request leaves the system after reception of k information symbols, denoted by the transition from state y to state $e_k(y) \triangleq y - e_{k-1}$. In summary, we have shown that, in case of an arrival or a server availability, the state of the request queue transitions from the current state y to the next state $e_i(y)$ whenever $y_{i-1} > 0$, where the state $e_i(y)$ is defined by

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

We can alternatively think of the number of requests y as customers that want k different services, with y_i being the number of customers waiting for service i. Once, a customer receives service i, it joins the queue for service $(i + 1)$. Then, we can think of this multi-dimensional Markov chain as a sequence of k queues in tandem, with an external arrival process. These queues are being served by a pool of servers. In this setting, the next state $e_i(y)$ denotes an arrival of a customer in queue i due to a departure from queue $i - 1$.

D. Rate Matrix

In this section, we specify the transition rates of continuous-time process $Y(t)$. We denote the rate from current state y to state $e_i(y)$ by $Q(y, e_i(y))$. It follows from the Poisson model that arrivals in the request queue occurs at a rate λ. That is, transitions from state y to state $e_0(y)$ happen at rate $Q(y, e_0(y)) = \lambda$, irrespective of state y.

Under the push service model where servers have independent exponential service, each with rate $\frac{k}{n}$, the aggregate service rate is k. The emergence of a service event can lead to three distinct possibilities. First, a request with $k-1$ pieces gets the final piece it needs and, subsequently, exits the system. Second, a request with $i - 1$ pieces is promoted to a request with i pieces. Third, all the requests in the system already have the content available at the server, and this service opportunity is lost.

The jump rates for $Y(t)$ associated with service events can be somewhat challenging to describe. One possible approach to characterize these rates leverages the chain property of Lemma 2. To this end, we first define information level i to contain all those requests which have acquired exactly i symbols. Specifically, we can identify the amount of resources devoted to serving requests at information level i, a task which can be accomplished in a straightforward manner. Then, we obtain the exact service rates for particular system states based on the fact that, when some information levels are devoid of requests, the corresponding resources cascade down to the next occupied state in the chain.

Consider an information chain, ordered by set inclusion, tailored to $\mathcal{S}(t)$ and that contains k elements. Let N_i be the number of servers that can supply requests at information level i. Since the number of information symbols gathered by devices increases with i, the number of servers that can offer additional symbols to these devices decreases with i; that is, $N_0 \geq N_1 \geq \cdots \geq N_{k-1}$. Moreover, owing to our priority scheme, the number of servers devoted to requests at information level i is given by the number of server that can serve level i but not level $i + 1$. Mathematically, this number is equal to $N_i - N_{i+1}$ for $0 \leq i < k - 1$. Under the push model with exponential waiting times, the nominal service rate attributed to information level i is equal to

$$\gamma_i = \frac{k}{n} \left(N_i - N_{i+1} \right). \qquad (1)$$

We note that, although $\mathcal{S}(t)$ can change over time, rate γ_i associated with information level i only depends on the coding scheme, and is independent of time. Of course, the packets depart when the entire message is recovered, that is $N_k = 0$.

For repetition coding, the number of servers that can provide messages to requests at information level i is $N_i = (k-i)n/k$. This yields nominal jump rates

$$\gamma_i^{\mathrm{rep}} = \frac{k}{n}\left(N_i - N_{i+1}\right) = 1 \qquad i = 0, \ldots, k-1.$$

On the other hand, the number of servers that can serve requests at information level i under MDS coding is $N_i = (n-i)$ for $i \leq k-1$. Altogether, this produces nominal service rates

$$\gamma_i^{\mathrm{mds}} = \frac{k}{n}\left(N_i - N_{i+1}\right) = \begin{cases} \frac{k}{n}, & i < k-1 \\ \frac{k}{n}(n-k+1), & i = k. \end{cases}$$

As mentioned above, the rates defined in (1) are nominal information level rates. Yet, when some information levels are devoid of requests, their resources dynamically cascade down to the next occupied information level. Accordingly, to find the instantaneous rate at level i, assuming $Y_i(t) > 0$, we must keep track of its inherited resources. Define

$$l_i(t) = k \wedge \min\{l > i : Y_l(t) > 0\}.$$

In words, $l_i(t)$ is the next occupied information level above i at time t and, hence, the first level whose resources will not trickle down to information level i. Consequently, the instantaneous rate at level i or, equivalently, the transition rate into level $i+1$ can be written as

$$Q(y, e_{i+1}(y)) = \sum_{j=i}^{l_i(t)-1} \gamma_j, \tag{2}$$

whenever $Y_i(t) > 0$. Of course, when $Y_i(t) = 0$, no resources can be used at information level i. Moreover, $Q(y, e_0(y)) = \lambda$, as mentioned earlier. We emphasize that rate operator $Q(\cdot,\cdot)$ depends on the state of the system. However, to avoid an overly cluttered notation, we leave this dependence implicit in (2). Altogether, these equations capture all the non-zero transition rates for this continuous-time Markov chain.

We observe that the k-dimensional Markov chain $Y(t)$ is a sequence of k queues in tandem, with an external Poisson arrival at rate λ and service rate γ_i at ith queue. These servers are coupled in the following sense. When one of the queue is empty, its associated server can pool its resources to the first non-empty queue preceding it. We have depicted this tandem queue in Figure 3.

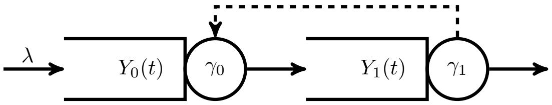

Figure 3. This block diagram displays two queues in tandem with external arrival rate λ and service rates γ_0 and $\gamma_1 = 2 - \gamma_0$, respectively. The ith queue length is denoted by $Y_i(t)$ for $i \in \{0, 1\}$. The dashed line from the server at queue 1 to the server at queue 0 represents the fact that server 1 pools with server 0 whenever queue 1 is empty.

IV. Performance Analysis

In general, a coupled k-dimensional queue such as $Y(t)$ is difficult to analyze. However, we can take advantage of the fact that it is equivalent to a series of k-tandem queues. Below, we find two stochastic systems that are respectively better and worse than these pooled tandem queues, in a sample-path sense. Specifically, we find uncoupled tandem queues that bound the performance of the actual tandem queues with server pooling. Series of queues in tandem with Poisson external arrival and independent exponential service rates are well studied objects. It turns out that even though the departures from one queue act as arrivals to the next queue, the queue distributions are statistically independent [22]. In Lemma 4, we provide uniform bounds for the service rate at each queue, independent of the states of other queues.

Lemma 4. *The transition rate $Q(y, e_{i+1}(y))$ is bounded below and above by*

$$\gamma_i \leq \sum_{j=i}^{l_i(t)-1} \gamma_j \leq \sum_{j=i}^{k-1} \gamma_j \triangleq \Gamma_i,$$

where equality holds for $i = k-1$.

Proof: This follows from the fact that rates are non-negative and $i < l_i(t) \leq k$. ■

Since, the service rate Γ_i upper bounds the service rate of queue i in our original coupled tandem queue, we have the following lower bound on performance.

Theorem 5 (Lower Bound)**.** *Consider a continuous-time Markov chain $X(t) \in \mathbb{N}_0^k$ with rate transition matrix Q, where*

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

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 $X(t)$ is path-wise less congested than $Y(t)$. Furthermore, the equilibrium distribution of $X(t)$ is given by

$$\pi(y) = \prod_{i=0}^{k-1} \left(1 - \frac{\lambda}{\Gamma_i}\right)\left(\frac{\lambda}{\Gamma_i}\right)^{y_i}, \tag{3}$$

and the mean sojourn time in the system is equal to

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

Proof: Stochastic dominance is a consequence of the fact that Markov process $X(t)$ is derived from process $Y(t)$ by adding extra servers for requests at information level i. To find the equilibrium distribution of $X(t)$, we take advantage of its tandem structure. Each queue i has Poisson arrivals at rate λ and independent random service times distributed exponentially with mean $1/\Gamma_i$. Therefore, the joint distribution of $X(t)$ is the product of the marginal distribution of each queue $X_i(t)$, yielding (3). ■

A corresponding tandem queue is depicted in Figure 4.

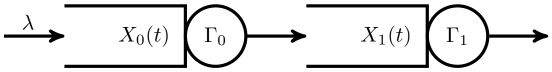

Figure 4. This block diagram displays two queues in tandem with external arrival rate λ and service rates $\Gamma_0 = 2$ and $\Gamma_1 = 2 - \gamma_0$, respectively. This choice of the rate parameters leads to a lower bound on the queue congestion.

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

Theorem 6 (Upper Bound)**.** *Consider a continuous-time Markov chain $X(t) \in \mathbb{N}_0^k$ with rate transition matrix Q, where*

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

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

$$\pi(y) = \prod_{i=0}^{k-1} \left(1 - \frac{\lambda}{\gamma_i}\right) \left(\frac{\lambda}{\gamma_i}\right)^{y_i}, \qquad (4)$$

and the mean sojourn time in the system is equal to

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

Proof: Stochastic dominance follows from the fact that the Markov process $X(t)$ is derived from the process $Y(t)$, by letting the servers be idle when their respective queues are empty, rather than cascading their resources. Notice that, in the original formulation, available servers can transfer their content to any request at a lower information level; whereas in the current setting, servers only provide messages at their current information level. To find the equilibrium distribution of $X(t)$, we again embrace the tandem structure. Each queue i has Poisson arrivals at rate λ and independent random service times distributed exponentially with mean $1/\gamma_i$. Hence, the joint distribution of $X(t)$ is the product of the marginal distribution of each queue $X_i(t)$, as seen in (4). ■

This tandem queue is depicted in Figure 5.

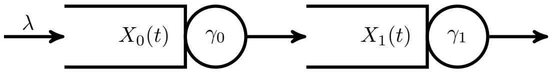

Figure 5. This block diagram displays two queues in tandem with external arrival rate λ and service rates γ_0 and $\gamma_1 = 2 - \gamma_0$, respectively.

This upper bound is only valid for $\lambda < \min_{i \in [k]} \gamma_{i-1}$. Although this bound is tight for repetition codes, it is somewhat loose for MDS coding, especially when the arrival rate λ approaches $\frac{k}{n}$. This is due to the fact that MDS coding prioritizes the service for request with $k-1$ pieces, and allocates a large service rate of $\frac{k}{n}(n-k+1)$ at that level. Contrastingly, requests with smaller number of pieces get significantly smaller service rate of $\frac{k}{n}$ when the system is congested. This bounding technique suggests that MDS coding may not perform as well as repetition coding due to unequal allocation. However, in reality, MDS coding scheme performs very well. This enhanced performance is attributable to the fact that, the extra service available to the request with $k-1$ pieces is transferred to other requests when $Y_{k-1}(t) = 0$. This mechanism is not reflected in the derivation of the upper bound. We use it as the basis for the following approximation.

Approximation 7. We can approximate the original queue with a series of uncoupled queues in tandem

$$X(t) = (X_0(t), \ldots, X_{k-1}(t)),$$

with external Poisson arrivals at rate λ and independent random service times distributed exponentially with rates $\bar{\gamma}_i$ for queue $i \in \{0, 1, \ldots, k-1\}$. We denote the equilibrium distribution of this approximate system by π. For this approximate system, the average service rate $\bar{\gamma}_i$ to queue i is its dedicated service rate γ_i plus the rate available from queue $i+1$, when it's empty. That is,

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

For each i, we can find the probability $\pi_i(0)$ of queue i being empty, and recursively compute the total service rate $\bar{\gamma}_{i-1}$ available to the queue $i-1$. This is formalized in the following theorem.

Theorem 8 (Approximation)**.** *Consider a continuous-time Markov chain $X(t) \in \mathbb{N}_0^k$ with transition matrix Q, where*

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

This Markov chain $X(t)$ is positive recurrent when

$$\lambda < \min_i \left\{ \frac{\Gamma_i}{(k-i)} \right\}.$$

The equilibrium distribution of $X(t)$ is given by

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

and the mean sojourn time in the system is equal to

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

Proof: We observe that Markov process $X(t)$ forms a series of uncoupled queues in tandem. We know that the joint distribution of the process $X(t)$ is the product of the marginal distribution of each queue $X_i(t)$. For all values of arrival rate λ such that $X(t)$ is positive recurrent, each queue i has Poisson arrivals with rate λ and independent random service times distributed exponentially with rate $\bar{\gamma}_i$. Thus, the marginal distribution π_i of queue i is

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

We can inductively find the total service rate to queue i as

$$\bar{\gamma}_i = \Gamma_i - (k-i-1)\lambda, \qquad i \in \{0, 1, \ldots, k-1\}.$$

The stability region for queue i is $\lambda < \bar{\gamma}_i$. Therefore, the stability region for the process $X(t)$ is $\lambda < \min_i \{\Gamma_i/(k-i)\}$. ■

V. Numerical Studies

We examine a $(4, 2)$ linear code for four servers and two 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. We see in Figure 6 that the analytical bounds for repetition code are quite tight, uniformly across a range of system loads. This is due to the symmetry of the service available to all partially fulfilled requests. Contrastingly, we see in Figure 7 that the upper bound is not as tight for MDS code. The lower bound also becomes loose as the load increases. Yet, the proposed approximations work well for both coding schemes.

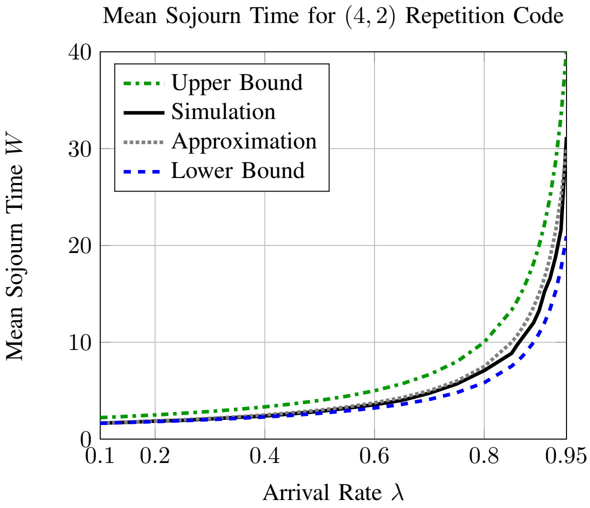

Figure 6. This plot depicts the simulated mean sojourn time W as a function of arrival rate λ for a $(4, 2)$ repetition code, along with the corresponding upper and lower bounds, and the proposed approximation.

Figure 8 displays the mean sojourn time of the requests in the actual system under repetition and MDS coding, as a function of system load. As expected, MDS coding significantly outperforms repetition coding. Consider the case, where we increase the number of servers while maintaining the code rate at $n/k = 2$ for a fixed arrival rate $\lambda = 0.3$. We can see in Figure 9 that the mean delay for MDS coding does not vary greatly with the number of servers n, while it increases for repetition code. Thus, MDS coding is more amenable to scaling than repetition coding.

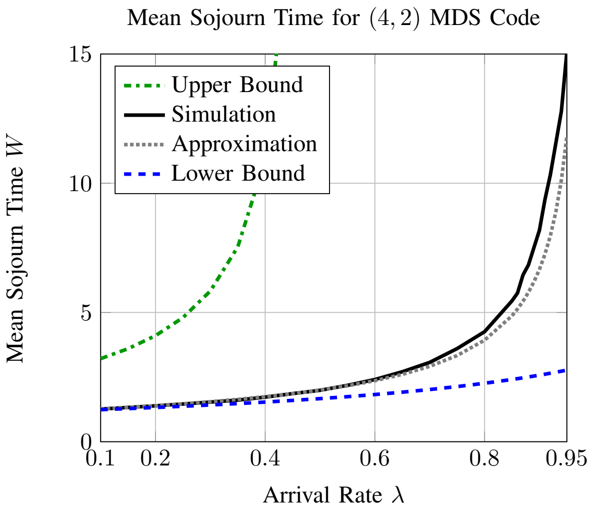

Figure 7. This plot depicts the simulated mean sojourn time W as a function of arrival rate λ for a $(4, 2)$ MDS code, along with the corresponding upper and lower bounds, and the proposed approximation.

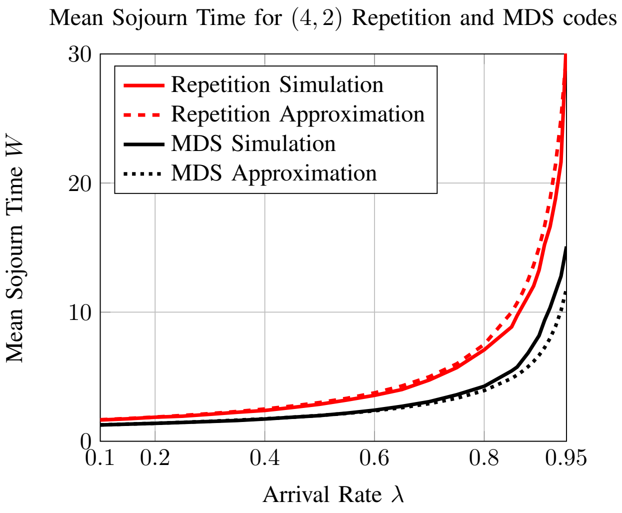

Figure 8. This figure focuses on a distributed storage system with n servers, where server i stores information symbol x_i associated with an (n, k) linear code. The plot depicts the mean delay to acquire the k coded symbols required to reconstruct the original message m under repetition and MDS coding. Approximated curves are also included for the sake of completeness.

In Figure 10, we vary k, the total length of the message, for both repetition and MDS codes. For arrival rate $\lambda = 0.45$ and $n = 24$ servers, we observe that the mean delay increases with reduced redundancy for repetition code, whereas it has a unique minimum for the MDS code. Furthermore, we see that MDS coding is robust to redundancy reduction, and this code can be utilized for efficient storage without significantly impacting the latency performance.

VI. Conclusion

We proposed an analytical framework to study latency redundancy tradeoff for homogeneous servers and two different coding schemes. We also proposed two stochastically

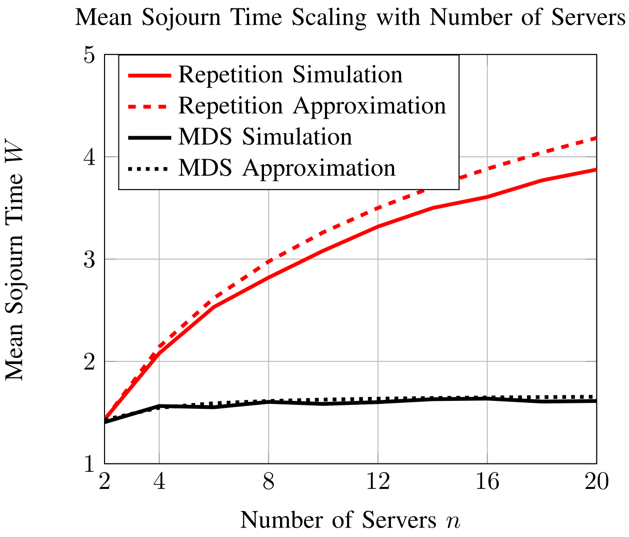

Figure 9. For a fixed arrival rate $\lambda = 0.3$ and coding overhead $\frac{n}{k} = 2$, this graph displays the plot of mean sojourn times for MDS and repetition codes as the number of servers n increases.

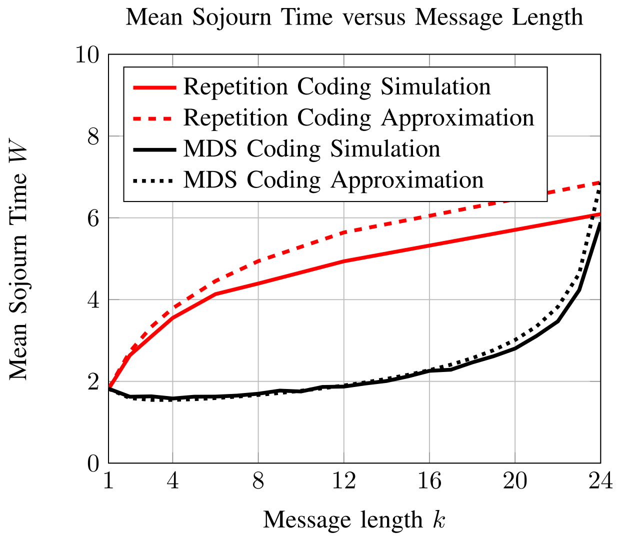

Figure 10. This graph plots mean sojourn times for MDS and repetition coding as functions of message length for rate $\lambda = 0.45$ and $n = 24$ servers.

dominating systems that bound the actual system performance. Based on these dominating systems, we provided uniform upper and lower bounds on the mean waiting time of the requests in the queue, for all system loads and both the coding schemes. We observe that the bounds are fairly tight for the repetition code, though not so tight for the MDS code at higher arrival rates. This motivated another stochastic system, that approximates the mean behavior of the actual system well. Using this approximation, we can make quantitative statements about the performance gains for various system parameters. In future work, we would like to extend this framework to a wider class of codes, and with heterogeneous servers. We would also like to explore the large deviation behavior of the actual queue system, in terms of bounding and approximating systems.

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, 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, 2001.

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

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

[6] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *48th Annual Allerton Conference on Communication, Control, and Computing*, 2010, pp. 1510–1517.

[7] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard, and M. Effros, "Resilient network coding in the presence of byzantine adversaries," *IEEE Trans. Inf. Theory*, vol. 54, no. 6, pp. 2596–2603, 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, Sept 2010.

[9] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, "Codes can reduce queueing delay in data centers," in *International Symposium on Information Theory Proceedings (ISIT)*, July 2012, pp. 2766–2770.

[10] N. B. Shah, K. Lee, and K. Ramchandran, "The MDS queue: Analysing the latency performance of erasure codes," *CoRR*, vol. abs/1211.5405, 2012.

[11] ——, "The MDS queue: Analysing the latency performance of erasure codes," in *International Symposium on Information Theory*, June 2014, pp. 861–865.

[12] ——, "When do redundant requests reduce latency?" *IEEE Trans. Commun.*, vol. 64, no. 2, pp. 715–722, Feb 2016.

[13] G. Joshi, Y. Liu, and E. Soljanin, "Coding for fast content download," in *50th Annual Allerton Conference on Communication, Control, and Computing*, Oct 2012, pp. 326–333.

[14] ——, "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 Performance Evaluation Review*, vol. 43, no. 2, pp. 54–56, Sep. 2015.

[16] D. Wang, G. Joshi, and G. Wornell, "Efficient task replication for fast response times in parallel computation," in *ACM International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS. New York, NY, USA: ACM, 2014, pp. 599–600.

[17] ——, "Using straggler replication to reduce latency in large-scale parallel computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 3, pp. 7–11, Nov. 2015.

[18] 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.

[19] R. Rojas-Cessa, L. Cai, and T. Kijkanjanarat, "Scheduling memory access on a distributed cloud storage network," in *21st Annual Wireless and Optical Communications Conference (WOCC)*, April 2012, pp. 71–76.

[20] S. Chen, Y. Sun, U. C. Kozat, L. Huang, P. Sinha, G. Liang, X. Liu, and N. B. Shroff, "When queueing meets coding: Optimal-latency data retrieving scheme in storage clouds," in *IEEE Conference on Computer Communications (INFOCOM)*, April 2014, pp. 1042–1050.

[21] L. E. Schrage and L. W. Miller, "The queue M/G/1 with the shortest remaining processing time discipline," *Operations Research*, vol. 14, no. 4, pp. 670–684, 1966.

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