Simplex Queues for Hot-Data Download

ABSTRACT

In cloud storage systems, hot data is usually replicated over multiple disks/servers in order to accommodate simultaneous access by multiple users as well as increase the fault tolerance of the system. Recent cloud storage research has proposed using availability codes, which is a special class of erasure codes, as a more storage-efficient way to store hot data. These codes enable data recovery from multiple, small disjoint groups of servers. The number of the recovery groups is referred to as the *availability* and the size of each group as the *locality* of the code. Up till now, we have very limited knowledge on how code locality and availability affect data access time. Data download from these systems involves multiple fork-join queues operating in-parallel, making the analysis of access time a very challenging problem.

In this paper, we present an analysis of average data access time in storage systems employing simplex codes, which are an important, in certain sense optimal, class of availability codes. We generalize the analysis for codes with locality 2 and any degree of availability. Specifically, using a queueing theoretic approach, we derive bounds and approximations on the average response time for two different Poisson request arrival models. We also compare two scheduling strategies for reduced access time and load balancing.

CCS CONCEPTS

•Mathematics of computing → Queueing theory; Renewal theory; Markov processes;

KEYWORDS

Availability, Download with Redundancy

ACM Reference format:

. 2017. Simplex Queues for Hot-Data Download. In *Proceedings of ACM Sigmetrics conference, Urbana-Champaign, Illinois USA, June 2017 (SIGMETRICS'17),* 17 pages.

DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In distributed computing and storage systems, data is traditionally replicated across multiple nodes[1, 8]. Compared to erasure correcting codes, replication is a storage-inefficient option to provide system reliability. However, it is simple and allows a straightforward regeneration of data stored on a failed node from one of its replicas on other nodes. Another important feature of replication is that it allows easy access to hot-data which multiple users can simultaneously download from different replicas, or each user can issue simultaneous requests to multiple replicas and wait for the first download to finish [6, 7, 15]. Data access systems arising in erasure coded storage have received a lot of attention for all-data (see for example [12, 14, 19, 24] and references therein) and some for hot-data download [16, 17], and usually involve single or multiple inter-dependent fork-join queueing systems.

Locally repairable codes (LRCs) are a recently proposed class of erasure codes that provides better storage/reliability tradeoff than repetition codes, while limiting the number of nodes that need to be accessed to repair a single failed node [9, 21]. Because of these properties, LRCs are already used in practice [11, 23]. However, very little is known about the hot data download time performance of these codes.

LRCs of interest for storage are systematic, that is, they append $n-k$ parity symbols to k data or systematic symbols. We will refer to the nodes that store systematic (data) symbols as systematic (data) nodes. If a data symbol can be repaired by accessing at most r other symbols, the symbol is said to have *locality* r. If all the code symbols have locality r, the code is said to have locality r. If the code allows repairing any data symbol by independently accessing one of t disjoint repair groups, it is said to have availability t. Availability is useful for distributed storage in two ways: 1) multiple repair groups decrease the probability that simultaneous node failures prevent the user from accessing data and 2) users may request a subset of data more frequently than others and the requests for this "hot-data" can be simultaneously served by disjoint groups of nodes. LRCs with good locality and availability are explored and several construction methods are presented in e.g., [22, 26].

Our goal is to understand the effect of LRCs with availability on time to access hot-data. We focus on two data request scheduling strategies: (1) *Replicate-to-all*: arriving requests for downloading a data symbol are forked to the systematic node and all its repair groups and (2) *Select-one*: requests are forwarded to either the systematic node or one of the repair groups. An analysis is given in [16] for these strategies for general LRCs. However, their results are mainly for low-traffic regime where download request arrival rate is low enough that the system can finish serving a download request before the next one arrives. When the low-traffic assumption does not hold, analysis involves queueing which gets intractable because of complex system dynamics. A requested data symbol is reconstructed from a repair group by accessing coded symbols from each repair node in the group. Therefore, there is a fork-join queueing sub-system for downloads from a repair group, and analysis of fork-join queues is a notoriously hard problem. Therefore, analyzing access time in systems employing availability codes is challenging for high-traffic regime.

Contribution: In this paper, we focus on availability codes with locality 2. There are two main reasons for studying these codes: 1) Codes with locality 2 include simplex codes, an important and, in several ways optimal, class of codes, e.g., they meet the upper bound on the distance of LRCs with a given locality [2] as well as the Griesmer bound and are therefore linear codes with lowest possible length given the code distance [18]. 2) Codes with locality 2 give rise to fork-join queues with two servers, which are better understood than the general fork join queueing models [20].

We consider two request arrival scenarios. In *fixed-arrival* scenario, requests that arrive in a busy period are for only one data symbol. In *mixed-arrival* scenario, requests may arrive for downloading any data symbol during a busy period. We give an analysis for the system under *replicate-to-all* scheduling strategy. Starting with the simplest possible simplex code with availability 1, we obtain close lower and upper bounds on the average access time to hot-data. Then we argue that extending these results further to codes with higher availability is hard. However, using ideas from queueing and renewal theory, we obtain lower and upper bounds, and an approximation for the average access time to hot-data. Finally, *select-one* scheduling is compared with the replicate-to-all scheduling in terms of access time.

2 SYSTEM MODEL

Data $[d_1, \dots, d_k]$ to be stored consists of elements of a finite field. A systematic $(n, k, r = 2, t)$-LRC is assumed to be used to generate coded symbols $[d_1, \dots, d_k, c_1, \dots, c_{n-k}]$ in which each d_i has t disjoint repair groups of size 2 where $t \in \mathbb{Z}^+$. Each code symbol is then stored in a separate node.

Download requests arrive for data symbols (e.g., a) rather than the complete data set $[a, b, c]$. Each server (node) can serve only one download request at a time and the arriving requests get enqueued locally. Content download at each server takes a random amount of time, which is referred to as *service time*. We refer to a request for downloading a data symbol as a *job* and the time that a job spends in the system between its arrival and its departure is referred to as the *system time*.

Two request scheduling strategies are analyzed. The first type of system *replicate-to-all* creates replicas of the arriving jobs and forwards them to the systematic server and all the repair groups. Job is completed when any of its replicas has been serviced. Within each repair group there is a fork-join sub-system that forks the incoming job replica into sibling *tasks*, which are then sent to two repair servers. To complete a job replica in a repair group, both sibling tasks must finish service. When a job replica is completed at the systematic node or at a repair group, all the remaining job replicas with their sibling tasks get canceled. Redundant job replicas are expected to reduce the average time to access data. The second type of system *select-one* forwards the arriving jobs to either the systematic node or one of the repair groups. This load-balancing behavior trades access time with resource usage efficiency. Fig. 1 illustrates these access schemes.

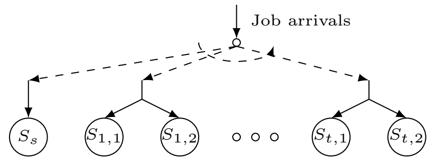

Figure 1: Arriving jobs can either be replicated to systematic server and all repair groups, replicate-to-all or be forwarded to one of them, select-one.

We make the following assumptions throughout. Arriving tasks at the servers get serviced according to First Come First Served (FCFS) scheme. Jobs arrive according to a Poisson process of known rate. Service time at each server is exponential and independent between servers and tasks. We name the resulting system as the *simplex queue* and denote it throughout as Simplex(t). The aforementioned concepts can be illustrated as follows: a Simplex($t = 3$) code encodes data $[a, b, c]$ into $[a, b, a+b, c, a+c, b+c, a+b+c]$ where each code symbol is stored on nodes $1, \dots, 7$, respectively. Each data symbol can be recovered from three disjoint repair groups, e.g., besides the systematic {node-1}, a can be recovered from {node-2, node-3}, {node-4, node-5}, {node-6, node-7}. In a *replicate-to-all* setup, a request for a is split among the four node sets and it departs whenever one of them is done. As for the select-one setup, the request for a is directed to one of these node sets only. Finally, we assume throughout that the decoding procedure for data recovery takes negligible time compared to download time.

3 EXPECTED HOT-DATA DOWNLOAD TIME

We are interested in analyzing how the availability provided by the simplex setup affects the download time of hot-data. To illustrate the complexity of the simplex queueing system, a possible system snapshot is given in Fig. 2.

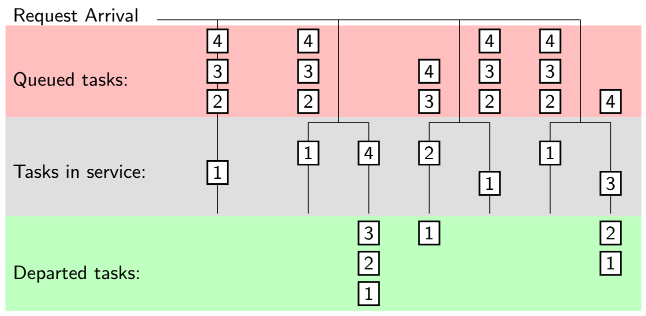

Figure 2: System snapshot of a queue corresponding to the $(7, 3)$ simplex code i.e., Simplex($t = 3$).

This queueing system, namely the Simplex($t = 3$), uses $(7, 3)$ binary simplex code. Two-server fork-join queues (FJ-2) are implemented within a repair group for reconstructing a data symbol. Steady-state behavior of FJ-2 is analyzed in [3, 4]. Using this result, [20] presents an exact expression for the average system time for FJ-2 and a very good approximation for FJ-$n > 2$.

The state space of simplex queues is complex. Not only the number of tasks waiting or in service at each server but also the order and the jobs to which the tasks belong to must be identified by the system state. Let T be the random variable denoting the time to download a data symbol (i.e., job system time). Derivation of $E[T]$ for the low-traffic regime is presented in [16]. Using these results, under low traffic regime for Simplex($t = 3$) $E[T] = \frac{\beta(2, 0.5)}{2\mu} \approx \frac{0.46}{\mu}$. In [16], an upper bound on $E[T]$ is found by using the more restrictive split-merge (SM) scheme in which all servers are blocked until the job in service is completed, thus multiple jobs cannot be in service simultaneously. The simulation results in [16] show that the upper bound suggested by SM-model is loose unless the arrival rate is low.

Analysis given until Section 7 is for the *fixed-arrivals* scenario in which servers are just for fetching and streaming data to the user while the data is stored across a pool of resources shared by the servers. Thanks to this, we can fix one as the systematic server while others as the repair servers within the fixed repair groups. Therefore, all the arriving jobs can be treated the same even though they may possibly be for downloading different symbols. Even though this reduces the practicality of the system under consideration, it greatly simplifies the analysis of the system time by significantly reducing the state space complexity. Without this assumption, besides the arrival order, every task needs to be associated with a data symbol where for different symbols different nodes must act as the systematic or repair servers. Another system level consideration which serves the same simplification is as follows. Suppose the coded symbols are both stored and served at the servers. However, only one symbol is frequently accessed in a busy period while other symbols are rarely requested. This also allows us to assume that download requests arrive for only one symbol in a busy period. As an example for how this can be the case in practice, suppose the data of interest are pieces of movies and web pages. During the day mostly web pages are expected to be requested while at night traffic for movies is expected to dominate.

In Sec. 7, results for the fixed-arrivals are shown to be upper-bounds for the *mixed-arrival* scenario. This latter scenario is more natural in that coded symbols are stored and served by the corresponding servers, and the arriving requests are mixed i.e., each may be for downloading one of the data symbols.

4 DOWNLOAD WITH AVAILABILITY ONE

As the simplest version of the simplex queue, consider Simplex(t = 1) where a data symbol can be downloaded from either the systematic server or the repair group. Let the service rates at the systematic server and two repair servers be respectively γ, α and β. We refer to the systematic server as S_s and the repair servers as $S_{1,1}$ and $S_{1,2}$. Let $\boldsymbol{N}(t) = [N_s(t), N_{1,1}(t), N_{1,2}(t)]$ be the system state where $N_*(t)$ is the number of tasks waiting or in-service in S_* and denote $Pr\{\boldsymbol{N}(t) = [k, i, j]\}$ as $p_{k,i,j}(t)$. $\boldsymbol{N}(t)$ is a Markov process where the state space consists of tuples $(k, i, j), k, i, j \geq 0$. Job arrival process is assumed to be Poisson with arrival rate λ ($Poisson(\lambda)$). Suppose that the stability is imposed and $lim_{t\to\infty} p_{k,i,j}(t) = p_{k,i,j}$. Balance equations for $\boldsymbol{N}(t)$ are,

$$\begin{aligned}&[\gamma\mathbb{1}(k \geq 1) + \alpha\mathbb{1}(i \geq 1) + \beta\mathbb{1}(j \geq 1)]p_{k,i,j} = \\ &\lambda\mathbb{1}(k \geq 1, i \geq 1, j \geq 1)p_{k-1,i-1,j-1} + \\ &\gamma p_{k+1,i+1,j+1} + (\gamma + \alpha)p_{k+1,i+1,j} + (\gamma + \beta)p_{k+1,i,j+1}.\end{aligned} \tag{1}$$

where $k, i, j \geq 0$ and $\mathbb{1}$ is the indicator function. Computing the generating function $P_{w,x,y} = \sum p_{k,i,j} w^k x^i y^j$ from the balance equations in (1) to find the exact analysis of the steady state behavior is intractable. Our approach is to look into approximating the average system time $E[T]$.

For every job in Simplex(t = 1), when the last task starts service, two situations are possible: either all of the three tasks are in service simultaneously or only two of them (at S_s, $S_{1,1}$ or S_s, $S_{1,2}$, and the third task being already delivered). We call the former job starting setup where all the tasks start service at the same time as *complete-start* while the latter one as *partial-start*. Fig. 3 shows an example where all the tasks of job 1 starts service simultaneously while one of the task of job 2 departs earlier than its siblings and the remaining tasks will start service simultaneously upon the completion of job 1. Overall, we define *job service start time* as the instant when all tasks of the job start service.

Lemma 4.1. *Simplex(t = 1) is an M/G/1 queue with job service time V. Given $E[V]$ and $E[V^2]$, the Pollaczek-Khinchin (PK) formula gives the average system time as,*

$$E[T] = E[V] + \frac{\lambda E[V^2]}{2(1 - \lambda E[V])} \tag{2}$$

$$E[V] = f_c E[V_c] + f_p E[V_p] \text{ and } E[V^2] = f_c E[V_c^2] + f_p E[V_p^2]. \tag{3}$$

where V_c and V_p represent job service time for respectively complete and partial-start, and f_i is the limiting probability that the starting setup J of an arbitrary job is $j \in \{p, c\}$.

Proof. According to the definition of job service start time given above, multiple jobs cannot be in service simultaneously. For example, while a job is in service, at most a single task of the next job can be in service. The system preserves the job order and multiple job departures at a time instant is not possible. Therefore, Simplex(t = 1) can be modeled as an M/G/1 queue.

Job service time distribution for an arbitrary job satisfies $Pr\{V \geq v\} = f_c Pr\{V_c \geq v\} + f_p Pr\{V_p \geq v\}$ where $Pr\{V_c \geq v\} = Pr\{V \geq v | J = c\}$ and $Pr\{V_p \geq v\} = Pr\{V \geq v | J = p\}$. Random variables V, V_c and V_p are non-negative, $E[V]$ and $E[V^2]$ follow from $E[V] = \int_0^\infty Pr\{V \geq v\}ds$ and $E[V^2] = \int_0^\infty 2sPr\{V \geq v\}ds$. □

In partial-start, one task starts service at S_s simultaneously with another starting service at either $S_{1,1}$ or $S_{1,2}$. For simplicity, assume $\alpha = \beta = \mu$. Completion of either of these tasks signal the job termination; $V_p = min\{Exp(\gamma), Exp(\mu)\} \sim Exp(\gamma + \mu)$. In complete-start, all tasks start service simultaneously; $V_c = min\{Exp(\gamma), max\{Exp(\mu), Exp(\mu)\}\}$, then one can find $E[V_c] = 2/(\gamma + \mu) - 1/(\gamma + 2\mu)$ and $E[V_c^2] = 4/(\gamma + \mu)^2 - 2/(\gamma + 2\mu)^2$.

Starting setup J_i of job-i depends on the state of the system $\boldsymbol{N}(a_i)$ seen by the job at arrival at time a_i. Since Poisson arrivals see time averages [28] under stability, we have $\lim_{i\to\infty} Pr\{J_i = j\} = f_j$ where f_j denotes distribution of starting setups for an arbitrary job. An exact expression for f_j is found as follows. The sub-sequence of the job arrivals that see an empty system forms a renewal process ([5], Theorem 5.5.8) and we define $R_j(t) = \mathbb{1}\{J(t) = j\}$ as a renewal-reward function where $J(t) = j$ is the event that job in service at time t made a type-j start and $\mathbb{1}$ is the indicator function.

$$\begin{aligned}f_j &= \lim_{\tau\to\infty} Pr\{R_j(\tau) = 1\} = \lim_{\tau\to\infty} E[R_j(\tau)] \\ &\overset{(a)}{=} \lim_{\tau\to\infty} \frac{1}{\tau} \int_{-\infty}^{\tau} R_j(\tau) \overset{(b)}{=} \frac{E[R_n]}{E[X]}.\end{aligned}$$

where (a) and (b) are due to the equality of the limiting time and ensemble averages of $R_j(t)$ by ([5], Theorem 5.4.5), $R_n = \int_{S_{n-1}^r}^{S_n^r} R_j(t)dt$, and S_{n-1}^r, S_n^r are the $(n-1)$th, nth renewal epochs (i.e., epochs for job arrivals that find the system empty), and X is the iid inter-renewal interval. To compute f_j, we need $E[R_n]$ and $E[X]$, finding which is a hard problem.

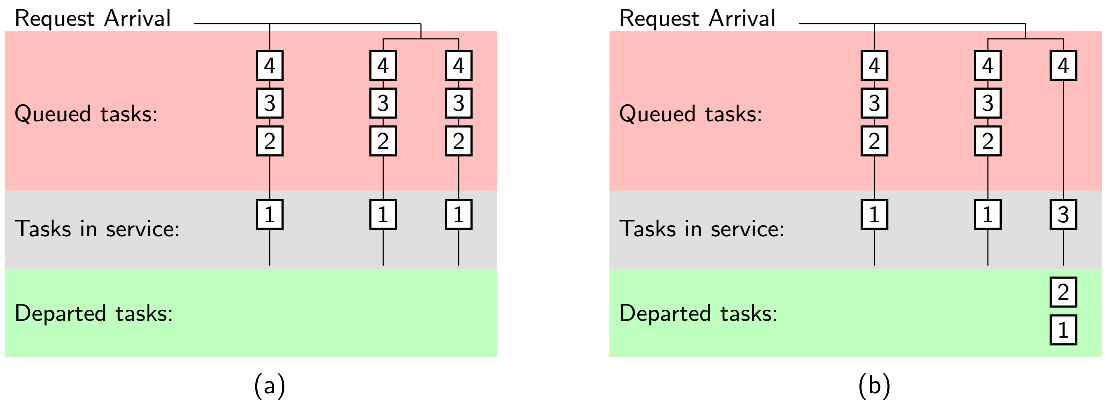

Figure 3: (a) complete-start for job-1 and (b) partial-start for job-2 upon the completion of job-1 in Simplex(t = 1).

4.1 Markov process for the system

Imagine that the synchronization between the tasks of every job is handled by a join queue at the tail of the system that queues the departures from each server. Join queue merges the incoming tasks according to their job ids. Since task departure from S_s terminates the job immediately, a job waiting for synchronization must be initiated by the first task departing from $S_{1,1}$ or $S_{1,2}$. State of the join queue can be defined as the pair $\boldsymbol{n}(t) = (n_{1,1}(t), n_{1,2}(t))$ where n_* denotes the number of jobs in the join queue that are initiated by S_*. Observe that $n_{1,1}(t)n_{1,2}(t) = 0$ holds for all t since the order of the tasks in repair servers are preserved and departure of both sibling repair tasks terminates the job. Together with the total number of jobs $N(t)$ at time t waiting or in service in the system; $(N(t), \boldsymbol{n}(t))$ defines a Markov process for Simplex(t = 1) as shown in Fig. 4. However, this markov process is tedious to analyze (see Appendix 9.2). Therefore, we estimate f_j by studying a different system approximating the actual setup which is given next.

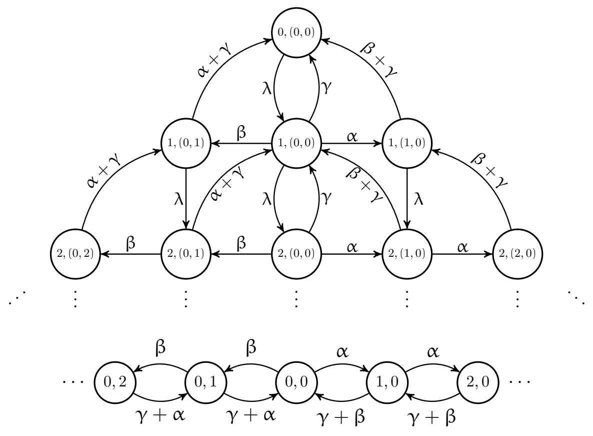

Figure 4: Markov process (Top) for Simplex(t = 1), and (Bottom) its high traffic approximation.

4.2 Analysis under high-traffic regime

Suppose λ is very close to its critical value for stability and the system is started at $-\infty$. Starting to observe the system state at $t = 0$, we assume that queues S_s, $S_{1,1}$ and $S_{1,2}$ are never empty, which is a rather crude assumption and holds only when system is unstable. We will refer to this set of working conditions as "high-traffic regime". Using the assumption that queues are never empty, state of the join queue $\boldsymbol{n}(t)$ itself is a simple Markov process shown on the bottom in Fig. 4. Note that after every transition, an exponential service time is restarted for each server and the first task completion causes the process to transit to another state. Here task cancellation at a server due to job completion is not considered as a task completion.

Steady-state balance equations for high-traffic approximation is

$$\alpha p_{i,0} = (\gamma + \beta)p_{i+1,0}, \qquad \beta p_{0,i} = (\gamma + \alpha)p_{0,i+1},\ i \geq 0. \quad (4)$$

where $lim_{t\to\infty}Pr\{\boldsymbol{n}(t) = [i,j]\} = p_{i,j}$. Solving balance equations:

$$p_{i,0} = (\frac{\alpha}{\beta+\gamma})^i p_{0,0}, \qquad p_{0,i} = (\frac{\beta}{\alpha+\gamma})^i p_{0,0},\ i \geq 1. \quad (5)$$

By the axiom of probability, $(1 + \sum_{i=1}^{\infty}(\frac{\beta}{\alpha+\gamma})^i + (\frac{\alpha}{\beta+\gamma})^i)p_{0,0} = 1$ assuming $\beta < \alpha + \gamma$ and $\alpha < \beta + \gamma$,

$$p_{0,0} = (1 + \frac{\beta}{\alpha+\gamma-\beta} + \frac{\alpha}{\beta+\gamma-\alpha})^{-1} = \frac{\gamma^2 - (\alpha-\beta)^2}{\gamma(\alpha+\beta+\gamma)}$$

from which $p_{i,0}$ and $p_{0,i}$ is found by substituting $p_{0,0}$ in (5). Other useful quantities are

$$\sum_{i=1}^{\infty} p_{i,0} = \frac{\alpha(\alpha+\gamma-\beta)}{\gamma(\alpha+\beta+\gamma)}, \qquad \sum_{i=1}^{\infty} p_{0,i} = \frac{\beta(\beta+\gamma-\alpha)}{\gamma(\alpha+\beta+\gamma)}$$

For tractable analysis, we continue with the assumption $\alpha = \beta = \mu$, then we have $p_{0,0} = \frac{\gamma}{\gamma+2\mu}$ and $\sum_{i=1}^{\infty} p_{i,0} = \sum_{i=1}^{\infty} p_{0,i} = \frac{\mu}{\gamma+2\mu}$. Next, we use the steady-state probabilities for the high-traffic assumption to obtain estimates for the quantities of interest.

4.3 Winning frequencies

We define the fraction of the jobs terminated by a server as the "winning frequency" of that server where a server wins when it terminates a job. Theorem 4.2 gives bounds on the winning frequencies for servers in Simplex(t = 1).

Theorem 4.2. *In Simplex(t = 1), let w_s, w_r be respectively winning frequencies of the systematic server with Exp(γ) service time and identical repair servers with Exp(μ) service time. Then bounds* *on winning frequencies are given as*

$$w_s \geq \frac{\gamma\nu}{\gamma\nu + 2\mu^2},\ w_r \leq \frac{\mu^2}{\gamma\nu + 2\mu^2},\ \textit{where } \nu = \gamma + 2\mu.$$

Proof. The fraction of the jobs terminated by a server can be computed from the steady-state probabilities of the Markov chain (MC) that is embedded in the process for high-traffic approximation. System state is $\boldsymbol{n}(t) = (n_{1,1}(t), n_{1,2}(t))$ where n_* denotes the number of jobs waiting in the join queue that are initiated by S_*. We stay in each state an exponential amount of time with rate $\nu = \gamma + 2\mu$. Therefore, steady-state probabilities of $\boldsymbol{n}(t)$ (p_i: limiting fraction of the time the process spends in state i) and the embedded chain (π_i: limiting fraction of the state transitions into state i) are equal as seen by $\pi_i = \frac{p_i \nu}{\nu \sum_i p_i \nu} = p_i$.

Limiting fraction of state transitions $f_s, f_{1,1}, f_{1,2}$ that represent job terminations by the respective servers $S_s, S_{1,1}, S_{1,2}$ are

$$f_s = \pi_{0,0}\frac{\gamma}{\nu} + \sum_{i=1}^{\infty}\pi_{i,0}\frac{\gamma}{\nu} + \sum_{i=1}^{\infty}\pi_{0,i}\frac{\gamma}{\nu} = \frac{\gamma}{\nu},$$

$$f_{1,1} = \sum_{i=1}^{\infty}\pi_{0,i}\frac{\alpha}{\nu} = (\frac{\mu}{\nu})^2, \qquad f_{1,2} = \sum_{i=1}^{\infty}\pi_{i,0}\frac{\beta}{\nu} = (\frac{\mu}{\nu})^2.$$

Limiting fraction of state transitions that represent job departures is $f_{jd} = f_s + f_{1,1} + f_{1,2} = \frac{\gamma\nu + 2\mu^2}{\nu^2}$. Winning frequencies for high-traffic approximation as

$$\hat{w}_s = \frac{f_s}{f_{jd}} = \frac{\gamma\nu}{\gamma\nu + 2\mu^2}, \quad \hat{w}_r = \frac{f_{1,1}}{f_{jd}} = \frac{f_{1,2}}{f_{jd}} = \frac{\mu^2}{\gamma\nu + 2\mu^2}.$$

Queues are never empty under high-traffic while fraction of the time repair servers are idle is non-zero under stability. Therefore, winning frequencies of repair servers are smaller under stability than they would be under high-traffic, so $w_s \geq \hat{w}_s$, $w_r \leq \hat{w}_r$. □

Fig. 5 shows simulated winning frequencies for $\gamma = \mu$. As λ increases, high-traffic assumption becomes more accurate and the simulated values converge to values $\hat{w}_s = 0.6$, $\hat{w}_r = 0.2$.

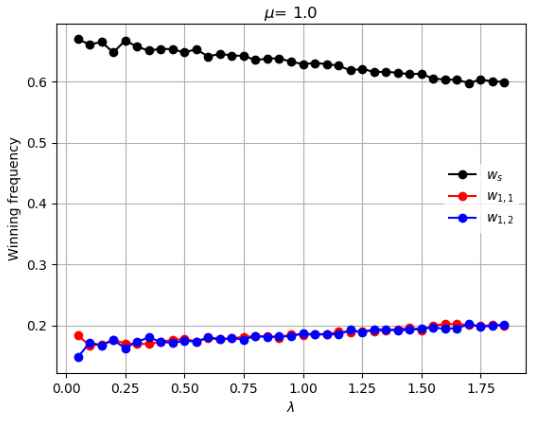

Figure 5: Simulated winning frequencies of the servers in Simplex(t = 1).

4.4 Average system time

Lemma 4.3. *Average system time $E[\hat{T}]$ under high-traffic is a lower-bound on the average system time $E[T]$ for Simplex($t = 1$).*

Proof Sketch. Comparing the two Markov processes in Fig. 4, one can see that high traffic approximation can be obtained from the actual one as follows: (1) Introduce additional transitions of rate α from state $(i, (i, 0))$ to $(i+1, (i+1, 0))$ and transitions of rate β from state $(i, (0, i))$ to $(i+1, (0, i+1))$ for $i \geq 0$, (2) Gather the states $(i, (m, n))$ for $i \geq 0$ into a "super state", (3) Observe that the process with these super-states is the same as the process for high-traffic regime. Therefore, high-traffic assumption has the effect of placing extra state transitions for the system to serve jobs faster. □

Theorem 4.4. *In Simplex($t = 1$), let V be job service time. Lower-bounds on the first and second moments of V are,*

$$\begin{aligned} E[V] \geq E[\hat{V}] &= \frac{\gamma\nu}{\gamma\nu + 2\mu^2}(\frac{2}{\gamma + \mu} - \frac{1}{\gamma + 2\mu}) + \frac{2\mu^2}{\gamma\nu + 2\mu^2}\frac{1}{\gamma + \mu}, \\ E[V^2] \geq E[\hat{V}^2] &= \frac{\gamma\nu}{\gamma\nu + 2\mu^2}(\frac{4}{(\gamma + \mu)^2} \\ &- \frac{2}{(\gamma + 2\mu)^2}) + \frac{2\mu^2}{\gamma\nu + 2\mu^2}\frac{2}{(\gamma + \mu)^2}. \end{aligned} \tag{6}$$

Then a lower-bound on the average system time $E[T]$ is,

$$E[T] \geq E[\hat{T}_{LB}] = E[\hat{V}] + \frac{\lambda E[\hat{V}^2]}{2(1 - \lambda E[\hat{V}])} \tag{7}$$

Proof. Define f_{jd} as the fraction of state transitions representing job departures and $f_{\rightarrow c}$ as the fraction of state transitions into a complete-start for the subsequent job. Since the system is always busy, a complete-start occurs for every transition into state $(0, 0)$ while partial-start occurs for every transition into $(i, 0)$ and $(0, i)$ for $i \geq 1$. Then, given $\nu = \gamma + 2\mu$, limiting fraction of the jobs that make partial $\hat{f}_p$ or complete-start $\hat{f}_c$ under high-traffic approximation are found as

$$\begin{aligned} f_{jd} &= \pi_{0,0}\frac{\gamma}{\nu} + \sum_{i=1}^{\infty}(\pi_{0,i} + \pi_{i,0})\frac{\mu + \gamma}{\nu} = \frac{2\mu^2 + 2\mu\gamma + \gamma^2}{\nu^2} \\ f_{\rightarrow c} &= \pi_{0,0}\frac{\gamma}{\nu} + \pi_{1,0}\frac{\mu + \gamma}{\nu} + \pi_{0,1}\frac{\mu + \gamma}{\nu} \\ &= \pi_{0,0}(\frac{\gamma}{\nu} + \frac{2\mu}{\mu + \gamma}\frac{\mu + \gamma}{\nu}) = \pi_{0,0} = \frac{\gamma}{\gamma + 2\mu}, \\ \hat{f}_c &= \frac{f_{\rightarrow c}}{f_{jd}} = \frac{\gamma\nu}{\gamma\nu + 2\mu^2}, \qquad \hat{f}_p = 1 - \hat{f}_c = \frac{2\mu^2}{\gamma\nu + 2\mu^2}. \end{aligned}$$

Local queues at the servers are never empty under high-traffic approximation, which increases the fraction of the jobs that make partial-start. Under stability, system has to return to the initial state with no job. The job that arrives first when the system is in no-job state makes a complete-start. Therefore, stability allows the system to have such renewals and $\hat{f}_c$ obtained for high-traffic approximation is a lower-bound; $\hat{f}_c \leq f_c$ and $\hat{f}_p \geq f_p$. Substituting $\hat{f}_c, \hat{f}_p$ for f_c, f_p in (6) yields estimates $E[\hat{V}]$ and $E[\hat{V}^2]$ for $E[V]$ and $E[V^2]$. As shown in (6), these estimates are lower-bounds; $E[\hat{V}] \leq E[V]$ and $E[\hat{V}^2] \leq E[V^2]$ because $E[V_c] > E[V_p]$, $E[V_c^2] > E[V_p^2]$

and $\hat{f}_c \le f_c$. Finally, substituting $E[\hat{V}]$, $E[\hat{V}^2]$ for $E[V]$, $E[V^2]$ in (2) gives the lower-bound $E[\hat{T}_{LB}] \le E[T]$ as shown in (7). □

Fig. 6 shows that $E[\hat{T}_{LB}]$ is a close lower-bound on the simulated $E[T]$. As expected from high-traffic approximation, $E[\hat{T}_{LB}]$ is more accurate for increasing values of λ.

4.5 Service rate allocation

Now we have a close (especially for high arrival rate) estimate $E[\hat{T}_{LB}]$ for $E[T]$, which we use to argue about how to allocate limited service capacity between the systematic node and repair group. Denote the total service rate available as $C = \gamma + 2\mu$ and define $\rho = \gamma/\mu$. In Appendix 9.1, we show that $\frac{\partial E[\hat{T}_{LB}]}{\partial \rho} < 0$ holds under stability, which suggests to allocate all of the available service rate to the systematic node to achieve the minimum system time.

4.6 Matrix analytic solution

Here we find an upper-bound on $E[T]$ by analysing a different system approximating Simplex($t = 1$). We truncate the infinite dimensional Markov chain in Fig. 4 such that the pyramid is limited to only five central columns. This means we reduce the process $(N(t), \boldsymbol{n}(t))$ to $(N(t), \hat{\boldsymbol{n}}(t)) = (N(t), (\hat{n}_{1,1}(t), \hat{n}_{1,2}(t)))$, with $\hat{n}_{1,1}(t) \le 2$ and $\hat{n}_{1,1}(t) \le 2$. This choice is motivated by the simulation results and the analysis in Appendix 9.2, which show that the most visited states are located at the central columns.

4.6.1 Computing the limiting probabilities. Finding a closed form solution to steady-state probabilities for the truncated chain is as challenging as the original problem. However one can solve the truncated Markov chain numerically with an arbitrarily small error using the "Matrix Analytics" method described in ([10], chapter 21). In the following, we denote the vectors and matrices in bold font. Start by defining the limiting probability vector

$$\begin{aligned}\boldsymbol{\pi} =& [\pi_{0,(0,0)}, \pi_{1,(0,1)}, \pi_{1,(0,0)}, \pi_{1,(1,0)},\\ & \pi_{2,(0,2)}, \pi_{2,(0,1)}, \pi_{2,(0,0)}, \pi_{2,(0,1)}, \pi_{2,(2,0)},\\ & \pi_{3,(0,2)}, \pi_{3,(0,1)}, \pi_{3,(0,0)}, \pi_{3,(1,0)}, \pi_{3,(2,0)}, \cdots]\\ =& [\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{\pi}_3, \boldsymbol{\pi}_4, \cdots].\end{aligned}$$

where $\pi_{k,(i,j)}$ is the limiting steady-state probability for state k, (i,j) and

$$\begin{aligned}\boldsymbol{\pi}_0 &= [\pi_{0,(0,0)}, \pi_{1,(0,1)}, \pi_{1,(0,0)}, \pi_{1,(1,0)}],\\ \boldsymbol{\pi}_i &= [\pi_{i+1,(0,2)}, \pi_{i+1,(0,1)}, \pi_{i+1,(0,0)}, \pi_{i+1,(1,0)}, \pi_{i+1,(2,0)}], \quad i \ge 1.\end{aligned}$$

One can write the balance equations governing the limiting probabilities in the form below

$$\boldsymbol{\pi Q} = \boldsymbol{0} \tag{8}$$

For the truncated Markov chain, $\boldsymbol{Q}$ has the following form

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{F}_0 & \boldsymbol{H}_0 & & & & \\ \boldsymbol{L}_0 & \boldsymbol{F} & \boldsymbol{H} & & \boldsymbol{0} & \\ & \boldsymbol{L} & \boldsymbol{F} & \boldsymbol{H} & & \\ & & \boldsymbol{L} & \boldsymbol{F} & \boldsymbol{H} & \\ & \boldsymbol{0} & & \boldsymbol{L} & \boldsymbol{F} & \boldsymbol{H} \\ & & & & \ddots & \ddots \end{bmatrix}, \tag{9}$$

where the sub-matrices $\boldsymbol{F}_0$, $\boldsymbol{H}_0$, $\boldsymbol{L}_0$, $\boldsymbol{F}$, $\boldsymbol{L}$ and $\boldsymbol{H}$ are given in Appendix 9.3 in terms of α, β, γ and λ. Using (8) and (9) we get the following system of equations in matrix form,

$$\begin{aligned}&\boldsymbol{\pi}_0\boldsymbol{F}_0 + \boldsymbol{\pi}_1\boldsymbol{L}_0 = 0\\ &\boldsymbol{\pi}_0\boldsymbol{H}_0 + \boldsymbol{\pi}_1\boldsymbol{F} + \boldsymbol{\pi}_2\boldsymbol{L} = 0\\ &\boldsymbol{\pi}_i\boldsymbol{H} + \boldsymbol{\pi}_{i+1}\boldsymbol{F} + \boldsymbol{\pi}_{i+2}\boldsymbol{L} = 0, i \ge 1.\end{aligned} \tag{10}$$

In order to solve the system above, we assume the steady-state probability vectors to be of the form,

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_1\boldsymbol{R}^{i-1}, \quad i \ge 1. \tag{11}$$

where $\boldsymbol{R} \in R^{5\times5}$. Combining (10) and (11) we get the following:

$$\begin{aligned}&\boldsymbol{\pi}_0\boldsymbol{F}_0 + \boldsymbol{\pi}_1\boldsymbol{L}_0 = 0,\\ &\boldsymbol{\pi}_0\boldsymbol{H}_0 + \boldsymbol{\pi}_1(\boldsymbol{F} + \boldsymbol{RL}) = 0,\\ &\boldsymbol{\pi}_i(\boldsymbol{H} + \boldsymbol{RF} + \boldsymbol{R}^2\boldsymbol{L}) = 0, \quad i \ge 1.\end{aligned} \tag{12}$$

From (12) we have common conditions for the system to hold:

$$\boldsymbol{H} + \boldsymbol{RF} + \boldsymbol{R}^2\boldsymbol{L} = \boldsymbol{0}, \qquad \boldsymbol{R} = -\left(\boldsymbol{R}^2\boldsymbol{L} + \boldsymbol{H}\right)\boldsymbol{F}^{-1} \tag{13}$$

The inverse of $\boldsymbol{F}$ in (13) exists since $\det(\boldsymbol{F}) = -\delta^3(\delta-\alpha)(\delta-\beta) \ne 0$ assuming $\delta = \alpha + \beta + \gamma + \lambda$ and $\lambda > 0$. Using (13), an iterative algorithm to compute $\boldsymbol{R}$ is given in Algorithm 1. The norm $||\boldsymbol{R}_i - \boldsymbol{R}_{i-1}||$ corresponds to the absolute value of the largest element of the difference matrix $\boldsymbol{R}_i - \boldsymbol{R}_{i-1}$. Therefore, the algorithm terminates when the largest difference between the elements of the last two computed matrices is smaller than the threshold ϵ. The initial matrix $\boldsymbol{R}_0$ could take any value, not necessarily $\boldsymbol{0}$. The error threshold ϵ could be fixed to any arbitrary value, but the lower this value the slower the convergence. Computing $\boldsymbol{R}$, the vectors $\boldsymbol{\pi}_0$

Algorithm 1 Computing matrix $\boldsymbol{R}$

```
1: procedure ComputingR
2:     $\epsilon \leftarrow 10^{-6}$, $\boldsymbol{R}_0 \leftarrow \boldsymbol{0}$, $i \leftarrow 1$
3:     while true do
4:         $\boldsymbol{R}_i \leftarrow -\left(\boldsymbol{R}_{i-1}^2\boldsymbol{L} + \boldsymbol{H}\right)\boldsymbol{F}^{-1}$
5:         if $||\boldsymbol{R}_i - \boldsymbol{R}_{i-1}|| > \epsilon$ then
6:             $i \leftarrow i + 1$
7:         else return $\boldsymbol{R}_i$
```

and $\boldsymbol{\pi}_1$ are remaining to be found in order to deduce the values of all limiting probabilities. Recall that in (12) the first two equations are yet to be used. Writing these two equations in matrix form,

$$\begin{bmatrix} \boldsymbol{\pi}_0 & \boldsymbol{\pi}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{F}_0 & \boldsymbol{H}_0 \\ \boldsymbol{L}_0 & \boldsymbol{RL} + \boldsymbol{F} \end{bmatrix} = \boldsymbol{0}, \tag{14}$$

where $\boldsymbol{0}$ is a 1×9 zeros vector and $\Phi = \begin{bmatrix} \boldsymbol{F}_0 & \boldsymbol{H}_0 \\ \boldsymbol{L}_0 & \boldsymbol{RL} + \boldsymbol{F} \end{bmatrix} \in R^{9\times9}$. In addition, we have the normalization equation to take into account.

Denoting $\mathbf{1}_0 = [1,1,1,1]$, $\mathbf{1}_1 = [1,1,1,1,1]$, by (11), we get

$$\begin{aligned}\boldsymbol{\pi}_0\mathbf{1}_0^T + \sum_{i=1}^{\infty}\boldsymbol{\pi}_i\mathbf{1}_1^T &= 1\\ \boldsymbol{\pi}_0\mathbf{1}_0^T + \sum_{i=1}^{\infty}\boldsymbol{\pi}_1\boldsymbol{R}^{i-1}\mathbf{1}_1^T &= 1\\ \boldsymbol{\pi}_0\mathbf{1}_0^T + \boldsymbol{\pi}_1(\boldsymbol{I}-\boldsymbol{R})^{-1}\mathbf{1}_1^T &= 1\\ \begin{bmatrix}\boldsymbol{\pi}_0 & \boldsymbol{\pi}_1\end{bmatrix}\begin{bmatrix}\mathbf{1}_0^T\\ (\boldsymbol{I}-\boldsymbol{R})^{-1}.\mathbf{1}_1^T\end{bmatrix} &= 1,\end{aligned} \tag{15}$$

where $\boldsymbol{I}$ is the 5×5 identity matrix. In order to find $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$, we solve the following system

$$\begin{bmatrix}\boldsymbol{\pi}_0 & \boldsymbol{\pi}_1\end{bmatrix}\Psi = [1,0,0,0,0,0,0,0,0], \tag{16}$$

where Ψ is obtained by replacing the first column of Φ with $[\mathbf{1}_0, \mathbf{1}_1(\boldsymbol{I}-\boldsymbol{R}^T)^{-1}]^T$. Hence, (16) is a linear system of 9 equations with 9 unknowns. After solving (16), we obtain the remaining limiting probabilities vector using (11).

4.6.2 Approximating the average system time $E[T]$. We first notice that

$$\begin{aligned}Pr\{N=0\} &= \pi_{0,(0,0)},\\ Pr\{N=1\} &= \pi_{1,(0,0)} + \pi_{1,(0,1)} + \pi_{1,(1,0)} = \boldsymbol{\pi}_0\mathbf{1}_0^T - \pi_{0,(0,0)},\\ Pr\{N=i\} &= \pi_{i,(0,2)} + \pi_{i,(0,1)} + \pi_{i,(0,0)} + \pi_{i,(1,0)} + \pi_{i,(2,0)}\\ &= \boldsymbol{\pi}_{i-1}\mathbf{1}_1^T, \quad i \geq 2.\end{aligned}$$

Then, the average number of jobs $E[\hat{N}_{MA}]$ in the truncated system is computed as

$$\begin{aligned}E[\hat{N}_{MA}] &= \textstyle\sum_{i=0}^{\infty} iPr\{N_{MA}=i\}\\ &= \boldsymbol{\pi}_0\mathbf{1}_0^T - \pi_{0,(0,0)} + \sum_{i=2}^{\infty} i(\boldsymbol{\pi}_{i-1}\mathbf{1}_1^T)\\ &= \boldsymbol{\pi}_0\mathbf{1}_0^T - \pi_{0,(0,0)} + \sum_{i=2}^{\infty} i(\boldsymbol{\pi}_1\boldsymbol{R}^{i-2}\mathbf{1}_1^T)\\ &= \boldsymbol{\pi}_0\mathbf{1}_0^T - \pi_{0,(0,0)} + \boldsymbol{\pi}_1(\sum_{i=2}^{\infty}(i-1)\boldsymbol{R}^{i-2} + \boldsymbol{R}^{i-2})\mathbf{1}_1^T\\ &= \boldsymbol{\pi}_0\mathbf{1}_0^T - \pi_{0,(0,0)} + \boldsymbol{\pi}_1(\sum_{j=1}^{\infty} j\boldsymbol{R}^{j-1} + \sum_{i=0}^{\infty}\boldsymbol{R}^{i})\mathbf{1}_1^T\\ &= \boldsymbol{\pi}_0\mathbf{1}_0^T - \pi_{0,(0,0)} + \boldsymbol{\pi}_1((\boldsymbol{I}-\boldsymbol{R})^{-2} + (\boldsymbol{I}-\boldsymbol{R})^{-1})\mathbf{1}_1^T.\end{aligned} \tag{17}$$

Equation (17) shows that we only need $\boldsymbol{\pi}_0$, $\boldsymbol{\pi}_1$ and $\boldsymbol{R}$, thus no need to calculate infinite number of limiting probabilities.

THEOREM 4.5. *A strict upper-bound on the average system time of Simplex($t = 1$) is given by,*

$$E[T] < E[\hat{T}_{MA}] = \frac{E[\hat{N}_{MA}]}{\lambda}. \tag{18}$$

where $E[\hat{N}_{MA}]$ is given in (17).

PROOF. Markov chain that is analysed is obtained by truncating the actual chain for Simplex($t = 1$). It is equivalent to imposing a blocking on the repair group whenever one of the repair server leads by 2 tasks, which works slower than the simplex queue. Therefore, average system time found for the truncated system is an upper bound on $E[T]$ and Little's law gives (18). □

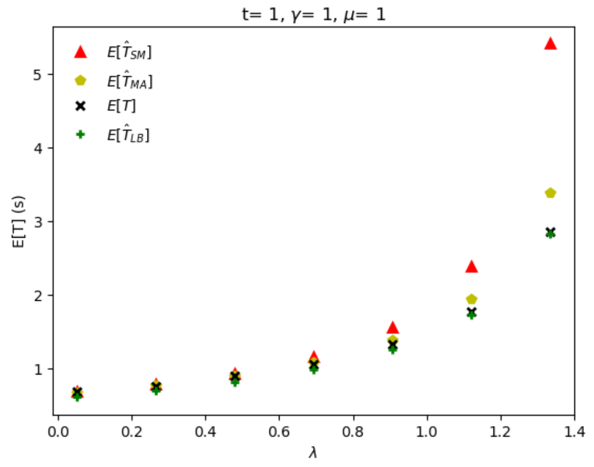

Figure 6: Split-merge upper bound $E[\hat{T}_{SM}]$, upper-bound $E[\hat{T}_{MA}]$ obtained by the matrix analytic analysis on the truncated MC, simulated $E[T]$ and the M/G/1 lower-bound $E[\hat{T}_{LB}]$ for the average system time in Simplex($t = 1$).

Fig. 6 shows that $E[\hat{T}_{MA}]$ is a close upper-bound on the simulated $E[T]$. This is due to the fact that truncation keeps the mostly visited system states and gives a good approximation.

5 DOWNLOAD WITH ANY AVAILABILITY

We start by generalizing some of the ideas developed for Simplex($t = 1$). Suppose that the service time at systematic server is $Exp(\gamma)$ and at each repair server it is $Exp(\mu)$. We index all possible service start types (instead of calling them complete or partial) starting from 0 up to t where type-i setup means that at the job starting instant, only one task of the job is in service at i repair groups while in the remaining $t-i$ repair groups both sibling tasks are in service. Given this definition, complete-start is equivalent to type-0 start. First and second moments $E[V_i]$, $E[V_i^2]$ for the job service completion time for type-i starting setup can be calculated as follows,

$$\begin{aligned}Pr\{V_i \geq s\} &= Pr\{E_\gamma \geq s\}Pr\{E_\mu \geq s\}^i Pr\{\max\{E_\mu, E_\mu\} \geq s\}^{t-i}\\ &= e^{-\gamma s}e^{-i\mu s}(1-(1-e^{-\mu s})^2)^{t-i}\\ &= e^{-(\gamma+t\mu)s}(2-e^{-\mu s})^{t-i}\\ &= e^{-(\gamma+t\mu)s}\sum_{k=0}^{t-i}\binom{t-i}{k}2^k(-e^{-\mu s})^{t-i-k}\\ &= \sum_{k=0}^{t-i}\binom{t-i}{k}2^k(-1)^{t-i-k}e^{-(\gamma+\mu(2t-i-k))s}\end{aligned}$$

$$
\begin{aligned}
E[V_i] &= \int_0^\infty Pr\{V_i \geq s\} ds \\
&= \sum_{k=0}^{t-i} \binom{t-i}{k} 2^k (-1)^{t-i-k} \frac{1}{\gamma + \mu(2t-i-k)} \\
E[V_i^2] &= \int_0^\infty 2s Pr\{V_i \geq s\} ds \\
&= \sum_{k=0}^{t-i} \binom{t-i}{k} 2^k (-1)^{t-i-k} \frac{2}{(\gamma + \mu(2t-i-k))^2}
\end{aligned}
\tag{19}
$$

where E_γ and E_μ denote Exponential service time respectively at the systematic server and each repair server.

Intuitively, $E[V_i]$ should decrease with i because type-i start means that at i repair groups, one sibling task already finished service. That is to terminate the job, only one task is left in these leading repair groups which is better than having to wait for both sibling tasks. In the following, we establish that $E[V_i]$ monotonically decreases with i.

$$
\begin{aligned}
E[V_i - V_{i+1}] &= \int_0^\infty (Pr\{V_i \geq s\} - Pr\{V_{i+1} \geq s\}) ds \\
= \int_0^\infty & e^{-(\gamma + t\mu)s} ((2 - e^{-\mu s})^{t-i} - (2 - e^{-\mu s})^{t-i-1}) ds > 0
\end{aligned}
\tag{20}
$$

where $(2 - e^{-\mu s})^{t-i} - (2 - e^{-\mu s})^{t-i-1} > 0$ since $2 - e^{-\mu s} > 1$. Same calculations can be carried out to show that $E[V_i^2]$ monotonically decreases with i.

Lemmas 4.1 and 4.3 generalize for simplex queue with any degree of availability as follows.

Lemma 5.1. *Simplex(t) is an M/G/1 queue with job service time V. Given $E[V]$ and $E[V^2]$, PK formula gives the average system time. Moments of V are,*

$$
E[V] = \sum_{j=0}^{t} f_j E[V_j], \qquad E[V^2] = \sum_{j=0}^{t} f_j E[V_j^2]. \tag{21}
$$

where $E[V_j]$ and $E[V_j^2]$ are given in (19)*, and f_j is the limiting probability that an arbitrary job makes type-j service start.*

Proof. According to our definition of job start, a job starts service the first time when all its tasks are in service. This guarantees that only one job can be in service at a time and multiple jobs cannot depart at the same time. In addition, jobs depart in the order they arrive. Service time of a job V depends on its service start setup J. We can write $Pr\{V \geq v\} = \sum_{j=0}^{t} f_j Pr\{V_i \geq v\}$ where $Pr\{V_i \geq v\} = Pr\{V \geq v | J = i\}$. Random variables V_i are non-negative, $E[V]$ and $E[V^2]$ $E[V] = \int_0^\infty Pr\{V \geq v\} ds$ and $E[V^2] = \int_0^\infty 2s Pr\{V \geq v\} ds$. Therefore, Simplex($t$) can be modeled as an M/G/1 queue. □

Lemma 5.2. *Average system time $E[\hat{T}]$ under high-traffic assumption is a lower-bound on the average system time $E[T]$ for Simplex(t).*

Proof. Local queues at the servers are never empty under high-traffic approximation while system has to visit the initial state with no-jobs many times under stability. While there is enough jobs in the system to keep all the servers busy under stability, high-traffic approximation becomes exact. We know by (20) that the more tasks are completed earlier at the leading repair servers, the faster the jobs get served on average. Therefore, system under stability can be thought as the blocking version of high-traffic approximation, which completes the proof. □

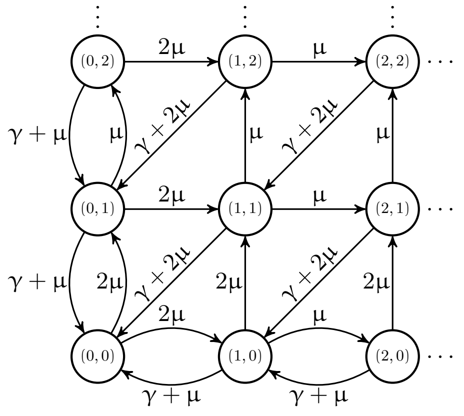

Figure 7: Markov process for Simplex(t = 2) under high-traffic.

5.1 Simplex Queue for Availability Two

In this section, we extend the ideas developed in Section 4 and find close estimates for average system time $E[T]$ for Simplex($t = 2$).

Using Lemma 5.1 for $t = 2$, we get

$$
\begin{aligned}
E[V] &= f_0 E[V_0] + f_1 E[V_1] + f_2 E[V_2] \\
E[V^2] &= f_0 E[V_0^2] + f_1 E[V_1^2] + f_2 E[V_2^2]
\end{aligned}
\tag{22}
$$

where $E[V_j]$ and $E[V_j^2]$ for $j \in \{0, 1, 2\}$ follow from (19).

We next estimate f_j under high-traffic. Consider a join queue at the tail of the system, in which the tasks that finish service wait for their siblings. In each repair group, only one server can be ahead with the task completion. Since the repair servers are assumed identical and there is no need to distinguish the leading server in a repair group, state of the join queue can be defined as $\boldsymbol{n}(t) = (n_1(t), n_2(t))$ where n_i represents the number of tasks waiting in the join queue at time t that departed from the leading server at repair group i. Under high-traffic approximation, $\boldsymbol{n}(t)$ is a Markov process shown in Fig. 7.

Embedded MC within $\boldsymbol{n}(t)$ allows us to find estimates $\hat{f}_0$, $\hat{f}_1$ and $\hat{f}_2$. Lemma 5.2 implies that $\hat{f}_0 < f_0$ and consequently $\sum_{j=1}^{t} \hat{f}_j > \sum_{j=1}^{t} f_j$ given (20). Using this fact and Lemma 5.1, we get lower-bounds $E[\hat{V}]$, $E[\hat{V}^2]$ on $E[V]$, $E[V^2]$ as shown in (23) by substituting $\hat{f}_0$, $\hat{f}_1$ and $\hat{f}_2$ for f_0, f_1 and f_2.

$$
\begin{aligned}
E[V] &\geq E[\hat{V}] = \hat{f}_0 E[V_0] + \hat{f}_1 E[V_1] + \hat{f}_2 E[V_2], \\
E[V^2] &\geq E[\hat{V}^2] = \hat{f}_0 E[V_0^2] + \hat{f}_1 E[V_1^2] + \hat{f}_2 E[V_2^2].
\end{aligned}
\tag{23}
$$

Finally, substituting $E[\hat{V}]$, $E[\hat{V}^2]$ for $E[V]$, $E[V^2]$ in PK formula gives a lower-bound on $E[T]$.

Steady-state probabilities $p_{i,j}$ for $\boldsymbol{n}(t)$ and those $\pi_{i,j}$ for the embedded MC are equal since the total transition rate ν out of every state in $\boldsymbol{n}(t)$ is equal to $\gamma + 4\mu$. Based on the same method for Simplex(t) with availability 1, one can compute the estimates $\hat{f}_0$, $\hat{f}_1$ and $\hat{f}_2$ by using the limiting fraction of state transitions for job departures f_{jd}, and those into type-0, type-1 and type-2 starting states $f_{\to 0}$, $f_{\to 1}$ and $f_{\to 2}$ for the subsequent job as

$$\begin{aligned}
f_{jd} &= \frac{\gamma}{\nu}\pi_{0,0} + \frac{\gamma+\mu}{\nu}\sum_{i=1}^{\infty}(\pi_{0,i}+\pi_{i,0}) \\
&\quad + \frac{\gamma+2\mu}{\nu}\Big(1-\pi_{0,0}-\sum_{i=1}^{\infty}(\pi_{0,i}+\pi_{i,0})\Big), \\
f_{\to 0} &= \frac{\gamma}{\nu}\pi_{0,0} + \frac{\gamma+\mu}{\nu}(\pi_{0,1}+\pi_{1,0}) + \frac{\gamma+2\mu}{\nu}\pi_{1,1}, \qquad (24)\\
f_{\to 1} &= \frac{\gamma+2\mu}{\nu}\sum_{i=2}^{\infty}(\pi_{i,2}+\pi_{2,i}), \\
\hat{f}_0 &= \frac{f_{\to 0}}{f_{jd}}, \quad \hat{f}_1 = \frac{f_{\to 1}}{f_{jd}}, \quad \hat{f}_2 = 1-\hat{f}_0-\hat{f}_1.
\end{aligned}$$

However (24) requires exact analysis of $\boldsymbol{n}(t)$, which is hard [1]. It is irreducible and for recurrence, $p_{i,j}$ should decrease with i and j. One can see this from the simulation or from the following informal argument. Expected drift at $(i, j),\ i, j > 0$ (i.e., inner states) towards $(i-1, j-1)$ is greater than the drift towards $(i+1, j)$ and $(i, j+1)$. Expected drift at $(i, 0),\ i > 0$ (i.e., states at the horizontal boundary) towards $(i-1, 0)$ and $(i, 1)$ is greater than the drift towards $(i+1, 0)$. Similar observation can be made for the states at the vertical boundary. Therefore, we expect the system to spend more time in the lower-left region of the state-space, which gives us the idea of doing the analysis by truncating the process. To keep the transition probabilities within the embedded MC same, we define the truncation as follows. Suppose the transition probability matrix of $\boldsymbol{n}(t)$ is $\boldsymbol{P}$ and the truncated process is $\tilde{\boldsymbol{n}}(t) = \{(i, j) | i, j \leq M\}$ where M defines the boundary at which the truncation starts. Then the transition probability matrix $\tilde{\boldsymbol{P}}$ of $\tilde{\boldsymbol{n}}(t)$ is defined as

$$\tilde{\boldsymbol{P}}_{i,j} = \begin{cases} 0, & j > M, i \leq M, \\ \sum_{j \geq M} \boldsymbol{P}_{i,j}, & j = i \leq M, \\ \boldsymbol{P}_{i,j}, & j \leq M, i \leq M. \end{cases} \qquad (25)$$

Substituting $\tilde{\pi}_{i,j}$'s of the MC embedded in $\tilde{\boldsymbol{n}}(t)$ for $\pi_{i,j}$ in (24) gives estimates $\hat{f}_0$, $\hat{f}_1$, $\hat{f}_2$. However one important consequence of truncation is that $\tilde{\pi}_{i,j}$'s have higher values for the remaining states i.e., $\tilde{\pi}_{i,j} > \pi_{i,j},\ i, j \leq M$, which is shown in Appendix 5.1. Therefore, we observe $\hat{f}_0 > f_0$, $\hat{f}_1 > f_1$. Therefore, there is no guarantee that $E[\hat{V}]$, $E[\hat{V}^2]$ by substituting $\hat{f}_0$, $\hat{f}_1$, $\hat{f}_2$ in (23) are lower-bounds on $E[V]$, $E[V^2]$, thus substituting $E[\hat{V}]$, $E[\hat{V}^2]$ in PK formula does not always give a lower-bound.

We can solve the truncated chain [2] to compute $\hat{f}_0$, $\hat{f}_1$, $\hat{f}_2$ and get $E[\hat{T}]$ using the PK formula. Fig. 8 compares $E[\hat{T}]$ attained by setting the truncation index $M = 5$ and $M = 2$ with the simulated $E[T]$. Lower-bound obtained by $M = 5$ performs pretty good. Note that as discussed in the previous paragraph, when M is low, the model is not a lower-bound anymore. This can be seen for $M = 2$. Models for lower values of M can be used only as an approximation.

[1]Fig. 7 shows that $\boldsymbol{n}(t)$ is multi-dimensional and has infinite number of states.
[2]Solving the truncated chain analytically is tedious but steady-state probabilities can be computed easily with a computer

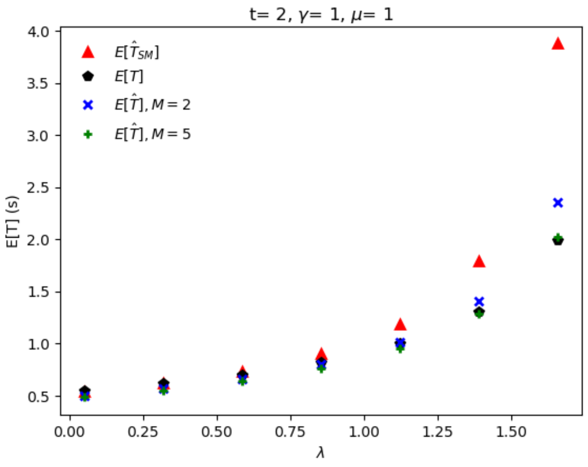

Figure 8: Split-merge upper-bound $E[\hat{T}_{SM}]$, the M/G/1 lower-bound $E[\hat{T}]$ attained by setting $M = 2$ and $M = 5$, and the simulated $E[T]$ for the average system time in Simplex($t = 2$).

5.2 Simplex Queue with any Availability

Analysis of the high-traffic approximation to find an estimate as in Sec. 4 and 5.1 is not possible for Simplex($t > 2$). State-space complexity for the join queue under high-traffic approximation exponentially increases with t and analyzing systems with higher number of repair groups becomes quickly intractable.

We previously observed for $t = 1, 2$ that the fraction of the jobs that make type-i start decrease with i. This can be explained with the following informal argument. For type-i start, at the job starting time instant, system should have i repair groups with a *leading* server that already finished servicing the assigned task of the job. We call such groups "leading groups". For a repair group, it is less likely to be leading because every job termination helps the slow servers to catch up with the leading server. Remember that completion of a job allows the slow servers that are still working on the tasks of the completed job to cancel the task and proceed with the next task waiting in the queue. In simplex queue, job termination can be signaled by the systematic server or by any repair group. However, a leading server should serve faster and compete with every other server to keep leading. Overall, it is intuitively expected to have more jobs making type-i start for lower values of i.

What is informally argued in the previous paragraph is further discussed in a slightly more formal way in Appendix 9.5. Even though we could not give a complete proof, the simulations suggest the following conjecture.

Conjecture 5.3. *In Simplex(t), we have $f_j > f_{j+1}$ for $0 \leq j \leq t-1$ where f_j denotes the limiting fraction of jobs that make type-j start. In other words, $f_{j+1} = \rho_j f_j$ where $\rho_j < 1$.*

Given f_j's, exact expression for average system can be found by Lemma 5.1. However, even for $t \leq 2$, we could only find estimates for f_j's. Theorem 5.4 gives a general lower-bound on average system time $E[T]$ of Simplex(t) by using an upper-bound on f_j's.

Theorem 5.4. *For $\rho \geq \max\{\rho_j | 0 \leq j \leq t-1\}$, f_j's are bounded as*

$$f_0 > \hat{f}_0(\rho) = \frac{1-\rho^{t+1}}{1-\rho}, \quad f_i < \hat{f}_i(\rho) = \rho^j \hat{f}_0(\rho), \quad 1 \leq i \leq t. \quad (26)$$

A lower-bound on $E[T]$ for Simplex(t) is given by

$$E[T] \geq E[\hat{T}(\rho)] = E[\hat{V}(\rho)] + \frac{\lambda E[\hat{V}^2(\rho)]}{2(1-\lambda E[\hat{V}(\rho)])} \quad (27)$$

for $E[\hat{V}(\rho)] = \sum_{j=0}^{t} \hat{f}_j(\rho) E[V_j]$ and $E[\hat{V}^2(\rho)] = \sum_{j=0}^{t} \hat{f}_j(\rho) E[V_j^2]$ where $E[V_j]$ and $E[V_j^2]$ are given in (19).

Proof. Using the normalization requirement $\sum_{i=0}^{t} f_i = 1$, f_i's in terms of ρ_i's are found as,

$$\sum_{i=0}^{t} f_i = \sum_{i=0}^{t} f_0 \prod_{j=0}^{i-1} \rho_j = 1 \text{ gives,}$$
$$f_0 = \frac{1}{1+\sum_{i=1}^{t} \prod_{j=0}^{i-1} \rho_j}, \qquad f_i = f_0 \prod_{j=0}^{i-1} \rho_j. \quad (28)$$

Substituting each ρ_i with an upper-bound ρ and solving for $\hat{f}_i(\rho)$'s gives $\hat{f}_0(\rho) = \frac{1-\rho^{t+1}}{1-\rho}$, $\hat{f}_i(\rho) = \rho^i \hat{f}_0$, which preserves Conjecture 5.3. We have

$$\frac{\hat{f}_i(\rho)-f_i}{f_i} = \frac{\hat{f}_i(\rho)}{f_i} - 1 = \frac{\hat{f}_0(\rho)\rho^k}{f_0 \prod_{j=0}^{i-1} \rho_j} - 1 = \frac{\hat{f}_0(\rho)}{f_0} \prod_{j=0}^{i-1} \frac{\rho}{\rho_j} - 1,$$

where $0 \leq j \leq t$ and $\frac{\rho}{\rho_j} > 1$ so we obtain $\hat{f}_i(\rho) > f_i$ and $\frac{\hat{f}_{i+1}(\rho)-f_{i+1}}{f_{i+1}} > \frac{\hat{f}_i(\rho)-f_i}{f_i}$. Overall, estimates $E[\hat{V}(\rho)]$, $E[\hat{V}^2(\rho)]$ for $E[V]$, $E[V^2]$ found by substituting $\hat{f}_i(\rho)$'s in (21) are lower-bounds as shown

$$\begin{aligned} E[V] - E[\hat{V}(\rho)] &= \sum_{j=0}^{t} (f_i - \hat{f}_i(\rho)) E[V_i] \\ &= (f_0 - \hat{f}_0(\rho)) E[V_0] + \sum_{j=1}^{t} (f_i - \hat{f}_i(\rho)) E[V_i] \\ &\overset{(a)}{=} \sum_{j=1}^{t} (\hat{f}_i(\rho) - f_i)(E[V_0] - E[V_i]) > 0. \end{aligned}$$

where (a) follows from

$$\sum_{i=0}^{t} f_i = \sum_{i=0}^{t} \hat{f}_i(\rho) \implies f_0 - \hat{f}_0(\rho) = \sum_{i=0}^{t} (\hat{f}_i(\rho) - f_i).$$

and (20). Same analysis can be done to show $E[\hat{V}^2(\rho)] < E[V^2]$. Substituting $E[\hat{V}(\rho)]$ and $E[\hat{V}^2(\rho)]$ in PK formula gives (27). □

The tighter the bound ρ on ρ_i's is, the better the estimates $E[\hat{V}(\rho)]$ and $E[\hat{V^2}(\rho)]$ are, so $E[\hat{T}(\rho)]$ is. The naive way is to simply set ρ to 1. Then $\hat{f}_i(1)$'s follow a uniform distribution and becomes equal to $1/(t+1)$. Then estimates become $E[\hat{V}(1)] = \sum_{i=0}^{t} E[V_i]/(t+1)$ and $E[\hat{V^2}(1)] = \sum_{i=0}^{t} E[V_i^2]/(t+1)$, which are then substituted in (27) to get the lower-bound $E[\hat{T}(1)]$. In Fig. 9, comparison between $E[\hat{T}(1)]$ and the simulated $E[T]$ is shown. Next, we improve the estimate ρ.

Corollary 5.5. *Estimate ρ in Theorem 5.4, holds the inequality,*

$$\frac{E[X]-E[\hat{V}(1)]}{E[X]} \rho^{t+1} - \rho + 1 - \frac{E[X]-E[\hat{V}(1)]}{E[X]} \geq 0 \quad (29)$$

where X is the inter-arrival interval time for job arrivals, $E[X] = 1/\lambda$, and $E[\hat{V}(1)] = \sum_{i=0}^{t} E[V_i]/(t+1)$.

Proof. See Appendix 9.6. □

Next we use inequality (29) to get an upper bound on ρ as follows. Unfortunately, solving for ρ in (29) does not yield a closed form solution, so to get one we take the limit as $\lim_{t\to\infty} \frac{E[X]-E[\hat{V}(1)]}{E[X]} \rho^{t+1} - \rho + 1 - \frac{E[X]-E[\hat{V}(1)]}{E[X]} \geq 0$, which gives $\rho \leq \hat{\rho} = E[\hat{V}(1)]/E[X]$. Note that taking the limit may make $\hat{\rho}$ not an upper-bound on ρ for small values of t. This may lead $E[\hat{T}(\hat{\rho})]$ to be not a lower-bound but only an approximation. However, as we see in Fig. 9, $E[\hat{T}(\hat{\rho})]$ is a lower-bound even for $t = 1$.

The expected relation $E[\hat{T}(1)] \leq E[\hat{T}(\hat{\rho})] \leq E[T]$ can be seen in Fig. 9. An interesting observation from these plots is that the relative gain achieved by $E[\hat{T}(\hat{\rho})]$ over $E[\hat{T}(1)]$ improves as the number of repair groups t increases.

5.2.1 An Approximation of $E[T]$ for Simplex Queues with Any Degree of Availability. Lower-bound $E[\hat{T}(1)]$ was obtained by setting the upper-bound ρ on ρ_i's to maximum value 1 in Theorem 5.4. Corollary 5.6 gives an approximate on $E[T]$ by using better upper-bounds on each ρ_i which are found incrementally. Note that, the given incremental computation aims at finding as tight bounds on ρ_i as possible but it does not guarantee that $E[\hat{T}(\hat{\boldsymbol{\rho}})]$ is a lower-bound on $E[T]$ but only an approximation. We observe in Fig. 9 that $E[\hat{T}(\hat{\boldsymbol{\rho}})]$ is almost equal to $E[T]$ for $t = 2$ and for higher values of t it serves as an upper-bound better than $E[\hat{T}_{SM}]$ that is obtained from split-merge approximation. Simulation comparison for Simplex($t = 1$) shows that the lower bounds $E[\hat{T}(1)]$, $E[\hat{T}(\hat{\rho})]$ and the approximation $E[\hat{T}(\hat{\boldsymbol{\rho}})]$ approximates the average system time for $t = 1$ almost exactly.

Corollary 5.6. *$E[T]$ for Simplex(t) is well approximated as*

$$E[T] \approx E[\hat{T}(\hat{\boldsymbol{\rho}})] = E[\hat{V}(\hat{\boldsymbol{\rho}})] + \frac{\lambda E[\hat{V}^2(\hat{\boldsymbol{\rho}})]}{2(1-\lambda E[\hat{V}(\hat{\boldsymbol{\rho}})])} \quad (30)$$

for $E[\hat{V}(\hat{\boldsymbol{\rho}})] = \sum_{j=0}^{t} \hat{f}_j(\hat{\boldsymbol{\rho}}) E[V_j]$ and $E[\hat{V}^2(\hat{\boldsymbol{\rho}})] = \sum_{j=0}^{t} \hat{f}_j(\hat{\boldsymbol{\rho}}) E[V_j^2]$ where $E[V_j]$ and $E[V_j^2]$ are given in (19)*, and*

$$\hat{f}_0(\hat{\boldsymbol{\rho}}) = \frac{1}{1+\sum_{i=1}^{t} \prod_{j=0}^{i-1} \hat{\rho}_j}, \qquad \hat{f}_i(\hat{\boldsymbol{\rho}}) = \hat{f}_0(\hat{\boldsymbol{\rho}}) \prod_{j=0}^{i-1} \hat{\rho}_j \quad (31)$$

Estimates $\hat{\rho}_j$ are computed recursively as

$$\hat{\rho}_i = \frac{E[X] - E[Y](1+\sum_{k=0}^{i-1} \prod_{l=0}^{k} \hat{\rho}_l)}{E[Y](t-i) \prod_{k=0}^{i-1} \hat{\rho}_k} \quad (32)$$

for $E[Y] = E[X] - E[\hat{V}(1)]$ and $\hat{\rho}_0 = \frac{E[X]-E[Y]}{tE[Y]}$.

Proof. See Appendix 9.7 □

In Appendix 9.8, another lower bound is found by analyzing an equivalent model for Simplex(t). It is shown as $E[\hat{T}_{fast-serial}]$ in

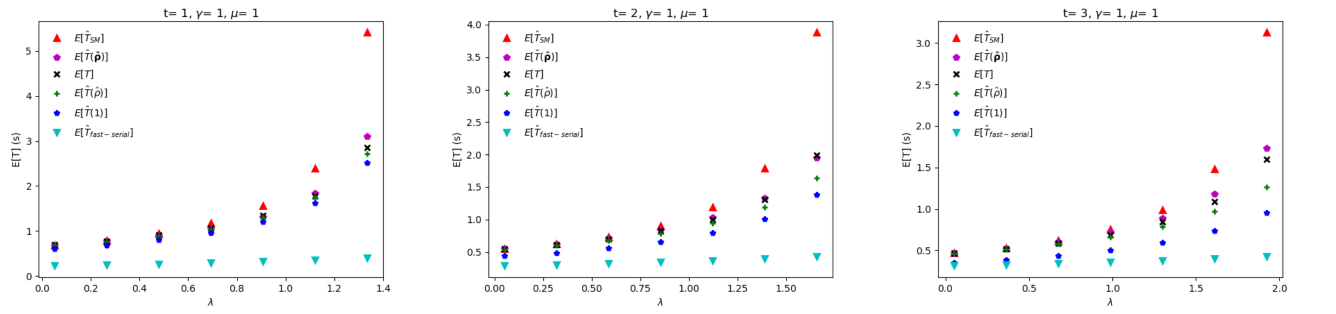

Figure 9: Comparison of the upper-bound $E[\hat{T}_{SM}]$ under split-merge assumption, the approximation $E[\hat{T}(\hat{\boldsymbol{\rho}})]$, the lower-bounds $E[\hat{T}(1)]$, $E[\hat{T}(\hat{\rho})]$, $E[\hat{T}_{fast-serial}]$ (cf. Appendix 9.8) and the simulated system time $E[T]$ for simplex queue where t is the number of repair groups.

Fig. 9 and is a much looser lower bound than $E[\hat{T}(1)]$ and $E[\hat{T}(\hat{\rho})]$, especially for higher job arrival rate.

6 SELECT-ONE SIMPLEX QUEUES

Load balancing is simply managing resources effectively to supply the demanded service capacity. This feature is desired for systems which are expected to work under high load. The system can experience either *symmetric* high load throughout the operation lifetime or *asymmetric* high load where the system may go through phases of high and low loads. For asymmetric load, the system benefits from a dynamic resource provisioning strategy, the simplest of which could have two operational states: one gets activated for low-load while the other is active for the high-load phase. For a distributed storage system, data access schemes are either imposed or limited by the way the data is encoded and distributed over the nodes. Data access schemes optimized for low-load and high-load may require the data to be re-encoded every time the system switches from one scheme to another. This introduces additional operational complexity, which makes the system prone to operational errors, and couples the problems of reliability and fast content access, which is against the "separation of concerns" principle for system design.

LRCs, and in particular simplex codes, provide the necessary flexibility in the data encoding layer for the system to seamlessly switch between different request scheduling strategies. Batch codes have been proposed for load balancing purposes in [13] and their connection to LRCs is studied in [25]. Simplex codes are linear batch codes. In Simplex(t), the simplest strategy for load balancing, namely select-one, is to forward the arriving requests either to the systematic server or one of the repair groups according to a scheduling distribution. Simplex setup allows previously analyzed replicate-to-all and the newly introduced select-one access strategies to be used interchangeably. For example, suppose that the request traffic for content follows an asymmetric load pattern. For low or middle-arrival rate regime, replicate-to-all strategy yields smaller average data access time, however with select-one strategy system operates under stability over a greater range of arrival rate. This may allow system to switch from replicate-to-all to select-one strategy seamlessly when the arrival rate increases beyond the critical point so the system can continue operating under stability. This is shown in Fig. 10 by comparing the average system time under replicate-to-all and select-one with uniform scheduling distribution in Simplex($t=1$). A closed form expression for the average system time $E[T]$ for Simplex(t) under select-one scheduling is given in Theorem 6.1.

THEOREM 6.1. *Given a scheduling distribution p such that p_i is the probability of forwarding a request independently to repair group-i where group-0 represents the systematic server, average system time for Simplex(t) under select-one access strategy is*

$$E[T] = \frac{p_0}{\gamma - p_0\lambda} + \sum_{i=1}^{t} p_i \frac{12\mu - p_i\lambda}{8\mu(\mu - p_i\lambda)} \tag{33}$$

PROOF. Every arriving job is independently sent either to systematic server with probability p_0 or repair group-i with probability p_i for $1 \leq i \leq t$. Therefore, given the job arrival process is Poisson(λ), the arrivals to repair group-i follow Poisson(λp_i). Then the systematic server is M/M/1 queue with service rate γ for which the average system time is $1/(\gamma - \lambda_0)$. Each repair group is a fork-join system of two servers (i.e., FJ-2) with the same service rate μ for which the average system time using the result in [20] is $(12 - \lambda_i/\mu)/(8\mu(\mu - \lambda_i))$. Putting these together, $E[T]$ is found. □

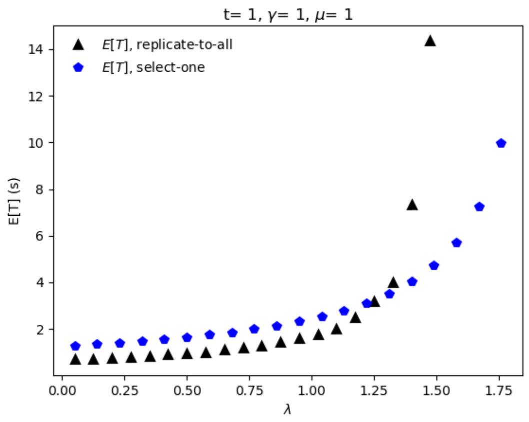

Figure 10: Comparison of the simulated average system time $E[T]$ with replicate-to-all and select-one Simplex($t=1$).

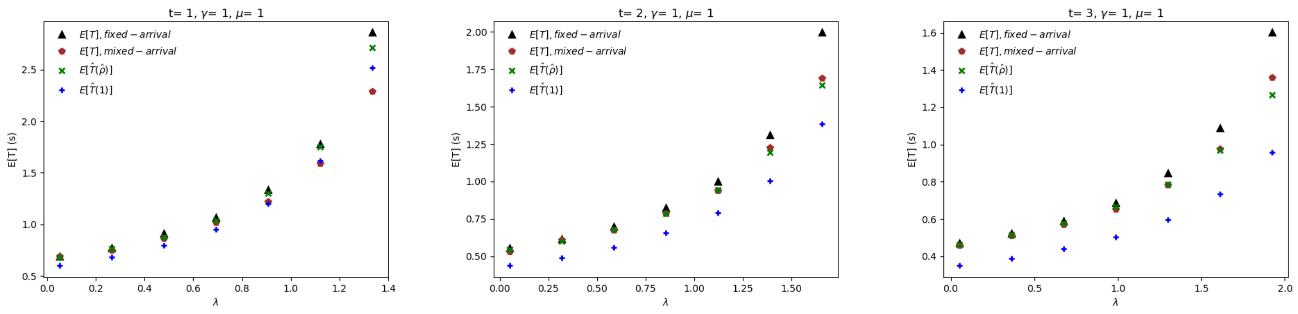

Figure 11: Comparison of the lower-bounds $E[\hat{T}(1)]$, $E[\hat{T}(\hat{\rho})]$ for $E[T_{FA}]$, and simulated average system times $E[T_{FA}]$ for fixed-arrival setup and $E[T_{MA}]$ for mixed-arrival setup in Simplex(t) for $t = 1, 2, 3$.

7 MIXED-ARRIVAL SCENARIO

Analysis given so far is for fixed-arrivals where roles of servers (systematic or repair) are for each job by assuming that the arriving jobs ask for always the same symbol throughout a busy period. In this Section, we discuss how the results for fixed-arrival scenario relate to the mixed-arrival scenario where the arriving jobs are uniformly asking for one of the data symbols stored. For example, in Simplex($t = 3$) data symbols $[a, b, c]$ are encoded into code symbols $[a, b, a + b, c, a + c, b + c, a + b + c]$ and each symbol is stored over nodes $1, \ldots, 7$. In fixed-arrival setup, all the arriving jobs will be either asking for only a or only b or only c. Thus, one of the nodes 1, 2, 3 is fixed to be the systematic server and the remaining servers compose three repair groups. However for mixed-arrival setup, one arriving job might ask for a while another one might ask for b. Hence, the systematic server and the repair groups will change depending on the arriving job.

Proposition 7.1. *Average system time for mixed-arrivals is a lower-bound for that of the fixed-arrivals.*

Proof. In Simplex(t), tasks of the job at the head of the system are served in one systematic server and t repair groups of size 2. One of the servers in each repair group may be leading such that it may be ahead with serving the tasks in its local queue compared to its sibling. Leading servers proceed with the tasks of the waiting jobs. For fixed-arrivals, these waiting tasks are one of the sibling tasks of a repair group and completion of tasks at the leading servers alone cannot terminate jobs, so jobs depart in order. For mixed-arrival, leading servers may start serving a systematic task for one of the waiting jobs and completion of a systematic task terminates the job, so jobs do not necessarily depart on the order of arrival. Therefore, a task completion at the leading servers may finish a job for mixed-arrivals while it only shortens the average service time of the consecutive jobs for fixed-arrivals. □

Proposition 7.1 shows that the results found on average system time for fixed-arrivals $E[T_{FA}]$ can be used as an upper bound for that of mixed-arrivals $E[T_{MA}]$. Note that, the fact that jobs don't necessarily depart in order and multiple jobs can depart at the same time makes the analysis of mixed-arrivals very challenging. Fig. 11 shows simulated values of $E[T_{FA}]$, $E[T_{MA}]$ and the lower-bounds $E[\hat{T}(1)]$, $E[\hat{T}(\hat{\rho})]$ obtained for $E[T_{FA}]$ in Subsection 5.2. The codes employed by the system are respectively (i) $[a, b] \longrightarrow [a, b, a + b]$ for $t = 1$, (ii) $[a, b] \longrightarrow [a, b, a+b, a+2b, a+3b]$ for $t = 2$, (iii) $[a, b, c] \longrightarrow [a, b, a + b, c, a + c, b + c, a + b + c]$ for $t = 3$. In the simulated mixed-arrival scenario, arriving jobs are independently and uniformly for one of the data symbols. An interesting observation here is that lower-bound $E[\hat{T}(\hat{\rho})]$ for $E[T_{FA}]$ is a close approximate of $E[T_{MA}]$ for $t = 2, 3$.

8 CONCLUSION

We studied the performance of codes with locality 2 and any degree of availability, specifically simplex codes, in terms of average download time of hot-data. We derived tight upper and lower bounds on the download time for codes with availability 1. We then generalized these results to systems with arbitrary availability. We argued that analyzing mixed-arrival scenarios is very challenging, and first gave an analysis on fixed-arrival scenario under replicate-to-all request scheduling strategy. Furthermore, we studied select-one scheduling strategy for load-balancing which allows the system to operate under stability over a greater range of arrival rates. The simplex setup allows the system to use both strategies interchangeably. While replicate-to-all achieves lower download time, the system can switch to select-one once the stability threshold is exceeded. Finally, we showed that the download time in mixed-arrival scenario is shorter than in the fixed-arrival scenario, and explained how the results presented for download time under fixed-arrivals relate to download time under mixed-arrivals.

REFERENCES

[1] Dhruba Borthakur. 2007. The hadoop distributed file system: Architecture and design. *Hadoop Project Website* 11, 2007 (2007), 21.
[2] Viveck Cadambe and Arya Mazumdar. 2013. An upper bound on the size of locally recoverable codes. In *2013 International Symposium on Network Coding (NetCod)*. IEEE, 1–5.
[3] Leopold Flatto. 1985. Two parallel queues created by arrivals with two demands II. *SIAM J. Appl. Math.* 45, 5 (1985), 861–878.
[4] Leopold Flatto and S Hahn. 1984. Two parallel queues created by arrivals with two demands I. *SIAM J. Appl. Math.* 44, 5 (1984), 1041–1053.
[5] Robert G Gallager. 2013. *Stochastic processes: theory for applications.* Cambridge University Press.
[6] Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, and Esa Hyytia. 2015. Reducing latency via redundant requests: Exact analysis. *ACM SIGMETRICS Performance Evaluation Review* 43, 1 (2015), 347–360.
[7] Kristen Gardner, Samuel Zbarsky, Mor Harchol-Balter, and Alan Scheller-Wolf. 2016. The Power of d Choices for Redundancy. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science.* ACM, 409–410.
[8] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. In *ACM SIGOPS operating systems review*, Vol. 37. ACM, 29–43.
[9] Parikshit Gopalan, Cheng Huang, Huseyin Simitci, and Sergey Yekhanin. 2012. On the locality of codeword symbols. *IEEE Transactions on Information Theory* 58, 11 (2012), 6925–6934.
[10] M. Harchol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action* (1st ed.). Cambridge University Press, New York, NY.
[11] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. 2012. Erasure coding in windows azure storage. In *Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12)*. 15–26.
[12] Longbo Huang, Sameer Pawar, Hao Zhang, and Kannan Ramchandran. 2012. Codes can reduce queueing delay in data centers. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on.* IEEE, 2766–2770.
[13] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2004. Batch codes and their applications. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing.* ACM, 262–271.
[14] Gauri Joshi, Yanpei Liu, and Emina Soljanin. 2012. Coding for fast content download. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on.* IEEE, 326–333.
[15] Gauri Joshi, Emina Soljanin, and Gregory Wornell. 2015. Queues with redundancy: Latency-cost analysis. *ACM SIGMETRICS Performance Evaluation Review* 43, 2 (2015), 54–56.
[16] Swanand Kadhe, Emina Soljanin, and Alex Sprintson. 2015. Analyzing the download time of availability codes. In *2015 IEEE International Symposium on Information Theory (ISIT).* IEEE, 1467–1471.
[17] Swanand Kadhe, Emina Soljanin, and Alex Sprintson. 2015. When do the availability codes make the stored data more available?. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton).* IEEE, 956–963.
[18] Andreas Klein. 2004. On codes meeting the Griesmer bound. *Discrete Mathematics* 274, 1fi?!3 (2004), 289 – 297. DOI:http://dx.doi.org/10.1016/S0012-365X(03)00201-2
[19] Guanfeng Liang and Ulaş C Kozat. 2014. Fast cloud: Pushing the envelope on delay performance of cloud storage with coding. *IEEE/ACM Transactions on Networking* 22, 6 (2014), 2012–2025.
[20] R Nelson and A N Tantawi. 1988. Approximate analysis of fork/join synchronization in parallel queues. *IEEE transactions on computers* 37, 6 (1988), 739–743.
[21] Dimitris S Papailiopoulos and Alexandros G Dimakis. 2014. Locally repairable codes. *IEEE Transactions on Information Theory* 60, 10 (2014), 5843–5855.
[22] Ankit Singh Rawat, Dimitris S Papailiopoulos, Alexandros G Dimakis, and Sriram Vishwanath. 2014. Locality and availability in distributed storage. In *2014 IEEE International Symposium on Information Theory.* IEEE, 681–685.
[23] Maheswaran Sathiamoorthy, Megasthenis Asteris, Dimitris Papailiopoulos, Alexandros G Dimakis, Ramkumar Vadali, Scott Chen, and Dhruba Borthakur. 2013. Xoring elephants: Novel erasure codes for big data. In *Proceedings of the VLDB Endowment*, Vol. 6. VLDB Endowment, 325–336.
[24] Nihar B Shah, Kangwook Lee, and Kannan Ramchandran. 2014. The MDS queue: Analysing the latency performance of erasure codes. In *2014 IEEE International Symposium on Information Theory.* IEEE, 861–865.
[25] Vitaly Skachek. 2016. Batch and PIR Codes and Their Connections to Locally-Repairable Codes. *arXiv preprint arXiv:1611.09914* (2016).
[26] Itzhak Tamo and Alexander Barg. 2014. A family of optimal locally recoverable codes. *IEEE Transactions on Information Theory* 60, 8 (2014), 4661–4676.
[27] E Varki. 2001. Response time analysis of parallel computer and storage systems. *IEEE Trans. on parallel and distributed systems* 12, 11 (2001), 1146–1161.
[28] Ronald W Wolff. 1982. Poisson arrivals see time averages. *Operations Research* 30, 2 (1982), 223–231.
[29] Roy D Yates and David J Goodman. 1999. Probability and stochastic processes. *John Willey & Sons* (1999).

9 APPENDIX

9.1 Service rate allocation in Simplex($t = 1$)

Algebra to show $\frac{\partial E[\hat{T}_{LB}]}{\partial \rho} < 0$ that is discussed in Subsection 4.5 is given here. Define $C = \gamma + 2\mu$ and $\rho = \gamma/\mu$, then the followings can be calculated

$$\hat{f}_c = \frac{1}{1 + \frac{2}{\rho(\rho+2)}}, \qquad \frac{\partial \hat{f}_c}{\partial \rho} = \frac{4(\rho+1)}{(\rho^2+2\rho+2)^2},$$
$$E[V_p] = \frac{\rho+2}{C(\rho+1)}, \qquad E[V_p^2] = \frac{2}{C^2}(\frac{\rho+2}{\rho+1})^2,$$
$$E[V_c] = \frac{2(\rho+2)}{C(\rho+1)} - \frac{1}{C}, \qquad E[V_c^2] = \frac{4}{C^2}(\frac{\rho+2}{\rho+1})^2 - \frac{2}{C^2},$$
$$\frac{\partial E[V_p]}{\partial \rho} = \frac{-1}{C(\rho+1)^2}, \qquad \frac{\partial E[V_p^2]}{\partial \rho} = \frac{-4(\rho+2)}{C^2(\rho+1)^3},$$
$$\frac{\partial E[V_c]}{\partial \rho} = \frac{-2}{C(\rho+1)^2}, \qquad \frac{\partial E[V_c^2]}{\partial \rho} = \frac{-8(\rho+2)}{C^2(\rho+1)^3},$$

$$\begin{aligned}
(i)\quad & E[\hat{V}] = E[V_p] + \hat{f}_c(E[V_c] - E[V_p]),\\
\frac{\partial E[\hat{V}]}{\partial \rho} =& \frac{\partial E[V_p]}{\partial \rho} + \frac{\partial \hat{f}_c}{\partial \rho}(E[V_c] - E[V_p]) \\
& + \frac{\partial (E[V_c] - E[V_p])}{\partial \rho}\hat{f}_c \\
=& \frac{-1}{C(\rho+1)^2} + \frac{4(\rho+1)}{(\rho^2+2\rho+2)^2}\frac{1}{C(\rho+1)} \\
& - \frac{1}{C(\rho+1)^2}\frac{\rho(\rho+2)}{\rho^2+2\rho+2} \\
=& \frac{1}{C}(\frac{-1}{(\rho+1)^2} + \frac{4}{(\rho^2+2\rho+2)^2} - \frac{\rho(\rho+2)}{(\rho+1)^2(\rho^2+2\rho+2)}) \\
=& \frac{-(\rho^2+2\rho+2)^2 + 4(\rho+1)^2 - (\rho^2+2\rho)(\rho^2+2\rho+2)}{C(\rho+1)^2(\rho^2+2\rho+2)} \\
=& \frac{-2(\rho+1)^2(\rho^2+2\rho+2) + 4(\rho+1)^2}{C(\rho+1)^2(\rho^2+2\rho+2)} \\
=& \frac{-2(\rho+1)^2(\rho^2+2\rho)}{C(\rho+1)^2(\rho^2+2\rho+2)} < 0,
\end{aligned}$$

$$
\begin{aligned}
(ii)\quad & E[\hat{V}^2] = E[V_p^2] + \hat{f}_c(E[V_c^2] - E[V_p^2]), \\
\frac{\partial E[\hat{V}^2]}{\partial \rho} =& \frac{\partial E[V_p^2]}{\partial \rho} + \frac{\partial \hat{f}_c}{\partial \rho}(E[V_c^2] - E[V_p^2]) \\
&+ \frac{\partial (E[V_c^2] - E[V_p^2])}{\partial \rho}\hat{f}_c \\
=& \frac{-4(\rho+2)}{C^2(\rho+1)^3} + \frac{8(\rho+1)}{C^2(\rho^2+2\rho+2)^2}((\frac{\rho+2}{\rho+1})^2 - 1) \\
&- \frac{4\rho(\rho+2)^2}{C^2(\rho+1)^3(\rho^2+2\rho+2)} \\
=& \frac{8}{C^2}(\frac{2\rho+3}{(\rho+1)(\rho^2+2\rho+2)^2} - \frac{\rho+2}{(\rho+1)^3}) \\
&- \frac{4\rho(\rho+2)^2}{C^2(\rho+1)^3(\rho^2+2\rho+2)} < 0,
\end{aligned}
$$

$$
\begin{aligned}
(iii)\quad & E[\hat{T}_{LB}] = E[\hat{V}] + \frac{\lambda E[\hat{V}^2]}{2(1-\lambda E[\hat{V}])} \\
\frac{\partial E[\hat{T}_{LB}]}{\partial \rho} =& \frac{\partial E[\hat{V}]}{\partial \rho} \\
&+ \frac{\lambda}{2}\frac{(\frac{\partial E[\hat{V}^2]}{\partial \rho}(1-\lambda E[\hat{V}]) + \frac{\partial E[\hat{V}]}{\partial \rho}\lambda E[\hat{V}^2])}{(1-\lambda E[\hat{V}])^2} \\
=& \frac{\partial E[\hat{V}]}{\partial \rho}(1 + \frac{\lambda^2 E[\hat{V}^2]}{2(1-\lambda E[\hat{V}])}) + \frac{\partial E[\hat{V}^2]}{\partial \rho}\frac{\lambda}{2(1-\lambda E[\hat{V}])}
\end{aligned}
$$

under stability $\lambda E[\hat{V}] < 1$ and we found above

$$\frac{\partial E[\hat{V}]}{\partial \rho} < 0, \frac{\partial E[\hat{V}^2]}{\partial \rho} < 0 \text{ which shows that } \frac{\partial E[\hat{T}_{LB}]}{\partial \rho} < 0.$$

9.2 Approximate analysis of Markov process for Simplex($t = 1$)

Here we give an exact solution of the pyramid Markov process for Simplex($t = 1$) shown in Figure 4 by using guess-based local balance equations. Consider the case $\alpha = \beta = \mu$ that makes the pyramid process symmetric i.e., $p_{k,(i,0)} = p_{k,(0,i)}$, $1 \le i \le k$. Using the symmetry, discussion in the following is given in terms of the states on the right side of the pyramid.

Observe that under low-traffic load, system spends almost entire time in states $(0,(0,0))$, $(1,(0,0))$, $(1,(0,1))$ and $(1,(1,0))$. Given this observation, notice that the rate entering into $(1,(0,0))$ due to job arrivals is equal to the rate leaving the state due to task completions at any server. To help with guessing the steady-state probabilities we start with the assumption that rate entering into a state due to job arrivals is equal to the rate leaving the state due to task completions. This gives us the following relation between steady-state probabilities of the column-wise subsequent states:

$$p_{k,(i,0)} = \frac{\lambda}{\gamma + 2\mu} p_{k-1,(i,0)}, \ 0 \le i \le k. \tag{34}$$

Define $\tau = \lambda/(\gamma + 2\mu)$. This relation allows us to write $p_{k,(i,0)} = \tau^{k-i} p_{i,(i,0)}$. However this obviously won't hold for higher arrival rates since at arrival rates some jobs wait in the queue, which requires the rate entering into a state due to job arrivals to be higher than the rate leaving the state due to task completions. To be used in the following discussion, first we write $p_{1,(1,0)}$ in terms of $p_{0,(0,0)}$ from the global balance equations as the following.

$$
\begin{aligned}
\lambda p_{0,(0,0)} &= \gamma p_{1,(0,0)} + 2(\gamma+\mu) p_{1,(1,0)}, \\
p_{1,(1,0)} &= \frac{\lambda - \gamma\tau}{2(\gamma+\mu)} p_{0,(0,0)}
\end{aligned}
\tag{35}
$$

For the nodes at the far right side of the pyramid, we can write the global balance equations and solve the corresponding recurrence relation as the following:

$$
\begin{aligned}
& p_{i,(i,0)}(\lambda + \mu + \gamma) = p_{i,(i-1,0)}\mu + p_{i+1,(i+1,0)}(\mu+\gamma), \ i \ge 1, \\
& p_{i+2,(i+2,0)} = b p_{i+1,(i+1,0)} + a p_{i,(i,0)}, \ i \ge 0 \text{ where} \\
& \quad b = 1 + \frac{\lambda}{\mu+\gamma}, \quad a = \frac{-\tau\mu}{\gamma+\mu}, \\
& \implies p_{i,(i,0)} = \frac{A}{r_0^i} + \frac{B}{r_1^i} \text{ where} \\
& \quad B = \frac{r_0 p_{0,(0,0)} + (p_{1,(1,0)} - b p_{0,(0,0)}) r_0 r_1}{r_0 - r_1}, \\
& \quad A = p_{0,(0,0)} - B \text{ where} \\
& \quad (r_0, r_1) = (\frac{-b - \sqrt{\Delta}}{2a}, \frac{-b + \sqrt{\Delta}}{2a}); \ \Delta = b^2 + 4a, \\
& p_{k,(i,0)} = p_{k,(0,i)} = \tau^{k-i}(\frac{A}{r_0^i} + \frac{B}{r_1^i}), \ 0 \le i \le k.
\end{aligned}
\tag{36}
$$

Even though the required algebra does not permit much cancellation, once can find the unknowns A and B above by computing $p_{0,(0,0)}$ as follows.

$$
\begin{aligned}
& \sum_{k=0}^{\infty} p_{k,(0,0)} + \sum_{i=1}^{\infty}\sum_{k=i}^{\infty}(p_{k,(i,0)} + p_{k,(0,i)}) \\
&= \frac{p_{0,(0,0)}}{1-\tau} + \frac{2}{1-\tau}\sum_{i=1}^{\infty} p_{i,(i,0)} \quad (\tau < 1) \\
&= \frac{p_{0,(0,0)}}{1-\tau} + \frac{2}{1-\tau}\sum_{i=1}^{\infty}(\frac{A}{r_0^i} + \frac{B}{r_1^i}) \\
&= \frac{p_{0,(0,0)}}{1-\tau} + \frac{2}{1-\tau}(\frac{A}{r_0 - 1} + \frac{B}{r_1 - 1}) \quad ((36), r_0, r_1 > 1) \\
&= \frac{p_{0,(0,0)}}{1-\tau} + \frac{2}{1-\tau}(\frac{(p_{0,(0,0)} - B)(r_1 - 1) + B(r_0 - 1)}{(r_1-1)(r_0-1)}) \\
&= \frac{p_{0,(0,0)}}{1-\tau} + \frac{2}{1-\tau}(\frac{B(r_0 - r_1) + p_{0,(0,0)}(r_1 - 1)}{(r_1-1)(r_0-1)}) \\
&= \frac{p_{0,(0,0)}}{1-\tau} + \\
& \quad \frac{2}{1-\tau}(\frac{(r_0 p_0 + r_0 r_1(p_{1,(1,0)} - b p_{0,(0,0)})) + p_{0,(0,0)}(r_1 - 1)}{(r_1-1)(r_0-1)}) \\
&= p_{0,(0,0)}(\frac{1 + 2(\frac{r_0 + r_0 r_1(\frac{\lambda - \gamma\tau}{2(\gamma+\mu)} - b) + r_1 - 1}{(r_1-1)(r_0-1)})}{(1-\tau)}) = 1, \\
& \implies p_{0,(0,0)} = \frac{(1-\tau)}{1 + 2(\frac{r_0 + r_0 r_1(\frac{\lambda - \gamma\tau}{2(\gamma+\mu)} - b) + r_1 - 1}{(r_1-1)(r_0-1)})}.
\end{aligned}
\tag{37}
$$

Simulation results show that the model for $p_{k,(i,0)}$ discussed above is proper in structure i.e., $p_{k,(i,0)}$ decreases exponentially as k or i increases. However, simulations show that $\tau(\lambda) = k(\gamma,\mu)\lambda/(\gamma + 2\mu)$.

For instance, for $\gamma = \mu$ we can find that $k(\gamma, \mu) \simeq 0.3$. Nevertheless this does not permit to find a general expression for $k(\gamma, \mu)$.

9.3 Matrix Analytic Solution for Simplex(t=1)

Defining $\delta = \alpha + \beta + \gamma + \lambda$, the sub-matrices forming $\boldsymbol{Q}$ are given below.

$$\boldsymbol{F}_0 = \begin{bmatrix} -\lambda & 0 & \lambda & 0 \\ \alpha+\gamma & \beta-\delta & 0 & 0 \\ \gamma & \beta & -\delta & \alpha \\ \beta+\gamma & 0 & 0 & \alpha-\delta \end{bmatrix},$$

$$\boldsymbol{H}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & \lambda & 0 \end{bmatrix},$$

$$\boldsymbol{L}_0 = \begin{bmatrix} 0 & \alpha+\gamma & 0 & 0 \\ 0 & 0 & \alpha+\gamma & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & \beta+\gamma & 0 \\ 0 & 0 & 0 & \beta+\gamma \end{bmatrix},$$

$$\boldsymbol{F} = \begin{bmatrix} \beta-\delta & 0 & 0 & 0 & 0 \\ \beta & -\delta & 0 & 0 & 0 \\ 0 & \beta & -\delta & \alpha & 0 \\ 0 & 0 & 0 & -\delta & \alpha \\ 0 & 0 & 0 & 0 & \alpha-\delta \end{bmatrix},$$

$$\boldsymbol{L} = \begin{bmatrix} 0 & \alpha+\gamma & 0 & 0 & 0 \\ 0 & 0 & \alpha+\gamma & 0 & 0 \\ 0 & 0 & \gamma & 0 & 0 \\ 0 & 0 & \beta+\gamma & 0 & 0 \\ 0 & 0 & 0 & \beta+\gamma & 0 \end{bmatrix},$$

$$\boldsymbol{H} = \begin{bmatrix} \lambda & 0 & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & 0 & \lambda \end{bmatrix}.$$

9.4 Truncated Markov process for $t = 2$

Here we show that truncating Markov process $\boldsymbol{n}(t)$ shown in Fig. 8 according to rule in (25) and getting $\tilde{\boldsymbol{n}}(t)$ imply $\tilde{\pi}_{i,j} > \pi_{i,j},\ i,j \leq M$.

Think of $\tilde{\boldsymbol{n}}(t)$ with two disjoint set of nodes $S_K = \{(i,j) | \min\{i,j\} \leq K\}$ and $S'_K = \tilde{\boldsymbol{n}}(t)/S_K$ where $K > 0, i,j \leq M$. Observe that transitions between S_K and S'_K exist only between the nodes that are on boundaries $B_K = \{(i,j) | \min\{i,j\} = K\}$ and B_{K+1}. Using ([29], Theorem 12.13), we have by letting $\Sigma_K = \sum_{(i,j)\in B_K} \pi_{i,j}$ and $\nu = \gamma + 4\mu$,

$$(\Sigma_0 - \pi_{0,0})\frac{2\mu}{\nu} = \Sigma_1 \frac{\gamma + 2\mu}{\nu},$$
$$(\Sigma_i - \pi_{i,i})\frac{\mu}{\nu} = \Sigma_{i+1}\frac{\gamma + 2\mu}{\nu};\ i > 0,$$
$$\text{Defining } \rho = 2 + \frac{\gamma}{\mu}, \Sigma_0 \geq \frac{\rho}{2}\Sigma_1, \quad \Sigma_i \geq \rho\Sigma_{i+1};\ i > 0.$$

Next think of the truncated chain with the two disjoint set of nodes $S_K = \{(i,j) | \max\{i,j\} \leq K\}$ and $S'_K = \tilde{\boldsymbol{n}}(t)/S_K$ where $K > 0, i,j \leq M$. Observe that transitions between S_K and S'_K exist only between the nodes that are on boundaries $C_K = \{(i,j) | \max\{i,j\} = K\}$ and C_{K+1}. We have by letting $\sigma_K = \sum_{(i,j)\in C_K} \pi_{i,j}$

$$(\sigma_0 - \pi_{0,0})\frac{2\mu}{\nu} \geq \sigma_1\frac{\gamma+\mu}{\nu},$$
$$(\sigma_i - \pi_{i,i})\frac{\mu}{\nu} \geq \sigma_{i+1}\frac{\gamma+\mu}{\nu},$$
$$\text{Defining } \tau = 1 + \frac{\gamma}{\mu}, \sigma_0 \geq \frac{\tau}{2}\sigma_1, \quad \sigma_i \geq \tau\sigma_{i+1};\ i > 0.$$

Overall for any $M > 0$, Σ_i and σ_i decrease at least exponentially with i. This is suggesting that $\tilde{\pi}_{i,j} - \pi_{i,j}$ due to truncation will be higher for lower i,j because the lower-left region of the chain is weighted exponentially more i.e., lower-left region gets higher share of increase.

9.5 On the Conjecture 5.3 for Simplex(t)

Here we do not give a complete proof for Conjecture 5.3. However Theorem 9.1 helps to build an intuition for Conjecture 5.3. Think of the system state S as the starting setup type of the job that is in service, so state space is $S \in \{0, \ldots, t\}$ where the state transition corresponds to departure of the job in service and immediate start of the next job's service. Given that Conjecture 5.3 holds i.e., $f_i > f_{i+1}$ for $0 \leq i \leq t$, one would expect the average drift at a state-i to be towards states-($< i$). Biggest challenge in proving the conjecture is that there is a transition with some probability between every possible value of state S. For instance, if the transitions were possible only between consecutive states i and $i+1$ as in birth-death chain, theorem below would have implied the conjecture.

THEOREM 9.1. *Let J_j represent the type of service start that job-j makes. Then $Pr\{J_{j+1} > i | J_j = i\} < 0.5$. In other words, for an arbitrary job that makes an arbitrary type-i service start, next job is more likely to make type-($\leq i$) start.*

PROOF. Define $L_k(t)$ as the number of tasks that the leading server at repair group k is leading by at time t. Suppose jth job makes a type-i start at time τ, namely $J_j = i$. We have the following inequality $Pr\{J_{j+1} > i | J_j = i, A\} > Pr\{J_{j+1} > i | J_j = i\}$ where A denotes the event that $L_k(\tau) > 1$ for every leading repair group k. Event A guarantees $J_{j+1} \geq i$ i.e., $Pr\{J_{j+1} \geq i | J_j = i, A\} = 1$, since even in case none of the leading servers advances before jth job terminates, next job will make at least type-i start. We will try to compute $Pr\{J_{j+1} > i | J_j = i, A\} = 1 - Pr\{J_{j+1} = i | J_j = i, A\}$. Suppose jth job terminates at time τ' and without loss of generality repair group k is leading if $k \leq i$ and non-leading otherwise. Events $\{J_{j+1} = i | J_j = i, A\}$ and $B_i = \{L_k(\zeta) < 2; \zeta \in [\tau, \tau'], i < k \leq t\}$ for $0 < i < t-1$ are equivalent since for $(j+1)$th job to make type-$(i+1)$ start, in at least one of the non-leading repair groups a server should advance by at least 2 tasks before jth job terminates. Event B_i can be expressed as $\bigcup_{l=0}^{t-i} C_l$ where $C_l = \{L_{k_j}(\tau) = 1; 1 \leq j \leq l, i < k_j \leq t\}$. Event C_l describes that l non-leading repair groups start leading by 1 before jth job terminates. Given that there exists i leading groups, denote the probability of an event that a new repair group starts to lead by 1 as p_i^{+1} and of the event that jth job terminates as p_i^T, so we can write $Pr\{C_l\} = p_{i+l}^T \prod_{k=i}^{i+l-1} p_k^{+1}$. Since events C_l for $0 \leq l \leq t-i$ are disjoint, $Pr\{B_i\} = \sum_{l=0}^{t-i} Pr\{C_l\}$ from which we can get the recurrence relation $Pr\{B_i\} = p_i^T + p_i^{+1} Pr\{B_{i+1}\}$. Since service times at the servers are assumed Exponential, probabilities are easy to find as $p_i^T = (\gamma + i\mu)/(\gamma + 2t\mu)$ and $p_i^{+1} = (2(t-i)\mu)/(\gamma + 2t\mu)$ where γ and μ are respectively service rates of the systematic server and repair servers.

We find,

$$\begin{aligned} Pr\{B_{t-1}\} &= p_{t-1}^T + p_{t-1}^{+1} p_t^T \\ &= \frac{\gamma + (t-1)\mu}{\gamma + 2t\mu} + \frac{2\mu}{\gamma + 2t\mu}\frac{\gamma + t\mu}{\gamma + 2t\mu} \\ &= \frac{\gamma + t\mu}{\gamma + 2t\mu} + \frac{\gamma\mu}{(\gamma + 2t\mu)^2} \\ &= 1 - \frac{t\mu}{\gamma + 2t\mu} + \frac{\gamma\mu}{(\gamma + 2t\mu)^2} > \frac{1}{2} \end{aligned} \tag{38}$$

Then suppose $Pr\{B_{i+1}\} > 0.5$,

$$\begin{aligned} Pr\{B_i\} &= \frac{\gamma + i\mu}{\gamma + 2t\mu} + \frac{2(t-i)\mu}{\gamma + 2t\mu} Pr\{B_{i+1}\} \\ &> \frac{\gamma + i\mu}{\gamma + 2t\mu} + \frac{(t-i)\mu}{\gamma + 2t\mu} \\ &= \frac{\gamma + t\mu}{\gamma + 2t\mu} = \frac{1}{2} + \frac{\gamma/2}{\gamma + 2t\mu} > \frac{1}{2} \end{aligned} \tag{39}$$

Knowing $Pr\{B_{t-1}\} > 0.5$ together with $Pr\{B_k\} > 0.5$ given that $Pr\{B_{k+1}\} > 0.5$ gives us $Pr\{B_i\} > 0.5$ for each i. Remember $Pr\{J_{j+1} = i | J_j = i, A\} = Pr\{B_i\}$, so we find $Pr\{J_{j+1} > i | J_j = i\} < Pr\{J_{j+1} > i | J_j = i, A\} = 1 - Pr\{B_i\} < 0.5$. This tells us that for any job and any type-i starting state, next job is more likely to make type-($\leq i$) start. □

9.6 Proof of Corollary 5.5

Proof. Under stability, sub-sequence of the job arrivals that see an empty system in Simplex(t) forms a renewal process ([5], Theorem 5.5.8). Since Simplex(t) is an M/G/1 queue, the expected number of job arrivals $E[J]$ between successive renewal epochs (busy periods) is $E[J] = E[X]/(E[X] - E[V])$ by ([5], Theorem 5.5.10). Jobs that see an empty system upon arrival definitely make type-0 start while within a busy period any type of start is probable. This observation reveals that $1/E[J]$ is a lower bound for f_0. Computing the value of $E[J]$ requires knowing $E[V]$, which we have been trying to estimate because it is hard to find. Therefore, we use the inequality $f_0 \geq 1/E[J] \geq 1/E[J]_{ub}$ where $E[J]_{ub} \geq E[J]$. An upper bound on $E[J]$ is $E[J]_{ub} = E[X]/(E[X] - E[V]_{lb})$ where $E[V]_{lb} \leq E[V]$. One possible value of $E[V]_{lb}$ is $E[\hat{V}(1)] = \sum_{i=0}^{t} E[V_i]/(t+1)$. Therefore, we get $f_0 \geq 1/E[J]_{ub} = (E[X] - E[\hat{V}(1)])/E[X]$.

Firstly, we have $\hat{f}_0(1) = 1/(1+t)$ by setting ρ to 1 in Theorem 5.4. In the system for which $\hat{f}_0(1)$ becomes exact (i.e., $f_0 = 1/(t+1)$), the lower-bound obtained from renewal theory $(E[X] - E[V])/E[X]$ holds as well under stability. For this system, $E[\hat{V}(1)]$ is exact and equal to $E[V]$ which gives $\hat{f}_0(1) = 1/(1+t) \geq (E[X] - E[\hat{V}(1)])/E[X]$.

Secondly, one can see that $(1-\rho)/(1-\rho^{t+1}) \geq 1/(1+t)$ for $0 < \rho < 1$ so we have $(1-\rho)/(1-\rho^{t+1}) \geq 1/(1+t) \geq (E[X] - E[\hat{V}(1)])/E[X]$ from which (29) is obtained. □

9.7 Proof of Corollary 5.6

Proof. Setting $\hat{f}_i = \hat{\rho}_0 \hat{f}_0$ for $1 \leq i \leq t$, $\hat{\rho}_0 \in (0,1]$ and using normalization requirement $\sum_{i=0}^{t} \hat{f}_i = 1$ one can find $\hat{f}_0 = 1/(1+t\hat{\rho}_0)$. Using the inequality $1/(1+t\hat{\rho}_0) \geq 1/(1+t) \geq (E[X] - E[\hat{V}(1)])/E[X]$, we get an upper bound on $\hat{\rho}_0$ as $\hat{\rho}_0 \leq (E[X] - E[Y])/E[Y]$ where $E[Y] = E[X] - E[\hat{V}(1)]$.

Fixing $\hat{\rho}_0 = (E[X] - E[Y])/E[Y]$, and setting $\hat{f}_1 = \hat{\rho}_0 \hat{f}_0$, $\hat{f}_i = \hat{\rho}_1 \hat{\rho}_0 \hat{f}_0$ for $2 \leq i \leq t$, we find an upper-bound on $\hat{\rho}_1$ executing the same steps done to find the upper-bound on $\hat{\rho}_0$. Normalization requirement gives $\hat{f}_0 = 1/(1 + \hat{\rho}_0(1 + \hat{\rho}_1(t-1)))$ and we have $\hat{f}_0 \geq 1/(1+t) \geq E[Y]/E[X]$ which yields $\hat{\rho}_1 \leq (E[X] - E[Y](1 - \hat{\rho}_0))/(\hat{\rho}_0 E[Y](t-1))$. The same process can be repeated by fixing $\hat{\rho}_0 = (E[X] - E[Y])/E[Y]$ and $\hat{\rho}_1 = (E[X] - E[Y](1-\hat{\rho}_0))/(\hat{\rho}_0 E[Y](t-1))$ to find an upper bound on $\hat{\rho}_2$. Generalizing this, fixing $\hat{\rho}_0$, ..., $\hat{\rho}_{i-1}$ to their respective upper-bounds, an upper-bound for $\hat{\rho}_i$ can be found as follows:

$$\hat{\rho}_i \leq \frac{E[X] - E[Y](1 + \sum_{k=0}^{i-1} \prod_{l=0}^{k} \hat{\rho}_l)}{E[Y](t-i) \prod_{k=0}^{i-1} \hat{\rho}_k}$$

Finally, setting each $\hat{\rho}_i$ to their respective upper-bounds lets us to compute the values of $\hat{f}_i$'s using which estimates $E[\hat{V}(\hat{\boldsymbol{\rho}})]$ and $E[\hat{V}^2(\hat{\boldsymbol{\rho}})]$ where $\hat{\boldsymbol{\rho}} = [\hat{\rho}_0, \ldots, \hat{\rho}_{t-1}]$ can be computed, and finally substitution in PK formula gives (32). □

9.8 A lower-bound for the expected hot-data download time in simplex queue by using an equivalent system model

We present another lower-bound for the expected system time $E[T]$ of Simplex(t). Main idea is to express the simplex setup consisting of multiple sub-systems working in parallel in terms of sub-systems cascaded in serial. Similar type of argument is used in [27] to find bounds and approximations on the system time in fork-join systems of n servers. Then the idea is generalized and used to find a lower-bound in [14] on the average access time to data encoded with MDS codes. In the following, we extend the idea introduced in these papers for Simplex(t) to derive a similar lower-bound as in [14].

The state space for Simplex(t) can be expressed by states represented as tuples $\boldsymbol{n} = (n_0, \boldsymbol{n}_1, \ldots, \boldsymbol{n}_t)$ where n_0 is the number of tasks in systematic server and each tuple $\boldsymbol{n}_i = (n_i^{(1)}, n_i^{(2)})$ for $1 \leq i \leq t$ denotes the number of tasks at the sibling servers in the repair group i. Since all repair servers are identical, state can be redefined as $\boldsymbol{n} = (n_0, (l_1, n_1), \ldots, (l_t, n_t))$ where l_i is the number of tasks the fast server is leading by and n_i is the number of tasks waiting in queue at the slow server in repair group i. In addition, tuples for the repair groups are ordered with respect to l_i such that $l_1 \geq l_2 \geq \ldots \geq l_t$.

The time that a job spends in the system can be thought as factored into *phases*. Every arriving job is split into $1 + 2t$ tasks in which t pairs of tasks are siblings that are sent to the same repair group. Until the job is completed, it may have some tasks from some repair groups finishing service earlier. For instance, in Simplex($t = 3$), at most 3 tasks of the job may depart earlier before the job finishes. We call a job being in phase-i if i tasks of the job already departed where phase-0 for a job means that all the tasks of the job are in system, waiting or in service. Overall, every job may go through possibly $t + 1$ phases, namely phase-0, phase-1, ..., phase-t, where jobs can move from one phase to only the next phase in order and at each phase, jobs may finish service and depart before moving to the next phase. For instance, in Simplex($t = 3$), a job in phase-0 may depart before moving to phase-1 if the task at the systematic server finishes service earlier, or similarly a job in phase-1 may depart before moving to phase-2 if the task at the systematic server or the remaining task at the leading repair group finishes service earlier.

The system can be thought as one with $i + 1$ queues cascaded in serial where jobs in phase-i are present in queue-i. We call this view of the system as the *serial model* while we refer to the structural view of the simplex queue as the *parallel model*. In serial model, jobs arrive to queue-0 and may possibly travel along the line of the following queues and the job is said to be in phase-i while it is waiting or in service in queue-i as illustrated in Fig. 12. Let N_i represent the number of jobs in queue-i which by construction represents the number of jobs in phase-i. Then the state space of the serial model consists of tuples $(N_0, \dots, N_t)$. Service time distribution is V_i at queue-i and after departing queue-i, a job leaves the system with probability $P_{i\to exit}$ or moves to queue-$i+1$ with probability $P_{i\to i+1}$. V_i, $P_{i\to exit}$ and $P_{i\to i+1}$ depend on $N_{i+1}, \dots N_t$. For instance for Simplex($t = 2$), if $N_0 > 0$, $N_1 = N_2 = 0$ at the systematic server $V_i\ Exp(\gamma + 4\mu)$, $P_{i\to exit} = \gamma/(\gamma + 4\mu)$, $P_{i\to i+1} = 4\mu/(\gamma + 4\mu)$ while if $N_0 > 0$, $N_1 = N_2 = 1$ $V_i\ Exp(\gamma + \mu)$, $P_{i\to exit} = \gamma/(\gamma+\mu)$, $P_{i\to i+1} = \mu/(\gamma+\mu)$. In general, $V_i\ Exp(\mu_i)$ with μ_i varying between $\gamma + \mu$ and $\gamma + i\mu + 2(t - i)\mu$ by increments of μ.

Next with a rather "graphical" argument (i.e., imagining the Markov chain for the parallel and serial models), we argue that the parallel model and the serial model we constructed above are equivalent. By construction, parameters n_i and l_i composing the state space of the parallel model can be expressed in terms of the parameter N_i composing the state space of the serial model as follows, $n_i = \sum_{k=i}^{t} N_k$, $l_i = \sum_{k=0}^{i-1} N_k$. Note that, n_0 is equal to the total number of jobs in the system as $n_i + l_i$ is for queue-i. Following the same argument given in [27], there is a one-to-one correspondence between the states of the parallel and serial models. In addition, system dynamics for both models are the same meaning that thinking of the Markov chain for both state space, each transition arc between a pair of states in parallel model is present with the same transition probability for the serial model. Thus, these two models are equivalent.

Equivalent serial model for the simplex system does not help much with finding an exact expression for the average system time $E[T]$ but gives a way to find a lower bound which is presented in Theorem 9.2.

Theorem 9.2. *A lower-bound for average system time in Simplex(t) is given by*

$$E[T] > E[\hat{T}_{fast-serial}] = \frac{1}{\gamma + 2t\mu - \lambda} + \sum_{i=1}^{t}\left(\frac{2(t-i)\mu}{\gamma + (2t-i)\mu}\right)\frac{1}{\gamma + (2t-i)\mu - \lambda\prod_{k=0}^{i-1}\frac{2(t-k)\mu}{\gamma+(2t-k)\mu}}. \tag{40}$$

Proof. In the serial model, consider ajob in service at queue-i. What this implies in the parallel model is that among the remaining $1 + 2t - i$ tasks of the job, at least one of them must be in service or at best all of the tasks can be in service. Thus the service rate μ_i at queue-i is at most $\gamma + (2t - i)\mu$. Setting all the μ_i's to their highest possible value results in a system that is faster in completing jobs. Fixing each μ_i, each queue becomes an M/M/1 using Burke's Theorem so an expression for the average system time $E[\hat{T}_{fast-serial}]$ can be easily found, which then can be used as a lower-bound for $E[T]$. Rate of arrivals λ_i to queue-i is $\lambda\prod_{k=0}^{i-1} P_{k\to k+1}$, $P_{i\to exit} = (\gamma + i\mu)/(\gamma + (2t - i)\mu)$, $P_{i\to i+1} = 1 - P_{i\to exit}$ and time that a job spends in queue-i is $1/(\mu_i - \lambda_i)$. Then $E[\hat{T}_{fast-serial}]$ can be found as

$$\begin{aligned} E[\hat{T}_{fast-serial}] &= \frac{1}{\mu_0 - \lambda} + \sum_{i=1}^{t}\frac{P_{i-1\to i}}{\mu_i - \lambda_i} \\ &= \frac{1}{\gamma + 2t\mu - \lambda} + \\ &\sum_{i=1}^{t}\left(\frac{2(t-i)\mu}{\gamma + (2t-i)\mu}\right)\frac{1}{\gamma + (2t-i)\mu - \lambda\prod_{k=0}^{i-1}\frac{2(t-k)\mu}{\gamma+(2t-k)\mu}} \\ &< E[T]. \end{aligned}$$

□

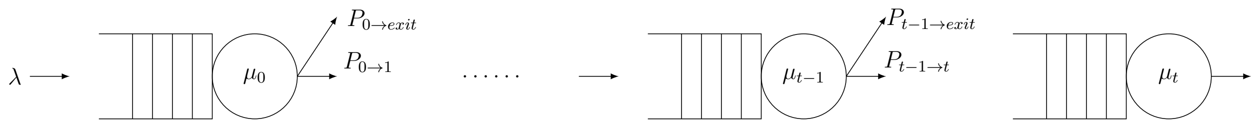

Figure 12: Representation of the serial model for Simplex(t).