Principles for Statistical Analysis in Dynamic Service Systems

Gal Mendelson
Stanford University

Kuang Xu
Stanford University

Abstract

Resource allocation control mechanisms and statistical analysis play key roles in the operation of dynamic service systems. Control is used to achieve desired performance and analysis is used for statistical tasks such as fault detection and parameter estimation. While control and analysis have been studied separately to a large extent, little is known on the interplay between the two.

In this paper we derive principles for performing online detection of server slowdown in dynamic service systems. We show that the choice of control has a drastic impact on the subsequent statistical analysis. Specifically, we show that congestion based statistics fail to detect server slowdown when applied to the data generated by systems which use popular adaptive control schemes.

We propose using the controller's action data as a new statistic for analysis and prove its effectiveness. Finally, we show that small changes to the control, such as how to break ties, can result in substantial gains in the effectiveness of the subsequent statistical analysis.

1 Introduction

Service systems typically involve the dynamic allocation of resources to serve incoming demand. These systems must meet stringent quality of service requirements, usually measured in timely service and availability. This can be achieved by proper capacity planning and good resource allocation schemes, which must take into account the fact that service systems, especially large scale ones, are susceptible to frequent abrupt changes that may hurt performance if left unattended, such as faults, server slowdowns and traffic intensity spikes.

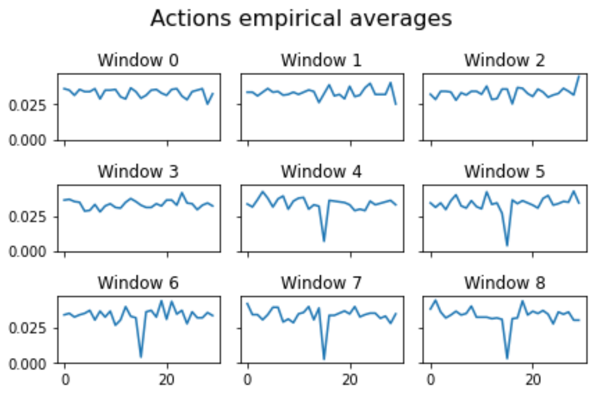
Alongside the control of such systems, it is therefore critical to quickly detect these changes to inform subsequent decisions, such as diagnosing and fixing faulty components. This can be done in the framework of statistical analysis of the available data that is generated by the system.

For example, consider a system such as a cloud service processing HTTP requests, or jobs, generated by users. In this system there are many servers, each with its own queue, which serve the incoming jobs. The control mechanism decides to which queue to send each job upon its arrival. Suppose that the statistical task is to quickly detect a sudden server slowdown.

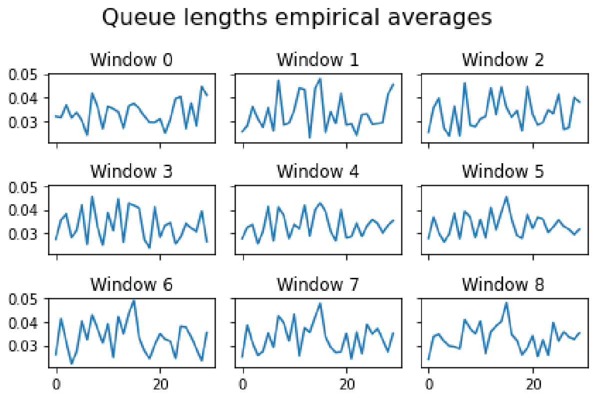
A reasonable statistic commonly used in practice for this purpose is looking at the queue lengths at the servers. If a queue starts to build up in one of them, a possible slowdown is declared. If the control randomly assigns jobs to servers, this method will work. However, if the control is adaptive, e.g. always sends an incoming job to the shortest queue, then queues will not build up and the method fails.

As demonstrated by this example, and as we later show, the choice of the control mechanism has a profound impact on the generated data and consequently may render the subsequent statistical analysis ineffective. In fact, it may seem that control and statistical analysis are conflicting with one another: on the one hand, statistical analysis performs best when the change in the data following an event is most noticeable. On the other hand, good adaptive control is designed to minimize the impact of such events on the performance, e.g. by rerouting traffic based on congestion signals. This may mask the change in the system and reflect less in the data.

While resource allocation schemes and statistical analysis of service systems data have been separately studied to a large extent, much less is known about the interplay between these two processes. Given the importance of good control and subsequent analysis, it is useful to have



(a) Actions (routing choices for arriving jobs) empirical averages.



(b) Queue lengths empirical averages.

Figure 1: Comparing Actions (routing choices for arriving jobs) and queue lengths empirical averages in a system with 30 servers over 9 consecutive time windows. The routing policy is Join the Shortest Queue and the load is 95%. Action data is very informative for slowdown detection in this case. Clearly, server number 15 slowed down in window 4. Queue lengths are noisy and are uninformative for slowdown detection in this case.

a coherent framework that incorporates both, i.e., one that lets us understand how to perform statistical analysis in the presence of dynamic scheduling and resource allocation. We do that in this paper.

For concreteness, we focus on the statistical task of online detection of service slowdown in a parallel-server system with a single dispatcher. This problem is mainly motivated by detecting mail-functioning servers in data centers, but also has applications in systems such as healthcare systems and ride-sharing systems. Our goal is to reason about how to do online slowdown detection in dynamical service systems and derive principles that can be used to guide the development and implementation of control and statistical analysis in practice.

Our main research questions are:

1. What type of information and summary statistics are useful for drawing statistical insight?
2. How does control impact statistical analysis? For instance, will a change in a routing policy alter the usefulness of a certain statistic? Should it change the inference algorithm?

We use two types of statistics to test for server slowdown: (1) Queue lengths, and (2) Action data. The action data is comprised of the previous routing decisions made by the dispatcher. To the best of our knowledge, action data was not considered before for the purpose of online detection. We consider three routing policies, which differ by their degree and nature of their adaptivity: (1) Random Routing (non-adaptive), (2) Proportional Random (rate-adaptive), and (3) Join the Shortest Queue (queue-adaptive).

We prove that a server slowdown *must* reflect in a change in the relative distributions of queue lengths, actions, or both, thus motivating their use as informative statistics. However, we also show that care must be taken when using either for detection.

Table 1 summarizes our main findings. We can see that the choice of control can drastically impact the subsequent statistical analysis and that congestion information such as queue lengths can be uninformative. On the other hand, the control's action data can be very powerful for statistical analysis. Figures 1a and 1b illustrate these points.

Finally, we further investigate the nature of action data and find that control decisions, such as what to do when queue lengths are tied, can introduce bias and/or noise. We prove that under any control policy the probability of ties is large and introduce a new tie breaking algorithm intended to eliminate biases and noise in the action data.

To summarize, our main contributions are:

1. We propose a framework to perform statistical anal-

	Queue length data	Action data
Queue-Adaptive	No detection in heavy traffic	Good in heavy traffic
Rate-Adaptive	No detection in any traffic	Good in any traffic
Non-Adaptive	No detection in heavy traffic	No detection at any traffic

Table 1: Comparing detection potential using queue lengths and action data for different policies.

ysis in dynamic service systems. It consists of two interconnected stages. In one, an engineer conducts scheduling and control. In the other, an analyst uses the resulting data to draw statistical insight.

2. Using this framework, we show that congestion based statistics fail to detect server slowdowns when applied to the data generated by systems which use popular adaptive control schemes.

3. We propose using the controller's action data as a new statistic for analysis and prove its effectiveness.

4. We show that small changes to the control can result in substantial gains in the effectiveness of the subsequent statistical analysis.

2 Control and statistical analysis framework

2.1 System and Model

We consider a parallel service system working in discrete time $t \in \mathbb{N}$, comprised of K servers labeled $\{1, \ldots, K\} := [K]$ and a single dispatcher. Each server has its own dedicated buffer in which a queue can form. At each time a job arrives at the dispatcher with probability λ, which must immediately route it to one of the servers. We assume that service times are geometrically distributed such that the service duration of a job processed by server i is distributed Geom(μ_i), with $0 < \mu_i < 1$. Servers serve jobs according to the FIFO rule.

Denote by $Q_i(t)$ the queue length at server i at time t, before any arrival or departure. The order of events at time t is as follows:

1. With probability λ a job arrives to the dispatcher, which then routes it to one of the servers.
2. For all i, if $Q_i(t) > 0$, then there is a departure with probability μ_i.

Notice that we do not allow a server to work on a job that had just arrived.

Denote by $A(t)$ the *potential action* of the dispatcher at time t. Namely, $A(t)$ encodes the routing decision at time t *if* a job arrives. The policies we consider may use current queue length information and randomization. Thus, $A(t)$ maps the current queue lengths and an additional independent random variable to a server in $[K]$. Define:

$$\mathbb{1}_i(t) = \begin{cases} 1 & \text{if } A(t) = i \\ 0 & \text{else} \end{cases}$$

The evolution of the queue lengths processes can be written using the following recursion:

$$Q_i(t+1) = Q_i(t) - \mathbb{1}_{s_i} \mathbb{1}_{\{Q_i(t)>0\}} + \mathbb{1}_a \mathbb{1}_i \qquad (1)$$

where $\mathbb{1}_{s_i}$ is the service indicator random variable (RV) which equals 1 w.p. μ_i and zero otherwise, $\mathbb{1}_a$ is the arrival indicator RV and $\mathbb{1}_i$ is the routing decision RV which equals 1 if the arrival was sent to server i by the dispatcher.

For simplicity we have suppressed the dependence of these RVs on time, and we assume that the arrival and service RVs are drawn from independent i.i.d sequences.

Denote the system load by:

$$\rho := \frac{\lambda}{\sum \mu_i}.$$

We assume that $\rho < 1$ before and after a possible server slow-down. Moreover, we assume that the control policies we consider are still able to stabilize the system after a server slowdown.

This queueing system is characterized by an observable process $X(t)$, which evolves according to some law. This is the process that encompasses all measurable quantities over time, such as previous and current queue lengths, idle times and arrivals.

2.2 Agents

There are two agents which interact with the system though the observable process $X(t)$.

Controller: Determines the dispatching policy. The Controller may use $X(t)$ to make routing decisions. For example, it may look at the current queue lengths and decide to route an incoming job to the server with the shortest queue. A key observation is that the Controller also changes the evolution of the observable process $X(t)$ through its decisions.

In this paper we consider three prevalent dispatching policies, which are representative of three classes of policies, differing by their adaptivity nature and degree:

1. **Queue adaptive: Join the shortest queue (JSQ).** An incoming job is sent to the server with the minimal queue length. Ties are broken by some rule, e.g. uniformly at random. In what follows, we show that the choice of this tie breaking rule can have a significant impact on analysis.

2. **Rate adaptive: Proportional random routing.** An incoming job is sent to server i w.p. $\mu_i/\sum\mu_j$. Notice that to implement this policy the dispatcher needs to know the actual service rates, even after a possible slowdown. We assume that the dispatcher has this knowledge either explicitly or implicitly using some automatic convergence mechanism, for example by obtaining service times measurements from servers. Importantly, we assume this information is not available for the statistical analysis.

3. **Non adaptive: Random routing.** An incoming job is sent to a server chosen uniformly at random.

Analyst: Performs statistical analysis by using data from the observable process $X(t)$. The goal of the analysis can be to infer system parameters such as traffic intensity or average waiting time. It can also be to detect changes in server speeds beyond some threshold, which is our focus in this work.

We allow the Analyst to use two types of data which are different functions of $X(t)$: Queue lengths $\{Q(t)\}$ and Actions $\{A(t)\}$. The relationship between these two conceptually different statistics is interesting and depends on the model, routing policy and what functions of these statistics are used for the statistical tests.

If arrival and service cannot occur at the same time (e.g. continuous time model), then one could reconstruct the action data based on the upward jumps of the queue lengths. The other direction is not true since action data does not contain information on service. In our model however, this reconstruction is impossible since actions can be masked by service completions.

But, regardless of the model, designing tests that use entire sample paths of queue lengths is difficult, possibly even intractable. Since our focus in this work is on practical simple and efficient tests, we will use empirical distributions as means for detection, which rule out such reconstructions.

Still, how different is queue lengths data from action data really? For routing policies that do not rely on queue lengths (e.g. rate or non adaptive) there is no direct connection between the two. Even for JSQ, $Q(t)$ uniquely determines $A(t)$ only when there are no ties and we later prove that ties happen often under any policy. We proceed with laying out the statistical framework the Analyst uses for detection.

2.3 Statistical analysis

In general, the goals of the statistical analysis determine what statistics to use and which tests to perform. We outline these in our context next.

Statistical task: Detect changes in system parameters. In this paper, we are specifically interested in detecting server slowdown using the framework of hypothesis testing, i.e. determining whether there is a slowdown of a certain magnitude or not. We distinguish between two types of tasks:

- *The detection task*: decide whether there was *a* server slowdown.

- *The identification task*: *given* there was a slowdown, identify *which* server slowed down.

In this preliminary report, we are interested in the latter, i.e. the identification task.

Choice of statistic / measurement: This is a function of the observable $X(t)$ which is used to perform the hypothesis testing. As mentioned above, we focus on two such statistics: (1) Queue lengths $\{Q(t)\}$: how long queues have been up to (and including) time t, which gives us previous and current congestion information. Queue lengths are widely used for adaptive control, as well as for analysis. (2) Action data $\{A(t)\}$: controller data information, which is the routing choices the dispatcher makes.

Window based online detection: To detect a change in queue lengths or actions caused by a sudden slowdown of a server, the Analyst gathers statistical information within disjoint consecutive windows of length L, and tests whether there was a change between consecutive windows.

Empirical distributions as means for detection: In general, the Analyst can use the temporal sample path information in each window, which contains the maximal amount of information for the observable statistics. However, as previously mentioned, the dimension of time complicates the tests' design considerably, as well as reduces the tractability of the entire procedure. It also requires understanding how the sample path changes after a slowdown event, which can be tricky.

Thus, instead of using the sample path, we can use the empirical distribution of the statistics over the window. These are defined as:

$$\hat{P}_Q(t,x) = \frac{1}{L}\sum_{s=t-L+1}^{t} \mathbb{1}_{\{Q(s)=x\}}, \quad \forall x \in \mathbb{Z}_+^K$$

$$\hat{P}_A(t,i) = \frac{1}{L}\sum_{s=t-L+1}^{t} \mathbb{1}_{\{A(s)=i\}}, \quad \forall i \in [K].$$

There are two main motivations for this choice. The first is that it considerably simplifies the test design and execution. Second, in the system we consider, by ergodicity, as L increases, the empirical distributions converge to steady state distributions, which we denote by P_Q^∞ and P_A^∞ .

Now, we prove in Conservation Theorem 3.1 that server slowdown *must* manifest in relative changes among the observables' steady state distributions. I.e, if server i slows down, it must be that the steady state distributions of Q_i and/or $\mathbb{1}_i$ change. Therefore, by association, it must manifest in changes in the empirical distributions $\hat{P}_Q$ and $\hat{P}_A$ as well.

Marginal empirical queue length distributions: Finally, even after moving from sample path to empirical distributions, $\hat{P}_Q$ is sill very complicated to work with because it keeps track of all possible combinations of queue values separately. While it might be useful to capture dependencies between the queues this way, we are able to show, for the model we consider, that when $\hat{P}_Q$ is potentially informative for detection, then so is the much simpler *marginal* empirical distributions of the queues. These are defined as:

$$\hat{P}_{Q_i}(t,x) = \frac{1}{L}\sum_{s=t-L+1}^{t} \mathbb{1}_{\{Q_i(s)=x\}}, \quad \forall x \in \mathbb{Z}_+, i \in [K]$$

Multiple hypotheses testing: Denote the service rate of servers which have not slowed down by $0 < \mu < 1$ and the slowdown factor by $0 < \alpha < 1$. Define the hypothesis:

$$H_0 := \{\mu_i = \mu, \forall i \in [K]\}$$

and for every $j \in [K]$:

$$H_j^\alpha := \{\mu_j = \alpha\mu \text{ and } \mu_i = \mu, \forall i \in [K] \setminus \{j\}\}.$$

The hypothesis H_j corresponds to the case where only server j slowed down by a factor of α. The hypothesis H_0 corresponds to the case where no server slowed down. We assume that one, and only one, of the hypotheses $H_0, H_1^\alpha, \dots, H_K^\alpha$ is true in a given window.

Remark 2.1. *In general, a slowdown can occur at any time during a window, not necessarily at the beginning. To overcome this complication, one can compare a window to two windows before it instead of one. The price is waiting an additional window for detection.*

Steady state samples approximation: Ideally, we would like to know how the quality of detection depends on the window size L. However, the empirical distributions over such windows are difficult to analyze. For the analysis, we approximate the empirical distributions by the empirical distributions of N independent samples from the steady state distributions of the statistics, P_Q^∞ and P_A^∞.

Denote by $Q^1, \dots, Q^N$ and $A^1, \dots, A^N$ N independent samples from P_Q^∞ and P_A^∞ respectively. With a slight abuse of notation, define the empirical distributions:

$$\hat{P}_Q(N,x) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{\{Q^n = x\}}, \quad \forall x \in \mathbb{Z}_+^K$$

$$\hat{P}_A(N,i) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{\{A^n = i\}}, \quad \forall i \in [K]$$

$$\hat{P}_{Q_i}(N,x) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{\{Q_i^n = x\}}, \quad \forall x \in \mathbb{Z}_+, i \in [K].$$

Test design: In this preliminary report we show negative results regarding the effectiveness of using queue lengths as means for detection regardless of the specific statistical test that is chosen. Thus, we will only specify the test which uses action data.

Since we consider the identification task, we design a test that given that we know *a* server slowed down, we declare *which* server it was. I.e., given H_0 is wrong, test for which of the hypotheses $H_1^\alpha, \dots, H_K^\alpha$ is true.

The test procedure is as follows:

1. Find the hypothesis under which the empirical average is minimal, i.e.:

$$i^* = \text{argmin}_i\{\hat{P}_A(N,i)\}$$

2. Identify server i^* as the server that slowed down.

The motivation for this simple test is that as the number of samples N gets larger, the empirical average converges to the steady state action probability values. Under adaptive policies, the server that slowed down will get less traffic, thus with a high probability, the minimal empirical average is that of the server that actually slowed down.

Alternative test: While we do not provide, for now, any results on tests other than the one specified above, it is useful to conceptually consider alternatives. A more complicated test, based on KL-divergence, can involve the entire distributions, not just the average:

1. Calculate the KL-divergence between the empirical steady state sample distribution and the steady state distribution under every hypothesis:

$$D_i := D_{KL}\big(\hat{P}_A(N,i) || P_A^\infty(i)\big)$$

2. Find the hypothesis under which the KL-divergence is minimal, i.e.:

$$i^* = \text{argmin}_i\{D_i\}$$

3. Identify server i^* as the server that slowed down.

The motivation for this simple test is that as the number of samples N gets larger, the empirical distribution converges to the steady state distribution under the correct hypothesis. Since the KL-divergence is a good measure for how different two probability distributions are, the hypothesis under which this distance is minimal should be the correct hypothesis with a high probability.

Naturally, in practice, one will use the actual empirical sample distribution over a window in this test and not samples from steady state.

Test error: The test error is the probability that we identified the wrong server as the one who slowed down, i.e.:

$$Error = \mathbb{P}(i^* \neq \text{actual server who slowed down})$$

3 Main Findings

3.1 Conservation law

In this section we motivate the use of action data by proving that a change in service rate must manifest in a change in the steady state distribution of the queue length, the action or both. In what follows, we assume that the control policy is stable before and after a change in service rate of one of the servers.

To state the theorem, denote by $Q^\infty = (Q_1^\infty, \dots, Q_K^\infty)$ a random vector distributed according to P_Q^∞, and similarly A^∞ a random variable distributed according to P_A^∞.

Theorem 3.1 (Conservation). *We have, for all $i \in [K]$:*

$$\mu_i \mathbb{P}(Q_i^\infty > 0) = \lambda \mathbb{P}(A^\infty = i) \tag{2}$$

Therefore, if μ_i changes, either $\mathbb{P}(Q_i^\infty > 0)$, $\mathbb{P}(A^\infty = i)$ or both must change as well.

Proof. The flow equation (1) holds if the queues lengths are initialized to their steady state distribution. We denote by Q^* the queue length process initialized in this way and by A^* the resulting potential action process. Taking the expected value of both sides of (1), we obtain:

$$\mathbb{E}[Q_i^*(t+1)] = \mathbb{E}[Q_i^*(t)] - \mu_i \mathbb{P}(Q_i^*(t) > 0) \\ + \lambda \mathbb{P}(A^*(t) = i), \quad (3)$$

where we have used the fact that arrival and service are driven by RVs independent of the state.

Since the queue length process Q^* is a Markov Chain and it is initialized using the steady state distribution, it is stationary in this case. Thus $Q_i^*(t)$ and $Q_i^*(t+1)$ are both distributed as Q_i^∞, and $A^*(t)$ as A^∞. In particular, we have $\mathbb{E}[Q_i^*(t+1)] = \mathbb{E}[Q_i^*(t)]$. Substituting in (3) yields the result. □

3.2 Congestion data can be uninformative

In this section we show that in the rate adaptive example, queue lengths are indistinguishable regardless of whether a server slows down or not, for any load. In the queue adaptive control example, we show that queue lengths are almost indistinguishable in high loads. Finally, for the non-adaptive case, we show that the queue length average and variance are prohibitively large in high loads. This implies the need for very large windows for convergence to steady state behaviour.

Proposition 3.2 (Rate-adaptive: Indistinguishable queue lengths)**.** *Under proportional routing, we have:*

$$\mathbb{P}(Q_i^\infty = x_i) = \rho(1-\rho)^{x_i}$$

$$\mathbb{P}(Q^\infty = x) = \Pi \mathbb{P}(Q_i^\infty = x_i) = \rho^K (1-\rho)^{\sum x_i}.$$

I.e., in steady state, the queue lengths are i.i.d. RVs, distributed Geometrically with parameter $1-\rho$.

Proof. Under proportional routing, the queue lengths are independent discrete time $M/M/1$ queues, where the input rate to queue j is $\lambda \cdot \mu_j / \sum \mu_i$. Thus, the load on server j, denoted by ρ_j, equals:

$$\rho_j = \frac{\lambda \cdot \mu_j / \sum \mu_i}{\mu_j} = \frac{\lambda}{\sum \mu_i} = \rho.$$

It is well known that the queue lengths of each server j is distributed Geometrically with parameter $1-\rho_j$ in steady state which concludes the proof. □

Thus, there is no way to distinguish between servers and deciding which one slowed down using queue length information.

Next, we consider the queue adaptive case, namely JSQ. It is well known that as the load approaches the maximum capacity of the system, the queue length processes under JSQ collapse to a single dimensional process, even when the service rates of the servers are different. This phenomena is usually referred to as 'State Space Collapse'. In other words, if in the rate adaptive example queue lengths were indistinguishable because they were i.i.d stochastic processes, under JSQ in heavy traffic, they are approximately *the same process* which makes them indistinguishable as well. We summarize this result and defer the technical details for a later version of this paper.

Proposition 3.3 (Queue-adaptive: Indistinguishable queue lengths in heavy traffic)**.** *Under JSQ, as the load increases, the joint queue length distribution approaches a single dimensional distribution. Informally:*

$$(Q_1^\infty, \ldots, Q_K^\infty) \overset{D}{\approx} (Q, \ldots, Q)$$

where Q is a random variable, same for all queues.

Finally, we consider the non-adaptive example, i,e, Random Routing, and argue that as the load increases the effectiveness of the queue lengths statistics deteriorates. It is well known that in this case the queue lengths are independent and that in steady state, each is distributed Geometrically(1-ρ_i), where $\rho_i = \frac{\lambda}{K\mu_i}$. We therefore know the expected value and variance for each queue.

Proposition 3.4 (Non-adaptive: Large queue length values and variance in heavy traffic)**.** *Under Random Routing, we have:*

$$\mathbb{E}[Q_i^\infty] = \frac{\rho_i}{1-\rho_i} = \frac{\lambda}{K\mu_i - \lambda}$$
$$\mathit{Var}(Q_i^\infty) = \frac{\rho_i}{(1-\rho_i)^2}$$

Thus, as λ increases, so do the ρ_i's, which in turn means higher expected values and variances.

3.3 Action data can be powerful

In this section we demonstrate the benefits of using action data in the presence of adaptive control. As we have shown in the previous section, queue lengths can be uninformative, especially in high loads. Action data on the other hand is very informative in the rate adaptive case, as well as in the queue adaptive case for high loads, since the action distribution must approach that of proportional routing for the system to remain stable.

It is clear that in these cases, the action steady state distribution separates well the slow server from the rest. The only question is whether the test we perform captures this. To this end, we now provide a preliminary analysis of the effectiveness of the empirical average test. Without loss of generality, assume that server 1 slowed down, i.e. hypothesis H_1^α is true. In the rate adaptive case, the steady state action probability distribution is:

$$P(A^\infty = i) = \begin{cases} \frac{\alpha}{K-1+\alpha} & \text{if } i = 1 \\ \frac{1}{K-1+\alpha} & \text{else} \end{cases}$$

Denote:

$$p = \frac{1}{K-1+\alpha}.$$

We take N samples from this distribution and get the empirical averages:

$$\hat{P}_A(N,i) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{\{A^n=i\}}, \quad \forall i \in [K].$$

We now check which i has the minimal empirical average $\hat{P}_A(N,i)$, and declare it as the server than slowed down.

Theorem 3.5 (Rate-adaptive: Bound on error)**.** *The error satisfies:*

$$Error \leq \frac{2K^2}{N(1-\alpha)^2}.$$

Proof. We have:

$$\begin{aligned} Error &= \mathbb{P}(\exists i > 1 : \hat{P}_A(N,i) < \hat{P}_A(N,1)) \\ &\leq \sum_{i=2}^{K} \mathbb{P}(\hat{P}_A(N,i) < \hat{P}_A(N,1)) \\ &= (K-1)\mathbb{P}(\hat{P}_A(N,2) < \hat{P}_A(N,1)) \end{aligned}$$

Where the last equality is by symmetry of the servers that did not slow down.

Now, each sample landed on only one server, and on server i with probability $P(A^\infty = i)$. Thus the joint distribution of how many samples there are from each server is multinomial. The marginal distribution of each element is Binomial. Thus, each empirical average multiplied by N is a Binomial random variable, such that $N\hat{P}_A(N,i)$ is distributed Bin$(N, P(A^\infty = i))$, and these are dependent across servers.

For simplicity, denote $X = N\hat{P}_A(N,2)$, $Y = N\hat{P}_A(N,1)$ and $Z = Y - X$. We have:

$$\begin{aligned} Error &\leq (K-1)\mathbb{P}(Z > 0) \\ &= (K-1)\mathbb{P}(Z - \mathbb{E}[Z] > -\mathbb{E}[Z]) \\ &\leq (K-1)\frac{Var(Z)}{(\mathbb{E}[Z])^2} \end{aligned}$$

Where we have used Chebichev's inequality and the fact that $\mathbb{E}[Z] < 0$. Now,

$$\mathbb{E}[Z] = \mathbb{E}[Y] - \mathbb{E}[X] = -Np(1-\alpha)$$

and

$$\begin{aligned} Var(Z) &= Var(Y) + Var(X) - 2Cov(X,Y) \\ &= N(p(1-p) + \alpha p(1-\alpha p)) - 2Cov(X,Y) \\ &= Np(1 - p + \alpha - \alpha^2 p) + 2Np\alpha p \\ &= Np(1 - p + \alpha - \alpha^2 p + 2\alpha p) \\ &= Np(1 + \alpha - p(\alpha^2 - 2\alpha + 1)) \\ &= Np(1 + \alpha - p(1-\alpha)^2) \leq Np(1+\alpha) \end{aligned}$$

where we have used properties of the multinomial distribution for the covariance calculation. Thus, we obtain:

$$Error \leq (K-1)\frac{Np(1+\alpha)}{(Np(1-\alpha))^2} \leq \frac{2K^2}{N(1-\alpha)^2}.$$

□

While this bound is far from tight, it does show that the error probability decays at a rate of at least $1/N$. We expect the error to decay exponentially with N.

3.4 Optimizing action data

In this section we show that small changes to the control policy can drastically improve the finite time statistical analysis. Specifically, we observe that popular choices for the control policy's tie breaking rule, namely index based or random, introduce heavy bias and noise respectively to the action data.

We prove that ties are very common under any stable policy, which makes the impact of the tie breaking rule significant. Finally, we introduce a new simple and efficient tie breaking rule, which we refer to as Adaptive Round Robin, designed to optimize the action data empirical distribution.

Theorem 3.6 (Ties are prevalent)**.** *For a homogeneous system, under any stable policy, we have:*

$$\mathbb{P}(\textit{There is a tie}) \geq 1 - \Big(\frac{K}{K-1}\Big)\rho.$$

For example, for any system with at least 10 servers the chance for a tie is at least 10% for any load under 80%, and at least 40% for any load under 50%.

Proof. The probability that there is a tie is larger than the probability that there are two or more servers with queue lengths zero. Denote by X_f^* the fraction of idle servers in steady state, namely:

$$X_f = \frac{1}{K}\sum_{i=1}^{K} \mathbb{1}_{\{Q_i^\infty = 0\}}.$$

Thus, we want to prove that:

$$\mathbb{P}(X_f > \frac{1}{K}) \geq 1 - \Big(\frac{K}{K-1}\Big)\rho$$

Since the server rates are equal, by symmetry we must have $\mathbb{P}(A^\infty = i) = 1/K$, for all $i \in [K]$. Plugging this into Equation (2) and summing over all servers, we obtain:

$$\frac{1}{K}\sum_{i=1}^{K}\mathbb{P}(Q_i^\infty > 0) = \frac{\lambda}{K\mu} = \rho.$$

We have:

$$\begin{aligned}\mathbb{E}[X_f] &= \frac{1}{K}\sum_{i=1}^{K}\mathbb{E}[\mathbb{1}_{\{Q_i^\infty=0\}}] = 1 - \frac{1}{K}\sum_{i=1}^{K}\mathbb{P}(Q_i^\infty > 0)\\ &= 1-\rho\end{aligned}$$

Now,

$$\begin{aligned}1-\rho = \mathbb{E}[X_f] &\leq \frac{1}{K}\mathbb{P}(X_f \leq \frac{1}{K}) + 1\cdot(1-\mathbb{P}(X_f \leq \frac{1}{K}))\\ &= 1 - (1-\frac{1}{K})\mathbb{P}(X_f \leq \frac{1}{K})\end{aligned}$$

Using the fact that $\mathbb{P}(X_f > \frac{1}{K}) = 1 - \mathbb{P}(X_f \leq \frac{1}{K})$ and rearranging completes the proof. □

We proceed by considering two common tie breaking rules and their impact on the action data.

Proposition 3.7 (Random introduces noise)**.** *Breaking ties randomly introduces substantial noise to the empirical action data distribution.*

Argument outline. By Theorem 3.6, ties occur often for any control policy. Therefore, for each arriving job, there is a high probability that a tie must be broken. If this is done randomly, then a large percentage of actions are distributed randomly between at least two servers. This introduces random fluctuations in the action data which we refer to as noise.

Figure 2b illustrates the finite time action data when a random tie breaking rule is used. We can see that there is substantial noise which masks the slowdown of server number 15 at window number 4.

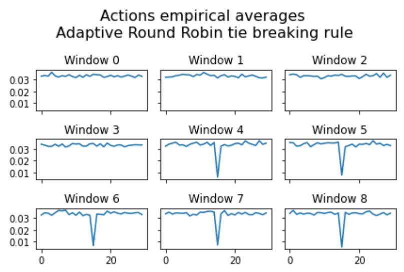
Proposition 3.8 (Index based introduces bias)**.** *Breaking ties by index introduces a substantial bias to the empirical action data distribution.*

Argument outline. If ties occur often, then smaller index servers are given substantially more jobs than higher indexed servers.

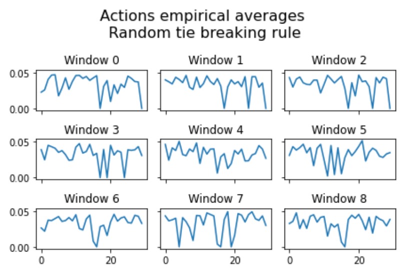
Figure 2c illustrates that there is a substantial bias using this rule, such that lower index servers are preferred. Interestingly, there is almost no noise, and the slowdown of server 15 is apparent.

Adaptive Round Robin tie breaking rule. We now introduce a new, simple and efficient method to break ties. It is inspired by the round robin policy and conceptually works as follows:

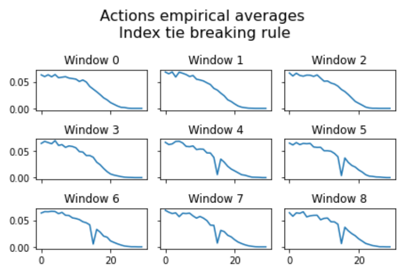
- The servers are are put on a ring.



(a) With Adaptive Round Robin there is less noise and no bias.



(b) Breaking ties randomly introduces noise.



(c) Breaking ties by index introduces bias.

Figure 2: Comparing action empirical averages over 9 consecutive time windows with different tie breaking rules. The simualtion has 30 servers, the policy is JSQ and the load is 50%. Server number 15 slows down during window number 4. The tie breaking rule drastically impacts the finite time action data distribution.

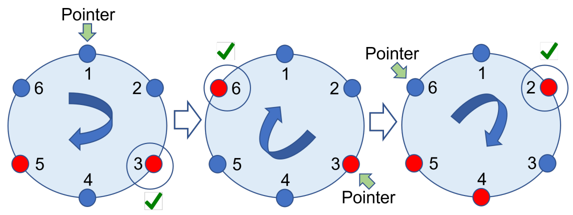


Figure 3: The Adaptive Round Robin tie breaking rule. The servers in red are tied. The chosen server is the one clockwise closest to the current position of the pointer.

- There is a pointer that points to one of the servers, initialized arbitrarily.
- When a tie needs to be broken, the chosen server is the one which is clockwise closest to the pointer.
- The pointer position is updated to the chosen server.

Figure 3 illustrates the Adaptive Round Robin tie breaking rule. At first (left) the pointer points to server 1, and there is a need to break the tie between servers 3 and 5 (marked in red). Since 3 is clockwise closest to 1, it is chosen and the pointer is updated accordingly. Next, the tie between servers 3 and 6 is broken and server 6 is chosen. Notice that the server the pointer points to is considered to be with the farthest distance from itself, so that a different server is always chosen to avoid choosing it repeatedly, which will introduce bias. Finally, server 2 is closest to 6 and therefore is chosen.

Efficient implementation. Given a set of size $2 \leq k < K$ of tied servers $\{s_1, s_2, \ldots, s_k\}$ and the current pointer position p, the algorithm does the following:

- Initialize two numbers which we call low (initialized to p) and $high$ (initialized to $K + 1$).
- For each element s_i, if $p < s_i < high$ we set $high = s_i$. Otherwise, if $s_i < low$ we set $low = s_i$.
- After considering the entire set $\{s_1, s_2, \ldots, s_k\}$, if $high = K + 1$, return low as the winner of the tie. Otherwise, return $high$.

Correctness. After considering the entire set $\{s_1, s_2, \ldots, s_k\}$, if $high$ was ever updated, at termination it must contain the closest server larger than p. In this case, it is the clockwise closet to p. If it was not updated (i.e. its value remained $K + 1$), it must mean there are no servers in $\{s_1, s_2, \ldots, s_k\}$ that are larger than p. In this case, at termination, low must contain the smallest server in the set $\{s_1, s_2, \ldots, s_k\}$, which is the clockwise closet to p.

Complexity. The algorithm always terminates using exactly k operations and the memory overhead is storing the numbers low and $high$, which need $\lfloor log_2(K) \rfloor + 1$ bits each.

Figure 2a illustrates the finite time action data when using the Adaptive Round Robin tie breaking rule. Clearly, the server slowdown in window 4 is apparent, and the data is unbiased and with substantially reduced noise.