The Twin Measure for Queueing System Predictability

David Raz
School of Computer Science
Tel-Aviv University, Tel-Aviv, Israel
davidraz@post.tau.ac.il

Hanoch Levy
School of Computer Science
Tel-Aviv University, Tel-Aviv, Israel
hanoch@cs.tau.ac.il

Benjamin Avi-Itzhak
RUTCOR, Rutgers University
New Brunswick, NJ, USA
aviitzha@rutcor.rutgers.edu

ABSTRACT

Two identical jobs with deterministically identical processing times arrive at a Web server simultaneously (Twins), but leave the system thirty seconds apart. Is the service predictable? Is their sojourn time predictable? This issue arises in modern day networking systems such as call centers and Web servers as well as in other queueing systems. We propose a novel measure based on the principle that in a predictable system, "twin" jobs should not depart the system very far apart. We analyze this measure for a number of common scheduling policies and compare the results. We compare the results to those of other predictability approaches proposed recently and discuss its usefulness.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance Attributes—*Predictability*; F.2.2 [**Nonnumerical Algorithms and Problems**]: Sequencing and Scheduling; G.3 [**Probability and Statistics**]: Queuing Theory

General Terms

Performance, Measurement

Keywords

predictability, FCFS, LCFS, SJF, LJF, LAS, SRPT, LRPT, round robin, job scheduling, processor sharing, PS, queue disciplines, twin measure

1. INTRODUCTION

How does one measure the predictability of a queueing system?

The importance of the issues of predictability is widely recognized in many works and applications. For example the issue of predictability was recently discussed in [13], which provides references to related work discussing its importance.

The issue of predictability is also strongly related to the issue of *Fairness*. If the system isn't predictable, it is most likely since some jobs are treated unfairly. See for example [1] that surveys several studies discussing the subject of fairness, and some recent works quantifying job fairness.

In modern day systems such as call centers and computer systems, where the actual workings of the system are hidden from the customer, one practical way to measure and check for system predictability is to launch a pair of identical jobs with deterministically identical service times (*Twins*). For example, it is very natural that a Web user who submits two concurrent identical, or close to identical, requests to the same Web site, will appreciate the site predictability based on their relative response times. We propose a measure that is based on this notion. Obviously, a low "twin measure" is not a sufficient requirement for customer satisfaction. In fact, we do not even claim that it is sufficient for guaranteeing system predictability, as this issue is too complicated to be captured by this simple notion. However, we do claim that it is a *required* feature, and a simple one to measure and analyze, as we will show.

Specifically, we propose a measure based on the expected value, and possibly higher moments, of the difference between the waiting times ,or sojourn times, of identical, simultaneously arriving jobs, conditioned on the jobs' service requirement. By conditioning on the service requirement this measure also captures the notion of a policy which is not equally predictable to all job sizes.

The model and the measure we propose are described in Section 2. Analysis of the twin measure in single server systems, for common scheduling policies, is given in Section 3. We then (Section 4) discuss the measures obtained: in Section 4.1 we classify the scheduling policies into four classes; in Section 4.2 we compare the classification results to those obtained by another recently proposed predictability criterion; in Section 4.3 we discuss what optimality under the twin measure means, and we propose a policy which is optimal with respect to both the twin measure and the sojourn times.

In Section 5 we analyze the twin measure for some common multiple server systems.

Finally, in Section 6 we discuss several ways to extend the twin measure and we conclude with some concluding remarks (Section 7).

2. MODEL AND NOTATION

We use an $M/GI/s$ model, namely, a queueing system with s servers, each with one unit of service rate. Arrivals are Poisson at rate λ, and the service requirements are sampled independently with probability density function (pdf) $b(x)$ and cumulative distribution function (cdf) $B(x)$. B_x is the probability of a service requirement of exactly x, and of course for continuous $b(x), \forall x, B_x = 0$. However, for many practical distributions there is at least one value for which there is an accumulation of probability and $B_x \neq 0$. e.g. the maximum value.

The service requirement has expected value $\overline{x}$, and a second moment $\overline{x^2}$.The load (utilization) of the server is defined as

$$\rho \overset{def}{=} \frac{\lambda \overline{x}}{s} = \frac{\lambda}{s} \int_0^\infty tb(t)dt,$$

and for stability we require $\rho < 1$.

One useful quantity is the load made up by the jobs of size less than or equal to x, denoted $\rho(x)$, which is

$$\rho(x) \overset{def}{=} \frac{\lambda}{s} \int_0^x tb(t)dt.$$

We also define he load made up by the jobs of size strictly less than x, denoted $\rho(x^-)$, which is

$$\rho(x^-) \overset{def}{=} \frac{\lambda}{s} \int_0^{x^-} tb(t)dt.$$

Note that for a continuous pdf $b(t)$, $\rho(x) = \rho(x^-)$.

We use the notation $X \sim BP(x)$ to denote that X is distributed as a busy period starting with a job of size x. $X \sim BP_y(x)$ and $X \sim BP_{y^-}(x)$ denote the same, except the busy period is only composed of jobs of size not larger than y and smaller than y, respectively.

We use the notation $X \preceq F(t)$ to denote that a random variable X stochastically dominates $F(t)$, i.e. $\mathbb{P}\{X \leq t\} \leq F(t) \quad \forall t \geq 0$. We use the notation $\prec$ in a similar way.

We assume that the server is work conserving, i.e. a preempted job retains the service it was already given.

We use `Typrwriter-Style` to denote scheduling policies.

2.1 The Twin Measure

Let C_1 and C_2 be identical jobs, with service requirements x and $x + \delta$ respectively. We call such jobs *twins*. Let the arrival, departure, and first service epochs of twin i, $i = 1, 2$ be a_i, d_i, and s_i respectively. Assume that C_1 arrives when (a_1) the system is in steady state. Assume that the twins arrive ϵ time units apart, that is, $a_2 - a_1 = \epsilon, \epsilon > 0$.

Definition 2.1 (Twin Measure). *Define the random variable $Z(x, \epsilon, \delta) = |d_2 - d_1|$, given x, ϵ, and δ. Let $z^n(x, \epsilon, \delta)$ be the n-th moment of $Z(x, \epsilon, \delta)$, i.e. $z^n(x, \epsilon, \delta) = \mathbb{E}\{Z(x, \epsilon, \delta)^n\}$.*

For scheduling policy ϕ and job size x, the n-th twin measure $\mathcal{T}_n^\phi(x)$ is defined as the limit, when ϵ and δ tend to zero, of $z^n(x, \epsilon, \delta)$, assuming a single limit exists. Namely $\mathcal{T}_n^\phi(x) = \lim_{\epsilon \to 0, \delta \to 0} z^n(x, \epsilon, \delta)$.

The shortened term *twin measure*, denoted $\mathcal{T}^\phi(x)$, is used to describe the first twin measure, namely $\mathcal{T}^\phi(x) = \mathcal{T}_1^\phi(x) = \lim_{\epsilon \to 0, \delta \to 0} \mathbb{E}\{Z(x, \epsilon, \delta)\}$. While we will focus in this paper on the first twin measure, we find it useful to define it in the scope of higher moments, for future research.

Remark 2.1. *The n-th twin measure is only defined when a single limit exists for both $\delta \searrow 0$ and $\delta \nearrow 0$, while $\epsilon \to 0$. One can devise a policy for which there is no such single limit. For example, consider a policy ϕ that serves jobs in a First-Come-First-Served (`FCFS`) manner, unless the second job in the queue has a smaller size than the first one, in which case it servers the first two jobs in a Processor Sharing (`PS`) manner. In this case for $\delta \searrow 0$ the twins are served in a `PS` manner and $\mathcal{T}^\phi(x) = 0$, while for $\delta \nearrow 0$ the twins are served in a `FCFS` manner, and $\mathcal{T}^\phi(x) = x$. The twin measure in such a case can be chosen to be the maximum of all the limits, the mean value, or it can remain undefined, as appropriate for the application.*

Definition 2.1 has the benefit that it applies to service distributions for which same size arrivals are impossible. It also avoids a pitfall that a policy ϕ can artificially serve equal sized jobs in a different manner than non-equal sized ones. As in some size distributions equal sized jobs are extremely rare, this will not hinder the expected performance of ϕ, yet allow it to have an artificially low twin measure. However, this definition makes the analysis tedious so for the sake of analysis we use a simpler definition:

Definition 2.2 (Simplified Twin Measure). *Let $\delta = 0$, i.e. both jobs have the same size. Define the random variable $Z(x, \epsilon) = |d_2 - d_1|$, given x and ϵ. Let $z^n(x, \epsilon)$ be the n-th moment of $Z(x, \epsilon)$, i.e. $z^n(x, \epsilon) = \mathbb{E}\{Z(x, \epsilon)^n\}$.*

For scheduling policy ϕ and job size x, the n-th twin measure $\mathcal{T}_n^\phi(x)$ is defined as the limit, when ϵ tend to zero, of $z^n(x, \epsilon)$, assuming a limit exists. Namely $\mathcal{T}_n^\phi(x) = \lim_{\epsilon \to 0} z^n(x, \epsilon)$.

The shortened term *twin measure*, is again used to describe the first twin measure, namely $\mathcal{T}^\phi(x) = \lim_{\epsilon \to 0} \mathbb{E}\{Z(x, \epsilon)\}$.

Whether we use the original or the simplified definition will be clear from the context.

One can also choose to normalize the twin measure, the obvious normalization factor being x. This does not change the results in any significant manner.

3. ANALYSING COMMON SCHEDULING POLICIES FOR SINGLE SERVER SYSTEMS

3.1 Processor Sharing (`PS`)

When C_1 departs, C_2 can have at most $\epsilon + \delta$ remaining service time. Therefore $z(x, \epsilon, \delta) \leq (\epsilon + \delta)\bar{N}$ where $\bar{N}$ is the mean number of jobs in the system. Thus

$$\mathcal{T}^{PS}(x) = \lim_{\epsilon \to 0, \delta \to 0} z(x, \epsilon, \delta) \leq \lim_{\epsilon \to 0, \delta \to 0} \left((\epsilon + \delta)\bar{N}\right) = 0.$$

Intuitively, both jobs are identical and arrive simultaneously, so they will receive exactly the same service, and leave the system simultaneously.

3.2 First Come First Served (`FCFS`)

Let $W(x, \epsilon, \delta)$ be a random variable denoting the time elapsing between d_1 and s_2. For any non-preemptive scheduling policy we have $Z(x, \epsilon, \delta) = W(x, \epsilon, \delta) + x + \delta$. For `FCFS` $W(x, \epsilon, \delta)$ is the amount of work arriving in the interval between the arrival epochs of the twins and it has an expected value of $\rho\epsilon$, i.e. $z(x, \epsilon, \delta) = \mathbb{E}\{Z(x, \epsilon, \delta)\} = \rho\epsilon + x + \delta$ and

$$\mathcal{T}^{FCFS}(x) = \lim_{\epsilon \to 0, \delta \to 0} (\rho\epsilon + x + \delta) = x.$$

3.3 Last Come First Served (LCFS)

We start with the non-preemptive case. There are two possible orders of service, either (i) C_1 is served before C_2, i.e. either the server was idle on a_1, or the server finished serving the job that was served on a_1 at some epoch in the interval $[a_1, a_2)$, and no other job arrived between a_1 and that epoch, or (ii) C_2 is served first.

(i) We can ignore all jobs served before C_1. The service order from then onward is as follows. First C_1 is served for x units of time. C_2 then waits for a busy period created by all jobs arriving while C_1 was served, and is served at the completion of this busy period, for $x + \delta$ units of time. Let $V(x, \epsilon, \delta)$ be a random variable denoting the time elapsing between s_1 and s_2. Thus $Z(x, \epsilon, \delta) = V(x, \epsilon, \delta) + \delta$. Clearly $V(x, \epsilon, \delta) \sim BP(x)$, thus

$$z(x, \epsilon, \delta) = \frac{x}{1-\rho} + \delta.$$

For example, this can be derived from the transform of the distribution of a busy period starting with a job of size x, $G^*(s, x) = e^{-x[s+\lambda-\lambda G^*(s)]}$, where $G^*(s)$ is the transform of the distribution of the busy period length (see e.g. [5, p. 212]).

Taking the limit we have

$$\mathcal{T}^{LCFS}(x) = \lim_{\epsilon\to 0, \delta\to 0} \left(\frac{x}{1-\rho} + \delta \right) = \frac{x}{1-\rho}.$$

(ii) We can ignore all jobs served before C_2. The service order from then onward is as follows. First C_2 is served for $x + \delta$ units of time. Then a busy period created by all jobs arriving while C_2 was served is being served. Then a busy period created by all jobs arriving between a_1 and a_2 is served, followed by C_1 being served for x units of time. Thus $Z(x, \epsilon, \delta) = V(x, \epsilon, \delta) - \delta$, where $V(x, \epsilon, \delta) \sim BP(x + \delta + \epsilon)$. Therefore

$$z(x, \epsilon, \delta) = \frac{x+\delta+\epsilon}{1-\rho} - \delta$$

$$\mathcal{T}^{LCFS}(x) = \lim_{\epsilon\to 0, \delta\to 0} \left(\frac{x+\delta+\epsilon}{1-\rho} - \delta \right) = \frac{x}{1-\rho}.$$

For the preemptive case, all jobs arriving in the interval $[a_1, a_2)$, including C_1, are served in total for a period of length ϵ until they are preempted by C_2. C_2 and any preempting jobs are then served until d_2, and can be ignored. The period (d_2, d_1) is composed of C_1 and jobs arriving in the interval (a_1, a_2), and jobs preempting them, minus a service of ϵ units of time already done. Therefore $Z(x, \epsilon, \delta) \sim BP(x + \epsilon) - \epsilon$, so

$$\mathcal{T}^{LCFS}(x) = \lim_{\epsilon\to 0, \delta\to 0} \left(\frac{x+\epsilon}{1-\rho} - \epsilon \right) = \frac{x}{1-\rho}.$$

To summarize, for both the non-preemptive and the preemptive case

$$\mathcal{T}^{LCFS}(x) = \frac{x}{1-\rho}.$$

3.4 Shortest Job First (SJF)

The analysis for SJF is quite tedious if one uses Definition 2.1. Therefore, the analysis provided here uses Definition 2.2, i.e. we assume the twin jobs are of equal size. One can verify that this simplification does not alter the result as well. We will use this definition from this point onwards.

We start with the non-preemptive case.

Note that in general, SJF does not determine the order of service between equally sized jobs. Obvious choices are either FCFS or LCFS. We call these policies SJF-FCFS and SJF-LCFS respectively.

Starting with SJF-FCFS, C_1 is always served first. We can ignore all jobs served before C_1. The service order from then onwards is as follows. First C_1 is served for x units of time. This is followed by a busy period composed of jobs of sizes smaller than x arriving while C_1 was served. Following this, jobs of size x arriving in the interval (a_1, a_2) are served, followed by a busy period composed of jobs of size smaller than x arriving while they were served. Lastly, C_2 is served for x units of time. If we let $V(x, \epsilon)$ be a random variable denoting the time elapsing between s_1 and s_2 it is easy to see that $Z(x, \epsilon) = V(x, \epsilon)$. Considering the service order we have $V(x, \epsilon) \sim BP_{x^-}(x) + BP_{x^-}(\epsilon\lambda x B_x)$, and therefore

$$\mathcal{T}^{SJF-FCFS}(x) = \lim_{\epsilon\to 0} \left(\frac{x}{1-\rho(x^-)} + \frac{\epsilon\lambda x B_x}{1-\rho(x^-)} \right) = \frac{x}{1-\rho(x^-)}. \quad (1)$$

For SJF-LCFS, note that C_1 can still be served first, e.g. if the server is idle on a_1. Therefore, let C_f and C_s be the first and second twins to be served, respectively. The order of service starts with C_f, followed by a busy period composed of jobs of sizes *not larger* than x arriving while C_f was served. Following this, jobs of size x arriving in the interval (a_1, a_2) are served, followed by a busy period composed of jobs of size not larger than x arriving while they were served. Lastly, C_s is served for x units of time. Using the same notation $Z(x, \epsilon) = V(x, \epsilon)$ and $V(x, \epsilon) \sim BP_x(x) + BP_x(\epsilon\lambda x B_x)$, leading to

$$\mathcal{T}^{SJF-LCFS}(x) = \lim_{\epsilon\to 0} \left(\frac{x}{1-\rho(x)} + \frac{\epsilon\lambda x B_x}{1-\rho(x)} \right) = \frac{x}{1-\rho(x)}. \quad (2)$$

Note that for continuous service distributions SJF-FCFS and SJF-LCFS have the same twin measure.

We now move on to the preemptive case. Again we consider SJF-FCFS and SJF-LCFS.

For SJF-FCFS C_1 is served first. When C_1 finishes service there are no jobs of size smaller than x in the queue. The next to be served are jobs of size x arriving in (a_1, a_2), followed by C_2. Each of those can be interrupted, but only by jobs of size smaller than x. Therefore $Z(x, \epsilon) \sim BP_{x^-}(\epsilon\lambda x B_x) + BP_{x^-}(x)$ and (1) holds.

For SJF-LCFS C_1, and other jobs of size x arriving in (a_1, a_2), can be served for a total service no loner than ϵ before being preempted by C_2. Let $\beta(\epsilon)$ be a random variable denoting this amount of service. When C_2 finishes service the next to be served are jobs of size x arriving in (a_1, a_2), followed by C_1. Each of those can be interrupted by jobs of size *not larger* than x. Therefore $Z(x, \epsilon) \sim BP_x(\epsilon\lambda x B_x) + BP_x(x) - \beta(\epsilon)$. As $0 \le \beta(\epsilon) \le \epsilon$ we have $\lim_{\epsilon\to 0} \beta(\epsilon) = 0$ and (2) holds.

3.5 Longest Job First (LJF)

We define LJF-FCFS and LJF-LCFS in a similar manner to the ones defined for SJF.

Using the same arguments as in Section 3.4, and the fact that

$$\int_{x^+}^{\infty} tf(t)dt = \rho - \rho(x)$$
$$\int_{x}^{\infty} tf(t)dt = \rho - \rho(x^-)$$

we have

$$\mathcal{T}^{LJF-FCFS}(x) = \frac{x}{1-(\rho-\rho(x))}$$
$$\mathcal{T}^{LJF-LCFS}(x) = \frac{x}{1-(\rho-\rho(x^-))}.$$

This applies to both the non-preemptive and the preemptive case.

3.6 Least Attained Service (LAS)

In the LAS scheduling policy, also called FB, service is given to the jobs which received the least service so far. See [8] for a survey of results regarding this policy.

Note that in the LAS scheduling policy jobs with equal attained service share the processor. Upon arrival, C_1 will be served for at most ϵ, then C_2 will be served for an equal amount, and from then on they will have equal attained service, and keep sharing the processor, until they leave the system together. Therfore

$$\mathcal{T}^{LAS}(x) = 0.$$

3.7 Shortest Remaining Processing Time (SRPT)

Note that since both jobs start with equal jobs sizes, from the first epoch in which one of them is served, the other will not be served until the first one leaves the system. It can easy to observe that the twin measure is the same no matter which of the jobs is first served. We will therefore assume w.l.g. that C_1 is served first. Following [11], $Z(x,\epsilon)$ can be decomposed into the sum

$$Z(x,\epsilon) = W(x)^{SRPT} + R(x)^{SRPT}$$

where $W(x)^{SRPT}$ is a random variable denoting the waiting time for C_2 of size x, i.e. the time from d_1 to s_2, and $R(x)^{SRPT}$ is a random variable denoting the residence time for C_2 of size x, i.e. the time from s_2 to d_2. Both $W(x)^{SRPT}$ and $R(x)^{SRPT}$ do not depend on ϵ and therefore

$$\begin{aligned}\mathcal{T}^{SRPT}(x) &= \lim_{\epsilon\to 0}\left(W(x)^{SRPT} + R(x)^{SRPT}\right)\\ &= \mathbb{E}\{W(x)\}^{SRPT} + \mathbb{E}\{R(x)\}^{SRPT}, \quad (3)\end{aligned}$$

Starting with $\mathbb{E}\{W(x)\}^{SRPT}$, note that once C_1 enters service, we can divide the arriving jobs into three categories: 1) jobs with service requirement over or equal to x. These jobs are served after d_2 and therefore can be ignored. 2) jobs with service requirement below the remaining service requirement of C_1 on the epoch of their arrival. These jobs will preempt C_1 and be served before d_1. 3) jobs with service requirement lower than x, but above the remaining service requirement of C_1 on the epoch of their arrival. These jobs are served in the interval (d_1, s_2).

To carry out this analysis, observe jobs arriving in an infinitesimal interval of size dt when C_1 has remaining service requirement t. Category 2) jobs preempt C_1 and create a sub busy period of size $dt/(1-\rho(t^-))$ (including the initial dt interval). Category 3) jobs arriving in this sub busy period are to be served after d_1, and the work load created by these jobs is

$$\frac{dt}{1-\rho(t^-)}\lambda\int_{t^+}^{x^-} yb(y)dy = \frac{\big(\rho(x^-)-\rho(t)\big)\,dt}{1-\rho(t^-)}, 0 \le t < x,$$

and zero for $t = x$.

Note that any job arriving while these jobs are served, with service requirement below x, will also be served before s_2. Therefore we are facing a busy period of size

$$\frac{\big(\rho(x^-)-\rho(t)\big)\,dt}{1-\rho(t^-)}\frac{1}{1-\rho(x^-)}.$$

Integrating this yields the mean size of the waiting interval, namely

$$\mathbb{E}\{W(x)\}^{SRPT} = \int_0^{x^-}\frac{\rho(x^-)-\rho(t)}{(1-\rho(t^-))\,(1-\rho(x^-))}dt.$$

As for $\mathbb{E}\{R(x)\}^{SRPT}$, this is simply the residence time of a job with service requirement x under SRPT. This is true since like a regular job, jobs already in the system once C_2 begins service are guaranteed to have remaining processing time over x, and therefore will not affect the residence time. Thus,

$$\mathbb{E}\{R(x)\}^{SRPT} = \int_0^{x^-}\frac{dt}{1-\rho(t^-)}.$$

Using (3) we get

$$\begin{aligned}&\mathcal{T}^{SRPT}(x) =\\ &\int_0^{x^-}\frac{\rho(x^-)-\rho(t)}{(1-\rho(t^-))\,(1-\rho(x^-))}dt + \int_0^{x^-}\frac{dt}{1-\rho(t^-)}\\ &\qquad = \int_0^{x^-}\frac{1-\rho(t)}{(1-\rho(t^-))\,(1-\rho(x^-))}dt,\end{aligned}$$

which for a continuous pdf $b(t)$ is simply

$$\mathcal{T}^{SRPT}(x) = \frac{x}{1-\rho(x)}.$$

3.8 Longest Remaining Processing Time (LRPT)

Under LRPT all jobs leave the system at the end of the busy period in which they arrive. Specifically, C_1 and C_2 leave the system simultaneously, leading to

$$\mathcal{T}^{LRPT}(x) = 0.$$

3.9 Round Robin (RR)

We analyze the RR policy with service quantum Δ where $\Delta \ll x$, and for simplicity we assume that service times are multiples of Δ. We use a model quite similar to the one described in [6, Sec 4.4], except that newly arriving jobs join the queue after the last arriving job.

As $\Delta \ll x$ we can be certain that even if C_1 begins service before a_2, it cannot finish a single quantum before a_2. Therefore the only jobs between C_1 and C_2 in the queue are jobs arriving in the interval (a_1, a_2) and C_1 will never be more than one service cycle ahead of C_2. Let $N(\epsilon)$ be a random variable denoting the number of jobs arriving or in the interval (a_1, a_2). Some of these jobs will have shorter service requirement than x, and therefore $\Delta \le Z(x,\epsilon) \le (N(\epsilon)+1)\Delta$.

However, note that $N(\epsilon)$ has an expected value is $\lambda\epsilon$, and therefore $\Delta \leq y(x,\epsilon) \leq \Delta + \lambda\epsilon$ leading to

$$\mathcal{T}^{RR}(x) = \lim_{\epsilon \to 0} y(x,\epsilon) = \Delta. \qquad (4)$$

Note that (4) holds also for other models of RR, e.g. when arriving jobs join the queue in other positions, though the analysis in some cases is somewhat more complicated.

4. DISCUSSION ON THE TWIN MEASURE OF SCHEDULING POLICIES

In this section we discuss the twin measures obtained in the previous sections. We start with proposing a classification and comparing the policies in each class. We then compare this classification to the classification provided by [13, 12] which we call the *Conditional Response Time Criterion*. We finalize with a discussion of the meaning of optimality under the twin measure.

For simplicity, we use the measures obtained for continuous service distributions. This is also convenient since [13, 12] deals mainly with continuous service distributions (although the results can probably be extended to the non-continuous case).

4.1 Classifying the Scheduling Policies

Definition 4.1 (Twin Measure Classification). *A scheduling policy ϕ will be called Absolutely Twin Predictable if $\mathcal{T}^{\phi}(x) = 0$ for every x.*

A scheduling policy ϕ will be called Strongly Twin Predictable if $\mathcal{T}^{\phi}(x) \leq x$ for every x.

A scheduling policy ϕ will be called Weakly Twin Predictable if $\mathcal{T}^{\phi}(x) < x/(1-\rho)$ for every x.

A scheduling policy ϕ will be called Not Twin Predictable if $\mathcal{T}^{\phi}(x) \geq x/(1-\rho)$ for at least one value of x.

The reason we use this classification will be made clear once we discuss the scheduling policies within each class.

4.1.1 Absolutely Twin Predictable

This class includes all policies for which twin jobs will leave the system simultaneously, namely PS, LAS and LRPT.

This clearly demonstrates that the twin measure has very little to do with the efficiency aspects of performance, as LAS is obviously much more efficient than PS.

It also demonstrates that the twin measure is not sufficient to guarantee sojourn time predictability. For example LRPT has notoriously unpredictable sojourn times.

4.1.2 Strongly Twin Predictable

This class includes policies for which the twin measure for a job of size x isn't larger than x. Observe that the minimum twin measure one can expect from any non-preemptive scheduling policy is x. One can see that FCFS achieves this optimal value for non-preemptive scheduling policies.

The other policy in this class is RR which in the limit $\Delta \to 0$ becomes PS.

4.1.3 Not Twin Predictable

LCFS has the largest measure analyzed, $x/(1-\rho)$. We conjecture that LCFS had the largest twin measures amongst non-preemptive scheduling policies. Note that the difference between the conjectured best and worst non-preemptive scheduling policies, FCFS and LCFS, can be extreme in cases where $\rho \to 1$.

4.1.4 Weakly Twin Predictable

Policies in this class include SJF, LJF and SRPT.

For small jobs, such that $\rho(x) < \rho - \rho(x)$, SJF has the lower measure, and the opposite for long jobs.

Interestingly, SJF and SRPT have the same measure. This can be explained in the following way. We call a job *intervening* if the job gets served in the interval (d_1, d_2), or (d_2, d_1) if C_2 is served first, and is not C_1 or C_2. Observe that under SJF a job of size x is only intervened by jobs of size $y < x$ arriving in a period of size x. In the non-preemptive case this period is the period in which the first job is served, while in the preemptive case this period is the period in which the second job is served.

Now consider SRPT. Observe a period of time of length dt in which the first job to be served was already served for t units of time. Jobs intervening in this period are jobs with size $t \leq y < x$. Observe a period of time of length dt in which the second job to be served was already served for t units of time. Jobs intervening in this period are jobs with size $y < t$. So in total, intervening jobs for every such interval dt are of size $y < t$. Now consider that the total length of such intervals of length dt is x, so the jobs intervening a job of size x under SRPT are also jobs of size $y < x$ arriving in a period of size x.

This is an interesting observation as it shows a similarity un-observed before between SRPT and SJF.

4.2 Comparing Predictability Criteria

In this section we compare the results of the twin measure classification with that of the Conditional Response Time Criterion. The Conditional Response Time Criterion was proposed in [13]. Further results are provided in [12], specifically for the case $\mathbb{E}\{X^3\} = \infty$.

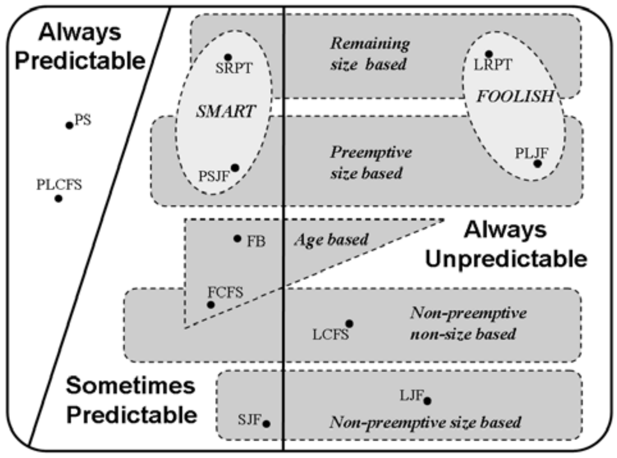

Figure 1: Classification According to the Conditional Response Time Criterion

We start by summarizing the Conditional Response Time Criterion in the settings and notation of this work.

Definition 4.2. *A job of size x is treated predictably under policy ϕ, service with pdf $b(x)$, and load ρ if the conditional variance in response time seen by a job of size x under* *policy ϕ,* $\mathrm{Var}\{\mathrm{T(x)}\}^{\phi}$*, follows*

$$\frac{\mathrm{Var}\{\mathrm{T(x)}\}^{\phi}}{x} \le \frac{\lambda \overline{x^2}}{(1-\rho)^3}.$$

A scheduling policy ϕ is predictable if every job size is treated predictably.

A scheduling policy ϕ is: (i) Always Predictable if ϕ is predictable under all loads and service distributions; (ii) Sometimes Predictable if ϕ is predictable under some loads and service distributions; and unpredictable under other loads and service distributions or (iii) Always Unpredictable if ϕ is unpredictable under all loads and service distributions.

Figure 1 summarizes the results of classifying common scheduling policies according to the Conditional Response Time Criterion.

Policy	Twin	Response Time
PS	Absolutely	Always
LAS	Absolutely	Sometimes
LRPT	Absolutely	Never
FCFS	Strongly	Sometimes
NP-LCFS	Not	Never
P-LCFS	Not	Always
NP-SJF	Weakly	Sometimes
P-SJF	Weakly	Sometimes
NP-LJF	Weakly	Never
P-LJF	Weakly	Never
SRPT	Weakly	Sometimes

Table 1: Comparing Predictability Criteria. The column "Twin" lists the class according to the Twin Measure Classification. The column "Response Time" lists the class according to the Conditional Response Time Criterion.

Table 1 compares the classification of policies according to the two criteria. A scheduling policy name starting with `NP` denotes the non-preemptive variant of the scheduling policy, while a name starting with `P` denoted the preemptive variant. Note that for continuous service distributions, our analysis has shown same results for preemptive and non-preemptive policies, which is not the case with the Conditional Response Time Criterion.

Only policies for which analysis was provided in both works are listed.

For some policies the two criteria agree. For example `PS` is both Absolutely Twin Predictable and Always Predictable. `NP-LCFS` is both Not Twin Predictable and Never Predictable. However, for some policies the criteria totally disagree. `LRPT` is Absolutely Twin Predictable, yet Never Predictable. $P-LCFS$ is Not Twin Predictable, yet Always Predictable. `LAS` is Absolutely Twin Predictable and Sometimes Predictable, but this predictability is only in the case $\mathbb{E}\{X^3\} = \infty$ (see [12]), so for most service distributions the criteria disagree.

This dissimilarity suggests that for guaranteeing predictability one might want to combine the two criteria, or require both.

4.3 Optimality under the Twin Measure

As several policies have zero twin measure, these policies can all be considered twin measure wise optimal. One might therefore want to find a policy that is both optimal in the twin measure sense, and has low sojourn time. Consider the following variant of `SRPT`, called $\mathtt{SRPT}_\alpha$. Assume the job with the shortest remaining processing time has remaining processing time x. Service is given in a processor sharing manner, to all jobs with remaining processing time not larger than $x+\alpha$. It is easy to see that if α can be arbitrarily small, this policy's sojourn times are arbitrarily close to `SRPT`. On the other hand, if $\alpha \ge \epsilon + \delta$ (which since $\epsilon \to 0, \delta \to 0$ can be done for arbitrarily small α), twin jobs will be served in processor sharing manner from a_2 until $\min(d_1, d_2)$, at which point one of them will leave the system and the other one will have at most $\epsilon + \delta$ service left. As $\epsilon \to 0, \delta \to 0$, this job will receive full service from that epoch onwards, and thus $Z(x, \delta, \epsilon) \le \epsilon + \delta$ and $\mathcal{T}^{SRPT_\alpha}(x) = 0$. Thus $\mathtt{SRPT}_\alpha$ has both optimal twin measure, and optimal sojourn time.

5. THE TWIN MEASURE IN MULTI-SERVER SYSTEMS

In this section we analyze the twin measure in some common multi-server settings. The scheduling policy between members of the same queue is `FCFS`.

5.1 Single Queue

We start our analysis with the simple single queue system, denoted `SingleQueue`, where the first job in the queue is served by the first server to become idle.

For simplicity we ignore effects caused by jobs arriving in the interval (a_1, a_2) in this analysis. As the scheduling policy is `FCFS`, it is easy to see that these effects would be negligible as $\epsilon \to 0$.

A simple observation is that the twin measure is smaller than $x + \epsilon$, or x when $\epsilon \to 0$. This is so because at the worse case (i) C_2 will be served right after C_1. However, there is a probability that either (ii) both twins arrive when two or more servers are idle, in which case both twins are served immediately, and leave the system simultaneously, or (iii) some other server than the one serving C_1 will become idle while C_1 is being served, in which case C_2 will be served partially in parallel to C_1, and leave system before $d_2 + x$. Letting $\epsilon \to 0$ and taking expectations we have

$$\mathcal{T}^{SingleQueue}(x) = (1-\alpha)\left(\int_0^x \beta(y) y dy + \gamma(x) x\right), \quad (5)$$

where α is the probability of arriving when two or more servers are idle. Given that no more than one server was idle on arrival, $\beta(y)$ is the probability that C_2 will be delayed for y units of time until another server is idle, and $\gamma(x)$ is the probability that no other server will be idle until C_1 is served, in which case C_2 is delayed for exactly x units of time.

For α we have

$$\alpha = \sum_{k=0}^{s-2} p_k,$$

where p_k is the probability of finding k jobs in the system. For example, in the case of Exponential service times ([5, Sec. 3.5]),

$$p_k = p_0 \frac{(sp)^k}{k!}, k \le m$$
$$p_0 = \left[\sum_{k=0}^{m-1} \frac{(s\rho)^k}{k!} + \frac{(s\rho)^s}{s!(1-\rho)}\right].$$

In the general distribution case one can map the distribution to a PH distribution (for example using [10]) and use matrix analytic methods ([7]) to obtain a good approximations of p_k.

For evaluating $\beta(y)$ and $\gamma(x)$ note that due to the Poisson arrivals the remaining service time on each of the servers at the epoch s_1 is the residual life of the service time, and has a pdf $\hat{b}(x) = (1 - B(x))/\bar{x}$ and a cdf $\hat{B}(x) = \int_0^x f(t)dt$. For C_2 to be delayed exactly y units of time due to one specific server, the other servers need to have a residual service time larger than y, so the probability of that event is $\hat{b}(y)(1-\hat{B}(y))^{s-2}$, and finally $\beta(y) = (s-1)\hat{b}(y)(1-\hat{B}(y))^{s-2}$. Using the same argument $\gamma(x) = (1 - \hat{B}(y))^{s-1}$.

To summarize

$$\mathcal{T}^{SingleQueue}(x) = \left(1 - \sum_{k=0}^{s-2} p_k\right) \times$$
$$\left(\int_0^x (s-1)\hat{b}(y)(1-\hat{B}(y))^{s-2} y dy + (1-\hat{B}(y))^{s-1} x\right).$$

One can observe that the twin measure is decreasing with s, as both α and $\beta(y)$ increase with s.

5.2 Multiple Queues

In this section we make some observation about the multiple queue system, where each queue is assigned one server, and that server serves the jobs in that queue in FCFS manner. If a job joins the system and finds an empty queue it joins that queue and is served immediately. Other wise, the job is assigned a queue using some queue assignment policy. Once a job is assigned a queue, it cannot jockey to another queue, even if the other queue's server is idle. We denote this setting MultipleQueue.

Note that analysis is dependent on the queue assignment policy. However, for all queue assignment policies there are three possible cases: (i) Two or more queues are empty upon arrival of the twins. In this case both twins are served in parallel and leave the system simultaneously. (ii) Both twins join the same queue. In this case C_2 departs x units after C_1, and (iii) the twins join different queues. In this case the twins will depart $|y|$ units of time apart, where y is the difference in the remaining work in the two queues, which can be negative.

Using very similar notation to (5)

$$\mathcal{T}^{MultipeleQueue}(x) = (1-\alpha)\left(\int_0^x \beta(y)|y|dy + \gamma(x)x\right),$$

where α is the probability of arriving when two or more queues are empty. Given that no more than one queue was empty, $\beta(y)$ is the probability that the difference in the remaining work in the two queues C_1 and C_2 joined is y. $\gamma(x)$ is the probability that the twins will join the same queue.

Evaluating α, $\beta(y)$ and $\gamma(x)$ is much more complicated in this case than it was for SingleQueue. We state some methods for this evaluation below. One can also resort to simulation methods, which are in fact not very complicated.

Evaluating α, after the general distribution is mapped to a PH distribution, involves a Markovian-chain which is infinite in more than one dimension. In these cases matrix analytic methods do not work and one needs to use other methods, such as the Dimensionality Reduction method proposed in [9], and see discussion there of other methods. This provides us with full state probabilities.

Analyzing $\beta(y)$ requires knowledge of the distribution of remaining work in a queue. This is in general much more complicated than the residual life of a single job, although it is possible, using the queue length distribution, which is obtainable from the state probabilities.

$\gamma(x)$ might be simple or complicated, depending on the job assignment policy. For example if the jobs are assigned to queues in random, $\gamma(x) = 1/s$. If jobs are assigned to the shortest queue, the queue length distribution can be used.

One queue assignment policy of interest is to assign jobs to the queue with the least remaining work. Although this joining policy might not always be practical, it is possible in some computing systems where the length of each job is predetermined. Note that in this case C_1 always departs first, and C_2 always departs x or less units of time after C_1. One can therefore easily observe that the twin measure is smaller than x. In general, this is not possible in other queue assignment policies.

6. EXTENDING THE TWIN MEASURE

In this section we discuss two ways to extend the twin measure: using more than two jobs, which we call *trains*, and not sending the jobs simultaneously.

6.1 Job Trains

One way to extend the results of the twin measure is to consider *Job Trains*, i.e. situations where more than two identical jobs are sent.

Packet Trains were proposed as means for measuring link bandwidth and available bandwidth (e.g. [2, 3, 4]). In these, packet trains are injected into the network. The dispersion of the probe packets at the receiver side is then used in different bandwidth estimation algorithms, using, for example, dispersion mean values or dispersion variance. However, this entire body of work assumes that packets are served using the FCFS policy, as indeed is the case with packet routers, at least for packets of the same flow. However, job schedulers may choose different policies, with quite different results.

We provide here only the simplified definition (parallel to Definition 2.2), as the non-simplified one is much more difficult to define rigorously and provides little benefit.

DEFINITION 6.1 (JOB TRAIN MEASURE). *Let $C_1, C_2, \ldots, C_m$ be m identical jobs, with equal service requirements x, arriving at epochs $a_1, a_2 = a_1 + \epsilon, a_3 = a_1 + 2\epsilon, \ldots, a_m = a_1 + (m-1)\epsilon$ where $\epsilon > 0$, and departing at epochs $d_1, d_2, \ldots, d_m$. Assume that C_1 arrives when (a_1) the system is in steady state. Define the random variable $Z(x, \epsilon, m) = \max_i d_i - \min_i d_i$, given x, m and ϵ. Let $z^n(x, \epsilon, m)$ be the n-th moment of $Z(x, \epsilon, m)$, i.e. $z^n(x, \epsilon, m) = \mathbb{E}\{Z(x, \epsilon, m)^n\}$.*

For scheduling policy ϕ and job size x, the n-th job train measure $\mathcal{T}_n^\phi(x, m)$ is defined as the limit, when ϵ tends to zero, of $z^n(x, \epsilon, m)$, assuming a limit exists. Namely $\mathcal{T}_n^\phi(x, m) = \lim_{\epsilon \to 0} z^n(x, \epsilon, m)$.

The shortened term *job train measure*, denoted $\mathcal{T}^\phi(x, m)$, is used to describe the first job train measure, namely $\mathcal{T}^{\phi}(x,m) = \mathcal{T}_1^{\phi}(x) = \lim_{\epsilon\to 0}\mathbb{E}\{Z(x,\epsilon,m)\}$.

Analysis of the job train measure for the policies analyzed in Section 3 is quite straightforward. In fact, for all the policies $\mathcal{T}^{\phi}(x,m) = (m-1)\mathcal{T}^{\phi}(x)$. However, this isn't always the case.

Consider for example a synchronous server setting. All jobs are of size x and the server works in service cycles of length $2x$ which serve either one or two customers, depending on availability of jobs. Each job receives a service rate of 1/2. If p_{odd} is the probability of finding an odd number of customers in the system upon arrival, then the job train measure is $p_{odd}2x\lceil(m-1)/2\rceil+(1-p_{odd})2x\lceil(m-2)/2\rceil$ which is definitely not linear with m. For example, for $m=2$ we get $p_{odd}2x$, for $m=3$ we get $2x$.

A second example is a round robin policy, where jobs join the queue at a random location. One can observe that all the jobs in the job train will be served in the same service cycle, so the job train measure is at most as large as the service cycle length. This will grow with m, but not linearity.

6.2 Non-Simultaneous Twins

A second way to extend the results of the twin measure is to consider twins which do not arrive at the system concurrently. For example, the twins can arrive exactly x units apart, i.e. $a_2 - a_1 = x$. This might provide more insight on the predictability of the system. For example, note that only for `LRPT` the measure is still zero.

Another case of specific interest is the case where the second twin enters the system when the first one departs, i.e. $a_2 = d_1$. For example this setting can represent a customer refreshing a Web page just as it finished loading, expecting a similar load time. It might be more interesting to measure the difference or ratio between the sojourn times, i.e. $|(d_1 - a_1) - (d_2 - a_2)|$ or $\max(d_1 - a_1, d_2 - a_2)/\min(d_1 - a_1, d_2 - a_2)$.

7. CONCLUDING REMARKS

We proposed a novel way of measuring the predictability of a queueing system, utilizing pairs of identical customers. This measure was analyzed for several common scheduling policies. The results show that the measure in itself is unrelated to the utilization or waiting time performance of the system. The results also suggest that the measure in itself cannot guarantee service time predictability. Comparing these results with the results of the Conditional Response Time Criterion show that in several cases they contradict, leading to the proposal that to guarantee predictability maybe both of the measures should be used as criteria.

Analysis of the measure for multiple servers shows that for single queue the measure improves when the number of server increases. For multiple queues this is dependent on the queue assignment policy.

We proposed several ways in which the measure can be extended, namely by using more than two customers, or by considering jobs that do not arrive concurrently. Analysis of these is left for future research.

8. ACKNOWLEDGEMENTS

This work was supported in part by the Israeli Ministry of Science and Technology, grant number 380-801, and by EURO-NGI network of excellence

9. REFERENCES

[1] B. Avi-Itzhak, H. Levy, and D. Raz. Quantifying fairness in queueing systems: Principles, approaches and applicability. *Probability in the Engineering and Informational Sciences (PEIS)*, 2006. To appear.

[2] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27-28:297–318, October 1996.

[3] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, August 2003.

[4] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Transactions on Networking*, 11(4):537–549, August 2003.

[5] L. Kleinrock. *Queueing Systems, Volume 1: Theory.* Wiley, 1975.

[6] L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications.* Wiley, 1976.

[7] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling.* ASA-SIAM, Philadelphia, 1999.

[8] M. Nuyens and A. Wierman. The foreground-background queue: a survey. Under submission, 2007.

[9] T. Osogami. *Analysis of Multiserver Systems via Dimensionality Reduction of Markov Chains.* PhD thesis, School of Computer Science, Carnegie Mellon University, 2005.

[10] T. Osogami and M. Harchol-Balter. Closed-form solutions for mapping general distributions to quasi-minimal PH distributions. *Performance Evaluation*, 63(6):524–552, June 2006.

[11] L. E. Schrage and L. W. Miller. The queue M/G/1 with the shortest processing remaining time discipline. *Operations Research*, 14:670–684, 1966.

[12] A. Wierman. *Scheduling for Today's Computer Systems.* PhD thesis, School of Computer Science, Carnegie Mellon University, 2007.

[13] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to higher moments of conditional response time. In *Proceedings of ACM Sigmetrics 2005 Conference on Measurement and Modeling of Computer Systems*, pages 229–239, Banff, Alberta, Canada, June 2005.