CS 162 Operating Systems **Fall 2017**

Queuing Theory - Little's Law

Lecturer: Prof. Stoica Scribe: Antares Chen

1 Introduction

When we began studying queuing theory, we asked how to determine a queue's average waiting time, average length, and the average processing rate. Despite the simplicity of these questions, they still act as informative metrics on the performance of our system: we would hope that the average processing rates for lines at Disney Land rides are relatively high! First introduced by [1] and proven in [2], Little's Law provides an elegant answer to these three questions. To that end, these notes will discuss the theorem, its applications, and its proof.

2 Queueing Systems

We will concern ourselves with *queueing systems* which are devices that processes discrete items in some fashion. The behavior of this system is as follows:

Queueing System:
The following occurs for each time step $t = 1, \ldots, T$:

1. Some number of items *arrive* at an unknown rate to the system.
2. Items are then placed in one or more queues within the system.
3. Items must be serviced from the queue(s) before *departing* from the system.

It is important to note what this model does not assume. First, it treats the servicing process as a black box. The system makes no assumptions on how many queues there are, in what order each queue is serviced, and how long it takes to service them. Additionally, we know not what state the items are in while they are inside the system. They may be waiting in queues, are being serviced, or a combination of both.

What items are will also depend on the context of the problem. For example, if we model the line for Splash Mountain as a queueing system, then the items would be the riders. Riders would arrive at the line, and depart the line when they are allowed to enter into the ride's boat. This example explicitly contains a queue that riders wait in, but we can model other scenarios as queueing systems as well.

Suppose your friend enjoys consuming wine on Wednesdays. Occassionally, your friend will stop by the local liquor store and purchase a few bottles from his favorite winery, take them home, and store them in his cellar until they are uncorked on a particularly celebratory Wednesday. In the case that we are interested in determining how long your friend's wine is allowed to age, we may model this process as a queueing system. The items would be bottles of wine, which would arrive at the system (the cellar), and then would depart upon their retrieval for consumption. Notice that in this case, the wine bottles can be retrieved in any order – not necessarily in the order for which they are bought.

2.1 Importance

Why would this model be useful for the study of operating systems? When we design an operating system, it is important that we also measure the performance of its components. Queueing systems provide a natural method of modeling many components of an operating system. A thread scheduler is a queueing system in that threads enter the

waiting queue and are processed in some order. Since there are many ways that we can schedule threads (round robin, MLFQS, shortest remaining time first, etc.) it may make sense for us to consider the scheduler as a black box to calculate performance metrics.

We can also consider caches as a queueing system where the items are data being cached. Upon retrival from disk, the data arrives into the system and is placed in a cache line. The data is then processed (evicted from cache) based on some cache replacement policy upon which they then depart from the system. Similar to thread schedulers, we may wish to treat the cache replacement policy as a black box as there are many different policies: least recently used, clock algorithm, first in first out, etc.

3 Little's Law

Given a queueing system, there are three questions we can ask:

1. What is the average number of items in the system?
2. What is the average waiting time for each item that has entered the system?
3. What is the average processing rate for the system?

Provided very sparse assumptions regarding the queuing system and answers to two of the above three questions, Little's Law will allow us to easily calculate the third value. The statement of Little's Law is decievingly simple.

*G*iven a queueing system, let L denote the average number of items in the system, λ the average processing rate, and W the average waiting time for each item. If there exists a steady state, then the following holds:

$$L = \lambda W$$

It's interesting to think about this law in relation to Stoichiometry. Recalling a bit of chemistry, the stoichiometric ratio measures the amount of regeant formed from a chemical reaction. The calculations themselves rely upon conservation of mass: matter cannot be created or destroyed in a closed system.

Our queueing system exhibits the same conservation of "mass." Items that enter the system cannot magically disappear, nor can they magically appear inside the system – the items themselves must be conserved! To that end, we may look at the units for each of the quantities L, λ, W as if performing a stoichiometry calculation and note that both sides match in units as they are supposed to.

$$L \text{ items} = \frac{\lambda \text{ items}}{1 \cancel{\text{ unit time}}} \cdot W \cancel{\text{ unit time}}$$

Yet, one key difference between this statement and Stoichiometry is that not all items need be processed within our (potentially infinite)time frame. Furthermore, this linear relation is surprisingly simple for the lack of assumptions we make regarding the behavior of the queueing system. Regardless of the distribution of arriving items and how the system chooses to process them, it always holds that $L = \lambda W$ if there exists a steady state!

Little's Law is fairly useful for "back of the envelope" calculations. Usually two of the three questions above are easy to estimate in a system. The third can then be quickly derived using Little's Law. Consider the following examples:

3.1 Wine Wednesdays

Recall your friend who enjoys consuming wine on Wednesdays. Wine tastes better as it ages, thus he has decided to track how long his bottles of wine get to age in the cellar before they are consumed. Suppose the cellar can hold 240 bottles of wine. He estimates that at any given moment, the cellar seems to be roughly $\frac{2}{3}$ full. After going through previous receipts, your friend also finds that he buys about 8 bottles from the Liquor store per month.

As previously mentioned, we can treat the cellar as a queuing system. Our goal is then to find the average amount of time an item (bottle of wine) remains in the system (cellar). The description above informs us that the average

number of items in the queue is $L = \frac{2}{3}(240) = 160$ bottles of wine. Additionally, the arrival rate is $\lambda = 8$ bottles per month. Using Little's Law, we can calculate the following.

$$L = \lambda W \qquad \iff \qquad 160 = 8W \qquad \iff \qquad W = 20$$

Here we find that average amount of time a bottle gets to age is $W = 20$ months or 1 year and 8 months.

3.2 Oakland Bay Bridge

You forgot that you were supposed to fly home for Thanksgiving and must now rush to SFO. However, right as you get to the Oakland Bay Bridge, you get stuck at a seemingly endless line in front of the toll booth. Let us estimate how much time you will need to wait before you can pay your toll.

The California Department of Transportation estimates that 270,000 vehicles cross the Bay Bridge every day. The Bay Bridge also has two types of toll lanes. Drivers that have the California FasTrak can drive through the booth at 30 mph and sensors will automatically collect the toll. Otherwise, they will need pull up to a booth window and manually pay the fee. You estimate that it only takes drivers with FasTrak 10 seconds to drive through the automated booth, while other drivers spend roughly 2 minutes finding exact change for the toll. If roughly $\frac{1}{3}$ drivers have a FasTrak device, what is the average amount of time you will need to wait until you reach the toll booth?

The first thing to do is model this as a queuing system. We let the items be drivers. An item arrives when the driver enters a toll booth queue and is processed when the driver reaches the toll booth and pays the toll. It is critical to also understand that details that may *seem* to add complexity to our queuing model does not actually affect how we calculate the average waiting time.

For example, it is not important whether or not you have a FasTrak device as the queuing model makes no assumptions on the distribution of the type of items arriving into the queuing system. Additionally, the Oakland Bay Bridge has many toll booth lanes, but the number of such lanes also does not affect the calculation of the average wait time. This is because the queuing model is agnostic towards the internal details of how the system processes items.

Now to actually perform the calculations, notice that the average arrival rate is $\lambda = 270,000$ cars per day or $\lambda = 3.125$ cars per second. We also have the average wait time.

$$W = \frac{1}{3}(10) + \frac{2}{3}(120) = \frac{250}{3}$$

It is important to use the correct units. Since the most granular unit of time is the seconds it takes for a FasTrak driver to pass through the toll, we choose to represent W in seconds. However, this means that we need to represent 2 minutes in seconds and λ as a value of items per second. We can now use Little's Law to derive the following.

$$L = \lambda W \qquad \iff \qquad L = \frac{250}{3}(3.125) = 260.416$$

Thus $L = 260.416$ seconds or roughly 4 minutes and 21 seconds.

4 Proving Little's Law

We now prove Little's Law. Let i index the i-th arriving item to the system and denote t_i as its time of entry and W_i as the time it spends within the system. It then follows that the completion time, when i exits the system is:

$$c_i = t_i + W_i$$

Observe that since W_i is arbitrary, the i-th departing item need not be the same as the i-th arriving item. Now, define $N(t)$ as the number of items that has arrived in the system up to time t, and $D(t)$ as the number of departed items. Before showing the proof, we need to make formal what we mean by "steady-state." Since we may potentially need to work over an infinite time window, the average number of items, waiting time for each item, and the processing rate of the system can be expressed via a limit as time approaches infinity.

For the average number of items within the system, define $L(t)$ to be the number of items in the system at time t. By the Mean Value Theorem, the average number of items is then given by:

$$L = \lim_{t\to\infty} \frac{1}{t} \int_0^t L(s)\, ds$$

For the average wait time and processing rate, we let:

$$\lambda = \lim_{t\to\infty} \frac{N(t)}{t} \qquad\qquad W = \lim_{t\to\infty} \frac{1}{N(t)} \sum_{i=1}^{N(t)} W_i$$

If the system reaches a steady-state then these limits should converge to a constant. We define each of these limits to be the constant that they equate to above.

4.1 The Proof

Theorem 1 (Little's Law)**.** *Given a queueing system, where the average number of items in the system, average processing rate, and average waiting time for each item converges to L, λ, W respectively. Then...*

$$L = \lambda W$$

Proof. We begin by calculating $L(s)$ given fixed s. At time s, an item i is in the system if and only if it arrives on or before s and departs after s. That is $t_i \le s < d_i$, so if we let $\mathbb{I}\{d_i > s\}$ denote the indicator variable for $d_i > s$, the value of $L(s)$ is exactly.

$$L(s) = \sum_{i:t_i\le s} \mathbb{I}\{d_i > s\}$$

However, the departure time of i is $d_i = t_i + W_i$, thus the above is equivalent to:

$$L(s) = \sum_{i:t_i\le s} \mathbb{I}\{t_i + W_i > s\} = \sum_{i:t_i\le s} \mathbb{I}\{W_i > s - t_i\}$$

Now to calculate the total number of arrivals up to time t, we integrate from $0 \le s \le t$:

$$\int_0^t L(s)\, ds = \int_0^t \Big(\sum_{i:t_i\le s} \mathbb{I}\{W_i > s - t_i\} \Big)\, ds$$

It would be nice if we could express the above as a sum of integrals rather than the other way around. Observe that the terms of the sum are index over $i : t_i \le s$. Since we integrate from $0 \le s \le t$, the indices are the sum equate to $i : t_i \le s \le t$ or just $i : t_i \le s$. Hence, the equation above admits:

$$\begin{aligned}
\int_0^t L(s)\, ds &= \int_0^t \Big(\sum_{i:t_i\le s} \mathbb{I}\{W_i > s - t_i\} \Big)\, ds \\
&= \int_0^t \Big(\sum_{i:t_i\le t} \mathbb{I}\{W_i > s - t_i\} \Big)\, ds \\
&= \sum_{i:t_i\le t} \Big(\int_{t_i}^t \mathbb{I}\{W_i > s - t_i\} \Big)\, ds \\
&= \sum_{i:t_i\le t} \Big(\int_0^{t-t_i} \mathbb{I}\{W_i > s\} \Big)\, ds \\
&= \sum_{i:t_i\le t} \min\{W_i, t - t_i\}
\end{aligned}$$

To see why the last line follows, consider the figure below. In the case where $W_i \le t - t_i$ then the integral is exactly W_i since for $s \le W_i$, we have $\mathbb{I}\{W_i > s\} = 1$. Conversely, the integral evaluates to $t - t_i$ since for $W_i > t - t_i$, we have $\mathbb{I}\{W_i > s\} = 1$ for all $s \le t - t_i$.

We can then bound the total number of arrivals into the system up to time t via the following inequality.

$$\sum_{i:c_i \le t} W_i \le \int_0^t L(s)\, ds \le \sum_{i=1}^{N(t)} W_i \tag{1}$$

The right side of this inequality follows immediately as:

$$\int_0^t L(s)\, ds = \sum_{i:t_i \le t} \min\{W_i, t - t_i\} \le \sum_{i:t_i \le t} W_i$$

For the left inequality, observe that we can split the sum into two: one over indices where items arrive and depart before t and another where items depart after t.

$$\begin{aligned}
\int_0^t L(s)\, ds &= \sum_{i:t_i \le t} \min\{W_i, t - t_i\} \\
&= \sum_{i:c_i \le t} \min\{W_i, t - t_i\} + \sum_{i:t_i \le t, c_i > t} \min\{W_i, t - t_i\} \\
&= \sum_{i:c_i \le t} W_i + \sum_{i:t_i \le t, c_i > t} (t - t_i) \\
&\ge \sum_{i:c_i \le t} W_i
\end{aligned}$$

Now if we multiply inequality 1 by $\frac{1}{t}$ and take the limit as $t \to \infty$, we derive the following on the RHS.

$$\begin{aligned}
\lim_{t\to\infty} \frac{1}{t} \int_0^t L(s)\, ds &\le \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{N(t)} W_i \\
&= \lim_{t\to\infty} \frac{N(t)}{t} \Big(\frac{1}{N(t)} \sum_{i=1}^{N(t)} W_i \Big) \\
&= \lambda W
\end{aligned}$$

The left hand side is more challenging to demonstrate so we omit the proof of $\lambda W \le \lim_{t\to\infty} \frac{1}{t} \sum_{i:c_i \le t} W_i$. We now have that

$$\lambda W \le \int_0^t L(s)\, ds \le \lambda W \qquad \iff \qquad \lambda W \le L \le \lambda W$$

Thus $L = \lambda W$ as required. □

4.2 Final Remarks

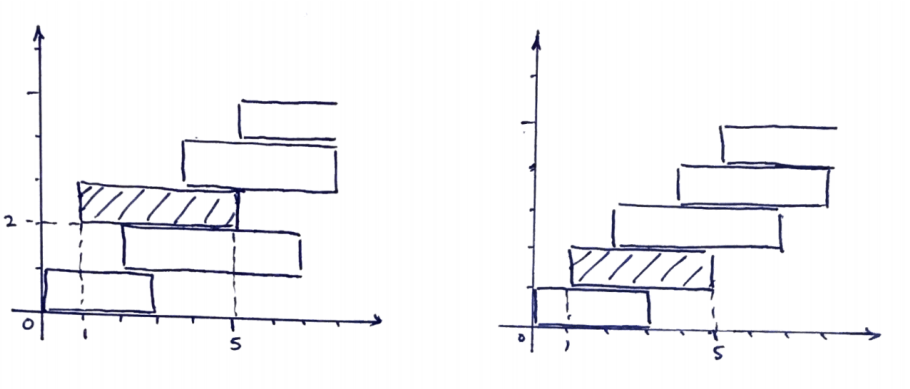
Recall back to when we demonstrated

$$\int_0^t L(s)\, ds = \sum_{i:t_i \le t} \min\{W_i, t - t_i\}$$

It may seem that like this resulted from a cute mathematical trick which interchanged the integrand and summation, but there is a more intuitive way too look at this equality. Determining the area under $L(t)$ can be done by summing vertical rectangles between a and b, which is exactly what a Riemannian integral like $\int_a^b L(t)\, dt$ does.

Since $L(t)$ is a step function, we can calculate this a discrete sum over time steps $a \le t \le b$. This sum can either be expressed as a discrete sum using vertical or horizontal rectangles. Expressing the sum using vertical rectangles would yield no additional information as the integral already does that. Instead, let us try to represent this using horizontal rectangles.

Suppose we graph the arrival process for a system in the following manner. Let the x-axis represent time and the y-axis represent the value of an index i. For the i-th arriving element, we plot a horizontal rectangle that is 1 unit in height starting at $y = i$ and W_i units in length starting at $x = t_i$. An example graph may look like that in the figure below. The highlighted block on the left represents job 2 arriving at time $t = 1$ and departing at time $t = 5$. When the blocks are sorted by arrival time, job 2 becomes indexed to job 1.

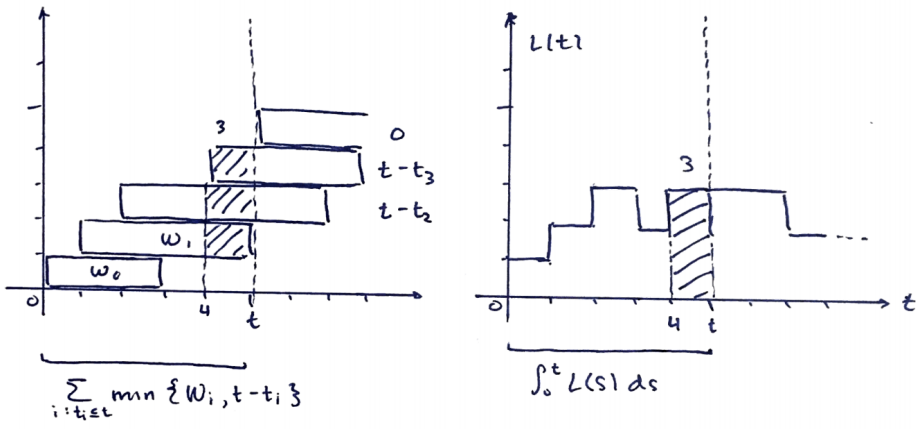


The horizontal rectangles in this plot can then be such that they are ordered by their arrival-time in the following manner while still preserving the total area within the sum of the rectangles. Notice that if we compute the sum of the areas within rectangles between times 0 to t, considering horizontal rectangles admits three different cases:

1. All items i with blocks that begin and end before t contribute exactly W_i to the sum.
2. All items i with blocks that begin before t and end after t contribute exactly $t - t_i$ to the sum.
3. All items i with blocks that begin after t contribute 0 to the sum.

These three cases imply that the sum of areas is exactly $\sum_{i:t_i \leq t} \min\{W_i, t - t_i\}$. Let us now consider vertical rectangles for this plot. Notice that the vertical slice starting at $x = t$ to $x = t + 1$ exactly determines how many items are in the system at time t – that is it is equivalent to $L(t)$. Thus the sume of areas from 0 to t is exactly $\int_0^t L(s)ds$.

In the figure below, notice that the horizontal blocks that end before t contribute W_i while the horizontal blocks intersection t contribute $t - t_i$. Finally the only block to the left of t, contributes 0. Additionally, the slice between time 4 and 5 has area 3 which thus the right graph denotes $L(4) = 3$.



The above constitutes two ways to compute the same area within these rectangles. It must be that these two quantities are equal!

$$\int_0^t L(s)\, ds = \sum_{i:t_i \leq t} \min\{W_i, t - t_i\}$$

From this discussion, we then get a nice way of looking at the calculations we performed in the proof for theorem 1. L is dependent on the area underneath the curve $L(t)$. By interchanging the integrand and summation, we switch the way we calculate this area from summing vertical rectangles to horizontal rectangles. Using a sum of horizontal rectangles then allows us to more easily bound upper and lower-bound the area underneath $L(t)$ such that we can squeeze the quantity between two sums in equation 1. After taking the limit and some algebraic transformation, we are able to squeeze L between λW thus showing $L = \lambda W$.

References

[1] Alan Cobham. Priority assignment in waiting line problems, *Journal of the Operations Research Society of America*, 1954, 70–76.

[2] John D.C. Little. A proof for the queuing formula: $L = \lambda W$, *Operations research*, 383–387, 1961.