

## Lecture 8: Transforms for Scheduling Policies

Scheduling is an extremely important topic in computer systems. Changing the scheduling policy can improve mean response time by an order of magnitude, without requiring any additional resources. Sometimes the goal is not improving mean response time, but rather the variability of response time. Here scheduling yields some counterintuitive results as well.

In this lecture we will derive the Laplace transform of response time for an M/G/1 queue under various scheduling policies. Every scheduling policy we consider will be work-conserving, i.e., whenever there is a job to be worked on, some job will receive service. The topics are ordered so that the sections increase in difficulty. We will make use of everything that we have learned up until now about transforms. Hang in there!

### 1 Announcements

1. Looking for a TA for Fall 2007 for 15-857a (same as 15-849a). Very little work is involved compared to most TA jobs. Helping with grading on 6-7 assignments, one or two office hours a week, and a little bit of guest teaching.
2. Reminder: Midterm next week during class. Closed notes. Yes, know your transforms by heart!

### 2 Homework Solutions from Last Time

1. **z-transform of  $N_Q$  for M/G/1:**

We express

$$\widehat{N}_S(z) = \sum_{i=0}^{\infty} \pi_i z^i$$

where

$$\pi_i = \mathbf{Pr} \{i \text{ jobs in system}\}.$$

Hence we have:

$$\begin{aligned}
\widehat{N}_Q(z) &= \pi_0 z^0 + \sum_{i=1}^{\infty} \pi_i z^{i-1} \\
&= \pi_0 + \frac{1}{z} \sum_{i=1}^{\infty} \pi_i z^i \\
&= \pi_0 + \frac{1}{z} \left( \widehat{N}_S(z) - \pi_0 \right) \\
&= (1 - \rho) \left( 1 - \frac{1}{z} \right) + \frac{1}{z} \widehat{N}_S(z)
\end{aligned}$$

Here, we have used the fact that  $\pi_0 = 1 - \rho$ .

2. **M/G/1 with initial setup costs:** In this model the first job to start each busy period experiences an initial cost  $I$  before its service is started. The service times of jobs will still be denoted by  $S$ .

We'll follow the M/G/1  $\tilde{T}_S$  derivation from lecture, working from the embedded DTMC. This goes almost exactly as in class, except the transition probabilities for leaving the 0 state are different to reflect the initial cost. Let

$$a_j = \mathbf{Pr} \{j \text{ arrivals in } S \text{ seconds}\}.$$

Let

$$a'_j = \mathbf{Pr} \{j \text{ arrivals in } S + I \text{ seconds}\}.$$

Then  $P_{ij} = a_{j-i+1}$  for  $i \geq 0$  and  $P_{0j} = a'_j$ . So we can write

$$\begin{aligned}
\pi_j &= \pi_0 a'_j + \sum_{i=1}^{j+1} \pi_i a_{j-i+1} \\
\sum_{j=0}^{\infty} \pi_j z^j &= \pi_0 \sum_{j=0}^{\infty} a'_j z^j + \sum_{j=0}^{\infty} \sum_{i=1}^{j+1} \pi_i a_{j-i+1} z^j \\
\tilde{N}_s(z) &= \pi_0 \hat{A}_{S+I}(z) + \frac{1}{z} \sum_{i=1}^{\infty} \pi_i z^i \sum_{j=i-1}^{\infty} a_{j-i+1} z^{j-i+1} \\
&= \pi_0 \hat{A}_{S+I}(z) + \frac{1}{z} (\tilde{N}_s(z) - \pi_0) \hat{A}_s(z) \\
\Rightarrow \hat{N}_s(z) &= \pi_0 \frac{z \hat{A}_{S+I}(z) - \hat{A}_s(z)}{z - \hat{A}_s(z)}.
\end{aligned}$$

To get  $\pi_0$ , we evaluate the limit of this as  $z \rightarrow 1$ , set the result equal to 1, and solve to find  $\pi_0 = \frac{1 - \lambda \mathbf{E}\{S\}}{1 + \lambda \mathbf{E}\{I\}}$ .

Also, we can use the fact that  $\hat{A}_S(z) = \tilde{S}(\lambda - \lambda z)$  for any  $S$  and change variables ( $s = \lambda - \lambda z$ ) to write

$$\tilde{T}_S(s) = \pi_0 \frac{(1 - s/\lambda) \tilde{S}(s) \tilde{I}(s) - \tilde{S}(s)}{1 - s/\lambda - \tilde{S}(s)}.$$

So

$$\tilde{T}_Q(s) = \frac{1 - \lambda \mathbf{E}\{S\}}{1 + \lambda \mathbf{E}\{I\}} \cdot \frac{\lambda - (\lambda - s)\tilde{I}(s)}{s - \lambda + \lambda \tilde{S}(s)}.$$

### 3 Non-preemptive, non-size-based policies

We will start by studying non-preemptive scheduling policies.

**Definition 1.** A **non-preemptive service order** is one which does not preempt a job once it starts service, i.e., each job is run to completion.

**FCFS:** When server frees up, it always chooses the job at head of queue to be served and runs it to completion.

**LCFS-non-preemptive:** When server frees up, it always chooses last job to arrive and runs it to completion.

**Random:** When the server frees up, it chooses a random job to run next.

**Question:** When would one use LCFS-non-preemptive?

**Answer:** Consider the situation where arriving jobs get pushed on a stack, and therefore it is easiest to access that job which arrived last, e.g. the task at the top of the pile on my desk!

**Question:** Which of the above 3 non-preemptive policies do you think has the best mean response time or mean waiting time?

**Answer:** It seems like FCFS should have the best mean response time because jobs are serviced most closely to the time they arrive, whereas LCFS may make a job wait a very long time. However, it turns out that all the above policies have exactly the *same* mean response time.

In fact, an even stronger statement can be made:

**Theorem 1 (Conway, Maxwell, Miller Section 8-5).** *All non-preemptive service orders that do not make use of job sizes have the same distribution on the number of jobs in the system.*

**Corollary 1.** *All non-preemptive service orders that do not make use of job sizes have the same  $\mathbf{E}\{N_S\}$  and the same  $\mathbf{E}\{T_S\}$ .*

**Question:** Any ideas for how the proof for Theorem 1 might go?

**Hint:** It will help to first recall the semi-Markov formulation for the M/G/1/FCFS queue. The idea is to look at the M/G/1 queue just at the point of departures. The “ $k$ th state” is the number of jobs that the  $k$ th departure leaves behind.

Let the state be the number of jobs in M/G/1 system at time of the last departure. The sequence of states  $\{X_i, i \geq 0\}$  is a semi-Markov process, where

$$\begin{aligned}
 P_{ij} &= \text{Probability that when leave state } i, \text{ we next go to state } j \\
 &= \Pr \left\{ \begin{array}{l} \text{At time of next departure there will be } j \text{ jobs in system,} \\ \text{given that at time of current departure there are } i \text{ jobs in system} \end{array} \right\} \\
 &= \Pr \{j - i + 1 \text{ jobs arrive during job's service time } \sim G\} \\
 &= \int_x \Pr \{j - i + 1 \text{ arrivals during time } x\} \Pr \{\text{service time is } x\} dx \\
 &= \int_x \frac{e^{-\lambda x} (\lambda x)^{j-i+1}}{(j-i+1)!} g(x) dx
 \end{aligned}$$

Recall that the solution to this semi-Markov process completely defines the number of jobs in the system for the M/G/1/FCFS, because, by PASTA, the limiting number of jobs seen by departures is the same as the true limiting number of jobs.

**Question:** So, having recalled this argument for M/G/1/FCFS, what would the argument be like to determine the limiting number of jobs in the system for M/G/1/LCFS?

**Answer:** The argument doesn't change at all when the service order is LCFS. In fact it's the same analysis for *any* service order that doesn't make use of job size.

**Question:** Why do we require that the scheduling policy not make use of size?

**Answer:** If you used size in determining which job got to serve next, then that would affect the distribution on the number of jobs which arrive during one service time.

**Question:** Consider again the set of all non-preemptive scheduling policies that don't make use of size. Is  $\text{Var}(T_S)$  the same for all these policies?

**Answer:** No. Observe that LCFS can generate some extremely high response times because we have to wait for system to become empty to take care of that first arrival. It turns out that, in agreement with intuition, we have

$$\text{Var}(T_S)^{\text{FCFS}} < \text{Var}(T_S)^{\text{Random}} < \text{Var}(T_S)^{\text{LCFS}}$$

We already know how to derive  $\text{Var}(T_S)^{\text{FCFS}}$ . We will now show how to derive  $\text{Var}(T_S)^{\text{LCFS}}$ . We do this by computing the Laplace transform of waiting time,  $\tilde{T}_Q^{\text{LCFS}}(s)$ .

Before we begin, we need to recall a few formulas and notation:

$$\begin{aligned}
S &= \text{job size} \\
f(x) &= \text{p.d.f. of job size} \\
F(x) &= \mathbf{Pr}\{S < x\} = \text{c.d.f. of job size}
\end{aligned}$$

$$\begin{aligned}
B_x &= \text{length of busy period started by a job of size } x \\
\widetilde{B}_x(s) &= e^{-x(s+\lambda-\lambda\widetilde{B}(s))} \\
B &= \text{length of busy period started by a job of size } S \\
\widetilde{B}(s) &= \int_{x=0}^{\infty} \widetilde{B}_x(s) f(x) dx \\
&= \int_{x=0}^{\infty} e^{-x(s+\lambda-\lambda\widetilde{B}(s))} f(x) dx \\
&= \widetilde{S}(s + \lambda - \lambda\widetilde{B}(s))
\end{aligned}$$

$$\begin{aligned}
S_e &= \text{Excess} \\
f_e(x) &= \text{p.d.f. of Excess} = \frac{\bar{F}(x)}{\mathbf{E}\{S\}} \\
\widetilde{S}_e(s) &= L_{f_e}(s) = \frac{1 - \widetilde{S}(s)}{s\mathbf{E}\{S\}}
\end{aligned}$$

Consider an arrival into the M/G/1/LCFS queue. Conditioning on whether the arrival sees an empty system or a busy system, we can immediately write:

$$\widetilde{T}_Q^{LCFS}(s) = (1 - \rho) \cdot \widetilde{T}_Q^{LCFS}(s|idle) + \rho \cdot \widetilde{T}_Q^{LCFS}(s|busy)$$

**Question:** What is  $\widetilde{T}_Q^{LCFS}(s|idle)$ ?

**Answer:** If the arrival sees an empty system, then the waiting time is zero, so

$$\widetilde{T}_Q^{LCFS}(s|idle) = 1$$

**Question:** In words, how long does the arrival wait if it finds the server busy?

**Answer:** At first one might think that the waiting time is just the excess of a service time,  $S_e$ , since the arrival has to wait for the job in service to finish serving. This is however not quite right, since more jobs may arrive during  $S_e$ , and those jobs have precedence over our arrival.

Thus the waiting time for our arrival is the length of a busy period started by a job of size  $S_e$ .

$$\begin{aligned}
\tilde{T}_Q^{LCFS}(s|busy) &= \int_{x=0}^{\infty} \tilde{B}_x(s) f_e(x) dx \\
&= \int_{x=0}^{\infty} e^{-x(s+\lambda-\lambda\tilde{B}(s))} \cdot f_e(x) dx \\
&= \tilde{S}_e(s+\lambda-\lambda\tilde{B}(s)) \\
&= \frac{1}{(s+\lambda-\lambda\tilde{B}(s))\mathbf{E}\{S\}} (1-\tilde{S}(s+\lambda-\lambda\tilde{B}(s))) \\
&= \frac{1}{(s+\lambda-\lambda\tilde{B}(s))\mathbf{E}\{S\}} (1-\tilde{B}(s))
\end{aligned}$$

Finally, unconditioning, we have:

$$\begin{aligned}
\tilde{T}_Q^{LCFS}(s) &= (1-\rho) \cdot \tilde{T}_Q^{LCFS}(s|idle) + \rho \cdot \tilde{T}_Q^{LCFS}(s|busy) \\
&= (1-\rho) + \rho \cdot \frac{1}{(s+\lambda-\lambda\tilde{B}(s))\mathbf{E}\{S\}} (1-\tilde{B}(s)) \\
&= (1-\rho) + \frac{\lambda(1-\tilde{B}(s))}{(s+\lambda-\lambda\tilde{B}(s))}
\end{aligned}$$

From the above transform, we can derive the second moment of waiting time,  $\mathbf{E}\{T_Q^2\}$ . We find that:

$$\mathbf{E}\{T_Q^2\}^{LCFS} = \frac{\lambda\mathbf{E}\{S^3\}}{3(1-\rho)^2} + \frac{((\lambda\mathbf{E}\{S^2\}))^2}{2(1-\rho)^3}$$

In comparison, for FCFS scheduling we saw:

$$\mathbf{E}\{T_Q^2\}^{FCFS} = \frac{\lambda\mathbf{E}\{S^3\}}{3(1-\rho)} + \frac{((\lambda\mathbf{E}\{S^2\}))^2}{2(1-\rho)^2}$$

So

$$\mathbf{E}\{T_Q^2\}^{LCFS} = \mathbf{E}\{T_Q^2\}^{FCFS} \cdot \frac{1}{1-\rho}$$

Thus while the mean waiting times are the same for FCFS and LCFS, the second moment of waiting time differs by a factor that depends on  $\rho$  but not on the job size distribution. Under high loads, the second moment of waiting time under LCFS becomes very bad as compared with the second moment of waiting time under FCFS.

## 4 Preemptive-Last-Come-First-Served

It is interesting to compare LCFS with the preemptive version of this policy: Preemptive-LCFS (or PLCFS).

Under PLCFS, whenever a new arrival enters the system, it immediately preempts the job in service.

**Key Observation:** Once job  $x$  is interrupted, it won't get back the processor until all jobs arriving after that point are completed (refer to Figure 1).

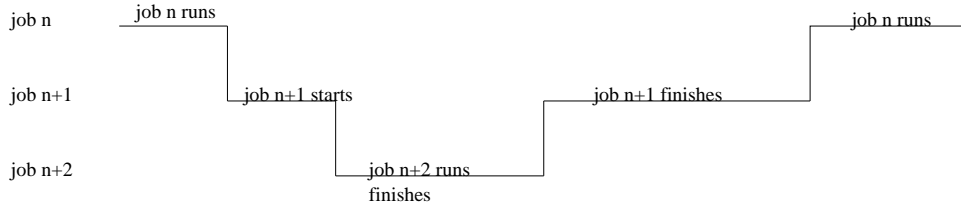


Figure 1: Jobs under Preemptive-LCFS

Let  $T(x)^{P-LCFS}$  = time in system for job of size  $x$  under P-LCFS.

Let  $T_S^{P-LCFS}$  = time in system under P-LCFS (for job of size  $S$ ).

Let  $A_x$  = number of jobs which arrive during time  $x$ .

Every time job  $x$  is interrupted, we get a contribution of  $T_S$ . Thus we have:

$$\begin{aligned}
 T(x)^{P-LCFS} &= x + \sum_{i=1}^{A_x} T_{S_i} \\
 \tilde{T}(x)(s)^{P-LCFS} &= e^{-sx} \cdot \hat{A}_x(\tilde{T}_S(s)) \\
 &= e^{-x(s+\lambda-\lambda\tilde{T}_S(s))} \\
 \tilde{T}_S(s)^{P-LCFS} &= \int_0^\infty \tilde{T}_x(s) f(x) dx \\
 &= \int_0^\infty e^{-x(s+\lambda-\lambda\tilde{T}_S(s))} f(x) dx \\
 &= \tilde{S}(s + \lambda - \lambda\tilde{T}_S(s))
 \end{aligned}$$

Observe that the transform obtained is exactly the same as length of a busy period. There's a reason for this. Suppose when we compute the time in system of a job under P-LCFS, we imagine the job as having entered an empty system (this is allowable, since jobs which entered before the job have no impact on the job). Now observe that our arrival cannot leave until the system is idle again. Thus the time until the arrival can leave is exactly the length of a busy period.

**Question:** How do LCFS and P-LCFS compare?

**Answer:** Looking at the transforms, we have:

$$\tilde{T}_S(s)^{P-LCFS} = \tilde{S}(s + \lambda - \lambda \tilde{T}_S(s))$$

whereas

$$\tilde{T}_S(s)^{LCFS} = \tilde{S}(s) \cdot \left( (1 - \rho) + \frac{\lambda(1 - \tilde{B}(s))}{(s + \lambda - \lambda \tilde{B}(s))} \right)$$

$T_S^{LCFS}$  is the excess of a busy period plus one service time, in the case that the arrival find the system busy. Otherwise it's just one service time.

By contrast,  $T_S^{P-LCFS}$  is the length of a busy period.

Thus it might seem initially that  $T_S^{LCFS} < T_S^{P-LCFS}$  always. This is not true. In fact, the mean response time under  $T_S^{LCFS}$  is lower exactly when  $C^2 < 1$  and greater when  $C^2 > 1$ .

## 5 Preemptive Priorities

For the remainder of the lecture, we will be discussing an M/G/1 queue with priority queueing. Here arriving jobs are divided into  $n$  priority classes, where class 1 is the highest priority and class  $n$  is the lowest. Class  $k$  job arrivals for a Poisson process with rate

$$\lambda_k = \lambda \cdot p_k.$$

The service time for jobs of class  $k$  will be denoted by the random variable  $S_k$ , and the load made up by jobs of class  $k$  is denoted by

$$\rho_k = \lambda_k \cdot E[S_k]$$

Imagine maintaining a separate queue for each class. When the server becomes free, always choose the job at the head of the highest-priority non-empty queue to work on. We will consider two types of priority queueing.

In this section, we look at **preemptive priority queueing**. Here, the job in service is preempted if a higher-priority job arrives, and the higher-priority job is served.

In the next section, we look at **non-preemptive priority queueing**. There, once a job starts running, it cannot be preempted, even if a higher-priority job comes along.

Preemptive priority queueing is very popular in computer operating systems. Here interactive jobs have preemptive priority over non-interactive jobs, and young jobs often have preemptive priority over old jobs. Preemptive priority queueing is also very popular in ATM networks where there are packet streams with different priorities trying to use the same communication link, and high-priority streams can cause lower-priority streams to suspend service, mid-stream.

Let  $T(k)$  denote the response time of a job of class  $k$ . Our goal will be to derive  $\widetilde{T}(k)(s)$ .

In analyzing *preemptive service disciplines*, it is helpful to view the time in system of a job as divided into two components:

1. The time until the job first starts serving (also called **waiting time**), and
2. The time from when the job first receives some service, until it leaves the system (also called **residence time**).

$$\widetilde{T}(k)(s)^{\text{Preemptive-Prio}} = \widetilde{Wt}(k)(s) \cdot \widetilde{Res}(k)(s)$$

## 5.1 Deriving residence time

We first derive the residence time component of  $\widetilde{T}(k)(s)$ .

**Question:** Once a job  $j$  of class  $k$  starts running, who can preempt it?

**Answer:** Only jobs of classes 1 through  $k - 1$  can preempt job  $j$  once it starts running.

So what we're looking for is the duration of a busy period, started by a job of class  $k$ , but where the only arrivals allowed are those of classes 1 through  $k - 1$ .

As always, to derive the length of any busy period, we first consider a busy period started by a job of size  $x$ .

Let  $B_{(k-1)+}$  denote the duration of a busy period made up of only jobs of class 1 through  $k$ . The  $+$  here means priority  $k - 1$  or "higher" priority.

Let  $B_{(k-1)^+}(x)$  denote the length of a busy period started by a job of size  $x$ , where we assume that the only arrivals allowed are those of class  $k - 1$  or higher priority.

Let  $B_{(k-1)^+}(S_k)$  denote the length of a busy period started by a job of size  $S_k$ , where we assume that the only arrivals allowed are those of class  $k - 1$  or higher priority. This is the *residence time* for a job of class  $k$ .

Let  $\lambda_{(k-1)^+}$  denote the total arrival rate of jobs of class  $k - 1$  or higher priority:

$$\lambda_{(k-1)^+} = \sum_{i=1}^{k-1} \lambda_i$$

Let  $A_x^{(k-1)^+}$  denote the number of arrivals during time  $x$  of jobs of class  $k - 1$  or higher priority.

Thus:

$$B_{(k-1)^+}(x) = x + \sum_{i=1}^{A_x^{(k-1)^+}} B_{(k-1)^+}^{(i)}$$

where each  $B_{(k-1)^+}^{(i)}$  is distributed identically as  $B_{(k-1)^+}$ .

Using the above equation,

$$\tilde{B}_{(k-1)^+}(x)(s) = e^{-sx} \cdot \hat{A}_x^{(k-1)^+}(\tilde{B}_{(k-1)^+}(s))$$

Now we know that

$$\hat{A}_x^{(k-1)^+}(z) = e^{-\lambda_{(k-1)^+}x(1-z)}$$

So,

$$\hat{A}_x^{(k-1)^+}(\tilde{B}_{(k-1)^+}(s)) = e^{-\lambda_{(k-1)^+}x(1-\tilde{B}_{(k-1)^+}(s))}$$

And so,

$$\tilde{B}_{(k-1)^+}(x)(s) = e^{-sx} \cdot e^{-\lambda_{(k-1)^+}x(1-\tilde{B}_{(k-1)^+}(s))} = e^{-x(s+\lambda_{(k-1)^+}-\lambda_{(k-1)^+}\tilde{B}_{(k-1)^+}(s))}$$

To get  $\tilde{B}_{(k-1)+}(S_k)(s)$  from  $\tilde{B}_{(k-1)+}(x)(s)$ , we just uncondition as below, where  $f_{S_k}(\cdot)$  denotes the probability density function of  $S_k$ .

$$\begin{aligned}\tilde{B}_{(k-1)+}(S_k)(s) &= \int_0^\infty \tilde{B}_{(k-1)+}(x)(s) f_{S_k}(x) dx \\ &= \int_0^\infty e^{-x(s+\lambda_{(k-1)+} - \lambda_{(k-1)+} \tilde{B}_{(k-1)+}(s))} f_{S_k}(x) dx \\ &= \tilde{S}_k \left( s + \lambda_{(k-1)+} - \lambda_{(k-1)+} \tilde{B}_{(k-1)+}(s) \right)\end{aligned}$$

Thus we've shown that:

$$\widetilde{Res}(k)(s) = \tilde{B}_{(k-1)+}(S_k)(s) = \tilde{S}_k \left( s + \lambda_{(k-1)+} - \lambda_{(k-1)+} \tilde{B}_{(k-1)+}(s) \right)$$

Observe that the first moment  $E[B_{(k-1)+}(S_k)]$  is given by:

$$E[Res(k)] = E[B_{(k-1)+}(S_k)] = \frac{E[S_k]}{1 - \sum_{i=1}^{k-1} \rho_i}$$

## 5.2 Deriving waiting time

We now derive the waiting time for a job of class  $k$ ,  $Wt(k)$ . This is the time until the job first receives service.

**Question:** What does waiting time for a job of class  $k$  depend on?

**Answer:** Clearly  $Wt(k)$  includes waiting for all those jobs in the system of class 1 through  $k$  when our tagged job arrives, since they have priority over our job. However, we also need to worry about all arrivals of priority 1 through  $k-1$  that arrive during  $Wt(k)$ , because they too get to complete before our tagged job gets to run.

We will need some notation:

Let  $W_{k+}$  denote the work in system of classes 1 to  $k$ .

Let  $\lambda_{k+}$  denote the arrival rate of classes 1 to  $k$ .

Let  $\rho_{k+}$  denote the load made up of classes 1 to  $k$ . That is:

$$\rho_{k+} = \sum_{i=1}^k \rho_i = \sum_{i=1}^k \lambda_i \mathbf{E}\{S_i\} = \lambda \sum_{i=1}^k p_i \mathbf{E}\{S_i\}$$

where  $p_i = \frac{\lambda_i}{\lambda}$ .

Let  $S_{k+}$  denote the size of an arbitrary job from those of classes 1 to  $k$ . Hence:

$$\tilde{S}_{k+}(s) = \sum_{i=1}^k \frac{p_i}{\sum_{j=1}^k p_j} \tilde{S}_i(s).$$

**Question:** So can you phrase  $Wt(k)$  as a busy period?

**Answer:**  $Wt(k)$  is the duration of a busy period, started by a phantom job of size  $W_{k+}$  where the only arrivals allowed are those of class  $k - 1$  or higher priority.

So, using our previous busy period notation:

$$\begin{aligned} Wt(k) &= B_{(k-1)+}(W_{k+}) \\ \widetilde{Wt}(k)(s) &= \widetilde{W}_{k+} \left( s + \lambda_{(k-1)+} - \lambda_{(k-1)+} \tilde{B}_{(k-1)+}(s) \right) \end{aligned}$$

where

$$\tilde{B}_{(k-1)+}(s) = \tilde{S}_{(k-1)+} \left( s + \lambda_{(k-1)+} - \lambda_{(k-1)+} \tilde{B}_{(k-1)+}(s) \right)$$

The only thing that remains is to determine  $W_{k+}(s)$ .

**Question:** What can we say about  $W_{k+}$ , the work that an arrival finds in the system, consisting of jobs of class  $k$  or higher priority?

**Hint:** Can you relate that work to queueing time?

**Another Hint:** Imagine an alternative universe, where there are only those jobs of class 1 through  $k$ . All other jobs are gone ....

**Answer:** Because jobs have *preemptive* priority, jobs of lower priority than  $k$  will never get worked on when there are jobs of priority  $k$  or better around. Hence:

$$\begin{aligned} W_{k+} &= \text{Work in alternative priority universe consisting of only jobs of class 1 through } k. \\ &= \text{Work in alternative FCFS universe consisting of only jobs of class 1 through } k. \\ &= \text{Time in queue in FCFS universe consisting of only jobs of class 1 through } k. \end{aligned}$$

Hence, we have:

$$W_{k^+} = \frac{(1 - \rho_{k^+})}{\lambda_{k^+} \widetilde{S}_{k^+}(s) - \lambda_{k^+} + s}$$

**Question:** With your current understanding, what can you say about  $\mathbf{E}\{Res(k)\}$  and  $\mathbf{E}\{Wait(k)\}$ ?

**Answer:**  $\mathbf{E}\{Res(k)\}$  is the mean length of a busy period, where all arrivals are jobs of priority  $k - 1$  or higher priority, but where the starting job has size  $S_k$ .

$\mathbf{E}\{Wait(k)\}$  is also the mean length of a busy period, whose jobs are only the first  $k - 1$  classes, but where the starting job has size equal to  $T_Q$  for an M/G/1/FCFS whose jobs are only the first  $k$  classes.

## 6 Non-preemptive Priorities

Non-preemptive priority queues are much more common in operations management. If a teller is handling airline reservations, she won't preempt a 2nd class customer, just because a 1st class customer comes along.

In non-preemptive priorities, once the job of class  $k$  gets to run, it can't be interrupted. So we only need to determine the waiting time of the job,  $Wt(k)$ , and then tack on the service time of the job,  $S_k$ .

$$\widetilde{T}(k)(s)^{\text{Non-Preemptive-Prio}} = \widetilde{Wt}(k)(s) \cdot \widetilde{S}_k(s)$$

Our goal is thus reduced to finding  $Wt(k)$ : Imagine a job of class  $k$  walks into the system. Let's tag this arrival. The tagged job is oblivious to any job in the system of priority worse than  $k$ , because it has priority over those jobs. The tagged job sees some work of classes 1 through  $k$  in the system. All those jobs have priority over it. The tagged job also may see a job in service, of any class, maybe even lower-prio than  $k$ . That job in service also has priority over the tagged job.

Let  $V_k$  denote the work seen by the job of class  $k$ , consisting all jobs of class  $k$  or higher priority, plus the job in service.

Note that unlike  $W_{k^+}$ ,  $V_k$  is not so easy to reason about. The problem is this extra job in service, which is more likely to be a high prio job, but could be any job. Thus we can't pretend we are in an alternative universe where there simply don't exist low priority jobs.

**Question:** Assuming that we are able to figure out  $V_k$ , how would we determine our tagged job's waiting time?

**Hint:** Who else does our tagged job need to wait for?

**Answer:** Observe that in addition to waiting behind  $V_k$ , the tagged job must wait behind all jobs arriving during  $Wt(k)$  whose priority is  $k - 1$  or better, since those jobs will run before the tagged job.

Thus, we are looking for a busy period, made up of only arrivals of class  $k - 1$  or better, which is started by a phantom job of size  $V_k$ . So,

$$Wt(k) = B_{(k-1)^+}(V_k)$$

So,

$$\widetilde{Wt}(k)(s) = \widetilde{V}_k \left( s + \lambda_{(k-1)^+} - \lambda_{(k-1)^+} \widetilde{B}_{(k-1)^+}(s) \right)$$

where

$$\widetilde{B}_{(k-1)^+}(s) = \widetilde{S}_{(k-1)^+} \left( s + \lambda_{(k-1)^+} - \lambda_{(k-1)^+} \widetilde{B}_{(k-1)^+}(s) \right)$$

So what remains is deriving  $V_k$ , which is of course the hard part.

We would like to express  $V_k$  as some kind of time in queue.

Imagine an alternative universe consisting of only 2 priority classes. In the alternative universe, class  $a$ , consists of all jobs of priority 1 through  $k$  in the regular world, while class  $b$  consists of all other jobs. Importantly, the jobs of class  $a$  are served in FCFS order, i.e., there is no differentiation between the jobs of class  $a$ . All jobs of class  $a$  have non-preemptive priority over jobs of class  $b$ .

**Question:** How can we express  $V_k$  in terms of our alternative universe?

**Answer:**

**Key idea:** Let  $T_Q(a)$  denote the time in queue for a job of class  $a$  in our alternative 2-priority universe. Then

$$V_k = T_Q(a).$$

**Question:** Why?

**Answer:** The work that the class  $k$  job finds is the same as the work that the class  $a$  job finds in the alternative universe. Because of the FCFS service among class  $a$  jobs in the alternative universe, a class  $a$  arrival has to wait behind all work in the system of class  $a$  plus any job in service, however it does not have to wait behind any future arrivals of class  $a$ .

We have thus reduced the problem of determining  $V_k$  to the problem of determining  $T_Q(a)$ , the time in queue for a class  $a$  job, in a 2-priority system, consisting of classes  $a$  and  $b$ . In Section 6.1, we will derive the z-transform of the number of class  $a$  jobs in a 2-priority system,  $N_a$ . In Section 6.2, we will show how to convert this into the Laplace transform of  $T_a$ , the response time of a class  $a$  job. From there, we will derive the queueing time for a class  $a$  job,  $T_Q(a)$ .

## 6.1 Two-prio system: Number of class $a$ jobs

Assume an M/G/1 with 2 non-preemptive priority classes, called  $a$  and  $b$ , where class  $a$  jobs have priority.

We will use the following notation:

$$\begin{aligned}
 \lambda_a &= \text{Arrival rate of class } a \text{ jobs} = \lambda p_a \\
 \lambda_b &= \text{Arrival rate of class } b \text{ jobs} = \lambda p_b \\
 S_a &= \text{Size of class } a \text{ job} \\
 S_b &= \text{Size of class } b \text{ job} \\
 \rho_a &= \lambda_a \cdot \mathbf{E}\{S_a\} \\
 a_{S_a}(k) &= \mathbf{Pr}\{k \text{ type } a \text{ jobs arrive during } S_a\} \\
 a_{S_b}(k) &= \mathbf{Pr}\{k \text{ type } a \text{ jobs arrive during } S_b\} \\
 \pi_k &= \mathbf{Pr}\{k \text{ jobs of type } a \text{ in the system}\} \\
 &= \mathbf{Pr}\{\text{departure (of any type) leaves behind } a \text{ jobs in system}\} \\
 \widehat{N}_a(z) &= \text{z-transform of number of type } a \text{ jobs in system} \\
 A_{S_a} &= \text{Number of type } a \text{ arrivals during } S_a
 \end{aligned}$$

We write balance equations:

$$\pi_k = \sum_{j=0}^{\infty} \pi_j p_{jk}$$

where the transition probability  $p_{jk}$  represents the probability that the next departure leaves behind  $k$  jobs, given that the current departure left behind  $j$  jobs.

Observe that:

$$p_{jk} = a_{S_a}(k - j + 1) \text{ if } j > 0$$

$$p_{0k} = a_{S_b}(k) \text{ with probability } \pi_0 - (1 - \rho)$$

$$p_{0k} = p_a a_{S_a}(k) + p_b a_{S_b}(k) \text{ with probability } (1 - \rho)$$

Here  $\rho$  represents the probability that the system is completely idle (no jobs of either type), while  $\pi_0 - (1 - \rho)$  represents the probability that there are zero jobs of type  $a$ , but a non-zero number of jobs of  $b$ . Recall that  $p_a = \frac{\lambda_a}{\lambda}$ , and likewise for  $p_b$ .

$$\begin{aligned} \pi_k &= (\pi_0 - (1 - \rho))a_{S_b}(k) + (1 - \rho)(p_a a_{S_a}(k) + p_b a_{S_b}(k)) + \sum_{j=1}^{k+1} \pi_j p_{jk} \\ \sum_{k=0}^{\infty} \pi_k z^k &= \sum_{k=0}^{\infty} (\pi_0 - (1 - \rho))a_{S_b}(k)z^k + \sum_{k=0}^{\infty} (1 - \rho)(p_a a_{S_a}(k) + p_b a_{S_b}(k))z^k \\ &\quad + \sum_{k=0}^{\infty} \sum_{j=1}^{k+1} \pi_j p_{jk} z^k \end{aligned}$$

$$\begin{aligned} \widehat{N}_a(z) &= (\pi_0 - (1 - \rho))\widehat{A}_{S_b}(z) + (1 - \rho)p_a \widehat{A}_{S_a}(z) + (1 - \rho)p_b \widehat{A}_{S_b}(z) \\ &\quad + \sum_{j=1}^{\infty} \sum_{k=j-1}^{\infty} \pi_j p_{jk} z^k \\ &= (\pi_0 - (1 - \rho))\widehat{A}_{S_b}(z) + (1 - \rho)p_a \widehat{A}_{S_a}(z) + (1 - \rho)p_b \widehat{A}_{S_b}(z) \\ &\quad + \sum_{j=1}^{\infty} \pi_j z^{j-1} \sum_{k=j-1}^{\infty} a_{S_a}(k - j + 1) z^{k-j+1} \\ &= (\pi_0 - (1 - \rho))\widehat{A}_{S_b}(z) + (1 - \rho)p_a \widehat{A}_{S_a}(z) + (1 - \rho)p_b \widehat{A}_{S_b}(z) \\ &\quad + \frac{1}{z} (\widehat{N}_a(z) - \pi_0) \sum_{u=0}^{\infty} a_{S_a}(u) z^u \\ &= (\pi_0 - (1 - \rho))\widehat{A}_{S_b}(z) + (1 - \rho)p_a \widehat{A}_{S_a}(z) + (1 - \rho)p_b \widehat{A}_{S_b}(z) \\ &\quad + \frac{1}{z} (\widehat{N}_a(z) - \pi_0) \widehat{A}_{S_a}(z) \end{aligned}$$

$$\begin{aligned}
\widehat{N}_a(z) \left(1 - \frac{\widehat{A}_{S_a}(z)}{z}\right) &= (\pi_0 - (1 - \rho))\widehat{A}_{S_b}(z) + (1 - \rho)p_a\widehat{A}_{S_a}(z) + (1 - \rho)p_b\widehat{A}_{S_b}(z) - \pi_0\frac{\widehat{A}_{S_a}(z)}{z} \\
&= (1 - \rho) \left(-\widehat{A}_{S_b}(z) + p_a\widehat{A}_{S_a}(z) + p_b\widehat{A}_{S_b}(z)\right) \\
&\quad + \pi_0 \left(\widehat{A}_{S_b}(z) - \frac{\widehat{A}_{S_a}(z)}{z}\right) \\
&= (1 - \rho)p_a \left(\widehat{A}_{S_a}(z) - \widehat{A}_{S_b}(z)\right) \\
&\quad + \frac{\pi_0}{z} \left(z\widehat{A}_{S_b}(z) - \widehat{A}_{S_a}(z)\right) \\
\widehat{N}_a(z) &= \frac{z(1 - \rho)p_a \left(\widehat{A}_{S_a}(z) - \widehat{A}_{S_b}(z)\right) + \pi_0 \left(z\widehat{A}_{S_b}(z) - \widehat{A}_{S_a}(z)\right)}{z - \widehat{A}_{S_a}(z)}
\end{aligned}$$

where  $\widehat{A}_{S_a}(z) = \widetilde{S}_a(\lambda_a - \lambda_a z)$  and  $\widehat{A}_{S_b}(z) = \widetilde{S}_b(\lambda_a - \lambda_a z)$ .

All that remains is to determine  $\pi_0$ . We do this by evaluating both sides of the above expression at  $z = 1$ , where we need to apply L'Hopital's Rule to the right hand side. This results in:

$$\pi_0 = 1 - p_a \rho$$

which makes intuitive sense.

## 6.2 Two-prio system: Queueing time for class $a$ jobs

Recall that our goal was to determine the time in queue for a class  $a$  job in the 2-prio non-preemptive queue. It is not immediately clear how this relates to the number of type  $a$  jobs in the system. We need another mental leap.

**Question:** How does the response time  $T_a$  for a job of class  $a$  relate to the number of type  $a$  jobs in the system?

**Answer:** Let  $N_a^{dep.a}$  denote the number of type  $a$  jobs seen by a departure of type  $a$ .

$$\begin{aligned}
A_{T_a} &= \text{Number of type } a \text{ jobs arriving during } T_a \\
&= \text{Number of type } a \text{ jobs seen by a departure of type } a \\
&= N_a^{dep.a}
\end{aligned}$$

Thus we can obtain an expression involving  $T_a$ , and hence one involving  $T_Q(a)$ , if we know  $N_a^{dep.a}$ .

We have already derived the transform for  $N_a$ , which is the number of type  $a$  jobs seen by an arbitrary departure. We just have to figure out how to translate  $\widehat{N}_a(z)$ , which we have, to  $\widehat{N}_a^{dep.a}(z)$ , which we want.

Observe that an arbitrary departure is of type  $a$  with probability  $p_a$ , and of type  $b$  with probability  $p_b = 1 - p_a$ . Thus

$$\widehat{N}_a(z) = p_a \widehat{N}_a^{dep.a}(z) + p_b \widehat{N}_a^{dep.b}(z)$$

At first this doesn't look too good, because we have introduced yet another variable into the mix:  $N_a^{dep.b}$ . However, we observe that we already know the value of this new quantity ...

**Question:** What do we know about  $N_a^{dep.b}$ ?

**Hint:** When do class  $b$  jobs get to run?

**Answer:** Class  $b$  jobs get to run only when there are zero class  $a$  jobs in the system. Suppose a class  $b$  job just started to run. Then the number of class  $a$  jobs seen by that departing class  $b$  job is exactly the number of class  $a$  arrivals during  $S_b$ . Hence:

$$\widehat{N}_a^{dep.b}(z) = \widehat{A}_{S_b}(z)$$

Thus:

$$\begin{aligned} p_a \widehat{N}_a^{dep.a}(z) &= \widehat{N}_a(z) - p_b \widehat{N}_a^{dep.b}(z) \\ &= \widehat{N}_a(z) - p_b \widehat{A}_{S_b}(z) \\ \widehat{N}_a^{dep.a}(z) &= \frac{\widehat{N}_a(z) - p_b \widehat{A}_{S_b}(z)}{p_a} \end{aligned}$$

Substituting in our expression for  $\widehat{N}_a(z)$  from the previous section, we have:

$$\begin{aligned}
\widehat{N}_a^{dep.a}(z) &= \frac{z(1-\rho)p_a \left( \widehat{A}_{S_a}(z) - \widehat{A}_{S_b}(z) \right) + \pi_0 \left( z\widehat{A}_{S_b}(z) - \widehat{A}_{S_a}(z) \right) - (1-p_a)\widehat{A}_{S_b}(z) \left( z - \widehat{A}_{S_a}(z) \right)}{p_a \left( z - \widehat{A}_{S_a}(z) \right)} \\
&= \frac{z(1-\rho)p_a \left( \widehat{A}_{S_a}(z) - \widehat{A}_{S_b}(z) \right) + (1-p_a\rho) \left( z\widehat{A}_{S_b}(z) - \widehat{A}_{S_a}(z) \right) - (1-p_a)\widehat{A}_{S_b}(z) \left( z - \widehat{A}_{S_a}(z) \right)}{p_a \left( z - \widehat{A}_{S_a}(z) \right)} \\
&= \frac{\widehat{A}_{S_a}(z) \left( z(1-\rho)p_a - (1-p_a\rho) + (1-p_a)\widehat{A}_{S_b}(z) \right) + \widehat{A}_{S_b}(z) \left( -z(1-\rho)p_a + (1-p_a\rho)z - (1-p_a)z \right)}{p_a \left( z - \widehat{A}_{S_a}(z) \right)} \\
&= \frac{\widehat{A}_{S_a}(z) \left( z(1-\rho)p_a - (1-p_a\rho) + (1-p_a)\widehat{A}_{S_b}(z) \right)}{p_a \left( z - \widehat{A}_{S_a}(z) \right)} \\
&= \frac{\widetilde{S}_a(\lambda_a - \lambda_a z) \left( z(1-\rho)p_a - (1-p_a\rho) + (1-p_a)\widetilde{S}_b(\lambda_a - \lambda_a z) \right)}{p_a \left( z - \widetilde{S}_a(\lambda_a - \lambda_a z) \right)} \\
&= A_{T_a}(z) = \widetilde{T}_a(\lambda_a - \lambda_a z)
\end{aligned}$$

We now make the substitution  $s = \lambda_a - \lambda_a z$ , which yields:

$$\begin{aligned}
\widetilde{T}_a(s) &= \frac{\widetilde{S}_a(s) \left( \frac{s-\lambda_a}{-\lambda_a} (1-\rho)p_a - (1-p_a\rho) + (1-p_a)\widetilde{S}_b(s) \right)}{p_a \left( \frac{s-\lambda_a}{-\lambda_a} - \widetilde{S}_a(s) \right)} \\
&= \frac{\widetilde{S}_a(s) \left( (s-\lambda_a)(1-\rho)p_a + \lambda_a(1-p_a\rho) - \lambda_a(1-p_a)\widetilde{S}_b(s) \right)}{p_a \left( s - \lambda_a + \lambda_a\widetilde{S}_a(s) \right)} \\
&= \frac{\widetilde{S}_a(s) \left( s(1-\rho) + \lambda_b \left( 1 - \widetilde{S}_b(s) \right) \right)}{\left( s - \lambda_a + \lambda_a\widetilde{S}_a(s) \right)}
\end{aligned}$$

Finally

$$\widetilde{T}_Q(a)(s) = \widetilde{T}_a(s) / \widetilde{S}_a(s)$$

So

$$\widetilde{T}_Q(a)(s) = \frac{s(1-\rho) + \lambda_b \left( 1 - \widetilde{S}_b(s) \right)}{\left( s - \lambda_a + \lambda_a\widetilde{S}_a(s) \right)}$$

Hence we have our expression for  $\widetilde{T}_Q(a)(s)$ , and we are done, since this equals the quantity  $V_k$  that we were searching for.

## 7 Homework

1. **Preemptive-Shortest-Job-First** The Preemptive-Shortest-Job-First (PSJF) scheduling policy always (preemptively) runs the job with the smallest size. Note that “size” here refers to the original service requirement that the job came in with, not its remaining size. Please derive the Laplace transform of the response time of the M/G/1/PSJF queue.
2. **(Optional)** Now that we’ve studied the transform for response time in the non-preemptive priority queue, try to recall the derivation of mean response time for the non-preemptive priority queue that we did last semester. Write this in a form where the busy period structure is clearly visible.

Thank you!