Donald Schartman
December 18, 2009
RT Final Report

Objective:

For this report I shall be discussing different scheduling algorithms and how they relate to each other. My goal is to compare and contrast these algorithms and to effectively do this I have chosen a movie downloading service as my example. I chose this example because I felt that it was easy to explain the different implementations of each algorithm and how they would apply. While these algorithms may not be the best way to implement an actual movie downloading service, I needed an avenue through which I could analyze the algorithms mathematically.

Implementation:

To compare and contrast these algorithms I have created five sets of 20 random movie times from 120 minutes to 30 minutes in length. Each set of the 20 numbers are then treated as if they are in a ready queue. There are also a few constants that I would like to mention. For the purposes of this paper, the download speed will be fixed at 1 MB/second (60 MB/min). The next constant is a conversion factor and it is the ratio of movie size over movie time. This constant is created assuming that there are 700 MB in a 120 minute movie thus providing me with 5.833 MB/min. To further simplify things I will be assuming that the compression ratios for the all movies are the same and that there is only 1 download slot. The decision of having only one download slot makes it feasible to analyze these algorithms mathematically for the purposes of this paper.

The first set of algorithms that I would like to talk about are First-Come First-Served, Shortest Job First, and Priority based. The analysis of these algorithms will be related and that is why I will be discussing them together. I then would like to discuss the use of a Round Robin algorithm. The Round Robin will have a slightly different analysis and that is why it will be discussed separately. Both sets of algorithms will have equations and statistics to further explain their behavior.

Results:

Before I begin discussing the results I would like to take the time to explain Table 1. There are 12 columns and the first two (H and L) represent the high and low values that were used in the random number generator. The next five columns, labeled MT R1 - MT R5, are the samples of random movie times that were generated. The five columns after those, labled WT R1 - WT R5, are the waiting times calculted based upon a FIFO schedule. For each column (except the H and L columns) I calculate the average, standard deviation, and the variance.

Equation 1

$$WT_n(\mathrm{min}) = \frac{\sum_1^{n-1} MT_n\,(min) * 5.833\,\frac{MB}{min}}{60\,\frac{MB}{min}}\ , for\ n > 1$$

Equation 1 explains the calclulation used to find the waiting time for each location in the download queue. Since I generate 20 random numbers, this equation calculates the waiting time based on the file size and download speed. Since all 20 random number are already in the ready queue, the last movie to has to wait for the sum of time it takes all the others to finish.

This will not always be the case, but it does simulate the situation of there being a high amount of traffic on the one download slot.

Table 1: Data Analysis 1

H	L	MT R1	MT R2	MT R3	MT R4	MT R5	WT R1	WT R2	WT R3	WT R4	WT R5
120	30	66	119	78	31	109	0.000	0.000	0.000	0.000	0.000
120	30	36	76	62	33	104	6.417	11.569	7.583	3.014	10.597
120	30	73	51	72	35	100	9.917	18.958	13.611	6.222	20.708
120	30	104	104	72	38	93	17.014	23.917	20.611	9.625	30.431
120	30	78	84	35	39	83	27.125	34.028	27.611	13.319	39.472
120	30	52	51	51	39	81	34.708	42.194	31.014	17.111	47.542
120	30	109	111	83	52	74	39.764	47.153	35.972	20.903	55.417
120	30	99	39	53	60	72	50.361	57.944	44.042	25.958	62.611
120	30	114	100	81	60	68	59.986	61.736	49.194	31.792	69.611
120	30	66	83	109	62	66	71.069	71.458	57.069	37.625	76.222
120	30	102	78	48	62	64	77.486	79.528	67.667	43.653	82.639
120	30	77	33	80	65	55	87.403	87.111	72.333	49.681	88.861
120	30	72	95	48	69	53	94.889	90.319	80.111	56.000	94.208
120	30	97	76	66	80	51	101.889	99.556	84.778	62.708	99.361
120	30	85	118	48	82	50	111.319	106.944	91.194	70.486	104.319
120	30	40	44	115	86	46	119.583	118.417	95.861	78.458	109.181
120	30	66	119	54	90	39	123.472	122.694	107.042	86.819	113.653
120	30	71	64	87	102	39	129.889	134.264	112.292	95.569	117.444
120	30	68	76	101	116	36	136.792	140.486	120.750	105.486	121.236
120	30	33	48	61	119	32	143.403	147.875	130.569	116.764	124.736
Average		75.400	78.450	70.200	66.000	65.750	72.124	74.808	62.465	46.560	73.412
StdDev		23.816	27.933	21.766	27.009	23.490	47.134	45.134	39.658	36.121	38.654
Var		567.200	780.261	473.747	729.474	551.776	2221.649	2037.057	1572.752	1304.732	1494.101

Looking at Table 1, MT R1 – MT R3 are strictly random movie times. MT R4 is random and sorted from smallest to largest, and MT R5 is random and sorted from largest to smallest. To analyze the First-Come First-Served approach let's look at MT R1 – MT R3. These are purely random cases and the results of applying equation 1 can be seen in the average wait times for WT R1 – WT R3. As you can see from the averages, WT R3 has the best average and WT R2 has the worst. The average wait time for the FIFO scheduling algorithm is heavily dependent on the order in which the jobs are processed. This brings us into the next algorithm, the shortest job first. Column WT R4 represents the best case scenario for the FIFO algorithm and is also a sample of the shortest job first algorithm. As you can see it turns out that WT R4 gives the lowest average wait time. But upon closer inspection that same column in MT R4 also has the lowest average movie time. While the SJF algorithm may provide a slight advantage in shorter average wait times over the FIFO, the real difference is in the perceived waiting per user. In Figure 1 below, I have created a graph of waiting time divided by movie time. What this graph represents is the perceived waiting time for a user downloading a movie. If we assume that a 1:1 ratio of waiting time to download time is fair, then the objective is to keep the ratio less than or equal to 1 at all times. Here we have the results for R2, R4, and R5. R2, in Table 1, had the worst results of all the three random number samples. R4 was my SJF sorted random sample and R5 was my sorted random sample from longest to shortest. The larger the ratio, the longer a user has to wait based on how long their movie is. As you can see, R4 maintains a ratio below 1 over its entire sample. This is the real benefit of the shortest job first algorithm. You can see R2 follow R4's trend but it has random spikes that show up in the average wait time. R5 is my worst case and as you can see that it is well above 1 for the majority of its sample. The shortest job first has a consistency that is superior to the other algorithms.

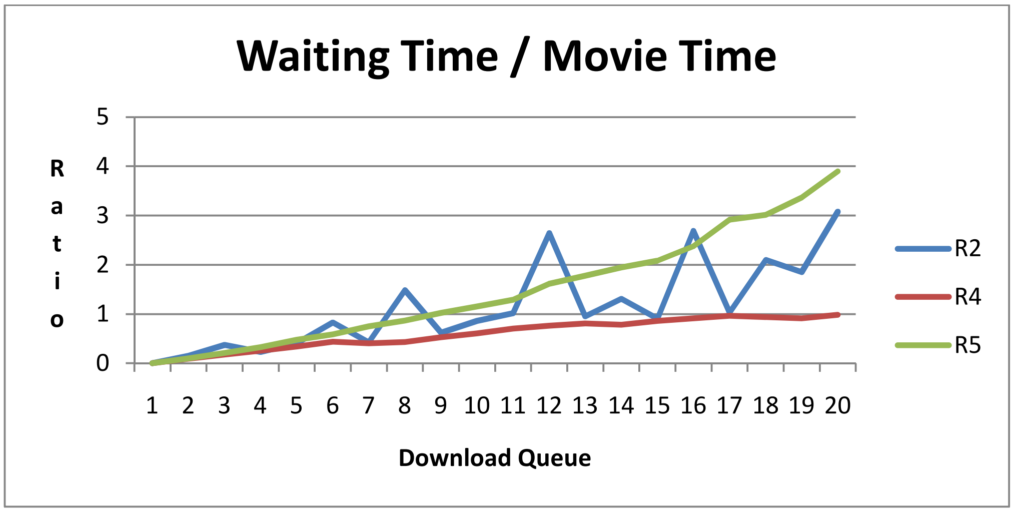

Figure 1: Perceived Waiting Time

This brings me to the next algorithm I want to discuss and that is the priority based. This algorithm takes into account the users priority while selecting a job. The only real modification that takes place is the fact that when the next download is chosen, the search involves priority. Instead of one ready queue, there will be two. One for the higher priority downloads and one for the others. These queues will be sorted independently of each other. Because this closely involves the same situations as the SJF data, I suspect that the priority based approach will have worse average wait times then the shortest job first algorithm. The other downside of using the priority based approach is the possibility of starvation. If higher priority downloads keep filling the queue, then the lower priority downloads will never complete. Downloads in the lower priority queue will only ever get a spot when the higher priority queue is empty.

The last algorithm I want to discuss is the Round Robin. For this paper the Round Robin will consist of the 20 slots in the ready queue and I will explore the effect that different time slices have on the overall impact of the algorithm and the average time for download completion. To figure out what is going on with the Round Robin algorithm, we will look at Table 2 below. In this table I have introduced a few new columns. Starting at the second column and moving right we have: Movie Times, Download Time, Number of RR cycles to complete, cycle1, …, cycle n, wait time1, …, wait time n, and finally time to completion. Let’s look at the column DLT R1. This is the time it takes to download the movie for that row. I have calculated some statistics for that row. The two most important that I want to discuss are the average and the standard deviation.

Table 2: Data Analysis 2

-	MT R1	DLT R1	Cycles @ 7.3	c1	c2	Wc1	Wc2	TTC	Eff
1	66.0	6.4	1.0	7.5	0.0	0.0	-	0	0.853333333
2	36.0	3.5	1.0	7.5	0.0	7.5	-	7.5	0.466666667
3	73.0	7.1	1.0	7.5	0.0	15.0	-	15	0.946666667
4	104.0	10.1	2.0	7.5	7.5	22.5	150	150	0.673333333
5	78.0	7.6	2.0	7.5	7.5	30.0	157.5	157.5	0.506666667
6	52.0	5.1	1.0	7.5	0.0	37.5	-	37.5	0.68
7	109.0	10.6	2.0	7.5	7.5	45.0	165	165	0.706666667
8	99.0	9.6	2.0	7.5	7.5	52.5	172.5	172.5	0.64
9	114.0	11.1	2.0	7.5	7.5	60.0	180	180	0.74
10	66.0	6.4	1.0	7.5	0.0	67.5	-	67.5	0.853333333
11	102.0	9.9	2.0	7.5	7.5	75.0	187.5	187.5	0.66
12	77.0	7.5	2.0	7.5	7.5	82.5	195.0	195.0	0.5
13	72.0	7.0	1.0	7.5	0.0	90.0	-	90	0.933333333
14	97.0	9.4	2.0	7.5	7.5	97.5	202.5	202.5	0.6
15	85.0	8.3	2.0	7.5	7.5	105.0	210.0	210.0	0.6
16	40.0	3.9	1.0	7.5	0.0	112.5	-	112.5	0.52
17	66.0	6.4	1.0	7.5	0.0	120.0	-	120	0.853333333
18	71.0	6.9	1.0	7.5	0.0	127.5	-	127.5	0.92
19	68.0	6.6	1.0	7.5	0.0	135.0	-	135	0.88
20	33.0	3.2	1.0	7.5	0.0	142.5	-	142.5	0.426666667
Average	**75.4**	**7.3**	**1.5**	**7.5**	**3.4**	**71.3**	**180.0**	**123.8**	**0.69700**
StdDev	**23.8**	**2.3**	**0.5**	**0.0**	**3.8**	**44.4**	**20.5**	**66.8**	**0.16862**
Var	**567.2**	**55.1**	**0.3**	**0.0**	**14.7**	**1968.8**	**421.9**	**4467.4**	**0.02843**
Min	**33.0**	**3.2**	**1.0**	**7.5**	**0.0**	**0.0**	**150.0**	**0.0**	**0.42667**
Max	**114.0**	**11.1**	**2.0**	**7.5**	**7.5**	**142.5**	**210.0**	**210.0**	**0.94667**

As shown in Table 2 (above), the average for DLT R1 is 7.3 and the standard deviation is 2.3. In my analysis in Table 2, I have chosen a time slice that is equal to the average movie download time (as well as 1 standard deviation above and below the average as will be explained later). For cycle 1, c1, every slot gets a time slice. Of the twenty slots, eleven downloads are completed in the first cycle. The last nine get completed in the second download cycle. To analyze the time it takes each download to finish, we will look at the column TTC. This represents how long each slot takes to start a time slice that will complete its download. This is calculated using a modified version of equation 1. The columns labeled Wc are the wait times as calculated before. But since there are multiple cycles the equation has been adapted to take that into consideration. Taking the greatest value for a row in all the columns for Wc, we get the column TTC. The average value for TTC represents the average time it takes for a download to start a cycle were it will finish using the round robin schedule as I have explained. There are a few apparent problems in using the round robin. The first is that the average wait time TTC, is much higher than the other algorithms. This made more sense once I started looking at the efficiency of all the download slots. As I have implemented the round robin, sometimes there is more of a time slice allocated then is actually needed. So to calculate efficiency I took the movie download time divided by the total time in time slices that it receives.

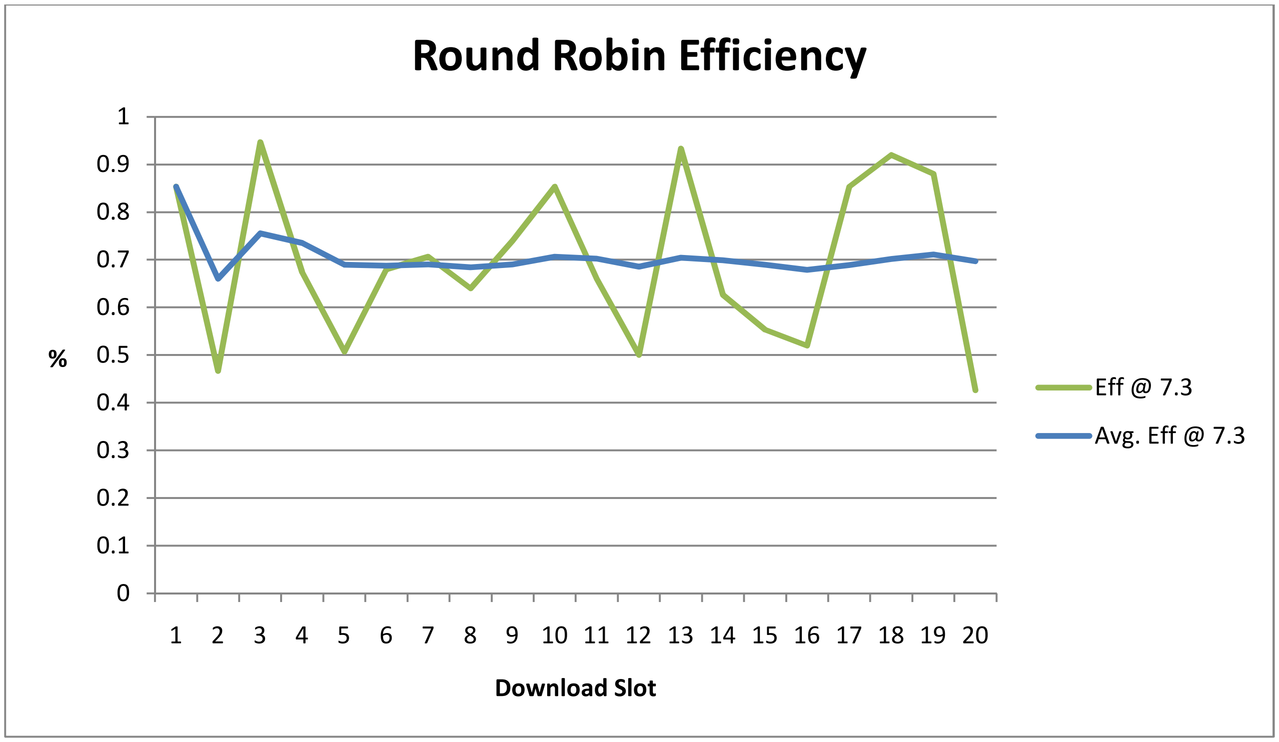

Figure 2

For the purposes of this paper we can assume that a FIFO style algorithm is 100% efficient. This is the big advantage over the round robin. Using a time slice equal to the average movie download time produced an average efficiency of 69.7%. It seems that in order to increase efficiency the time slice has to be reduced in size. The side effect of this is that the average wait time increases. Increasing the time slice gave better average wait times but reduced the efficiency. The decrease in efficiency will not allow the average wait times to approach those of a FIFO or SJF schedule. The results of modifying the time slice can be seen in Tables 3 and 4 in Appendix A. For these tables, I analyzed time slices 1 standard deviation above and below the average.

Conclusion:

In this paper, I've offered a statistical analysis of various scheduling algorithms as they apply to the example of a movie downloading service. From the data that I created, the strengths and weaknesses of each algorithm are much more apparent. The results show that the most efficient algorithm is the shortest job first because of the consistency that it offers. As expected the worst algorithm was the round robin. It's very clear that the efficiency of the algorithm is a big reason for its downfall. My movie downloading example had to be simplified with assumptions such as download speed, compression ratios, and the number of download slots so that analyzing the behavior of the algorithm could be done mathematically. As a result, I now have a much greater understanding of the application of these algorithms and think that this statistical analysis was extremely beneficial to my understanding of scheduling in general.

Appendix A:

Table 3: Data Analysis 3

-	MT R1	DLT R1	Cycles @ 9.6	c1	c2	Wc1	Wc2	TTC	Eff
1	66.0	6.4	1.0	9.6	0.0	0.0	-	0	0.666667
2	36.0	3.5	1.0	9.6	0.0	9.6	-	9.6	0.364583
3	73.0	7.1	1.0	9.6	0.0	19.2	-	19.2	0.739583
4	104.0	10.1	2.0	9.6	9.6	28.8	192	192	0.526042
5	78.0	7.6	1.0	9.6	0.0	38.4	-	38.4	0.791667
6	52.0	5.1	1.0	9.6	0.0	48.0	-	48	0.53125
7	109.0	10.6	2.0	9.6	9.6	57.6	201.6	57.6	0.552083
8	99.0	9.6	1.0	9.6	0.0	67.2	-	67.2	1
9	114.0	11.1	2.0	9.6	9.6	76.8	211.2	76.8	0.578125
10	66.0	6.4	1.0	9.6	0.0	86.4	-	86.4	0.666667
11	102.0	9.9	2.0	9.6	9.6	96.0	220.8	96	0.515625
12	77.0	7.5	1.0	9.6	0.0	105.6	-	105.6	0.8
13	72.0	7.0	1.0	9.6	0.0	115.2	-	115.2	0.729167
14	97.0	9.4	1.0	9.6	0.0	124.8	-	124.8	1.0
15	85.0	8.3	1.0	9.6	0.0	134.4	-	134.4	0.9
16	40.0	3.9	1.0	9.6	0.0	144.0	-	144	0.40625
17	66.0	6.4	1.0	9.6	0.0	153.6	-	153.6	0.666667
18	71.0	6.9	1.0	9.6	0.0	163.2	-	163.2	0.71875
19	68.0	6.6	1.0	9.6	0.0	172.8	-	172.8	0.6875
20	33.0	3.2	1.0	9.6	0.0	182.4	-	182.4	0.333333
Average	**75.4**	**7.3**	**1.2**	**9.6**	**1.9**	**91.2**	**206.4**	**99.4**	**0.65495**
StdDev	**23.8**	**2.3**	**0.4**	**0.0**	**3.9**	**56.8**	**12.4**	**59.0**	**0.18330**
Var	**567.2**	**55.1**	**0.2**	**0.0**	**15.5**	**3225.6**	**153.6**	**3485.3**	**0.03360**
Min	**33.0**	**3.2**	**1.0**	**9.6**	**0.0**	**0.0**	**192.0**	**0.0**	**0.33333**
Max	**114.0**	**11.1**	**2.0**	**9.6**	**9.6**	**182.4**	**220.8**	**192.0**	**1.00000**

Table 4: Data Analysis 4

-	MT R1	DLT R1	Cycles @ 5	c1	c2	c3	Wc1	Wc2	Wc3	TTC	Eff
1	66.0	6.4	2.0	5.0	5.0	0.0	0.0	100.0	-	100	0.64
2	36.0	3.5	1.0	5.0	0.0	0.0	5.0	-	-	5	0.7
3	73.0	7.1	2.0	5.0	5.0	0.0	10.0	105.0	-	105	0.71
4	104.0	10.1	3.0	5.0	5.0	5.0	15.0	110.0	185.0	185	0.673333
5	78.0	7.6	2.0	5.0	5.0	0.0	20.0	115.0	-	115	0.76
6	52.0	5.1	2.0	5.0	5.0	0.0	25.0	120.0	-	120	0.51
7	109.0	10.6	3.0	5.0	5.0	5.0	30.0	125.0	190.0	190	0.706667
8	99.0	9.6	2.0	5.0	5.0	0.0	35.0	130.0	-	130	0.96
9	114.0	11.1	3.0	5.0	5.0	5.0	40.0	135.0	195.0	195	0.74
10	66.0	6.4	2.0	5.0	5.0	0.0	45.0	140.0	-	140	0.64
11	102.0	9.9	2.0	5.0	5.0	0.0	50.0	145.0	-	145	0.99
12	77.0	7.5	2.0	5.0	5.0	0.0	55.0	150.0	-	150.0	0.8
13	72.0	7.0	2.0	5.0	5.0	0.0	60.0	155.0	-	155	0.7
14	97.0	9.4	2.0	5.0	5.0	0.0	65.0	160.0	-	160.0	0.9
15	85.0	8.3	2.0	5.0	5.0	0.0	70.0	165.0	-	165.0	0.8
16	40.0	3.9	1.0	5.0	0.0	0.0	75.0	-	-	75	0.78
17	66.0	6.4	2.0	5.0	5.0	0.0	80.0	170.0	-	170	0.64
18	71.0	6.9	2.0	5.0	5.0	0.0	85.0	175.0	-	175	0.69
19	68.0	6.6	2.0	5.0	5.0	0.0	90.0	180.0	-	180	0.66
20	33.0	3.2	1.0	5.0	0.0	0.0	95.0	-	-	142.5	0.64
Average	**75.4**	**7.3**	**2.0**	**5.0**	**4.3**	**0.0**	**47.5**	**140.0**	**190.0**	**140.1**	**0.733**
StdDev	**23.8**	**2.3**	**0.6**	**0.0**	**1.8**	**0.0**	**29.6**	**25.2**	**5.0**	**45.3**	**0.120**
Var	**567.2**	**55.1**	**0.3**	**0.0**	**3.4**	**0.0**	**875.0**	**637.5**	**25.0**	**2055.6**	**0.014**
Min	**33.0**	**3.2**	**1.0**	**5.0**	**0.0**	**0.0**	**0.0**	**100.0**	**185.0**	**5.0**	**0.510**
Max	**114.0**	**11.1**	**3.0**	**5.0**	**5.0**	**5.0**	**95.0**	**180.0**	**195.0**	**195.0**	**0.990**