CS457 Assignment 1

Daniel Burstyn (20206120)

March 2, 2009

Contents

1 Pseudocode

Below is a breif pseudocode description of my simulation code, modeled after notes 5.

1.1 State variables

- k - Item with ID k is to be transmitted next
- g - Keep track of when to handle i-requests (when $g = G$)
- nt - Number of users in the thinking state
- request_set - Contains t_arrival (time of arival) and item ID of a user's request for an f-item, and whether the the request is for an i-item or an f-item
- irequest_queue - Contains t_arrival of requests for i-items in a queueing structure

1.2 Event types

- F-Arrival
- I-Arrival
- Departure (parameters:t_arrival,type)
- Start_Transmit (paramters:k,g)
- End_Simulation

1.3 Initialization

- clock = 0
- $nt = N$
- initialize event_set to empty
- initialize request_set to empty
- initialize irequest_queue to empty
- for $n = 1, 2, \ldots, N$

    determine think_t

    schedule an F-Arrival event at clock + think_t

  end for

- k = 1
- g = 1
- determine interarrival_t (interarrival time for i-request)
- schedule a I-Arrival at clock + interarrival_t
- schedule a Start_Transmit(k,g) event at clock
- schedule an End_Simulation event at time term_sim

1.4 F-Arrival event

- nt = nt -1
- determine id of item requested by user
- enter entry (clock, id, f-item) to request_set

1.5 I-Arrival event

- enter entry (clock, 0, i-item) to request_set
- determine interarrival_t (interarrival time for i-request)
- schedule a I-Arrival at clock + interarrival_t

1.6 Start_Transmit event

- determine transmit_t, transmission time of an information item
- If g = G then

    g = 0

    if irequest_queue is non-empty then

        remove entry from queue, and retrieve t_arrival, and type

        schedule a departure(t_arrival, type) at clock + transmit_t

    end if

  end if
- else

    for each entry in request_set

if item_id = k then

remove entry from set, and retrieve t_arrival, and type

schedule a departure(t_arrival, type) at clock + transmit_t

end if!

end for

k = k + 1

g = g + 1

if k > N, then k = 1

- schedule a Start_Transmit(k,g) event at clock + transmit_t

1.7 Departure event

- If type = f-item

  record metrics

  nt = nt + 1

  determine think_t

  schedule an F-Arrival event at clock + think_t

- else

  record metrics

  end if

1.8 End_Simulation event

- Exit main loop

2 Part b) Length Of Run Determination

2.1 Analysis

For this problem, I chose the values $L = 50000$ and $m = 8$ to generate my confidence interval. The results of these simulations can be found on the following page.

From my 8 replications, I took the average of each of the R values to get a sample mean of ≈ 14.31. The standard deviation of the R values was also calculated to be $s \approx 0.063$. The t value is listed in the textbook as 2.365 as $t_{(7,0.975)}$ which is the value for a 95% confidence interval for 7 degrees of freedom.

The confidence interval was then computed using interval= $2 * t * \frac{s}{\sqrt{m}} \approx 0.105$. The $\pm 1\%$ inteval is simply 0.02∗sample mean ≈ 0.286. We can see that our 95% confidence interval is will without the $\pm 1\%$ interval and we can conclude that the chosen L of 50000 is a valid run length for our simulation in order to get sufficiently accurate results.

2.2 Raw Data

b)

Sample Mean	Standard Dev	t value	Confidence Interval	Width of 1% Interval
14.3069	0.0628	2.3650	0.1051	0.2861

	Replication 1	Replication 2	Replication 3	Replication 4	Replication 5	Replication 6	Replication 7	Replication 8
R	14.28	14.25	14.29	14.31	14.22	14.42	14.36	14.32
Ri	5.47	5.29	5.52	5.59	5.47	5.72	5.48	5.55
R1	16.65	16.68	16.71	16.79	16.7	16.9	16.81	16.74
R2	17	16.87	17.05	17.05	16.98	17.14	17.23	17.04
R3	16.63	16.7	16.61	16.66	16.48	16.73	16.77	16.64
R4	15.44	15.57	15.53	15.76	15.41	15.56	15.67	15.67
R5	14.79	14.87	14.73	14.79	14.73	15.01	14.83	14.92
R6	14.09	14.05	13.53	14	13.8	14.28	13.69	13.77
R7	13.37	13.17	13.19	13.13	13.26	13.57	13.58	13.7
R8	12.74	12.98	12.94	12.54	12.46	12.92	12.95	12.6
R9	12.34	12.32	12.22	12.59	12.34	12.7	12.65	12.44
R10	12.33	12.71	12.32	12.26	12.31	12.64	12.51	12.26
R11	12.09	12	12.19	12.19	11.96	11.91	12.19	12.49
R12	12.27	12.42	12.27	12.37	12.47	12.61	12.15	12.5
R13	12.74	12.78	12.61	12.57	12.45	12.3	12.66	12.69
R14	12.9	12.99	12.85	12.66	12.89	13.17	12.86	12.76
R15	12.89	12.97	13.22	13.17	13.25	13.02	13.46	12.98
R16	13.63	13.82	13.67	13.53	13.63	13.85	13.72	14.05
R17	14.23	14.35	14.48	13.9	14.24	14.05	14.02	14.14
R18	14.74	14.65	14.78	15.09	14.98	15.12	15.03	14.68
R19	15.25	15.12	15.33	14.8	14.97	15.43	15.55	15.37
R20	16.36	15.94	16.21	16.03	16	16.12	16.04	16.28

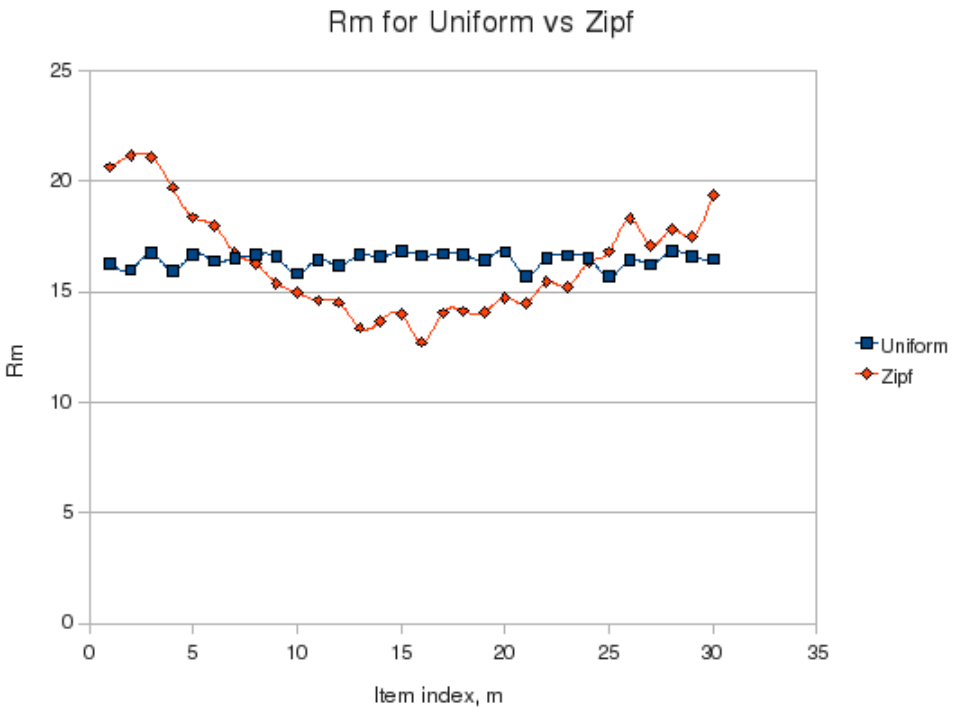
Table 1: Raw data for part b)

3 Part c) Data Retrieved from Varying Factors

The raw data is attached at the end of this report, and each graph will be presented when it is discussed in part (d).

4 Part d) Analysis of Varied Factors

4.1 Zipf vs Uniform Distributions



Looking at the graph above, we can see that uniform's R_m values are all approximatey the same. This is to be expected since the items are both requested with a uniform distribution (each item has the same likelihood of being chosen) and each item is served for the same amount of time on average since we serve them in a circular fashion for the same mean time.

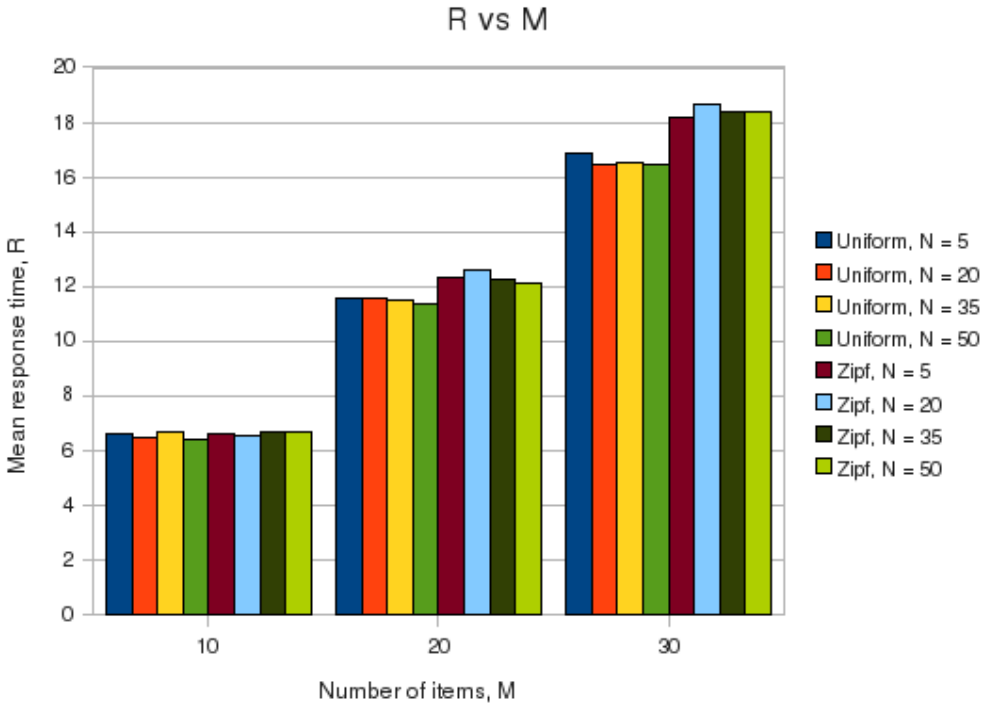
We can also see that the zipf distribution is quite different from the uniform distribution. In particular, notice that both the low R_ms and the high R_ms are higher than in the uniform distribution, and the middle range m produce lower R_ms for the zipf distribution.

The formula for the zipf distribution indicates that the first item has a particular probability of being chosen, c. The second item has half that chance of being picked, the third item one third of c, and so on. This would indicate that the lower id f-items are requested much more frequently than the high id ones. This is what causes the distribution that we see in the graph.

Since each user is most likely to pick item 1, then a large number of users will be waiting for item 1 to be served. Once it is served, each user will have a think time (with a mean of 5) and then they will place a request for a new item. In that time, about 5 f-items will be transmitted, and k will be about 6. Now, we can see that at this time, most of the users will again request item 1, since it is the most likely. They will have to wait for the whole cycle to complete before 1 is transmitted again, which is what causes the higher R_m for $m < 6$ in the zipf distribution. For higher m values, the probabilites are quite close (consider $\frac{c}{29}$ and $\frac{c}{30}$). And, since most of the requests are made while transmitting item 6 or so, the users that unfortunately requested higher id items will have to wait until the end of the cycle, which is what causes the high response time for large m. Lastly, of course the users that requested items in the middle range will get them quickly, because the requests are made right before that range.

The graph above uses $M = 30$ since having more data points allows us to more clearly see the relationship. The graphs for other M values (not included in this report) are quite similar, except that we can imagine the response time for higher id requests is lower, since the cycle is smaller.

4.2 Effect of M on R_m

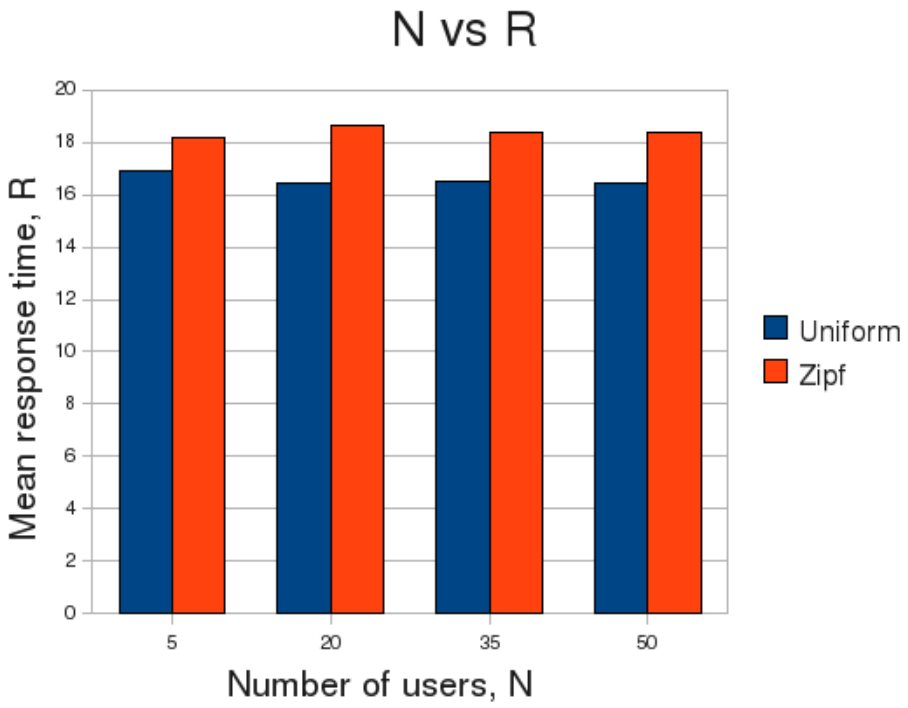


This chart clearly illustrates the effect that varying M (the number of f-items) has on the mean response time R. Larger values of M produce larger values of R regardless of N or the distribution used for p_m. This can be very easily explained.

Each f-item is transmitted in a cycle, with an average transmit time of 1. Since our simulation runs for a finite length of time, we can clearly see that if we have more f-items, each one will be transmitted less times, and thus less frequently. Of course, if each item is transmitted less frequently, then each individual request will be serviced slower, and thus that will increase the mean response time R, as we see in the chart above.

The slight difference between zipf and uniform distributions comes from the fact that large M values means more items, which means more requests are for items further away from 6. This slightly increases the mean response time.

4.3 Effect of N on R_m

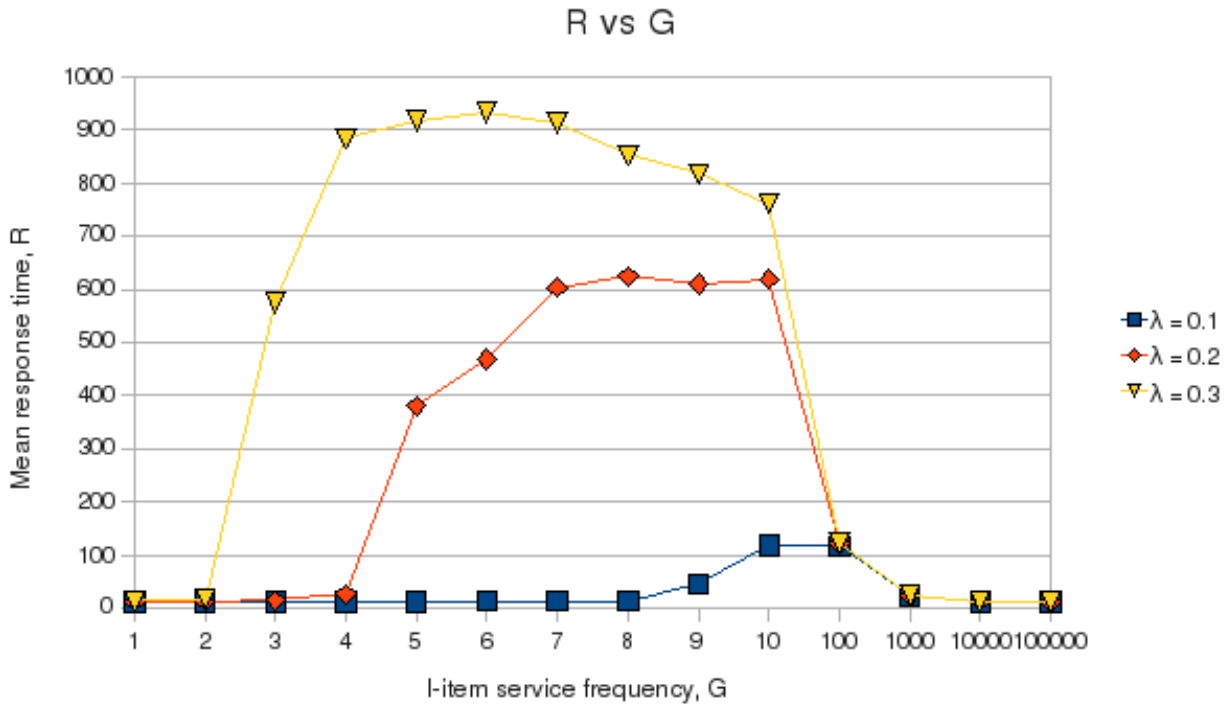


This chart shows how varying number of users, N, affects the mean response time R. For this chart, the value $M = 30$ was chosen arbitrarily since we saw in the previous section that varying N has the same effect on R independent of M.

It is quite obvious to see from the chart that varying the number of users has little influence on R. This can also be easily explained. The model works in way such that user's requests are stored on their workstation, and the server uses a broadcast method to serve up the frequently requested items. In broadcast mode, the server only broadcasts the items and any number of users can listen to the broadcast without impacting the transmit time. From there, it is logical to conclude that increasing the number of users will only increase the number of requests, and not how often each request is services. Since the distribution remains the same, the mean will be unaffected as we see in the chart.

5 Part e) Finding an Optimal G

5.1 Data



e)

	λ = 0.1		λ = 0.2		λ = 0.3	
G	**R**	**Ri**	**R**	**Ri**	**R**	**Ri**
1	12.27	2.38	13.05	2.95	14.39	4.25
2	12.27	3.35	13.28	5.33	17.66	26.76
3	12.27	4.55	13.97	12.81	575.38	4542.49
4	12.4	6.46	26.77	149.02	884.48	8921.83
5	12.53	9.65	380.43	4553.18	916.33	11170.58
6	12.91	16.49	468.82	6612.17	932.99	13312.45
7	13.08	22.98	604.08	9840.15	912.4	14971.65
8	13.92	43.66	623.78	11456.27	852.91	15733.52
9	46.5	719.25	610.4	12444.83	816.9	16794.49
10	118.47	2445.71	619.65	13976.28	761.21	17221.18
100	116.44	22442.36	122.74	23792.15	124	24060.09
1000	22.92	24730.88	23.14	25048.45	23.29	25285.62
10000	12.54	25061.79	12.91	29866.45	12.46	24980.52
100000	11.48	NaN	11.46	NaN	11.53	NaN

5.2 Analysis

The chart above shows the relationship between the frequency of servicing i-items, G, and the mean response time, R, for various values of λ. Note that R values for larger G are also provided at an exponential scale. That data used to create the chart is also provided above.

From the chart we see that for the most part, the smaller R is, the smaller G is. This would lead us to believe that $G = 1$ is the optimal choice. However, we must first consider a couple other special cases.

First, we notice that for $G > L$, we get an even smaller R. In this case, it is obvious that not a single i-request is serviced, and thus only f-requests are handled. Although strictly speaking this gives us a lower R, we should not have to completely stop sending i-items, and so this case is discarded. Similarily, for very large G (see $G = 10000$), only a handful of i-items are serviced. This causes little impact on the mean response time, however we see that the i-item response times are way too large and these cases should also be discarded.

We could also consider the case in which $G = 0$. In this case, only i-items are serviced and f-requests are completely ignored. This situation is so against our common sense that I did not evaluate R values for such a situation, and it to is ignored.

Ignoring these impractical cases, we are left with $G = 1$ being the ideal value to minimize mean response time.