Brian Dinh
CS 115
Fall 2017

Project Aloha:
Comparing the Slotted and Unslotted Aloha Protocols

Introduction

This simulation was coded in SimPy. The implementation was based on the ruleset given in the project pdf, but there were some rules that had to be implicitly assumed as well. Some of the aforementioned rules or guidelines were not listed in the pdf, but instead answered through an email or on Piazza. They are as follows:

- All simulations are run for 100,000 time units. Each run is repeated 1000 times and the results are averaged to produce the final value
- Transmission times are 1 time unit
- When a host generates a new packet:
  - if the host is idle the packet is transmitted immediately (however, this allows the Capture Effect and reduces fairness amongst hosts)
  - if the host is not idle the new packet is placed in a transmission queue
- The host is considered to be idle if it is not currently transmitting or retransmitting a packet
- Packets are chosen to be transmitted in a FCFS manner according to the transmission queue; packets cannot preempt
- Packets marked as collided are not discarded or destroyed; instead they will continually be retransmitted until successful.
- All throughput values were rounded up to the nearest hundredths place. Complete, un-rounded values can be found in the appendix

Brian Dinh
CS 115
Fall 2017

What is the Maximum Global Arrival Rate?

The first goal was to find the maximum rate of packet arrivals (R) that the system could maintain. In order to do that, I ran the simulation for both protocols ten times for R in [0.1, 1] with n = 10 hosts. Throughput was calculated as p / t, where:

- p = the number of packets successfully transmitted
- t = total elapsed simulation time

I did not calculate the throughput with R = 0 because that is trivial. If packets arrive every 0 time units then there would be 0 transmitted packets.

After 1000 complete runs at every R value, I computed the mean throughput. Figure 1 shows the results.

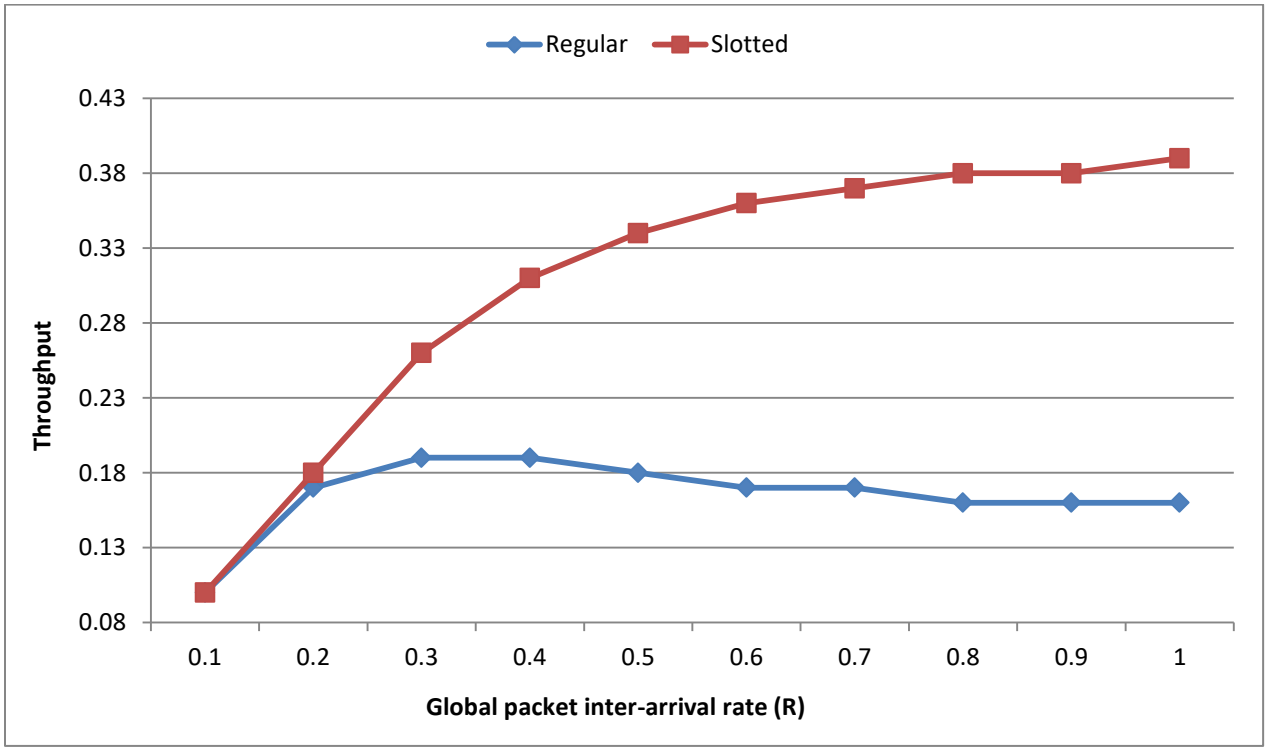

Figure 1: Mean throughput comparison of regular Aloha and slotted Aloha protocols as a function of R from 0.1 to 1 with n = 10 hosts

Brian Dinh
CS 115
Fall 2017

I chose R_{max} values corresponding to the highest value throughput at a peak or before a plateau on the graph. Throughput results were:

- Regular Aloha: Throughput = 0.19 at R_{max} = 0.4
- Slotted Aloha: Throughput = 0.39 at R_{max} = 1

In other words: in a system with 10 hosts, regular Aloha protocol can transmit 19% of all packets if the 0.4 packets arrive every time unit. Slotted Aloha can transmit 39% of all packets if 1 packet arrives every time unit.

Because the values were rounded up to show only two decimal places, some other values of R_{max} for regular Aloha were considered. For example, the actual highest throughput achieved for regular Aloha was actually 0.191476 at R = 0.3, compared to 0.188296 at R = 0.4. I decided to truncate all the other decimal places up to the hundredths because the results were too small to make a difference. From 0.5 and above, the rounded results were not close enough to consider. Therefore, I chose 0.4 as the R_{max} for regular Aloha.

Is R_{max} Independent of n?

In order to figure out if R_{max} was independent of n, I had to know if the results were similar with a large number of hosts. I compared the mean throughputs for both protocols at n = 10, n = 100, n = 1000, n = 10,000, and n = 100,000 hosts respectively. An attempt was made to run a simulation at n = 1,000,000 hosts but my computer could not handle it. I suspect it was due to how I coded my retransmission function, which was recursive. However, I believe 100,000 hosts is definitely enough to count as a “large” number of hosts. Figure 2 shows the results.

Brian Dinh
CS 115
Fall 2017

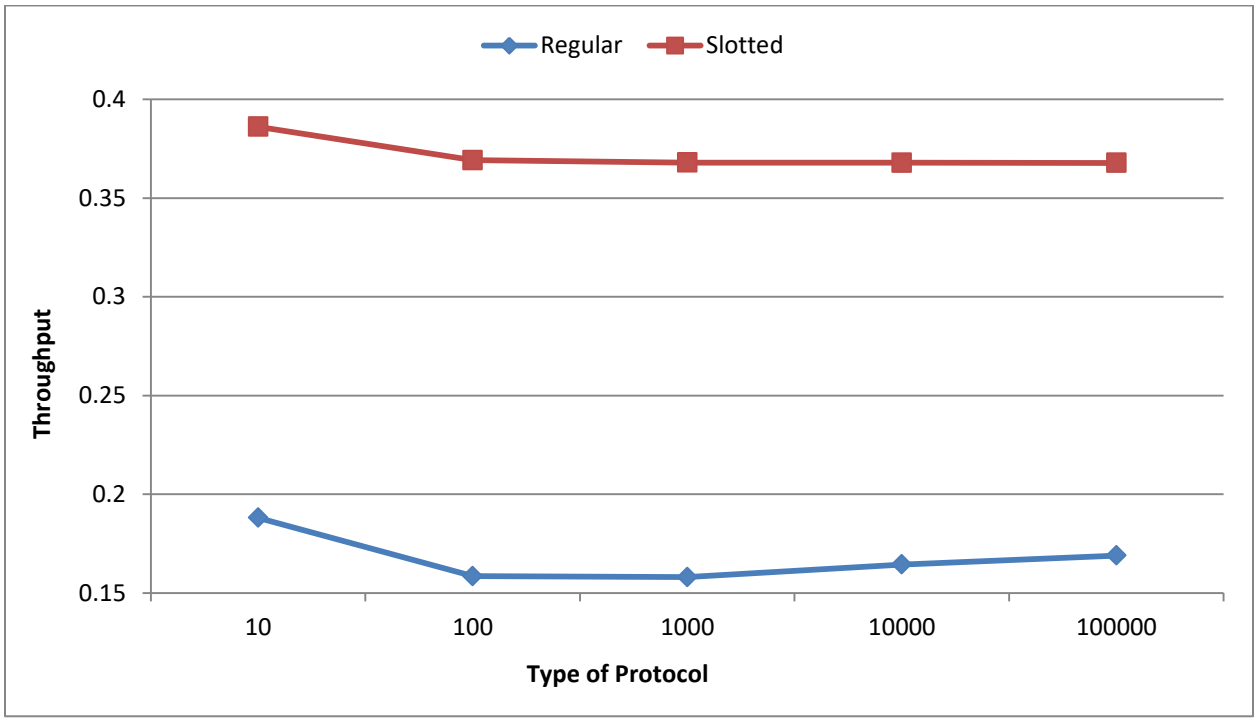

Figure 2: Mean throughput of regular Aloha and slotted Aloha protocols as a function of n from 10 to 10000 hosts. Both protocols are plotted using their R_{max} values (regular = 0.4, slotted = 1)

The results show that throughput is not independent of n. With a large number of hosts, throughput is definitely down. Intuitively, this makes sense. In a real system, if many people are trying to utilize one medium there are bound to be collisions and slow downs on the network.

Although a large number of hosts did slow down the system throughput, two peculiarities arose from these runs. Slotted Aloha throughput drops to be 0.37 and stays there. This plateau contrasts regular Aloha, where the throughput seems to increase as n gets larger; however, it never goes above the limit at 0.19. I suspect there to be a slight coding error for the regular Aloha protocol as I don't believe the throughput should rise. However, this might also mean that I should have run more than just 1000 runs. Maybe some higher value throughputs skewed the mean results.

Brian Dinh
CS 115
Fall 2017

What is the Optimal Retry Rate?

I tried four different retry rates over both protocols. All rates were parameters to the exponential distribution function, which I will define shorthand as *exp()* for brevity. The rates were:

1. *exp((1/n)/a)*
2. *exp(1/(n*a))*
3. *a*exp(1/n)*
4. *exp(r/n)*

The first three modified rates utilized the default rate *1/n* with the addition of a variable *a* in various configurations. I defined *a* as the number retransmission attempts from a host. If a host's packet keeps colliding, then they will wait longer and longer after each attempt to retransmit so that more successful hosts' packets can move on. The fourth rate is similar to the global packet arrival rate (more in-depth explanations later). Figure 3.1 shows the results.

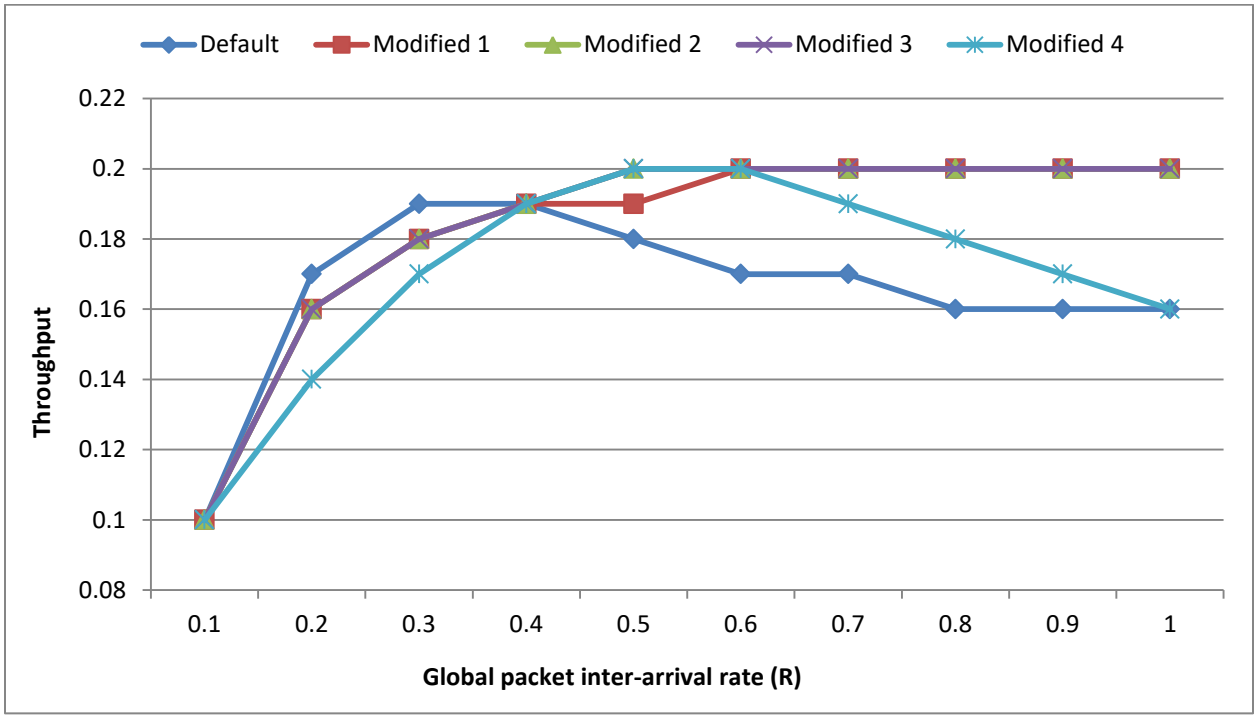

Figure 3.1: Mean throughput comparison of regular Aloha protocol retry rates. Plotted as a function of R from 0.1 to 1 with n = 10 hosts

Brian Dinh
CS 115
Fall 2017

The results for regular Aloha show that every modified rate either plateaus or decreases after R = 0.6. Rates 1, 2, and 3 basically had the same curve, meaning the addition of *a* did not change the resulting plot much no matter how it was used. Rate 4 acted more in line with the original rate plot where it grew to a peak and then declined.

The new R_{max} of 0.6 also yielded a new higher throughput of 0.20. Although all modified rates had the same result, I decided to choose rate #4 as the best choice because the curve acted more similarly to the original plot curve. In addition to that, it made more sense when subjecting and comparing it to the original project guidelines. The reasoning is that:

- The default rate of *1/n* means that on average, a host would be trying to retransmit every 1 time unit
- We already know from figure 1 that a global arrival rate R = 1 (one packet every 1 time unit) for regular Aloha produces a suboptimal throughput
- Therefore, *1/n* is not optimal, but could be tailored to increase R_{max} and throughput.

Thus, I took modified rate 4 and changed the *r* to be a fixed value. I tested the new retry rate with three *r* values of 0.19, 0.4, and 0.6. These specific values of *r* were chosen because of their connection to previous results. Values *r* = 0.19 and *r* = 0.4 were the previous highest throughput and R_{max} respectively. The idea was that the system could only push out as much as it could put in. If the system was not overloaded then the values would increase until a plateau or a peak. Finally, *r* = 0.6 was chosen because it was the newest R_{max} value. Figure 3.2 shows the results

Brian Dinh
CS 115
Fall 2017

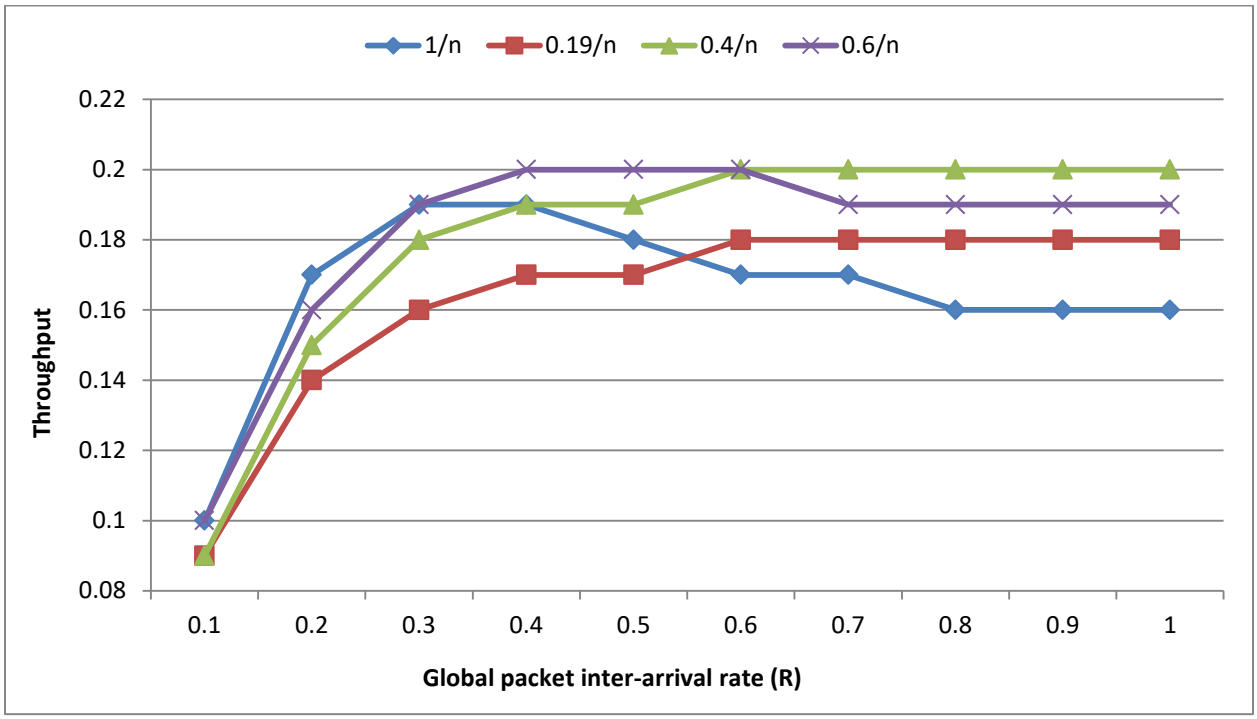

Figure 3.2: Mean throughput comparison of regular Aloha protocol retry rate #4. Plotted as a function of R from 0.1 to 1 with n = 10 hosts

We see that each *r* still peaks or plateaus at R = 0.6, but *r/n* with *r* of 0.6 produces the highest output curve. I believe this makes sense intuitively because, again, the system can only push out as much as it puts in. If the global arrival rate is maxed out at R = 0.6, then accordingly, retransmission rates must also be capped at 0.6.

For slotted Aloha, I did not do a test for R in [0.1, 1] because I knew the R_{max} was already at the limit of 1. The test then was to maximize the throughput at R = 1 if possible. I tried the same rates as I did for regular Aloha with the exception of modified rate #4. This is because the best value of *r*, according to the results from figure 3.2, must be R_{max}. However, in slotted Aloha this would mean *r* = 1, which is the same as the default rate. Therefore, rate #4 is omitted. Figure 3.3 shows the results.

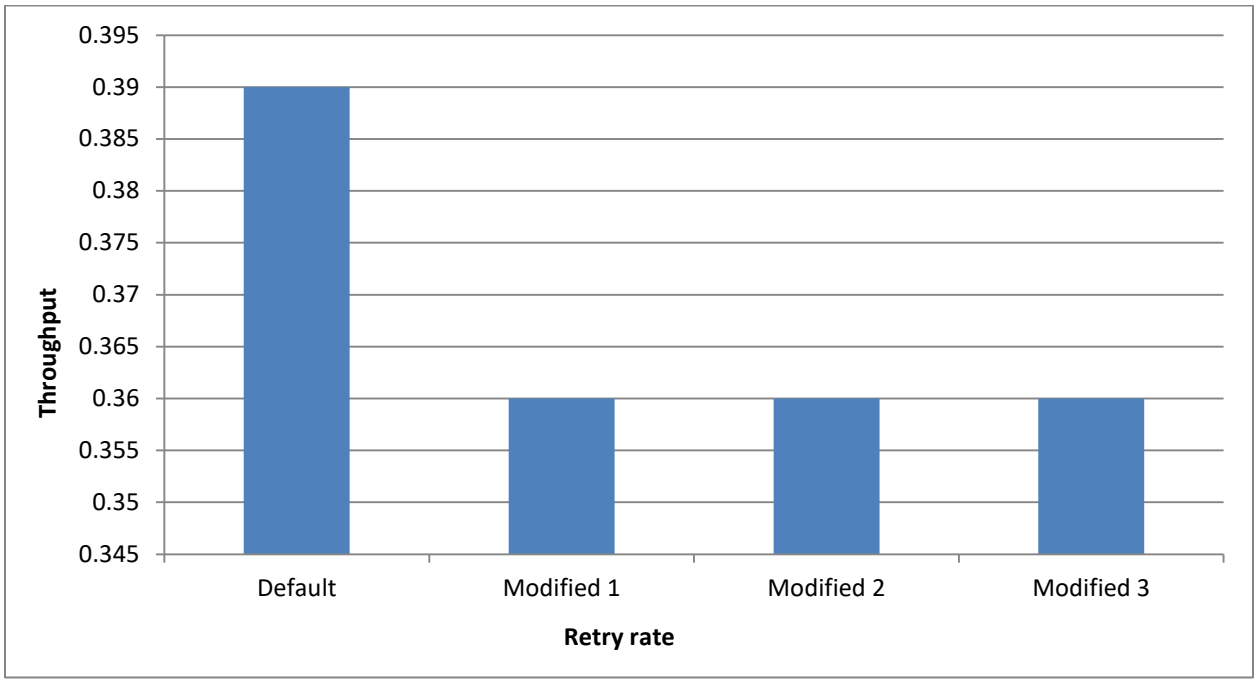

Figure 3.3: Mean throughput comparison of Slotted Aloha protocol retry rates. Plotted as a function of R = 0.1 with n = 10 hosts

Throughput for the modified rates is capped at 0.36. Again, it is similar across the board as was seen in regular Aloha. The addition of *a* did not make any difference. It seems as though there is no way to improve upon slotted Aloha's results if I were to continue with the same *r* technique I utilized in regular Aloha. It would produce the same results as the default value.

As it is now I do not have an answer for finding a better slotted Aloha retry rate than 1/n, which I believe is pretty good. Although slotted Aloha provides better general throughput, large numbers of collided packets can become an issue because they have to transmit in a slot instead of whenever they want. This causes a snowball effect and blocks many packets into a continuous retransmission state. That is why the three modified rates were so similar. When the number of attempts are high packets are never able to transmit, causing throughput to drop.

Brian Dinh
CS 115
Fall 2017

Conclusion

My R_{max} values seem pretty accurate when comparing them to values in various textbooks on computer networking and related literature online. However, my throughput values for the default retry rate of *1/n* are slightly higher than the maximum theoretical throughput of 0.18 and 0.36 for regular Aloha and slotted Aloha respectively.

The discrepancy is most likely due to programming error. I may have misinterpreted the guidelines in the project pdf and coded them slightly off. The results could have been shifted ever so slightly because of the program assumptions I had listed in the introduction. It could also be that the guidelines themselves are specific to this simulation project and is not meant to be a complete Aloha system.

I believe I could have found a better retry rate for slotted Aloha with more time or with closer study of the workings. Perhaps I was looking at the timing and rates the wrong way. Ultimately my results are close enough to the correct values so I am content.

Brian Dinh
CS 115
Fall 2017

Appendix

Figure 1 un-rounded values:

	Regular	Slotted
0.1	0.096052	0.097377
0.2	0.166575	0.184896
0.3	0.191476	0.256319
0.4	0.188296	0.307881
0.5	0.179436	0.341678
0.6	0.172107	0.362027
0.7	0.166623	0.373543
0.8	0.162403	0.380495
0.9	0.159427	0.384254
1	0.156617	0.38615

Figure 2 un-rounded values:

	Regular	Slotted
10	0.182093	0.37114
100	0.152602	0.298489

Figure 3.1 un-rounded values:

	Default	Modified 1	Modified 2	Modified 3	Modified 4
0.1	0.096052	0.095286	0.095252	0.095241	0.08285
0.2	0.166575	0.159833	0.15967	0.159592	0.137194
0.3	0.191476	0.184101	0.184258	0.18426	0.170312
0.4	0.188296	0.191553	0.19159	0.191631	0.188337
0.5	0.179436	0.194418	0.194561	0.194631	0.195679
0.6	0.172107	0.196098	0.195717	0.195872	0.195153
0.7	0.166623	0.196984	0.196881	0.196788	0.189556
0.8	0.162403	0.197552	0.197718	0.197589	0.180493
0.9	0.159427	0.198412	0.198336	0.198187	0.169165
1	0.156617	0.198867	0.198837	0.198927	0.156716

Brian Dinh
CS 115
Fall 2017

Figure 3.2 un-rounded values:

	1/n	0.19/n	0.4/n	0.6/n
0.1	0.096052	0.089304	0.093095	0.094785
0.2	0.166575	0.136966	0.152825	0.160199
0.3	0.191476	0.157345	0.178796	0.187869
0.4	0.188296	0.16688	0.188735	0.19528
0.5	0.179436	0.172333	0.192474	0.196003
0.6	0.172107	0.175675	0.194589	0.195158
0.7	0.166623	0.177928	0.195631	0.194156
0.8	0.162403	0.180214	0.196164	0.192845
0.9	0.159427	0.181789	0.196578	0.191844
1	0.156617	0.183381	0.196881	0.191089

Figure 3.3 un-rounded values:

	Values
Default	0.38615
Modified 1	0.35903
Modified 2	0.359354
Modified 3	0.359879