Low-Variance Retrial Times in a Multiserver Queueing Model

by

David W. Lubke

Thesis

Submitted to the Department of Mathematics

Eastern Michigan University

in partial fulfillment of the requirements

for the degree of

MASTER OF ARTS

in

Mathematics

Thesis Committee:

Andrew Ross, PhD, Chair

John Curran, PhD

Tanweer Shapla, PhD

January 8, 2010

Ypsilanti, Michigan

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my wife, Katie, for all of her patience and encouragement throughout my time in graduate school. She has spent an enormous amount of time with our children, Noah and Jessica, who know only of life with daddy in school. My advisor for this project, Dr. Andrew Ross, I count among the best teachers and mentors I have ever had. Thank you Dr. Ross for teaching me so much in class, on this project, and even when preparing me for my job that I started halfway through my thesis project. Finally, I want to thank my thesis committee, Dr. Tanweer Shapla and Dr. John Curran.

Abstract

A retrial queue model arises often in analyzing call centers and cell-phone systems where there is no organized queue. If all servers are busy when a customer calls, the customer must try again after some length of time. We use general low-variance retrial time distributions in a multi-server retrial queue model of moderate size (10-60 servers). We compare the error in predicting system performance when approximating with an exponential distribution and examine a counter-intuitive pattern.

This paper addresses the question of whether the exponential distribution is a good enough approximation for deterministic-retrial queue systems, and how large is the percentage error. We show that retrials have a substantial effect on multi-server queues and should not be ignored by assuming an exponential distribution, especially at low retrial rates. In addition, we investigate the cause of this difference by developing two schemes, which are a cross between deterministic and exponential retrial durations.

Introduction, Problem Statement, and Background

A typical example of a queueing system is a call center where calls arrive, wait on hold, and get served by customer service agents. Those calls waiting on hold are considered in the "queue" while those that are actually on the line with an agent are "in service". Upon completion of the call, the caller exits the system and presumably does not reenter. In some systems, though, there is no organized on-hold queue; if all of the servers are busy when a call arrives, the caller gets a busy signal and must call back at some random time in the future to try again. In this type of queueing system, this caller is considered in "orbit" while retrying to enter the system. In other words, orbit is an abstraction to account for retrial calls that could not enter the system each time they arrived because all of the servers were busy. Typical examples of retrial queueing systems include mobile phone cells (if all channels are busy when you try to place a call, it won't go through and you aren't placed on hold to wait your turn), dial-up internet access, computer networks using Carrier Sense Multiple Access with Collision Detection (CSMA-CD) like Wi-Fi and the Ethernet protocol, and an interesting phenomenon in programming called spin lock.

A few of the main attributes that define a queueing system are the rate at which calls arrive to the system, the rate at which they are served, and the number of servers. The arrival rate, λ, is the number of calls per hour arriving to the system from outside. This was set to 9, 16, 25, or 36 calls per hour. Similarly, the service rate, μ, is the number of calls per hour that each server can handle. In our study, we fixed μ at one call per hour to simplify the simulation. Since any queueing system can be scaled in time to

have $\mu = 1$, this is common practice for a study such as ours. The variable traffic, ρ, of the system is defined as:

$$\rho = \frac{\lambda}{\mu} \tag{1}$$

Hence with $\mu = 1$, traffic is equal to the arrival rate. Similarly, utilization, the fraction of time, on average, a server is busy, is defined as

$$\hat{U} = \frac{\lambda}{c * \mu} \tag{2}$$

where *c* is the number of servers in the system.

In real world applications of queueing theory, such as a set of toll collection booths on an expressway, there are large fluctuations in traffic over the course of the day (heavy volumes of traffic in the morning and late afternoon) and even the week (less traffic on the weekends than during the week). As is common in queueing theory, our study looks at constant arrival rates, leaving the more complicated question of time-varying arrival rates for later work.

In queueing theory literature, a common assumption is for the interarrival and service times to be exponentially distributed and independent of one another. In standard queueing notation this is an M/M/c queueing system where c denotes the number of servers. "M" is used for the exponential distribution due to the memoryless property, which is also called the Markovian property.[1] Various other distributions have been studied such as Erlang (Ek), deterministic (D), and a general distribution (G).[2]

In some of above mentioned systems, such as the case of dial-up internet access, a customer could use software to automatically redial, therefore creating a situation that is less variable than exponential. The software could even retry at precisely regular

intervals, i.e. using a deterministic distribution. Additionally, if a customer must manually redial to enter service, such as in the mobile phone cell case, the memoryless assumption of the exponential distribution may no longer be valid.

It is often convenient to assume that the time from one retrial to the next, per person, has an exponential distribution—that is, the chance that they will retry in the next minute is the same no matter how long it has been since the last retrial. This implies a moderately large variability in the retrial times. However, this is inaccurate in cases where the retrials are done in an automated fashion. There, the time from one retrial to the next is very close to constant rather than random. These low-variance distributions are difficult to treat using standard queueing computations, especially for systems with more than a few servers. However, it is much easier to analyze a retrial queueing system with much more variable intervals, such as an exponential distribution. **The question we ask then is whether the exponential distribution is a good enough approximation for deterministic-retrial queue systems, and how large is the percentage error?**

An excellent example of a system where the redials are done in an automated fashion can be seen with the PowerDialer^{TM} (http://www.technologyarts.com/). This device is specifically made to redial a number as fast as 25 times per minute depending on the speed of the phone line. This equates to an arrival every 1/1500th of an hour or, in other words, it retries every 2.4 seconds. While this is an extreme case, the technology does exist and is in use.

Purpose of the Study

Our research is unique in that much of the previous work on retrial queue systems assumes the retrial times to be exponentially distributed. While still keeping the arrival

and service times exponentially distributed, we are using deterministic retrial times and then comparing several performance measures such as the average number of calls that are retrying from orbit and probability of delay to the corresponding performance measures for exponentially distributed retrials. Additionally, many other retrial queue models are based on only one server. We are evaluating a much more complex system with multiple servers. This is a more realistic simulation of a real world retrial queue system that may be found in the telecommunications industry.

In the end, our goal is to quantify the difference between our non-exponential system and the exponential retrial system. Further, we provide our explanation of why we see these counterintuitive patterns, which have not yet been reported in the literature.

Justification and Significance

Until the mid-1970s, the effect of retrials in queueing systems were not widely studied. Multiple researchers point to the book by Kosten, where he states that "any theoretical result that does not take into consideration this repetition effect should be suspect" [3] to emphasize the point that standard queueing models do not always accurately depict the real world. Since then**,** a fair amount of research has been done on retrial queue systems. Two primary works are worth mentioning: *Retrial Queueing System: A Computational Approach*, by Artalejo[4] and *Retrial Queues*, by Falin and Templeton.[5] Thus far, however, much of the research on retrials has studied small-scale systems of typically one or two servers. For example, Artalejo[6] analyzed a single-server system in which customers arriving according to a Poisson process are sent to orbit if the server is busy. If the server is free, the customer is served according to a general distribution. Then, whenever the server is not busy, it searches out a customer from orbit. Artalejo

and Lopez-Herrero[7] also analyzed a discrete-time multiserver queueing system with retrials but used a geometric distribution for arrival times, service times, and retrial times. Various other retrial queue scenarios have been studied, a sampling of which can be found in a bibliography compiled by Artalejo[8].

Retrial queueing systems can be found in a wide variety of real world applications. Perhaps the most common in queueing literature are computer and telecommunication literature.[7] Multiple papers such as those by Kelly,[9] and Harris, et all,[10] apply retrial queues specifically to telephone systems. Others apply retrials to local area networks[11,6] or cellular telephone systems.[12,9,10] Interestingly, Chakravarthy[13] uses the analogy of a package delivery service with multiple stations, where each station is manned by a 2-person team. One person's sole function is to process packages while the other's main duty is to answer the telephone. The package handler is trained to also assist on the phone; however, the other attendant is not trained to process packages. "Any phone calls arriving when both servers are busy will go into orbit and compete for service at random intervals of time."[13] This is a fairly complex example of a two-server queueing system where two types of customers arrive, in this case packages and telephone calls.

Perhaps the most interesting application is a process called "spin lock," which occurs in computer programming, especially in low-level programming such as assembly language or operating system kernels. Spin locks are used, for example, in a multiprocessor operating system where perhaps one processor has control of a process or block of memory and another processor wants access to it. The waiting processor spins in a loop of waiting and checking to see if the resource is available while the other

processor has a "lock" on the resource. An early paper by Gilbert[14] applies queueing theory to spin locks in a multiprocessing operating system.

Research Question

In this paper, we show how we use our simulation to determine just what the percentage error is between the deterministic cases compared with an exponential retrial distribution for multiple performance measures. We quantify ranges for which the percentage error is either very large or negligible. Other questions we set out to answer are just how far from exponential is the worst case and for what retrial rates? We also investigate whether the discrepancies get larger or smaller as the system size increases.

Our goal is to determine if the exponential distribution is really a suitable approximation for deterministic-retrial queueing systems and to determine the percentage error. We accomplish this through repeated discrete-event simulations in MATLAB using deterministic and low-variance retrial distributions and comparing the results to those of an exponential-retrial system, computed using a Continuous Time Markov Chain (CTMC) rather than discrete-event simulation.

Methodology

Our research consists of utilizing MATLAB to simulate retrial queueing systems of moderate size (10-60 servers). We randomly simulate up to 40,000 calls arriving to a multi-server system with a buffer size of zero and process them to find how long each call waits, how many times it retries, and so on. The resulting average performance measures are then compared to what would have happened with exponentially distributed retrial times.

The main inputs to the system consist of the retrial rates, arrival rate, service rate, the number of servers and some flags to determine the method of generating retrial durations. We used the same retrial rate vector of [0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9, 3, 4, 6, 8, 10, 12, 15, 20, 30, 45, 60] retries per hour and the service rate, μ, was set to 1 call per hour. We simulated 4 different system sizes where we set the arrival rate equal to 9, 16, 25, and 36, then determined the number of servers, *c*, by a method proposed by Halfin and Whitt,[15]

$$c = \rho + z \times \sqrt{\rho} \tag{3}$$

where *z* is chosen to be either 1 or 2 based on an approximation from the normal distribution table. This method (designed for ordinary queues, not retrial queues) is known as the Square-Root Staffing (SRS) rule, and it is quite common in both theory and practice [16]. Using z = 1, gives us 12, 20, 30, and 42 servers respectively. In ordinary queues, setting z = 1 results in a probability of delay of approximately 0.25. Setting z=2 gives a much lower probability of delay (roughly 0.04 for an ordinary queue), which in our case makes retrials more rare and less important, so we do not study the z=2 case.

The retrial duration flags determined which of 4 different types of simulations were run for each system size. We devised two flags; one method we called "shared sequence" retrial times (SSRT) and another we called "personal" retrial times (PRT). Flags were set to either 1 for true or 0 for false. In the SSRT scheme, every customer has the same set of retrial durations, but that duration changes with each retrial. The opposite is the PRT scheme, where each customer has his or her own retrial duration which is used over and over for each retrial. The deterministic case is a composite of personal and shared sequence retrial durations, since each call uses the same retrial duration for every

retrial. Finally, we have the classic exponential case where retrial times are independent both in retrial duration and for each retrial. TABLE 1 illustrates the difference between the four cases.

TABLE 1. Sample Retrial Durations

EXPONENTIAL					
Cust.	Retrial Number				
Number	1	2	3	4	5
1	0.0105909	0.033035	0.045083	0.0070491	0.0027464
2	0.0085752	0.0068789	0.001576	0.0122603	0.0076149
3	0.0377211	0.0102219	0.0101105	0.041561	0.0003325
4	0.0392786	0.0016868	0.0014015	0.0162152	0.0725114
5	0.0328694	0.0241654	0.0121502	0.0054429	0.0042033

PRT					
Cust.	Retrial Number				
Number	1	2	3	4	5
1	0.0105909	0.0105909	0.0105909	0.0105909	0.0105909
2	0.0085752	0.0085752	0.0085752	0.0085752	0.0085752
3	0.0377211	0.0377211	0.0377211	0.0377211	0.0377211
4	0.0392786	0.0392786	0.0392786	0.0392786	0.0392786
5	0.0328694	0.0328694	0.0328694	0.0328694	0.0328694

SSRT					
Cust.	Retrial Number				
Number	1	2	3	4	5
1	0.0105909	0.0085752	0.0377211	0.0392786	0.0328694
2	0.0105909	0.0085752	0.0377211	0.0392786	0.0328694
3	0.0105909	0.0085752	0.0377211	0.0392786	0.0328694
4	0.0105909	0.0085752	0.0377211	0.0392786	0.0328694
5	0.0105909	0.0085752	0.0377211	0.0392786	0.0328694

DETERMINISTIC					
Cust.	Retrial Number				
Number	1	2	3	4	5
1	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667
2	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667
3	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667
4	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667
5	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667

Call durations were generated easily in the deterministic case by creating a vector with length of 40,000 at each retrial rate where the duration was the reciprocal of that retrial rate. Hence the deterministic table above is for the last retrial rate of one arrival every 60 hours corresponding to a retrial duration of 1/60, or 0.01666... seconds. In the PRT and SSRT cases, we generated the retrial durations using exponentials with the same rate as the desired retrial rate. We used an exponential random variable divided by the retrial rate for both, but in the SSRT case we did this to create a row vector of length 100 (one for each trial). That row vector was then replicated 40,000 times to create the 40,000 by 100 matrix of retrial durations in the pattern seen in the above table. In this way, each of the 40,000 arriving calls uses the same sequence of retrial durations, but the duration is different for each of the 100 trials. For the PRT case, we created a column vector of length 40,000, which was then used repeatedly for all 100 trials; hence each of the 40,000 calls has its own retrial duration that is used through all 100 trials.

A small constraint was imposed on the personal retrials case so that retrials could not happen any faster than once per second. Initially, we discovered that the PRT simulation would not complete due to memory errors in MATLAB. Upon investigation we discovered that some retrial times were being generated that were so fast that the system just could not handle storing and processing all of the resulting events. For example, in one instance a retrial duration was generated to be 6.572×10^{-8} hours, or approximately every 0.00023659 seconds. In some cases, this would cause the average number of times a call tried to enter service to reach well into the hundreds. In one instance, the average number of times a call tried to enter service was 232, while the average number of tries, conditional on being sent to orbit, was 960. Ultimately, this led to the path size being so large that MATLAB gave an out-of-memory error. The Power Dialer mentioned previously can redial as fast as every 2.4 seconds, so we feel safe with an assumption that nothing can retry faster than about once per second.

Random numbers were generated in such a way to lend some repeatability to the experiments. Given the above mentioned retrial rate vector, the simulation ran through each retrial rate 100 times. The random seed was generated each time based on the retrial rate, the number of servers, and the sample number (1 to 100). Each sample and retrial rate generated 40,000 calls. Additionally, the random seed was then changed slightly before generating arrival intervals, service durations, and retrial durations.

With any queueing system, there are time periods at the beginning and end of the simulation which must be accounted for and trimmed from the analysis. Since the initial state of our system began with all servers and orbit empty, there is a warm-up period as the system builds up to steady state. Then, at the end of the simulation, there is a cool-

down period as new calls stop arriving from outside the system, and the only calls entering service are coming from orbit. Eventually, the system would work its way back to being empty. However, analysis of the system should be done at steady state so a method is needed to trim off the warm-up and cool-down periods.

To find the end of the warm-up period in our simulation, we compared the average number of calls in orbit for the CTMC portion of the simulation with the number in orbit at each event for the simulation being run. Essentially, the simulation ran one of the four types mentioned previously - deterministic, PRT, SSRT, or exponential - followed by a comparative CTMC simulation using the same input parameters. Then, to find the end of the warm-up period, the simulator looked for the point at which the number of calls in orbit was greater than or equal to the average number of calls for the CTMC data. The time point for our useable data was then the simulated time at which this event happened.

To find the beginning of our cool-down period, we used a simpler method of trimming the last 5% of calls. Since we had a maximum of 40,000 calls, it could be up to call number 38000. The ending time point for our useable data was then the simulated time that this happened. Throughout most of our simulations, we found this value to be fairly consistent of approximately 4200 hours or 175 days.

Results

To analyze the output of our simulations, there were several key variables we were interested in. First, we looked at the average number of calls in orbit, L_o. One of the reasons we were interested in this was to verify that our chosen orbit size of 100 for the CTMC computation was sufficient. Similarly, we were interested in the average

number of calls in the system, or L. Since our system had a queue size of zero, this was really the average number of calls in service plus orbit. Another common measure is the average wait time per visit to orbit, or W_o, and the average wait time of the system, or W; however, in our study, the graphs for wait time in orbit and for average number of calls in orbit had the same shape. Therefore, we chose not to include W and W_o in our analysis. Other outputs we did look at, though, included the probability that an arrival from outside was delayed, *Pr(delay)*, and the probability that a retrial fails to enter the system, *Pr(retry fail)*.

Using our arrival rate, λ, the preceding variables can be related by Little's Law, which states the average number of customers in the system is equal to the arrival rate times the average time each customer spends in the system. This can be represented algebraically as:

$$L = \lambda * W \tag{4}$$

Waiting time in orbit can be seen in two different ways. One view is that a failed retrial does not count as leaving then re-entering orbit; the call just stayed in orbit. The second view is that a failed retrial counts as leaving and re-entering. The arrival rate to orbit under the first view is just the rate at which outside arrivals are sent to orbit; we call this λ_{o1}. The arrival rate to orbit under the second view, λ_{o2}, is that of the first view plus the rate at which calls retry and fail. This can be shown as

$$\lambda_{o2} = \lambda_{o1} + retryrate * \Pr(retryfail) \tag{5}$$

Little's Law can also be applied to these views and some insight can be gained by doing so. First, applying Little's Law we see the following:

$$L_o = \lambda_{o1} * W_o(total) \tag{6}$$

$$L_o = \lambda_{o2} * W_o(each) \quad (7)$$

where $W_o(total)$ is the average total time a customer spends in orbit conditional on at least one retrial, $W_o(each)$ is the average time spent in orbit per retrial (simply 1 over the retrial rate), and L_o is the average number of customers in orbit. First, we examined our output to verify that these equations hold true. We can see our results of this check in TABLE 2, which shows the percent difference between the average number of customers in orbit from our simulation and that calculated using equations (6) and (7) above. For example, the top left entry is 0.06% more than the calculated value for L_o. This also serves as a check on our simulation code.

TABLE 2. Percent Difference from Simulated Average Number in Orbit, Calculated using Little's Law

Lambda_o1					Lambda_o2			
Determ	PRT	SSRT	IID	Retrial Rate	Determ	PRT	SSRT	IID
0.06	0.04	0.10	0.02	0.1	0.01	0.04	0.03	0.01
0.03	0.02	0.04	0.01	0.3	-0.01	0.03	0.02	0.01
0.02	0.03	0.03	0.02	0.5	0.00	0.02	0.01	0.01
0.04	0.04	0.03	0.03	0.7	0.00	0.04	0.01	0.02
0.01	0.01	0.01	0.01	0.9	0.00	0.01	0.00	0.01
0.02	0.03	0.02	0.02	1.1	0.00	0.02	-0.01	0.00
0.01	0.02	0.01	0.01	1.3	0.00	0.01	0.00	0.00
0.03	0.04	0.03	0.02	1.5	0.00	0.02	0.01	0.01
0.02	0.02	0.02	0.02	1.7	0.00	0.02	0.01	0.01
0.02	0.03	0.03	0.02	1.9	0.00	0.02	0.00	0.01
0.02	0.02	0.01	0.01	3.0	0.00	0.01	0.00	0.00
0.01	0.02	0.01	0.01	4.0	0.00	0.02	0.00	0.00
0.03	0.03	0.02	0.02	6.0	0.00	0.02	0.00	0.00
0.01	0.01	0.00	0.01	8.0	0.00	0.01	0.00	0.00
0.02	0.02	0.02	0.02	10.0	0.00	0.01	0.00	0.00
0.02	0.02	0.02	0.02	12.0	0.00	0.01	0.00	0.00
0.02	0.02	0.02	0.02	15.0	0.00	0.00	0.00	0.00
0.01	0.01	0.01	0.01	20.0	0.00	0.00	0.00	0.00
0.01	0.01	0.02	0.01	30.0	0.00	0.01	0.00	0.00
0.01	0.01	0.01	0.01	45.0	0.00	0.01	0.00	0.00
0.02	0.03	0.02	0.02	60.0	0.00	0.02	0.00	0.00

Since our values are very close to the values our simulation returned for the average number of customers in orbit (± 0.1%), the next step is to set the equations equal to each other. Doing this, we can gather some insight into how these variables affect things like

probability that a call did not enter service immediately on arrival, *Pr(delay)*, and how they relate to our outside arrival rate, λ. Setting equations (6) and (7) equal to each other then, we have:

$$\lambda_{o1} * W_o(total) = \lambda_{o2} * W_o(each) \tag{8}$$

Also, we note that the arrival rate for customers coming to orbit from outside the system, λ_{o1}, can be related to the probability of delay as:

$$\lambda_{o1} = \lambda * \Pr(delay) \tag{9}$$

Because of our definition of $W_o(each)$, we can rewrite (7) as:

$$\lambda_{o2} = L_o * retryrate \tag{10}$$

More importantly we can calculate the probability that a retrial fails to enter service, *Pr(retry fail)*, which is difficult to do from the CTMC, but much easier through Little's Law. We can rearrange equation (5) to see that

$$\Pr(retryfail) = \frac{\lambda_{o2} - \lambda_{o1}}{retryrate} \tag{11}$$

Since λ_{o2} should be equal to the overall retrial rate since the rate customers flow into orbit should be equal to the rate customers flow out of orbit.

With a few exceptions, all of the graphs presented in this section are for the system size of 9 units of traffic and 12 servers. Graphs for the larger systems are available in the Appendix and show the same trends as those for the smaller system, just on different scales.

FIGURE 1 and FIGURE 2 show the average number of calls in orbit, L_o. The first shows L_o for the exponentially distributed retrials, and the second shows L_o for the deterministic system. In all figures, the error bars are at ±1 standard error. FIGURE 3 shows the same data but for the case of Shared-Sequence Retrial Times. The case of

Personal Retrial Times had values nearly identical to the exponential case and can be found in the Appendix. Note that the scale appears very large but this was done in order to plot all of the L_o graphs on the same scale for comparison. The largest peak was about 42 customers at the lowest retrial rate in the deterministic system with 36 units of traffic and 42 servers.

From the smallest system of 12 servers to the largest system of 42 servers then there was a difference of over 20 customers in orbit at the lowest retrial rate. For the system of 20 servers, there were about 26 customers and about 34 customers in the 30 server system. Similarly, the IID and PRT systems showed nearly the same peak starting values of about 15, 20, 26, and 32 customers in orbit for the corresponding systems of 12, 20, and 30, and 42 servers. The SSRT system, however, showed values closer to the Deterministic system with values of about 17, 25, 30, and 36 customers in orbit.

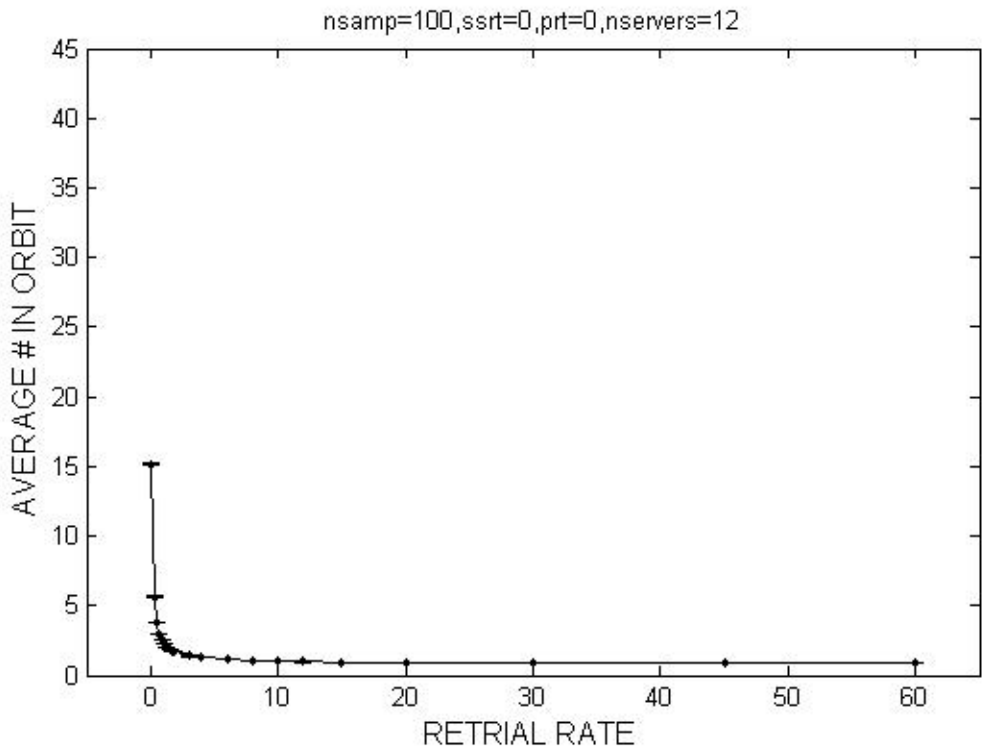


FIGURE 1. Average Number of Calls in Orbit for Exponential Retrials and 12 Servers

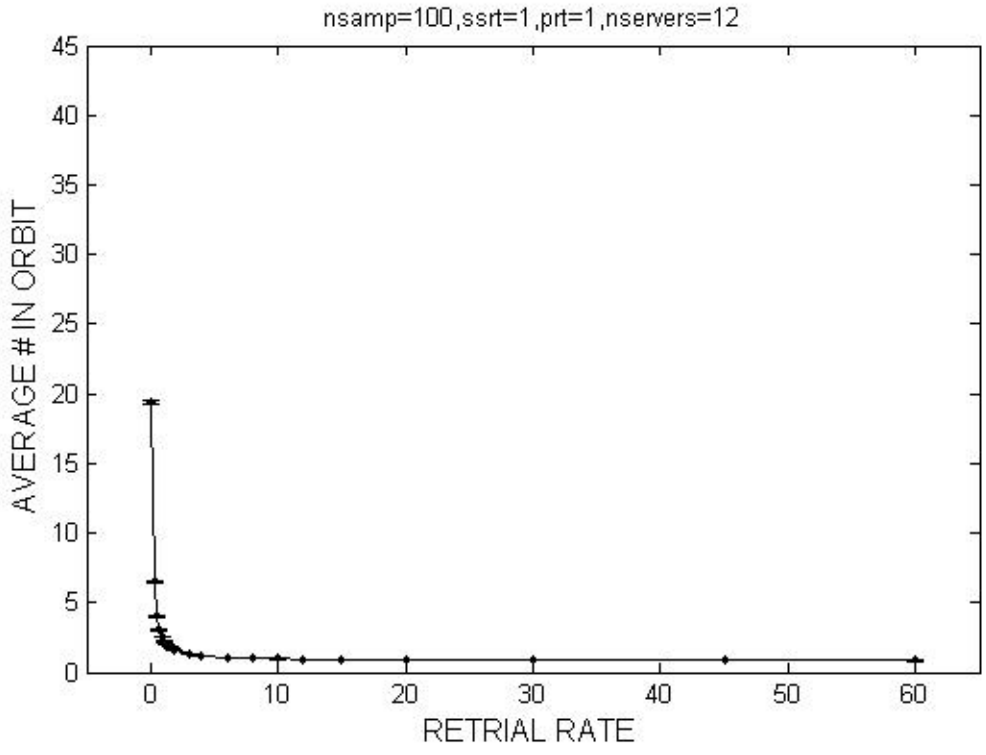


FIGURE 2. Number of Calls in Orbit for Deterministic Retrials and 12 Servers

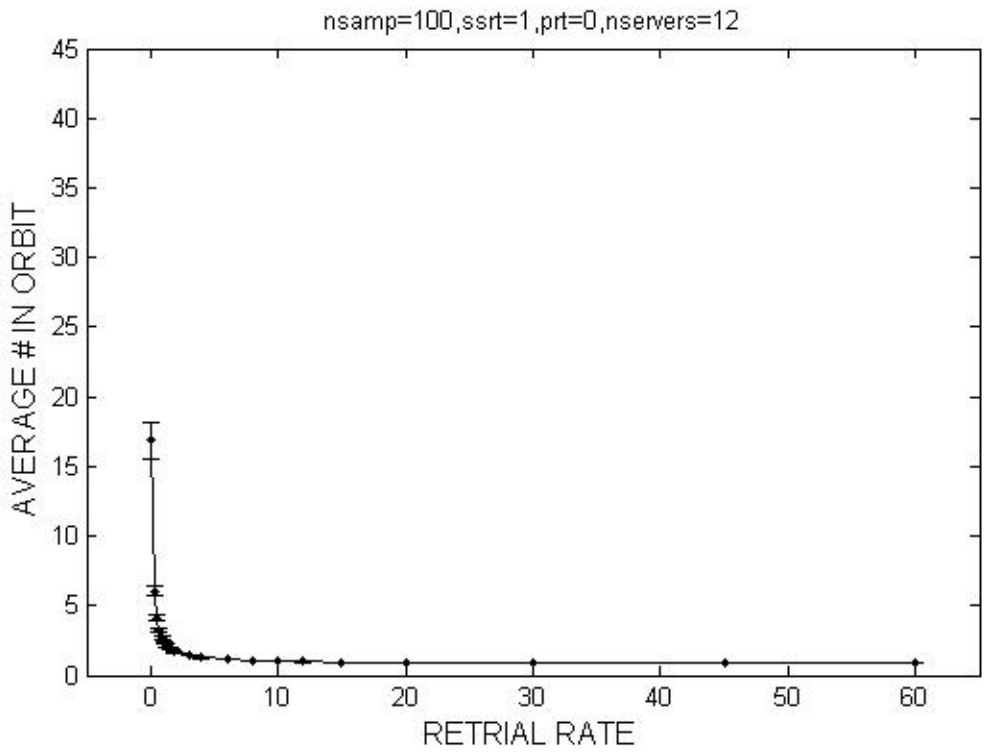


FIGURE 3. Number of Calls in Orbit for Shared Sequence Retrials and 12 Servers

While the above graphs display the average number of customers in orbit, what we are more interested in is how each case compares with the exponential retrials. The next set of graphs shows the average number of customers in orbit compared to what we would expect to see with exponentially distributed retrials. FIGURE 4, FIGURE 5, and FIGURE 6 show the percent difference from exponential for the average number of customers in orbit for the Deterministic, PRT, and SSRT systems respectively. Again, these graphs are for the system size of 9 units of traffic and 12 servers. The graphs for the remaining system sizes can be found in the Appendix.

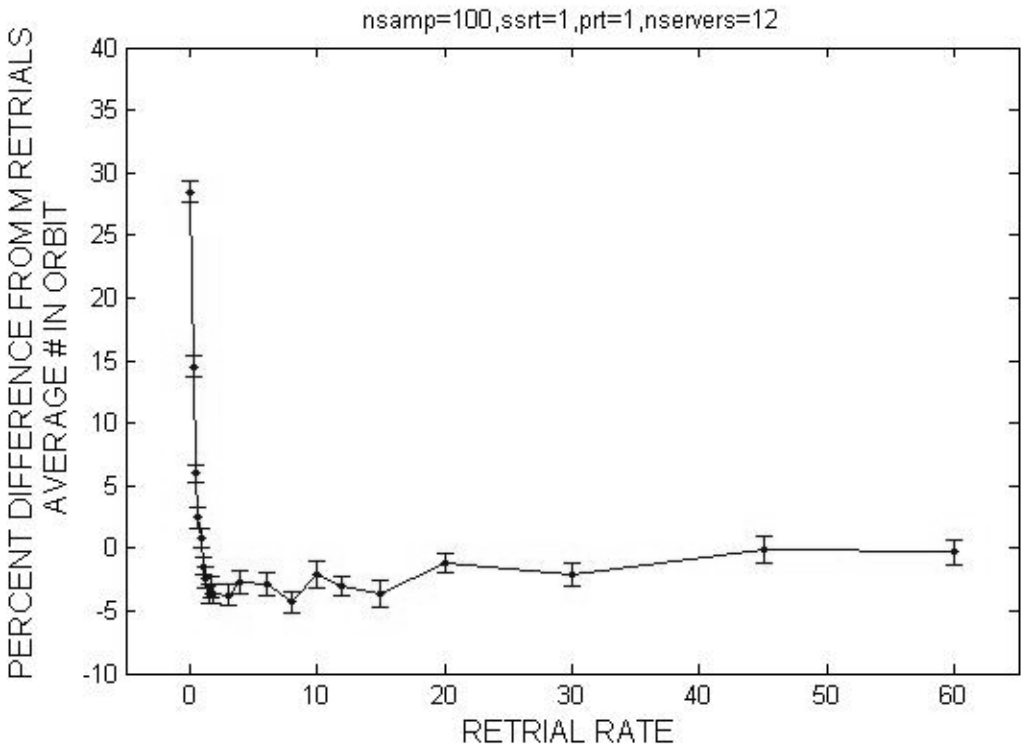


FIGURE 4. Percent Different from Exponential, Average Number of Calls in Orbit for Deterministic Retrials and 12 Servers

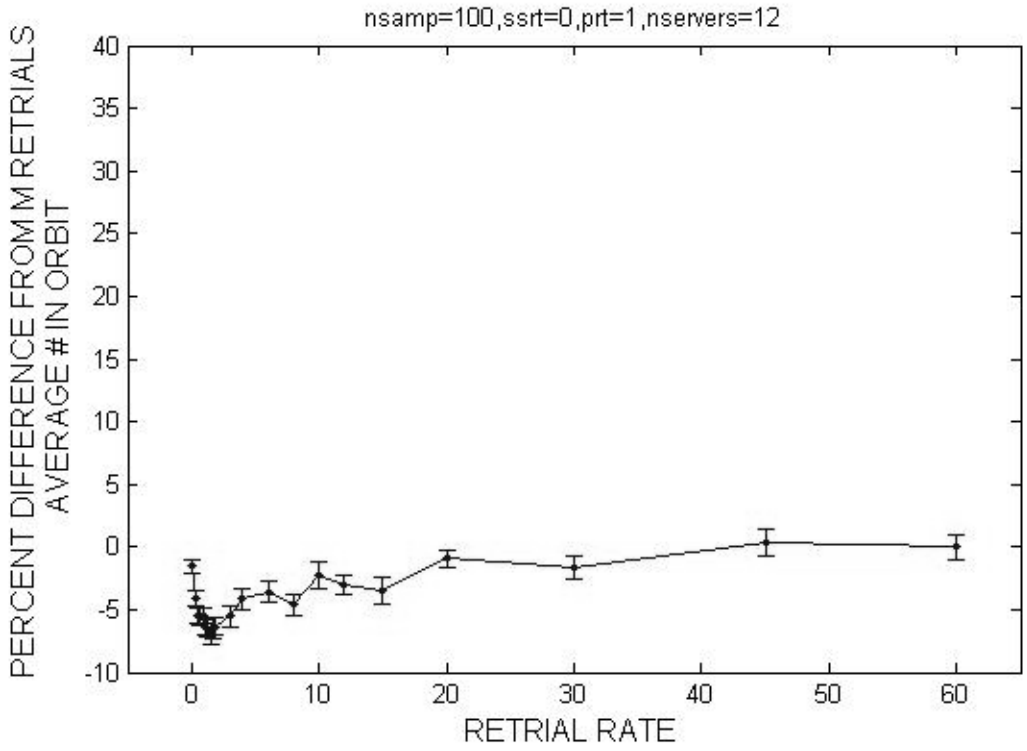


FIGURE 5. Percent Different from Exponential, Average Number of Calls in Orbit for Personal Retrial Times and 12 Servers

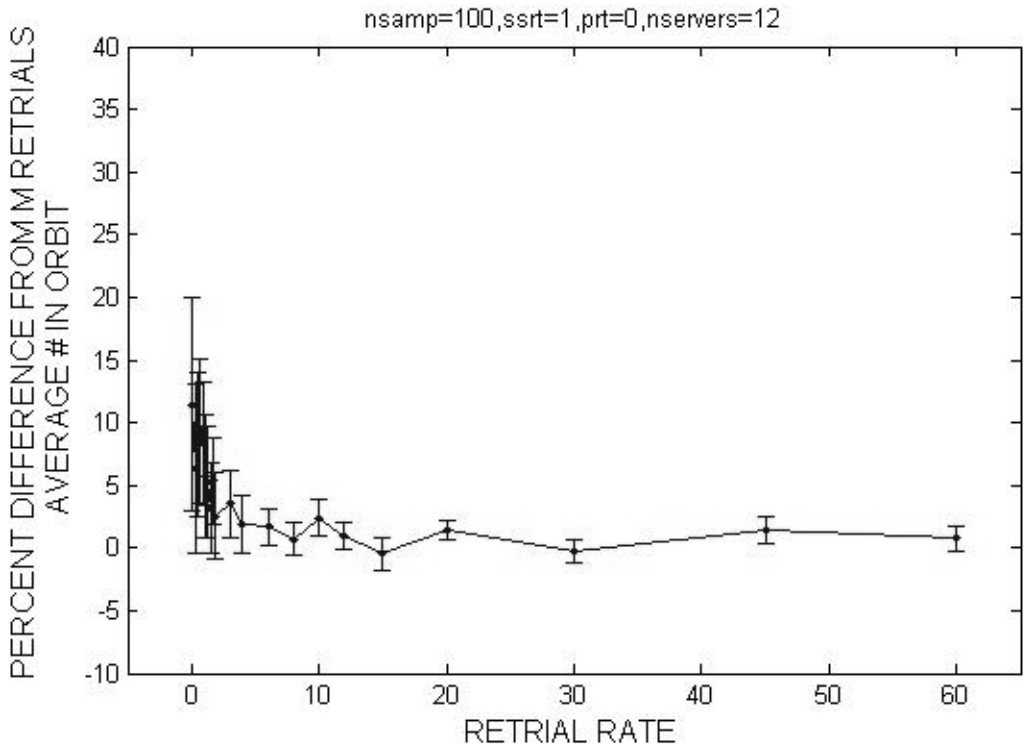


FIGURE 6. Percent Different from Exponential, Average Number of Calls in Orbit for Shared Sequence Retrial Times and 12 Servers

We notice in the above graphs that the deterministic system has far more customers in orbit than the exponential system for low retrial rates. All three graphs have roughly the same shape and eventually yield nearly the same values as the exponential system as the retrial rate increases. The main difference in each graph is for very low retrial rates. We notice that the PRT graph is fairly level, actually beginning and remaining below the exponential system until the retrial rate nears our fastest rate of one retrial every minute. Furthermore, SSRT seems to account for much more of the difference between deterministic and exponential than PRT does, but SSRT alone still does not show as large of a percent difference as deterministic does.

As the system size increases, the deterministic graph does not change much. The peak value only increases from about 27% in the 12-server system up to about 34% in the

42-server system. The PRT system also changes very little as the system size increases, the main difference being that the size of the error bars increase slightly. Oddly, in the SSRT case, the peak value increases from about 12% different in the 12 server system, doubles to about 24% in the 20 server system, then drops back down to about 17% and finally 14% as the system size increases.

The next result we examined was the probability of an outside arrival being delayed, *Pr(delay)*. In other words, a customer arrived from outside the system and all servers were full, causing the customer to bounce to orbit and retry later. FIGURE 7 shows a graph of *Pr(delay)* for all four run types of system size 9 units of traffic and 12 servers. From this, it appears that all four systems had roughly the same shape and did not vary significantly. We do note, however, that the curve for personal retrial times was in general the highest curve. When we see the individual graphs of the percent difference from exponential (FIGURE 8, FIGURE 9, and FIGURE 10) we realize that the personal retrial times give us a different shape than deterministic or shared-sequence. Again, graphs for the other system sizes can be found in the Appendix.

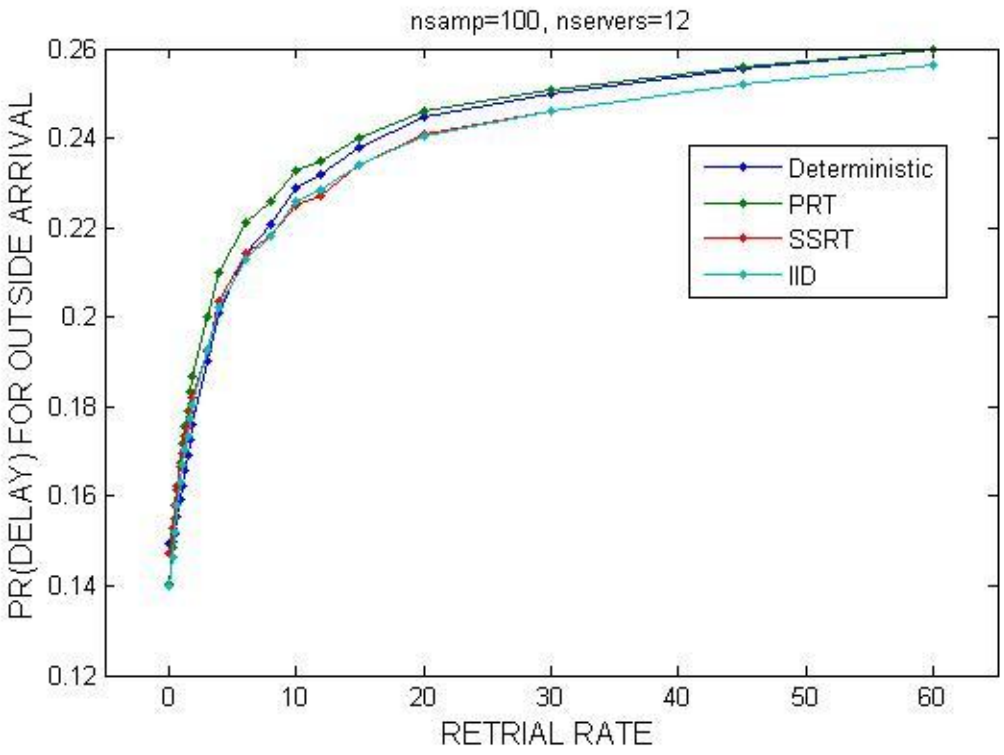


FIGURE 7. Probability of Delay of Outside Arrival for 12 Servers

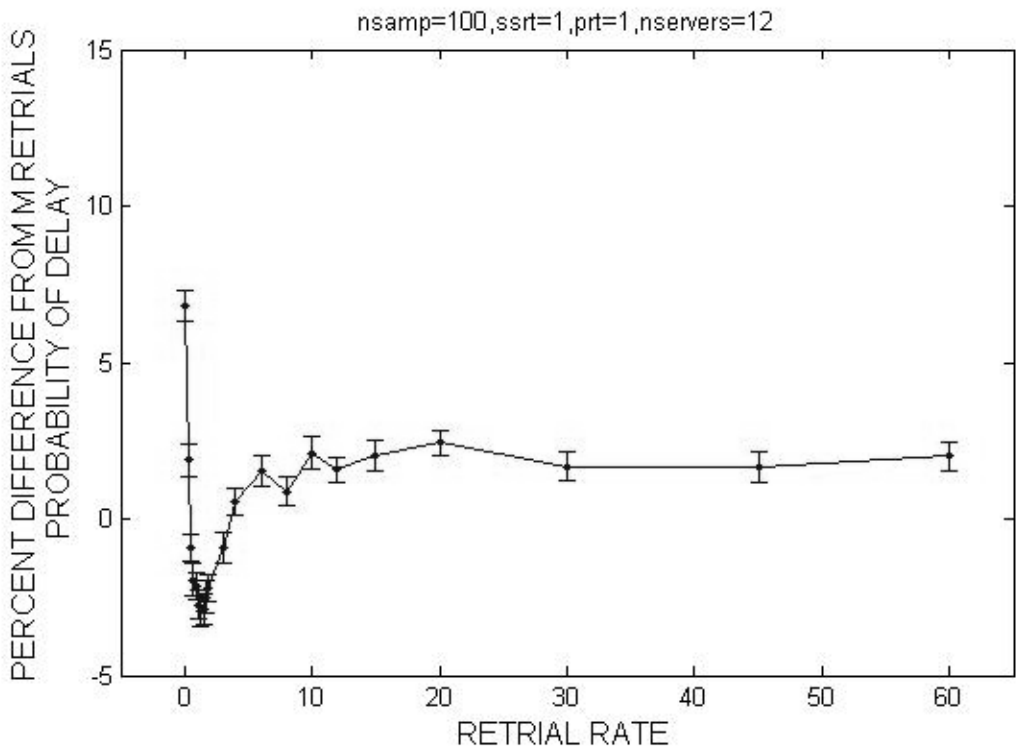


FIGURE 8. Percent Different from Exponential, Probability of Delay of an Outside Arrival, Deterministic System with 9 units of traffic and 12 servers

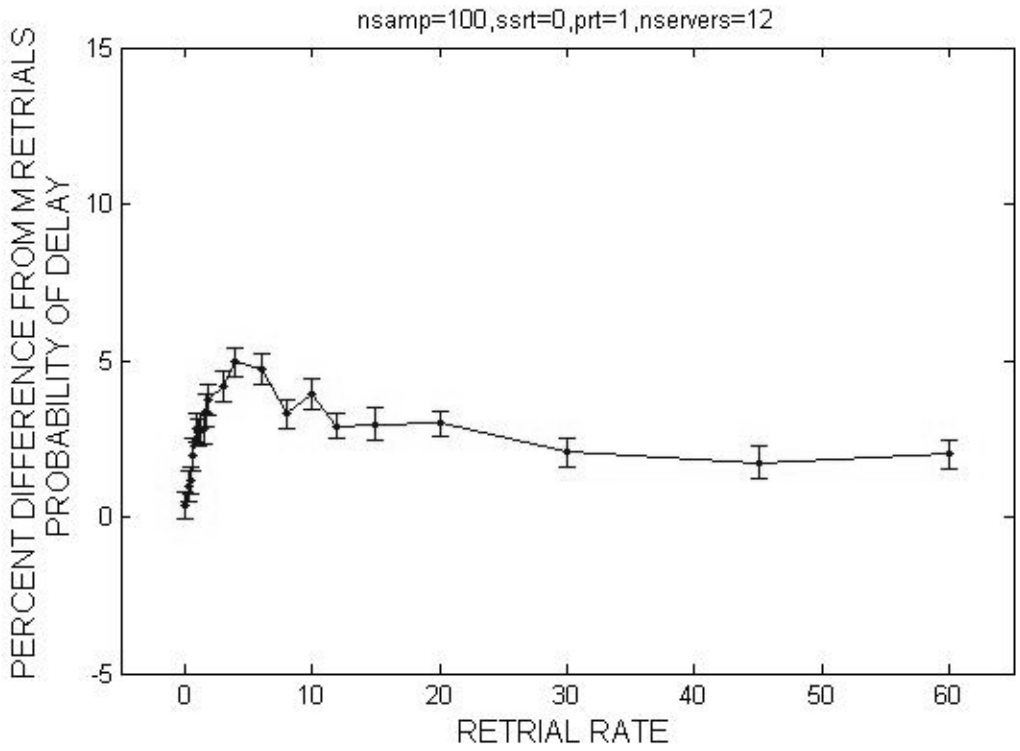


FIGURE 9. Percent Different from Exponential, Probability of Delay of an Outside Arrival, Personal Retrial Times with 9 units of traffic and 12 servers

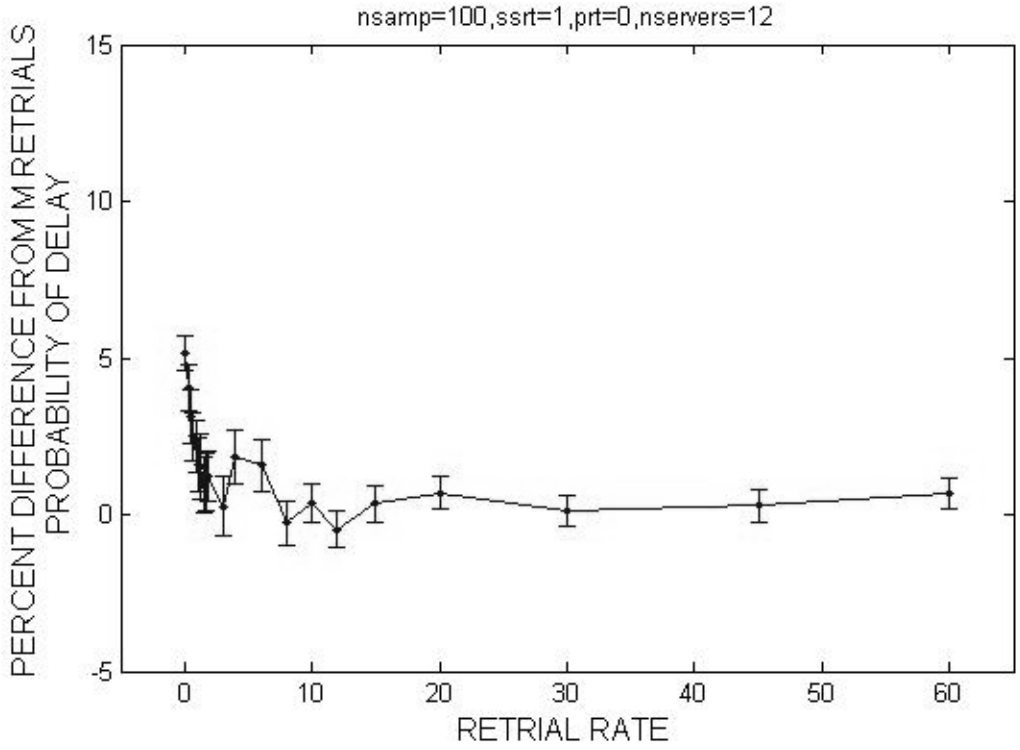


FIGURE 10. Percent Different from Exponential, Probability of Delay of an Outside Arrival, Shared-Sequence Retrial Times with 9 units of traffic and 12 servers

As the system size increased, the above graphs for Percent Difference from Exponential, Probability of Delay exhibited an interesting behavior. In the Deterministic case, they all showed the same pattern of quickly decreasing to below exponential, then increasing and eventually leveling out to be nearly the same as the exponential. However, as the system size increased, the maximum increased from about 7% to 12% and the minimum decreased from about -4% to -6% as the system grew from 12 to 42 servers, respectively. Although the PRT graphs had a slightly different shape than the others, the values showed almost no change as the system size increased. Finally, the SSRT graphs mainly increased at the beginning of the graph.

A similar measure of the system performance is the probability that a retrial fails to enter service, *Pr(retry fail)*. In other words, we want to examine the customers who have already failed to enter service on their initial arrival to the system and then look at their probability of failing to enter the system when they try again. Below, FIGURE 11 shows clearly that in the PRT case, *Pr(retry fail)* increases much more quickly than the either the SSRT or Deterministic case, which appear to closely follow exponential. This is probably because fast retrials in PRT can cause one or two calls to retry very quickly and repeatedly, thus causing many of their retrials to fail.

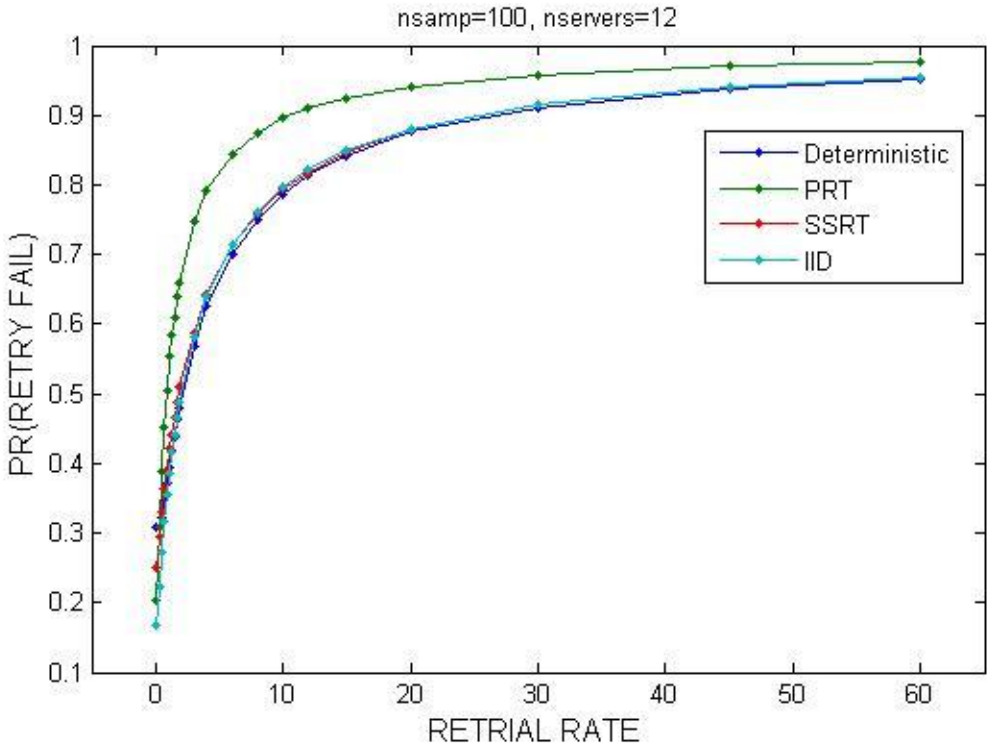


FIGURE 11. Probability That a Retrial Fails to Enter Service, 12 Servers

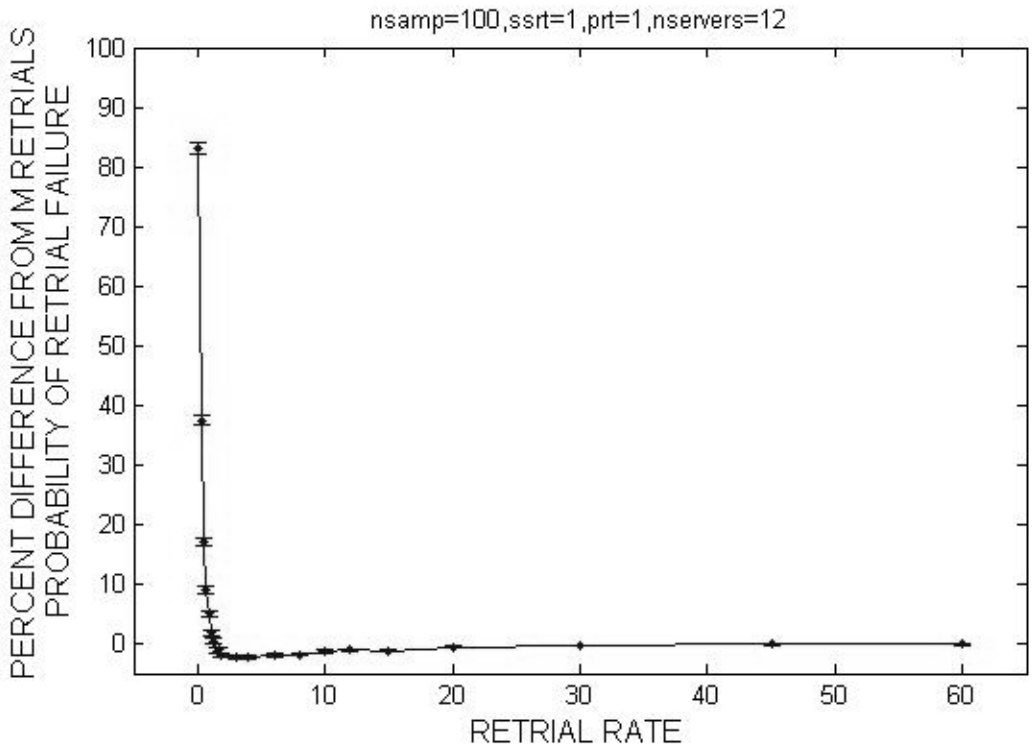


FIGURE 12. Percent Different from Exponential, Probability of Retrial Failing to Enter Service, Deterministic Retrial Times with 9 units of traffic and 12 servers

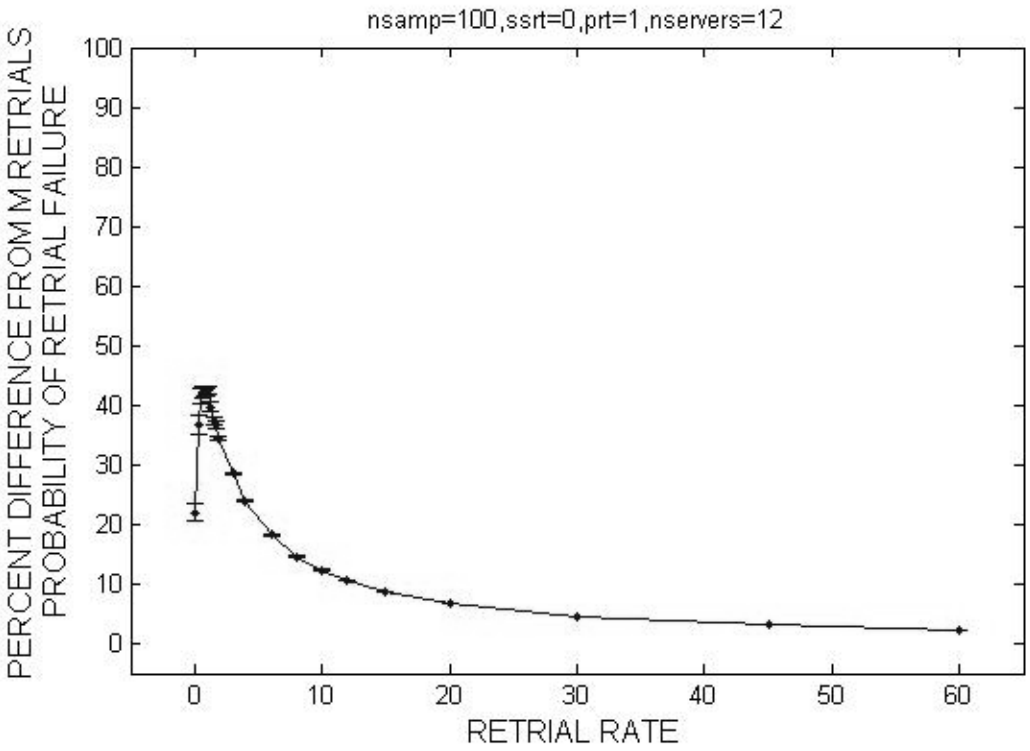


FIGURE 13. Percent Different from Exponential, Probability of Retrial Failing to Enter Service, Personal Retrial Times with 9 units of traffic and 12 servers

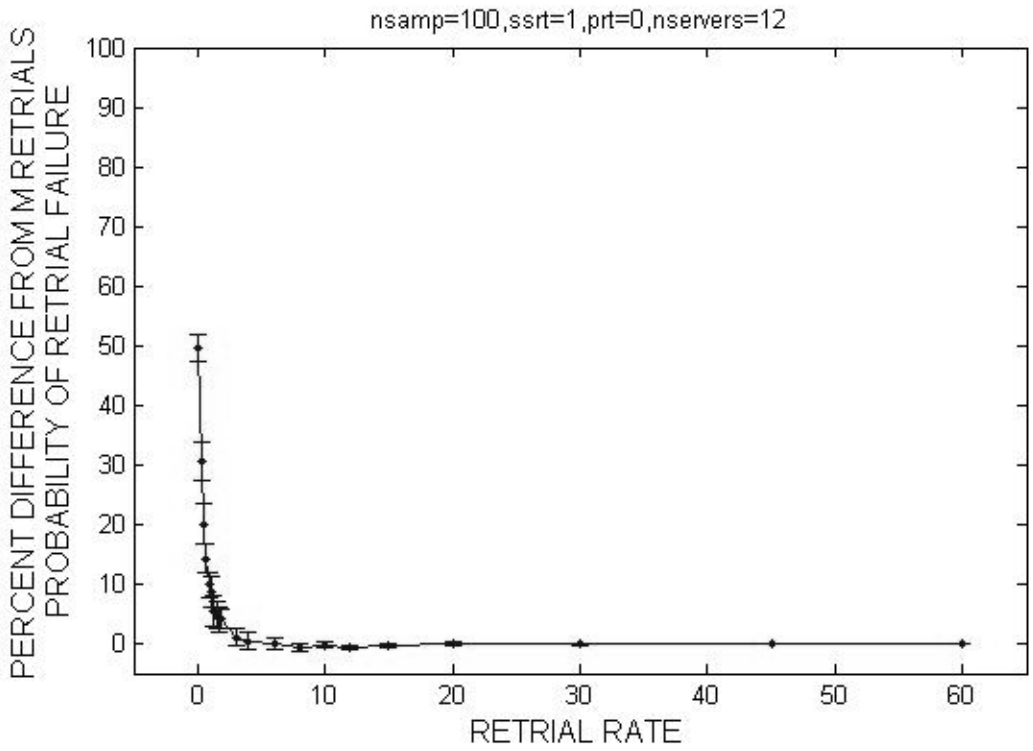


FIGURE 14. Percent Different from Exponential, Probability of Retrial Failing to Enter Service, Shared-Sequence Retrial Times with 9 units of traffic and 12 servers

As the system size increased, the above graphs for the Deterministic and SSRT systems exhibited roughly the same pattern as many of the earlier graphs, in that the maximum increased steadily while the graph retained its shape. In the Deterministic case, the maximum increased from about 80% more than Exponential for 12 servers to about 145% more than exponential for 42 servers. In the SSRT case, the maximum increased from about 50% to 100% for the 12- and 42-server systems respectively. As noted, the PRT has a completely different shape than Deterministic and SSRT and also exhibits the opposite trend as the system size increases. The maximum value decreases from about 40% above exponential in the 12-server system to about 30% in the 42-server system. It's also of interest that the peak is not at the lowest retrial rate, but instead anywhere from a retrial rate of 0.7 to 1.3.

An additional output measure we examined was the ratio of the probability that a retrial customer fails to enter service, *Pr(retry fail)*, to the probability that an outside arrival fails to enter service, or *Pr(delay)*. This ratio gives us a measure of the extent to which retrials see the time-average delay probability. An important fundamental approximation for multi-server retrial queueing systems is retrials see time averages, or RTA. This stems from the better known Poisson arrivals see time averages (PASTA) property. In short, "PASTA says that if the arrival process is Poisson, then the steady state probability that an arriving customer sees the system in state *i* is the same as the limiting probability that the system is in state *i"*.[4] RTA assumes then that the proportion of customers returning from orbit who find *i* customers in service is equal to the time-average probability of *i* customers in service.[4] It is important to note that the RTA assumption is meant only for slow retrial rates, but few papers in the literature have analyzed how slow the retrial rate must be for RTA to be fairly accurate. Figure 16 shows that the ratio of *Pr(retry fail)/Pr(delay)* is fairly large (above 2) for retrial rates as slow as 1/10 even with exponential retrials, and the ratio is even larger for our PRT, SSRT, and deterministic-retrial systems.

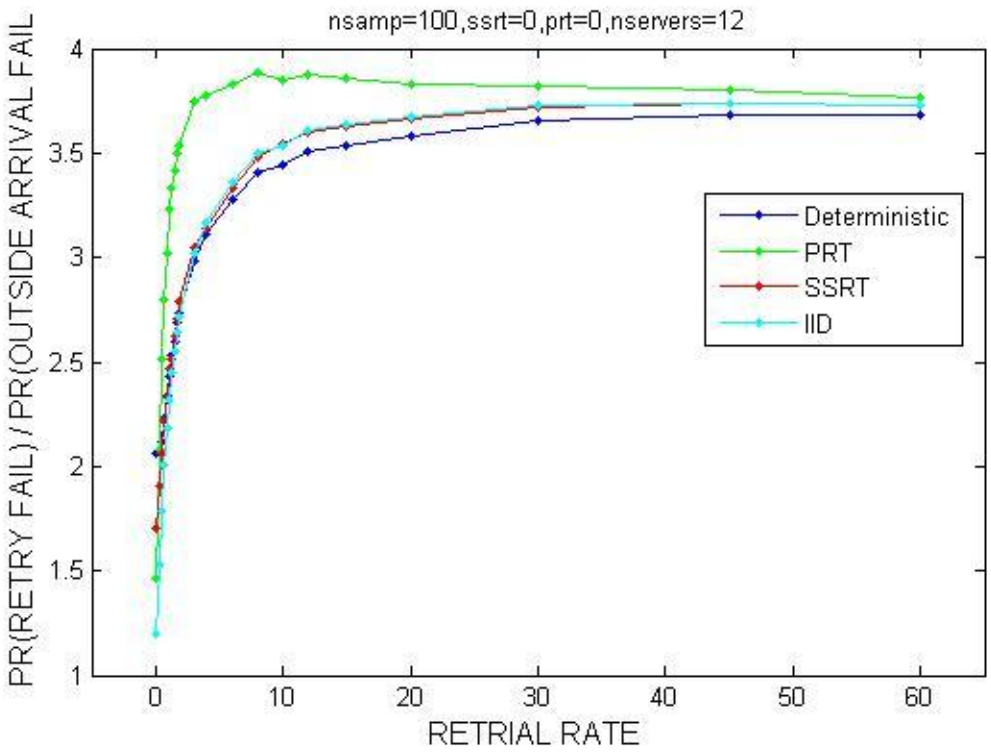


FIGURE 15. Ratio of *Pr(retry fail)* to *Pr(delay)*, 9 units of traffic and 12 Servers

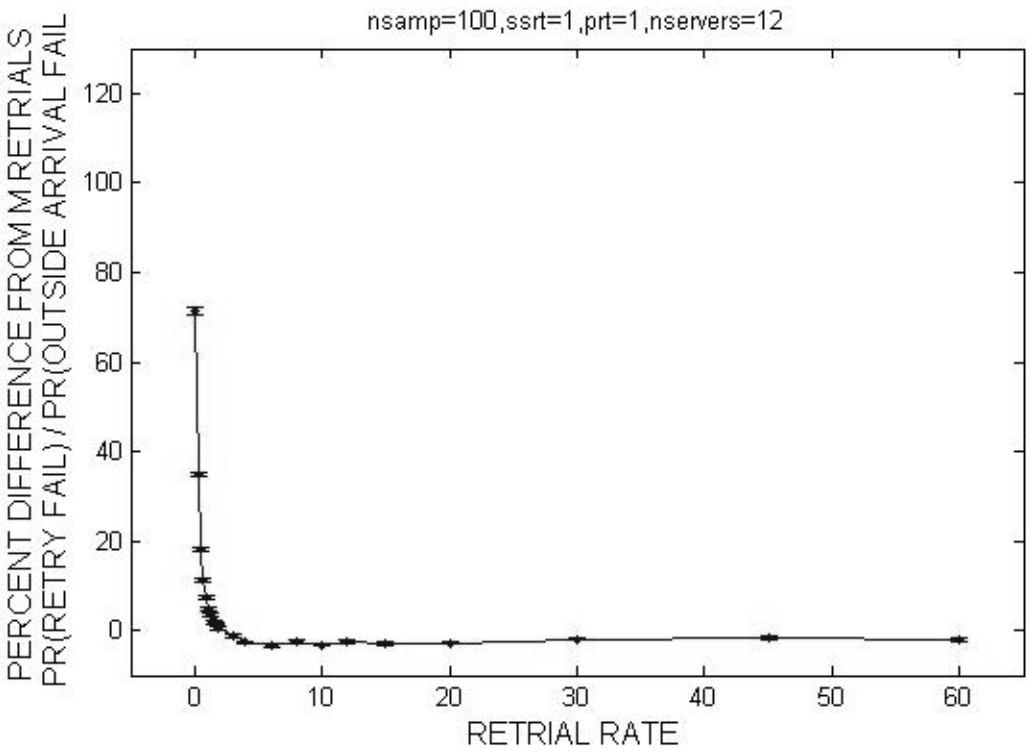


FIGURE 16. Percent Different from Exponential for ratio of *Pr(retry fail)* to *Pr(delay)*, Deterministic Retrial Times, 9 units of traffic and 12 Servers

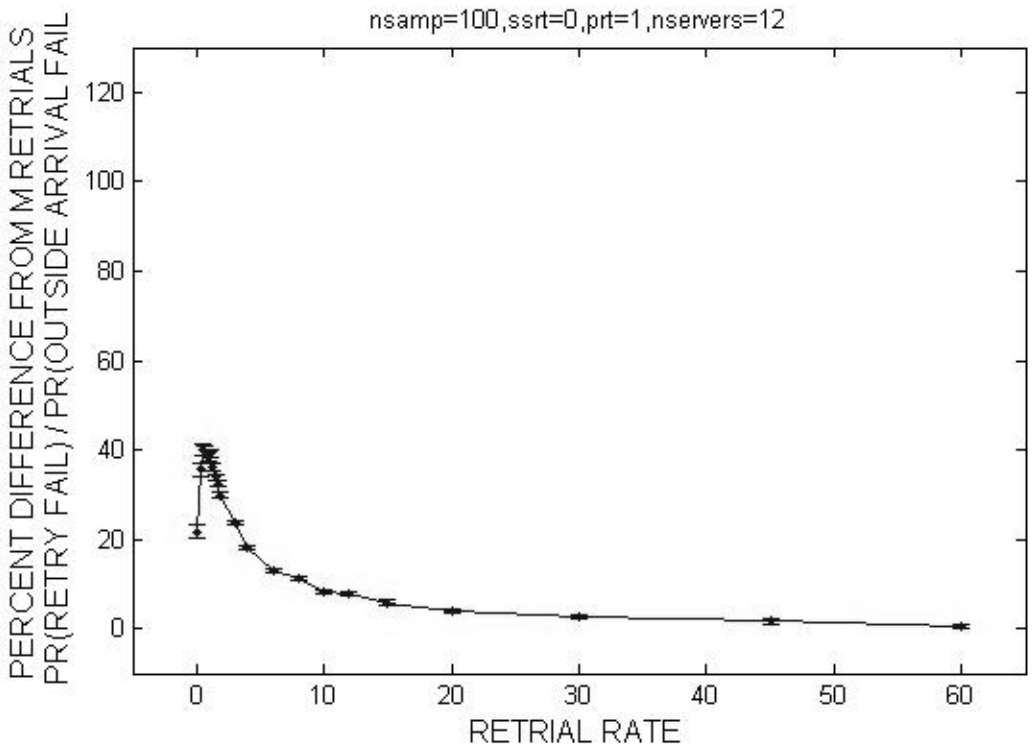


FIGURE 17. Percent Different from Exponential for ratio of *Pr(retry fail)* to *Pr(delay)*, Personal Retrial Times, 9 units of traffic and 12 Servers

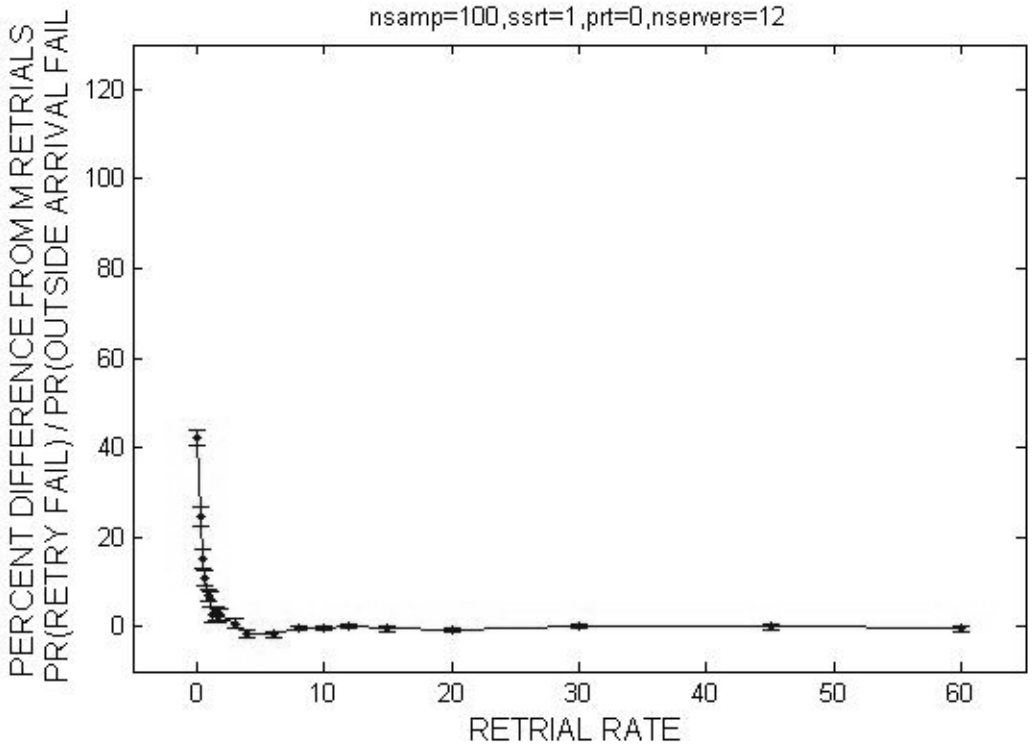


FIGURE 18. Percent Different from Exponential for ratio of *Pr(retry fail)* to *Pr(delay)*, Shared-Sequence Retrial Times, 9 units of traffic and 12 Servers

Again, the above graphs are for the smallest system of 12 servers. As the system size increased, the percent difference, in most cases, became even worse. As can be seen in FIGURE 16, the peak for the Deterministic graph is about 70% different from Exponential for the system of 12 servers. However, for the 42 server system, this peak stretches up to about 120% different and does not drop below the 20% mark until the retrial rate hits 1.5 retries per hour. The SSRT systems followed the same pattern as the Deterministic but on a smaller scale. The peak went from about 40% in the 12-server system up to about 80% in the 42-server system.

A different pattern can be seen with the PRT graphs, though. FIGURE 15 and FIGURE 17 show that the Personal Retrial Times ratios follow a different pattern than the others. The trend as the system size increases is also different than the others. The peak percent difference actually decreases from about 40% to about 26% as the system size increases from 12 to 42 servers. Also, the peak is not at the lowest retrial rate like the Deterministic and SSRT graphs. Instead, for the PRT graphs, the peak is at 0.5 retries per hour in the 12-server system and 1.3 retries per hour in the 42-server system.

Conclusions

We have shown that low-variance retrial times have **a substantial effect** on the performance of multi-server retrial queues **at low retrial rates**, and they should not be approximated away using exponential retrial times. The **effect gets worse as the systems grow larger**, in our data set. The SSRT property seems responsible for more of the observed differences than the PRT property, but SSRT alone still does not account for the total size of the differences.

Limitations/Delimitations of the Study

The major limitations of our research stem from the nature of doing simulations. That is, a simulation does not provide the convenient mathematical equations that standard analysis of queueing systems provides. The queueing community tends to favor results in the form of mathematical equations that can be analyzed, explained, and manipulated to apply to other systems. Another disadvantage is the length of the simulations. Due to limitations in computing power, our initial simulations lasted from 3 hours to as much as 8 hours in length. The length increases with the size of the system, and additional simulations could last even longer as we vary the parameters and increase the system size even further. Similarly, we would like to experiment with even larger systems, to see if the growth in the percent error levels off.

References

[1] Sheldon M. Ross, Introduction to Probability Models, edited by Anonymous 9th ed. (Academic Press, 2007), pp. 782.

[2] Mario Lefebvre, Applied stochastic processes, edited by Anonymous (Springer, New York, 2007), pp. 382.

[3] L. Kosten, Stochastic theory of service systems, edited by Anonymous (Pergamon Press, Oxford, 1973), pp. 168.

[4] J. R. Artalejo and A. Gómez-Corral, Retrial Queueing Systems, A Computational Approach, edited by Anonymous (Springer, 2008), pp. 318.

[5] Falin, G. I. and Templeton, J. G. C., Retrial Queues, edited by Anonymous (Chapman and Hall, London, 1997).

[6] J. R. Artalejo, "Analysis of an M/G/1 queue with constant repeated attempts and server vacations," Comput.Oper.Res. 24 (6), 493-504 (1997).

[7] J. R. Artalejo and M. J. Lopez-Herrero, "A simulation study of a discrete-time multiserver retrial queue with finite population," Journal of Statistical Planning and Inference 137 (8), 2536-2542 (2007).

[8] J. R. Artalejo, "Accessible bibliography on retrial queues," Math.Comput.Model. 30 (3-4), 1-6 (1999).

[9] F. P. Kelly, "On Auto-Repeat Facilities and Telephone Network Performance," Journal of the Royal Statistical Society.Series B (Methodological) 48 (1), 123-132 (1986).

[10] Carl M. Harris, Karla L. Hoffman and Patsy B. Saunders, "Modeling the IRS Telephone Taxpayer Information System," Oper.Res. 35 (4), 504-523 (1987).

[11] David J. Houck and Wai Sum Lai, "Traffic modeling and analysis of hybrid fiber–coax systems," Comput.Networks ISDN Syst. 30 (8), 821-834 (1998).

[12] M. Ajmone Marsan *et al.*, "Efficient Estimation of Call Blocking Probabilities in Cellular Mobile Telephony Networks with Customer Retrials," IEEE J.Sel.Areas in Commun 19, 332-346 (2001).

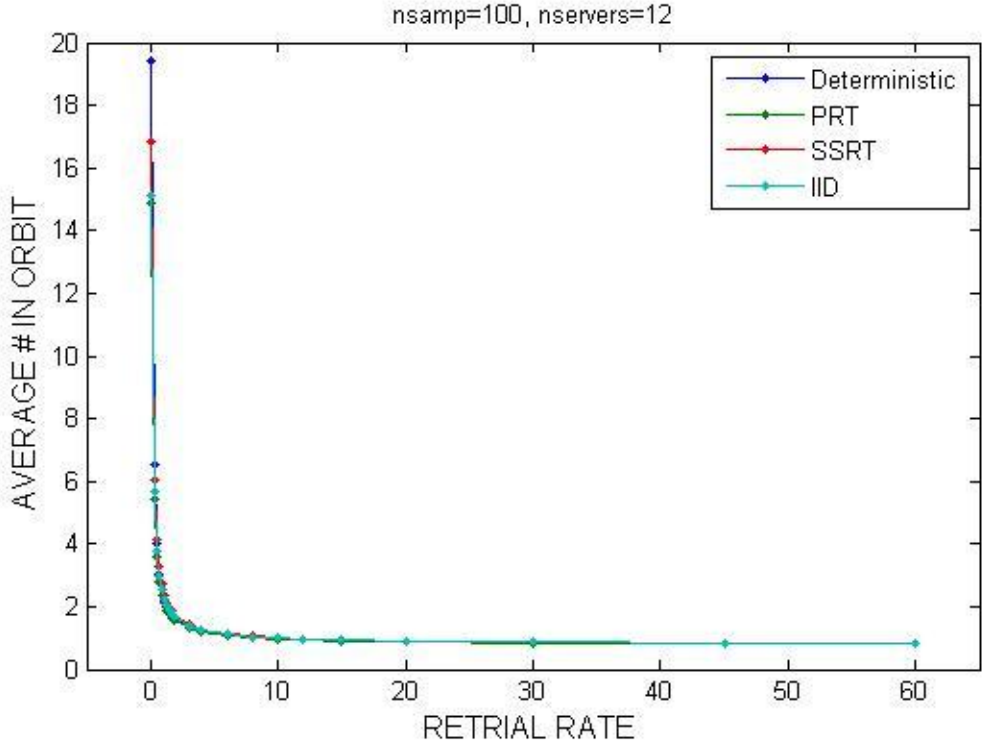
[13] S. R. Chakravarthy and A. Dudin, "Analysis of a retrial queuing model with MAP arrivals and two types of customers," Math.Comput.Model. 37 (3-4), 343-363 (2003).

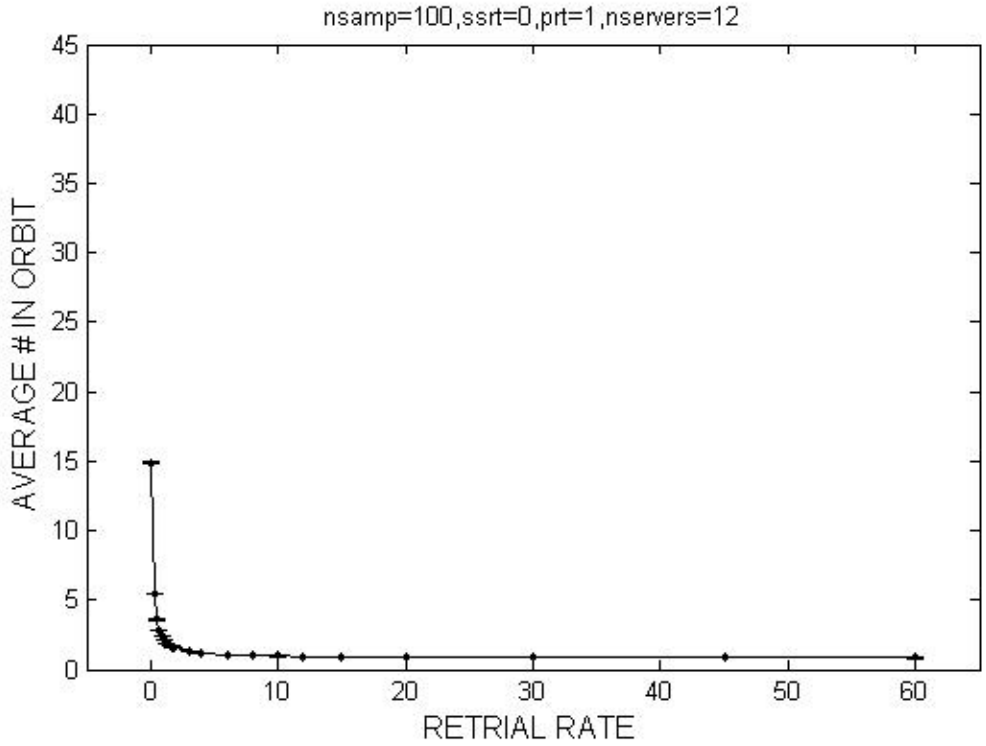
[14] D. C. Gilbert, "Modeling spin locks with queuing networks," SIGOPS Oper.Syst.Rev. 12 (1), 29-42 (1978).

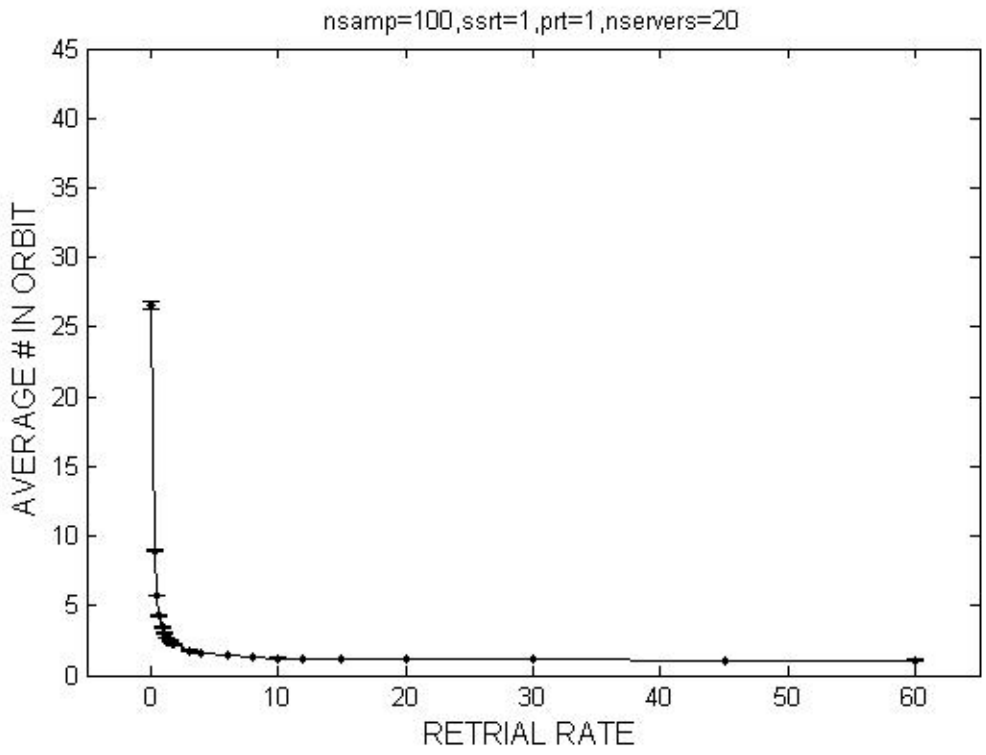
[15] Shlomo Halfin and Ward Whitt, "Heavy-Traffic Limits for Queues with Many Exponential Servers," Oper.Res. 29 (3), 567-588 (1981).

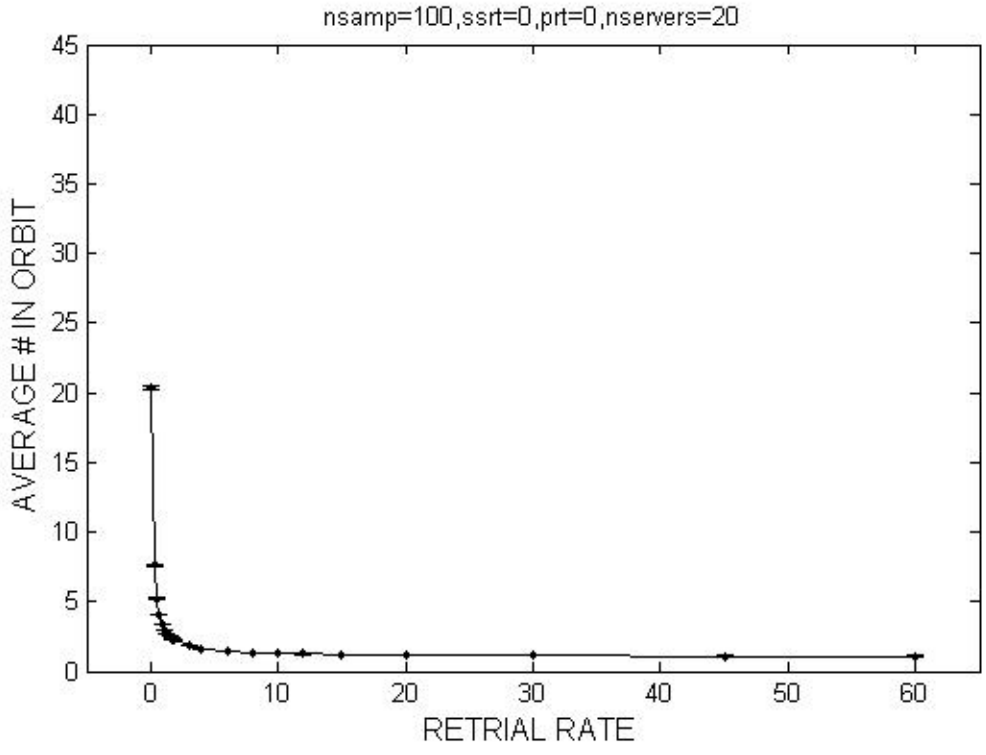
[16] Peter J. Kolesar and Linda V. Green, "Insights On Service System Design From a Normal Approximation To Erlang's Delay Formula," Production and Operations Management 7 (3), 282-293 (1998).

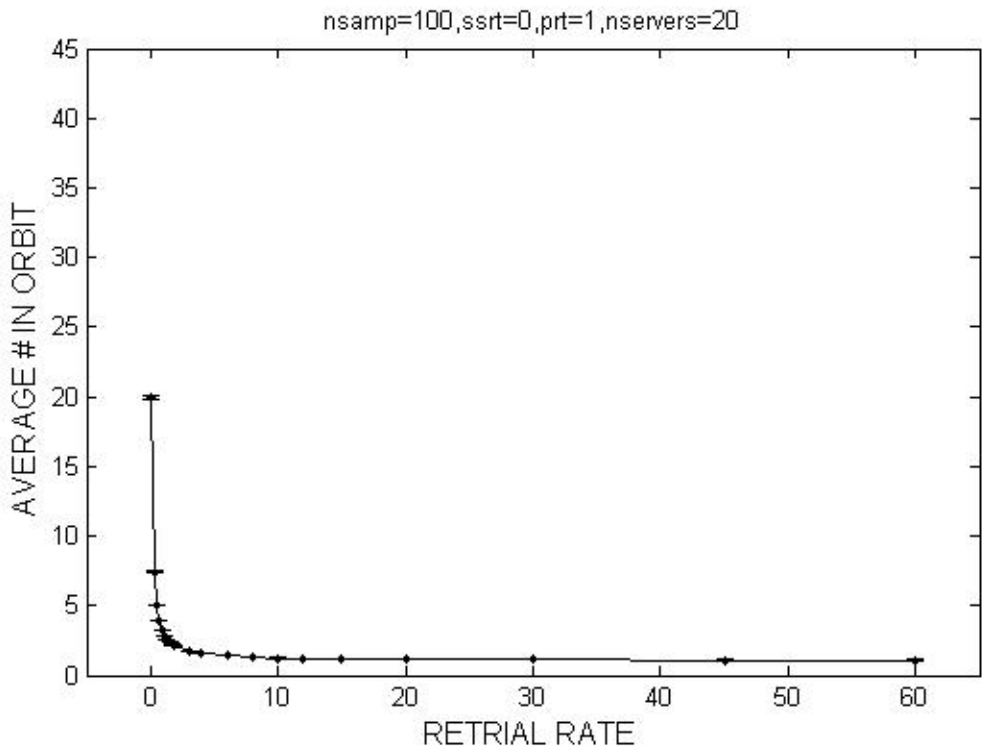
Appendix A: Additional Graphs

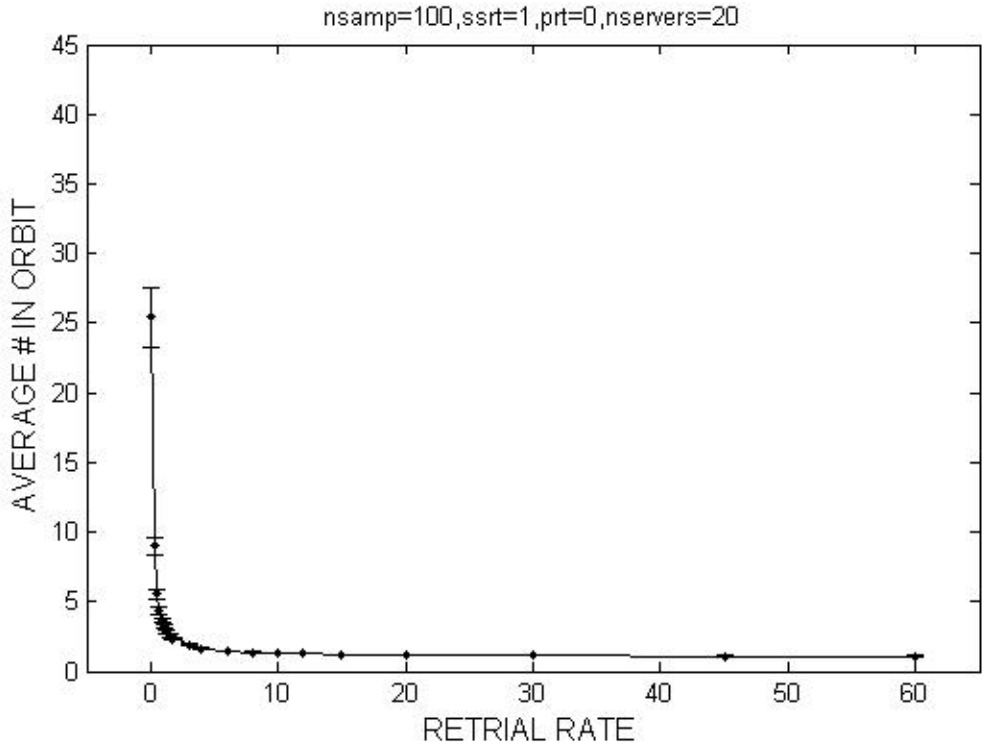


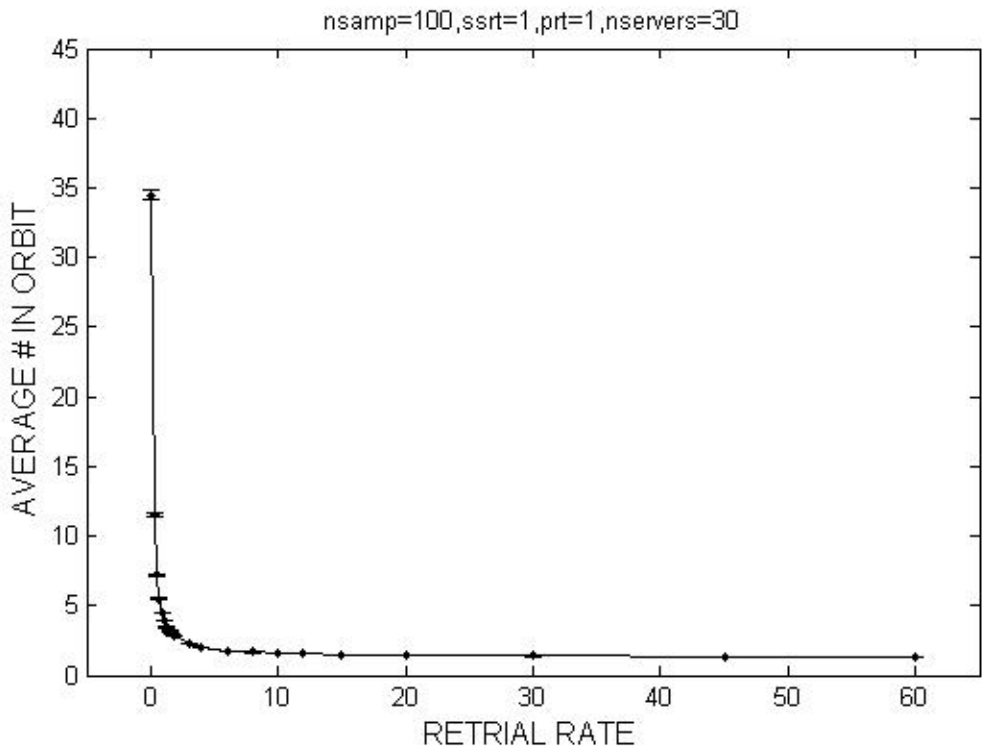


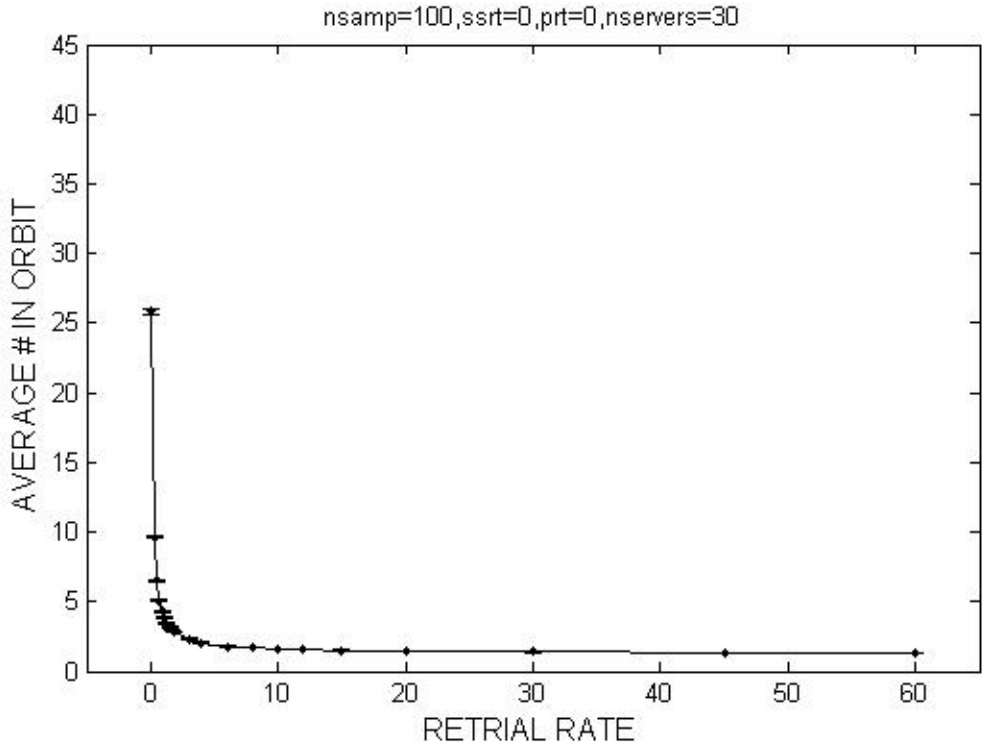

nsamp=100,ssrt=1,prt=1,nservers=20
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

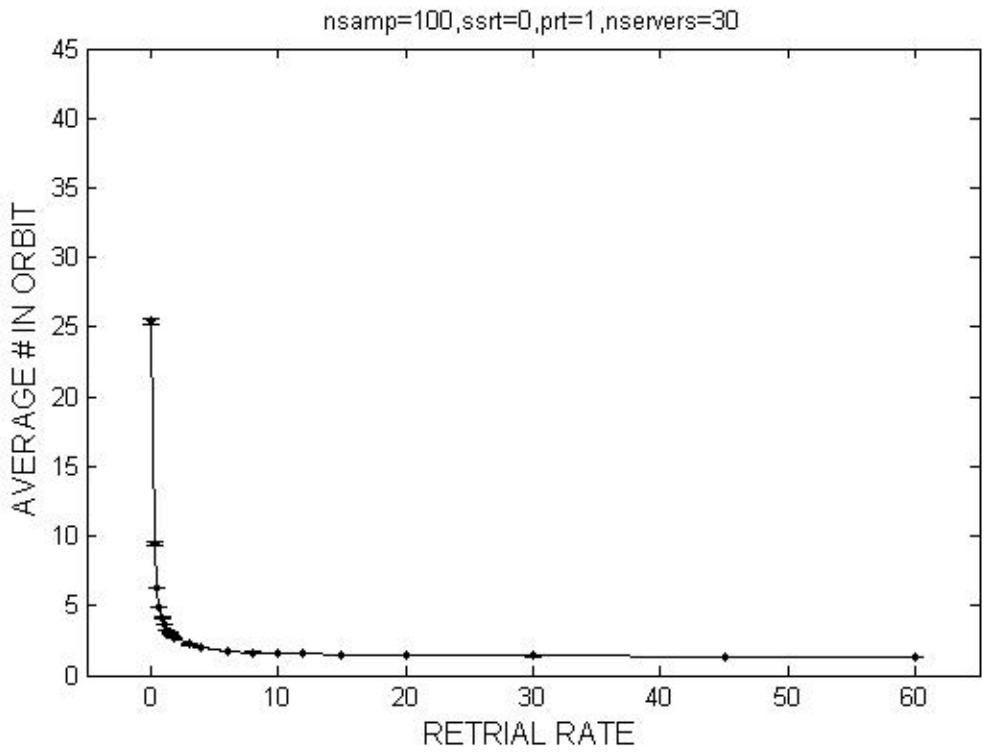

nsamp=100,ssrt=0,prt=0,nservers=20
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

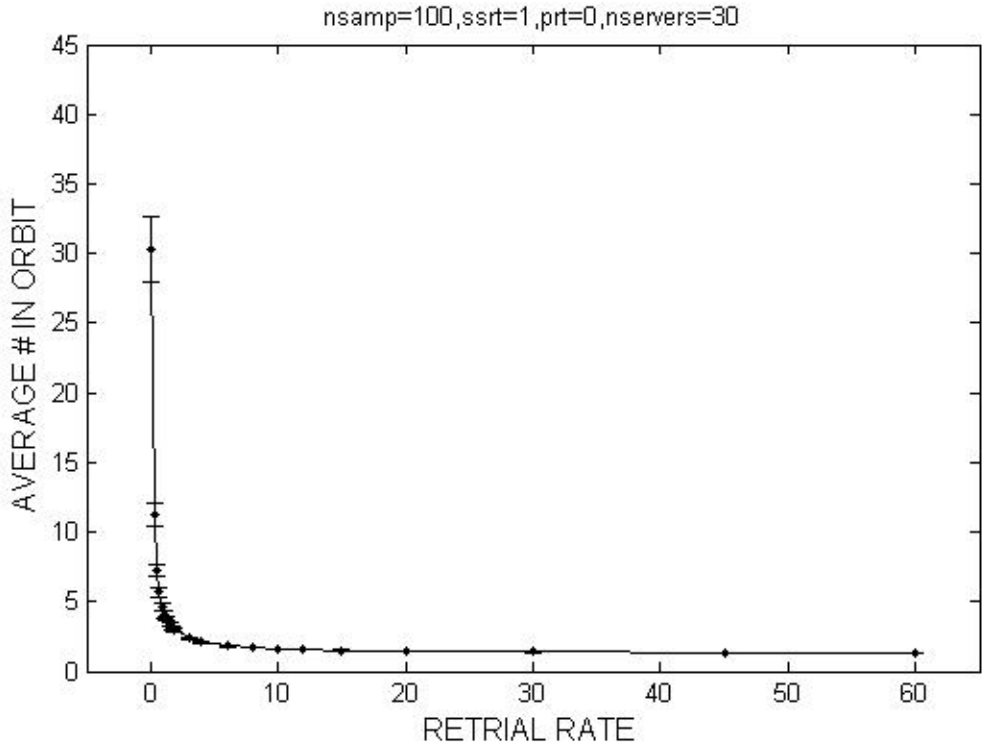

nsamp=100,ssrt=0,prt=1,nservers=20
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

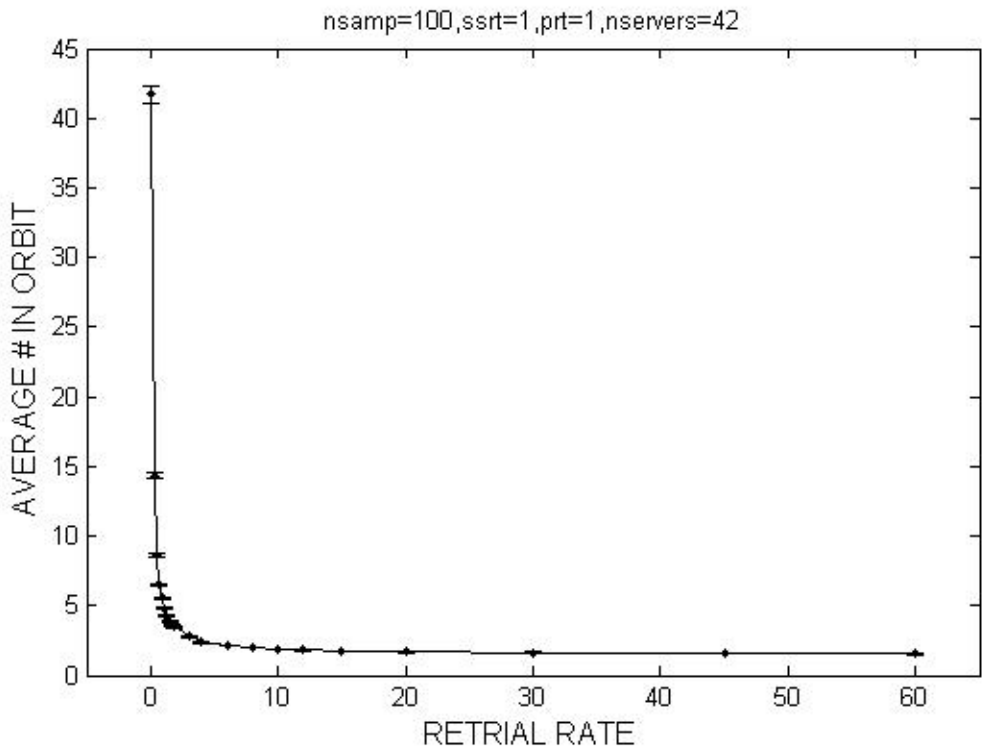

nsamp=100,ssrt=1,prt=0,nservers=20
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

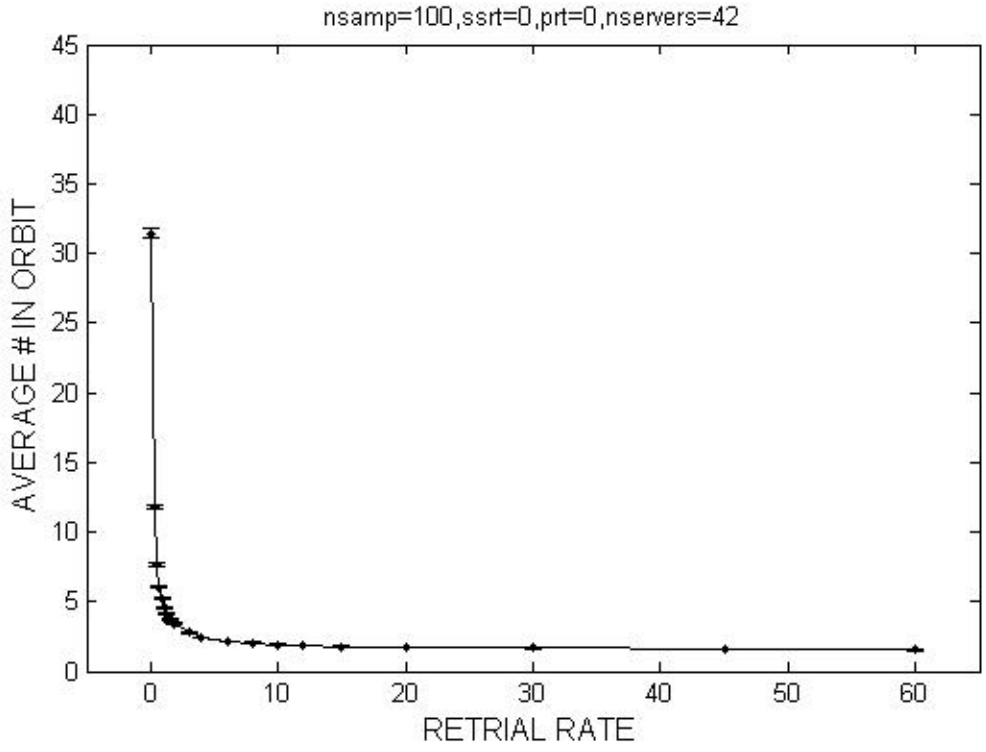

nsamp=100,ssrt=1,prt=1,nservers=30
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

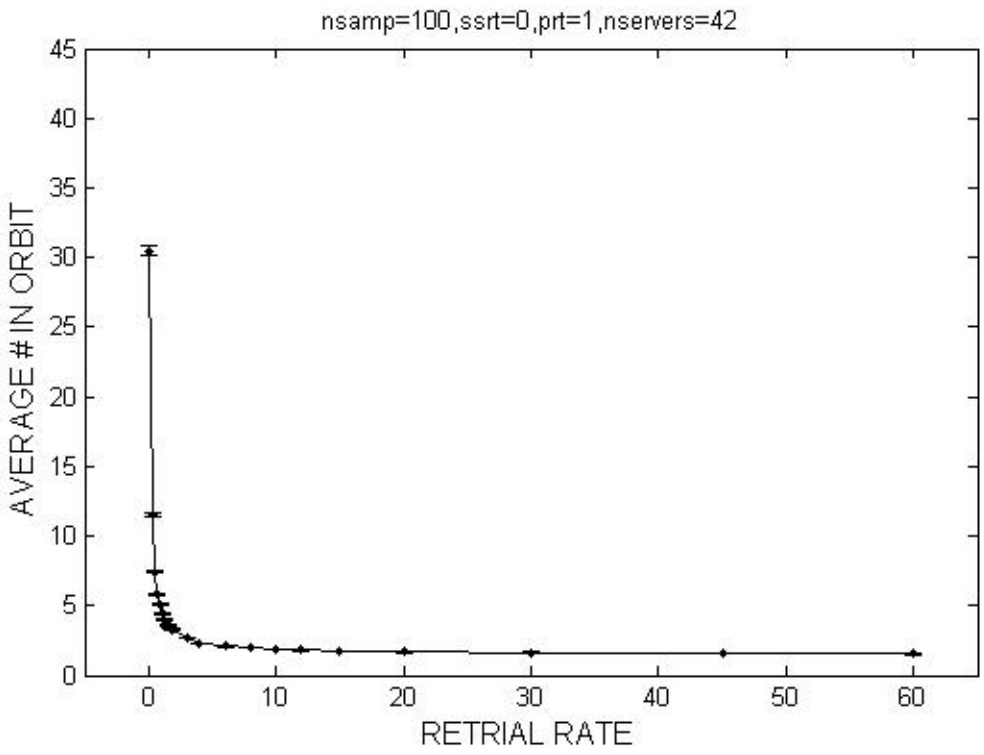

nsamp=100,ssrt=0,prt=0,nservers=30
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

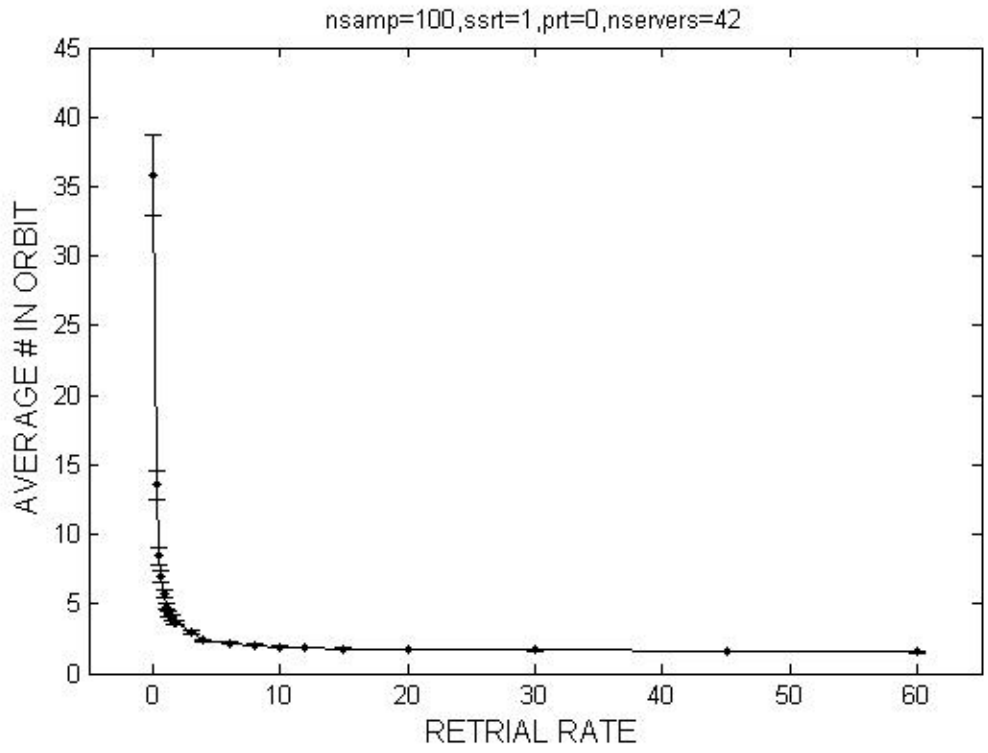

nsamp=100,ssrt=0,prt=1,nservers=30
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

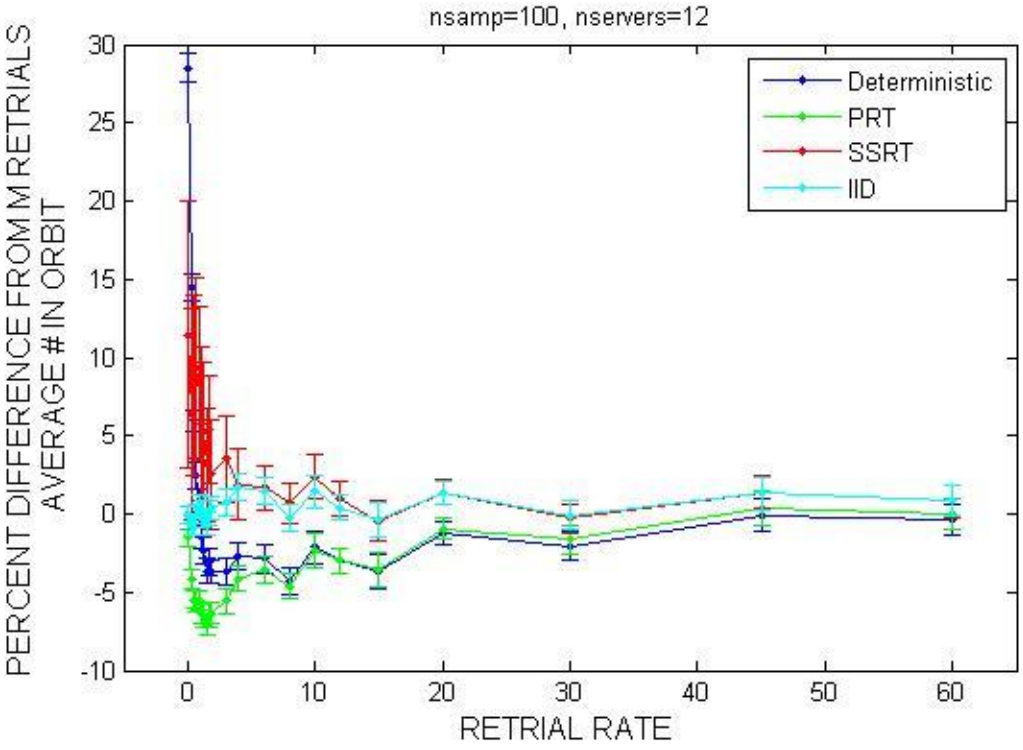

nsamp=100,ssrt=1,prt=0,nservers=30
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

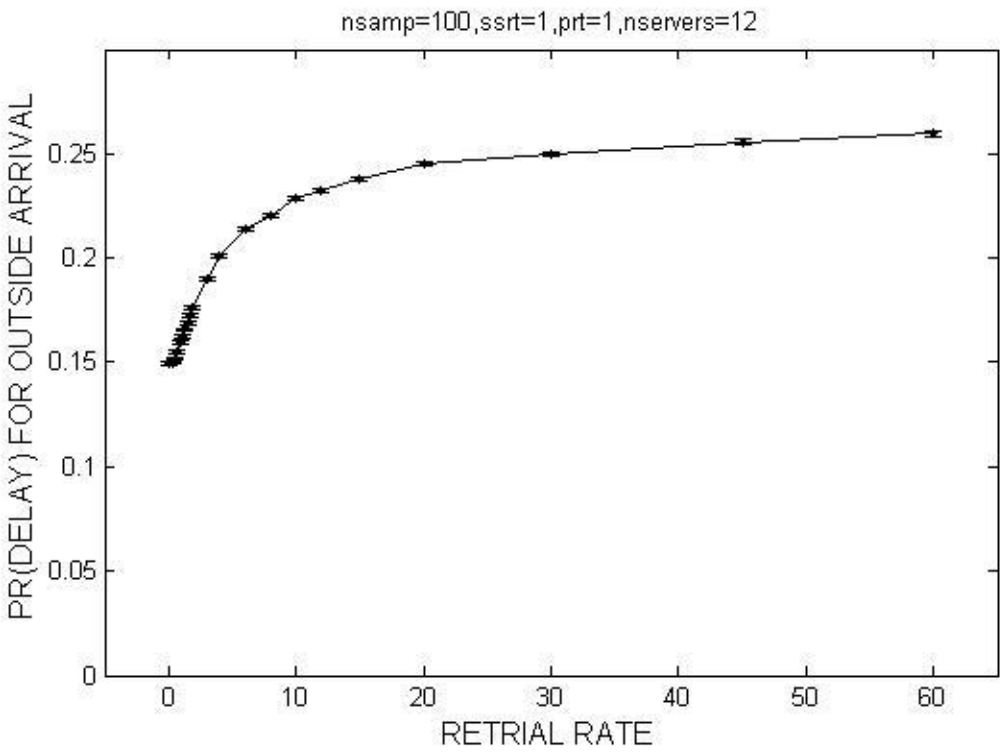

nsamp=100,ssrt=1,prt=1,nservers=42
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

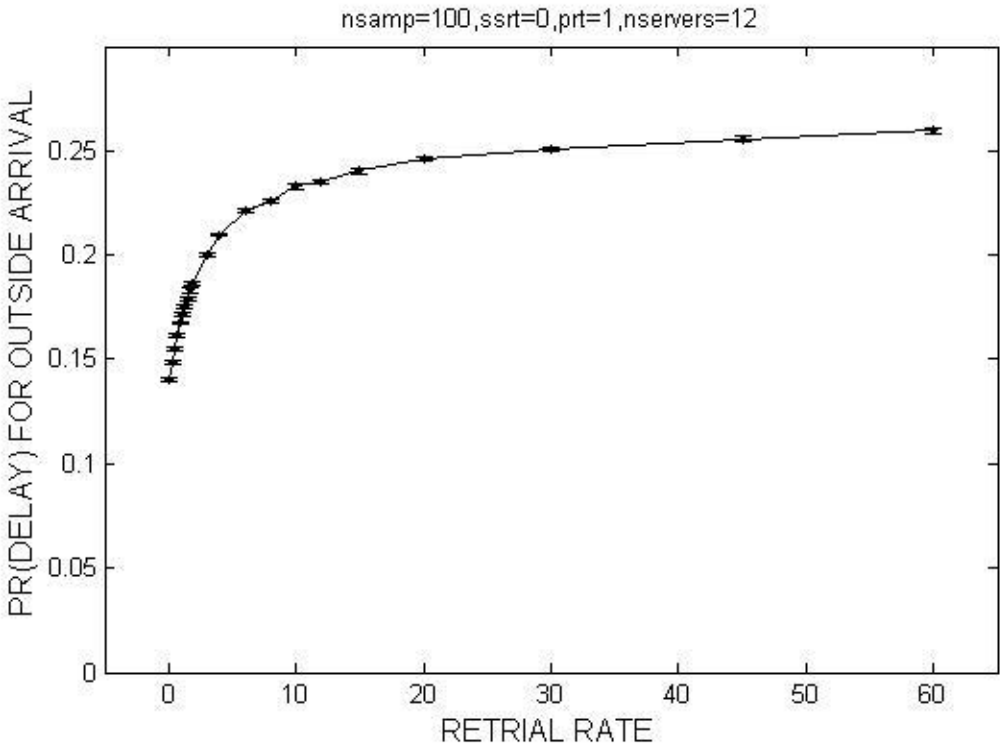

nsamp=100,ssrt=0,prt=0,nservers=42
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

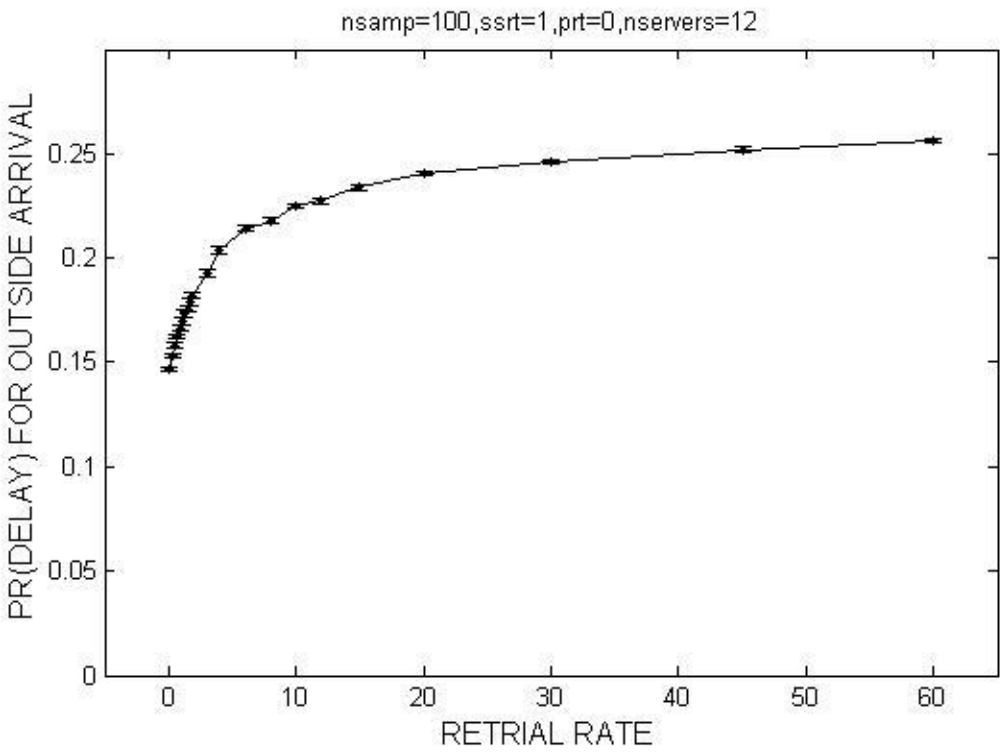

nsamp=100,ssrt=0,prt=1,nservers=42
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

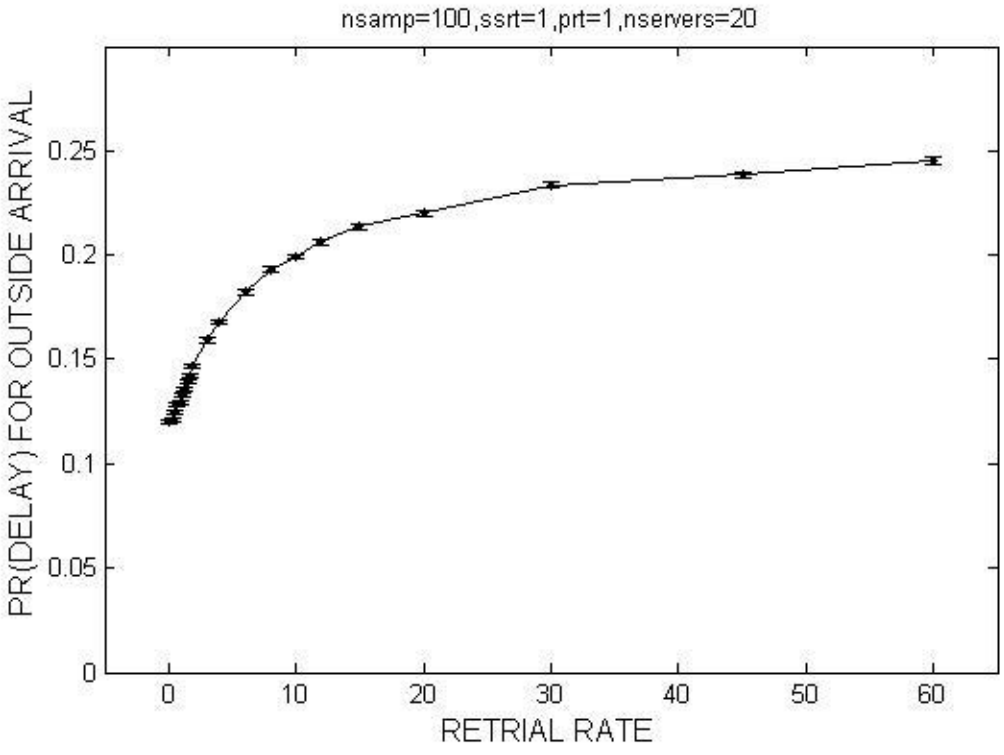

nsamp=100,ssrt=1,prt=0,nservers=42
AVERAGE # IN ORBIT
RETRIAL RATE
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60

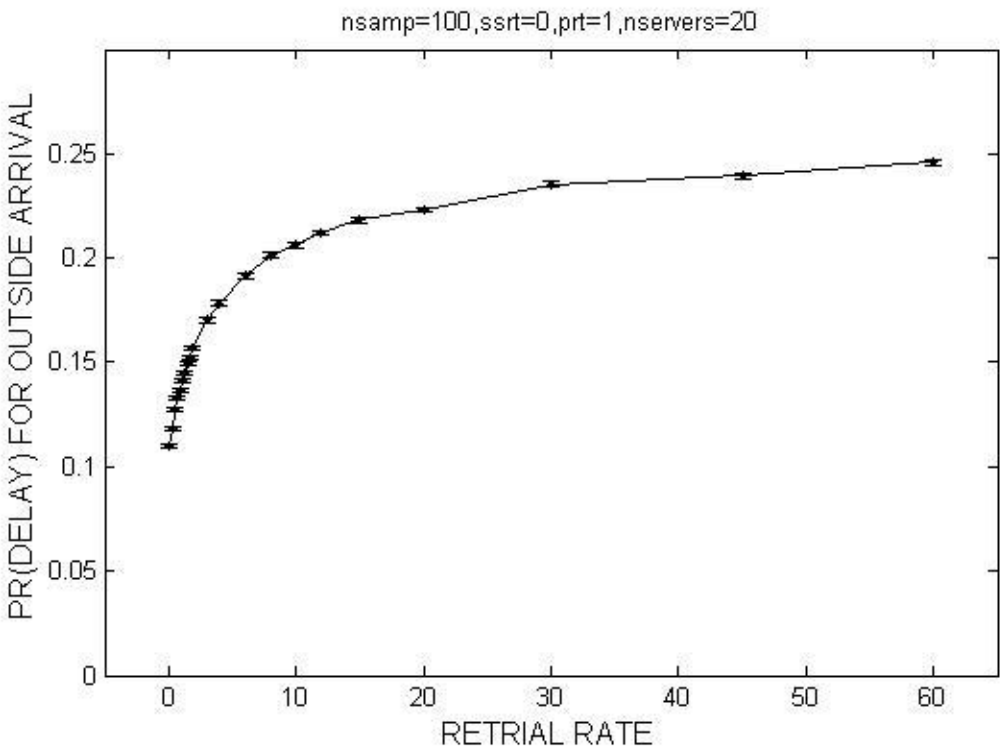

nsamp=100, nservers=12
PERCENT DIFFERENCE FROM M RETRIALS AVERAGE # IN ORBIT
RETRIAL RATE
Deterministic
PRT
SSRT
IID
-10
-5
0
5
10
15
20
25
30
0
10
20
30
40
50
60

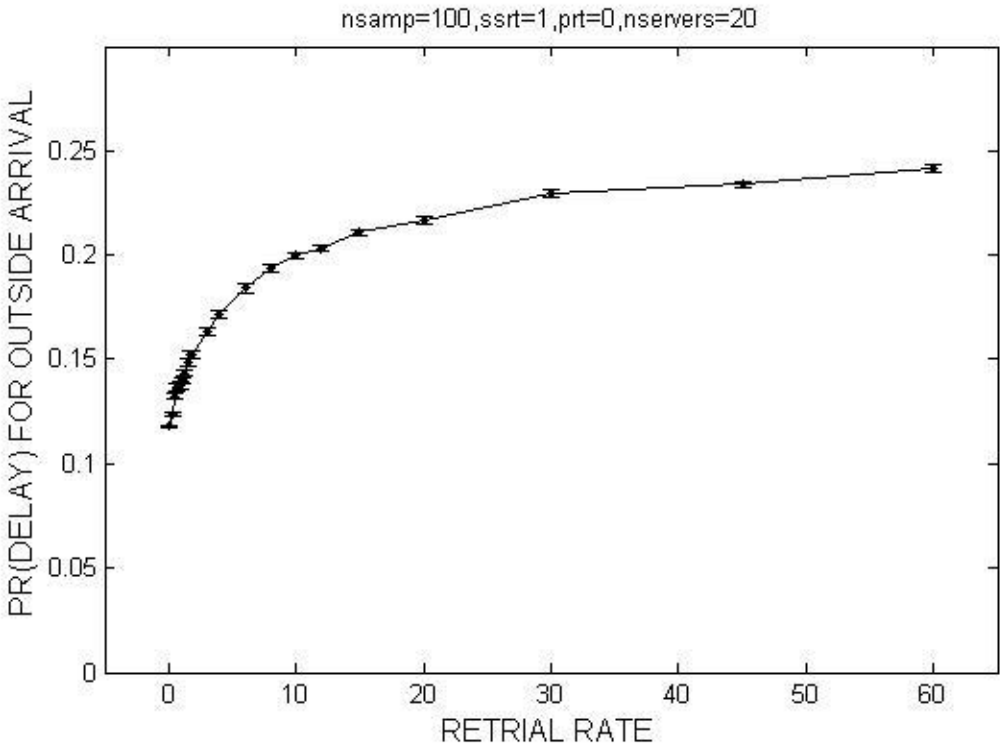

nsamp=100,ssrt=1,prt=1,nservers=12
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

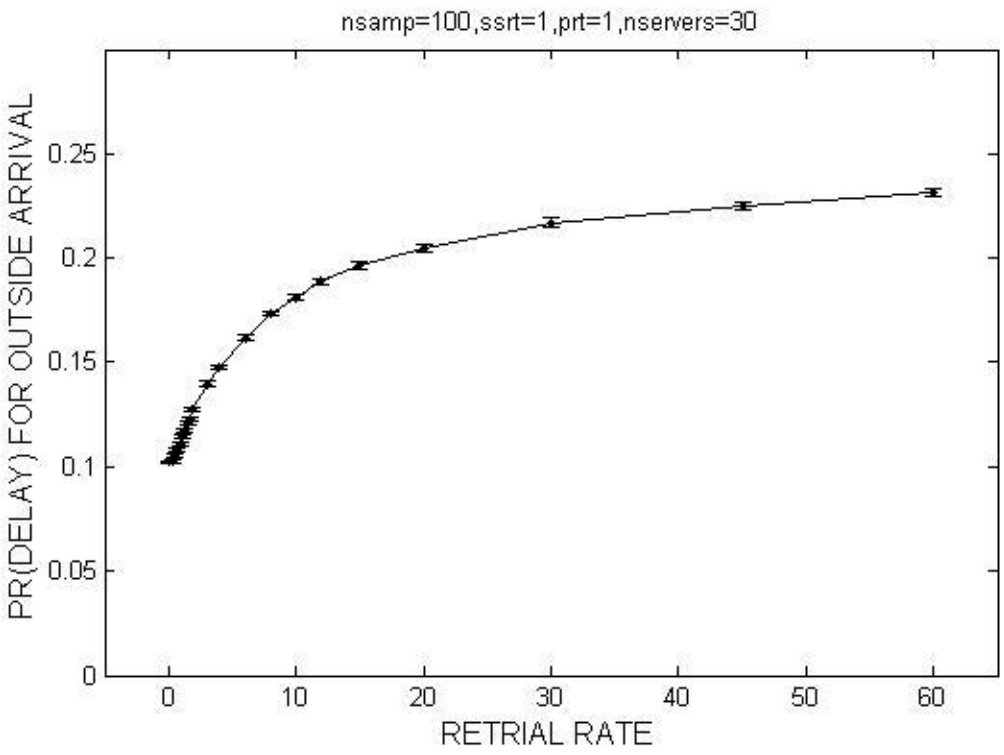

nsamp=100,ssrt=0,prt=1,nservers=12
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

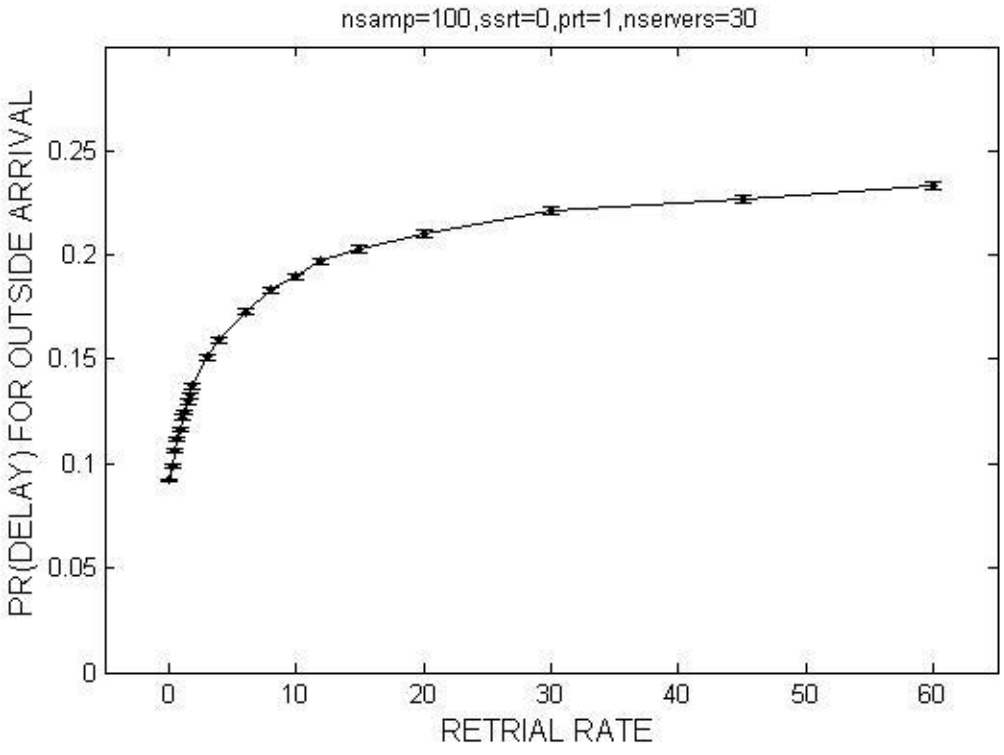

nsamp=100,ssrt=1,prt=0,nservers=12
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

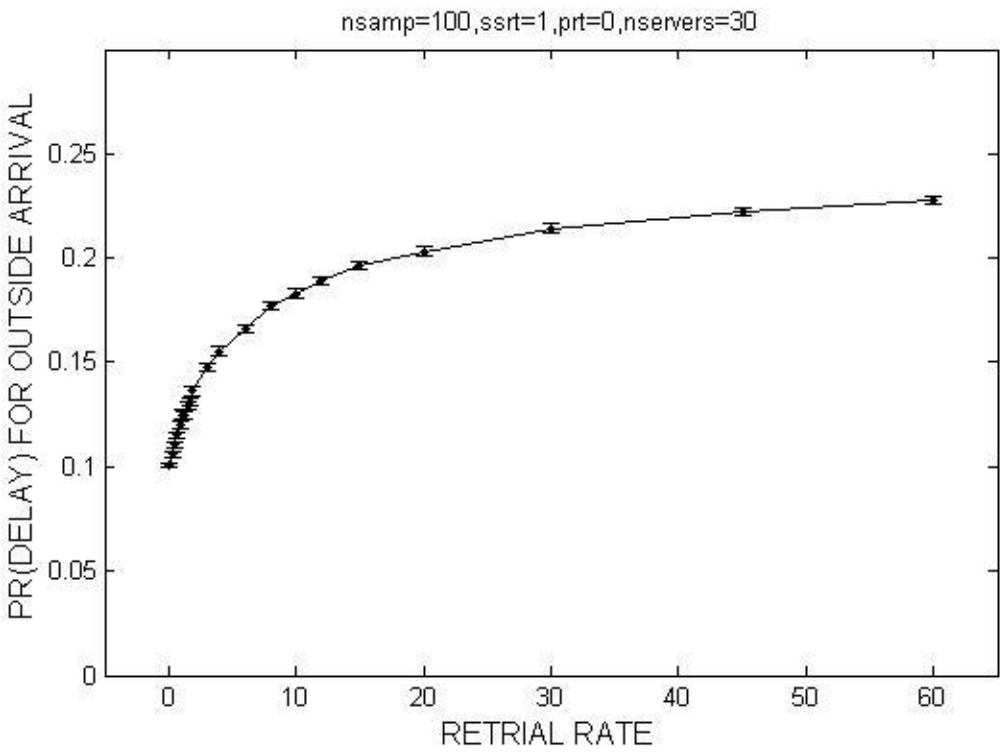

nsamp=100,ssrt=1,prt=1,nservers=20
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

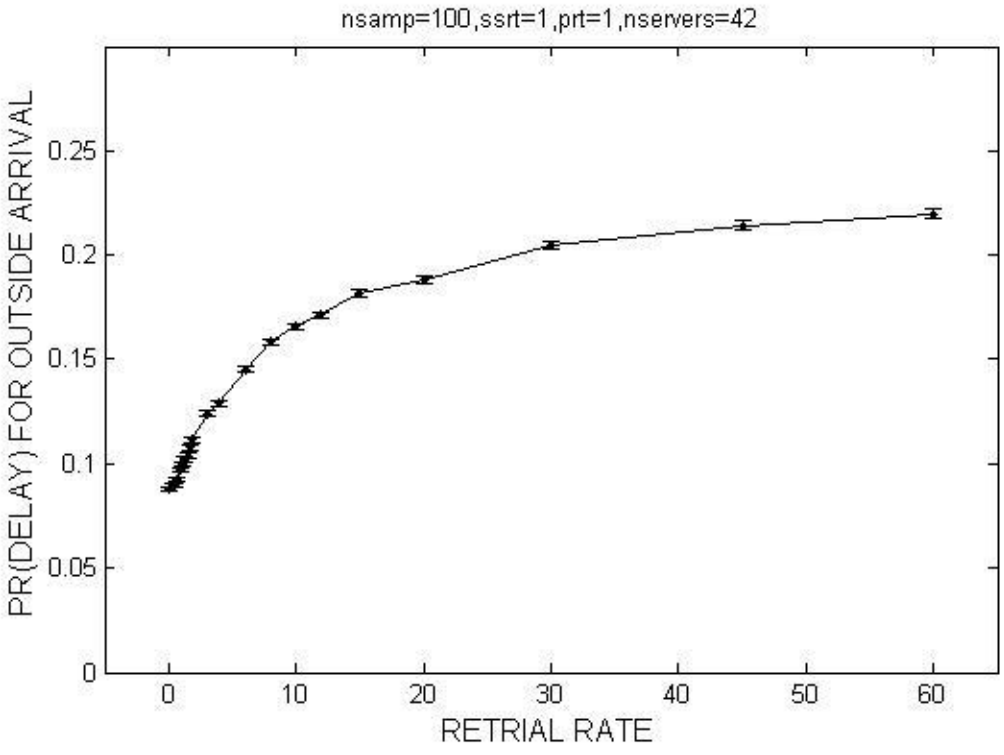

nsamp=100,ssrt=0,prt=1,nservers=20
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

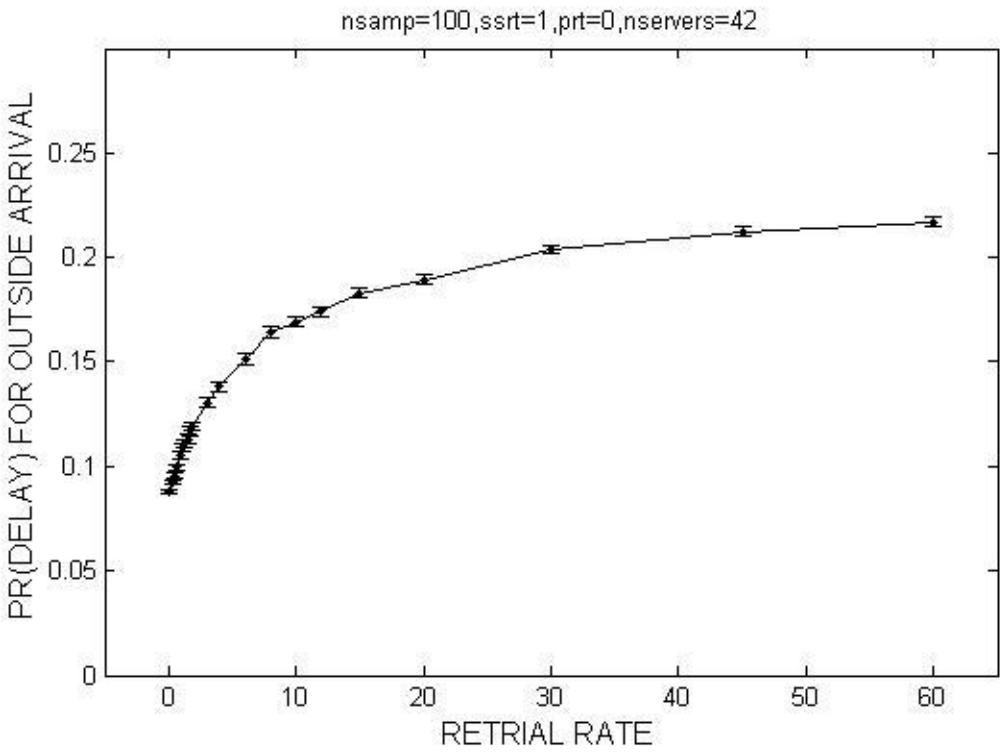

nsamp=100,ssrt=1,prt=0,nservers=20
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

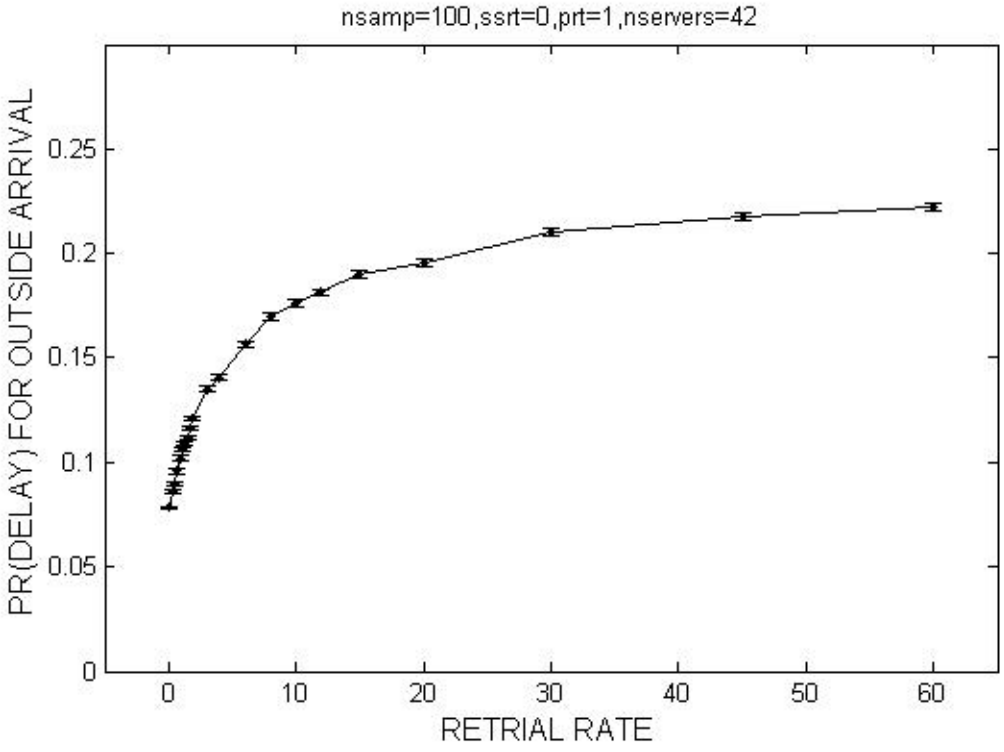

nsamp=100,ssrt=1,prt=1,nservers=30
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

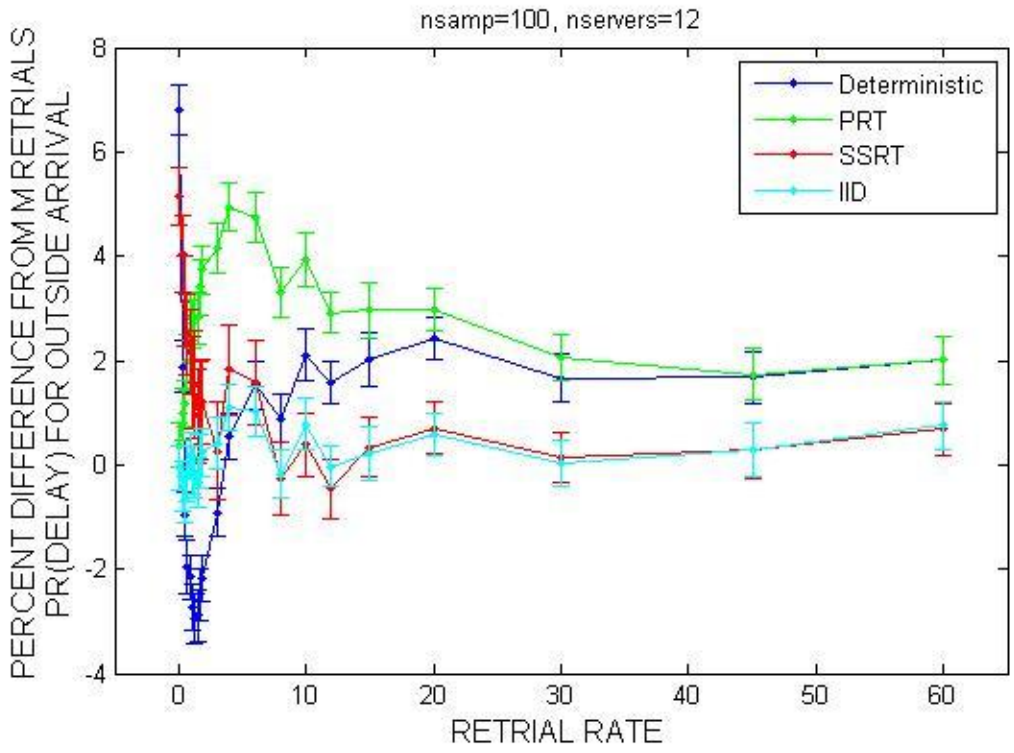

nsamp=100,ssrt=0,prt=1,nservers=30
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

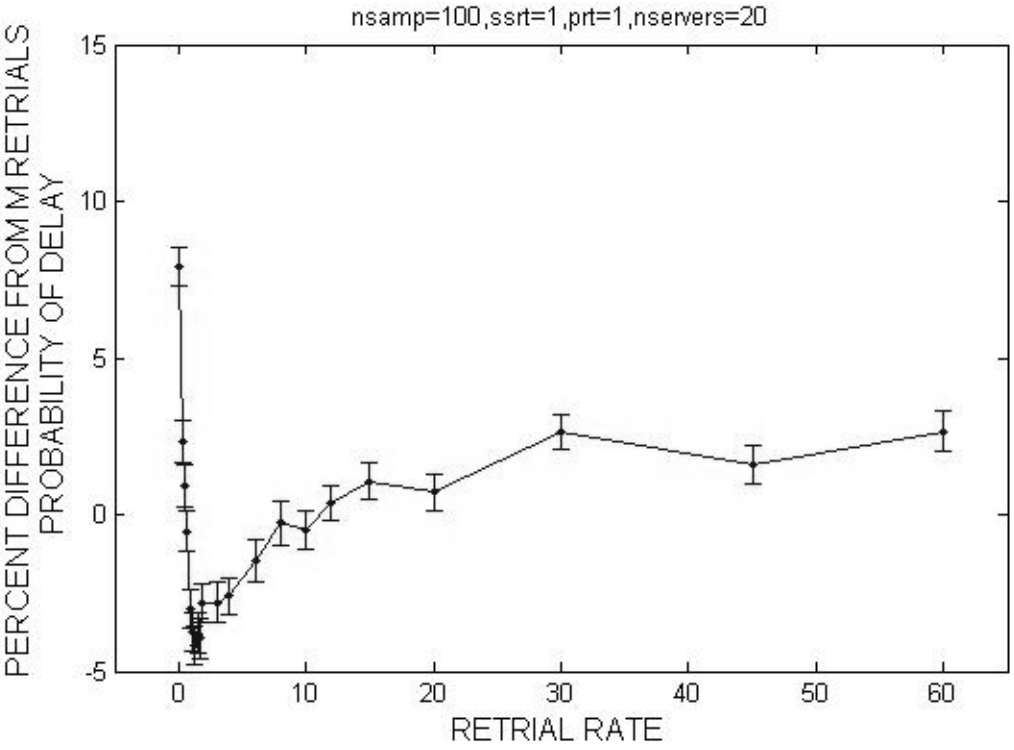

nsamp=100,ssrt=1,prt=0,nservers=30
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

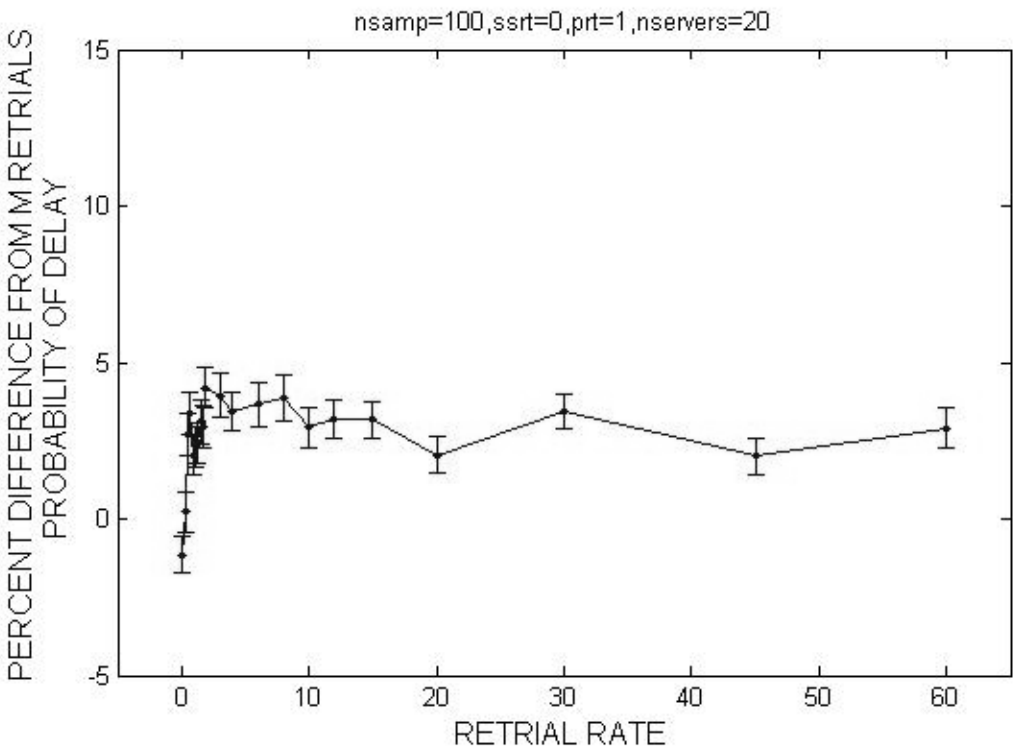

nsamp=100,ssrt=1,prt=1,nservers=42
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

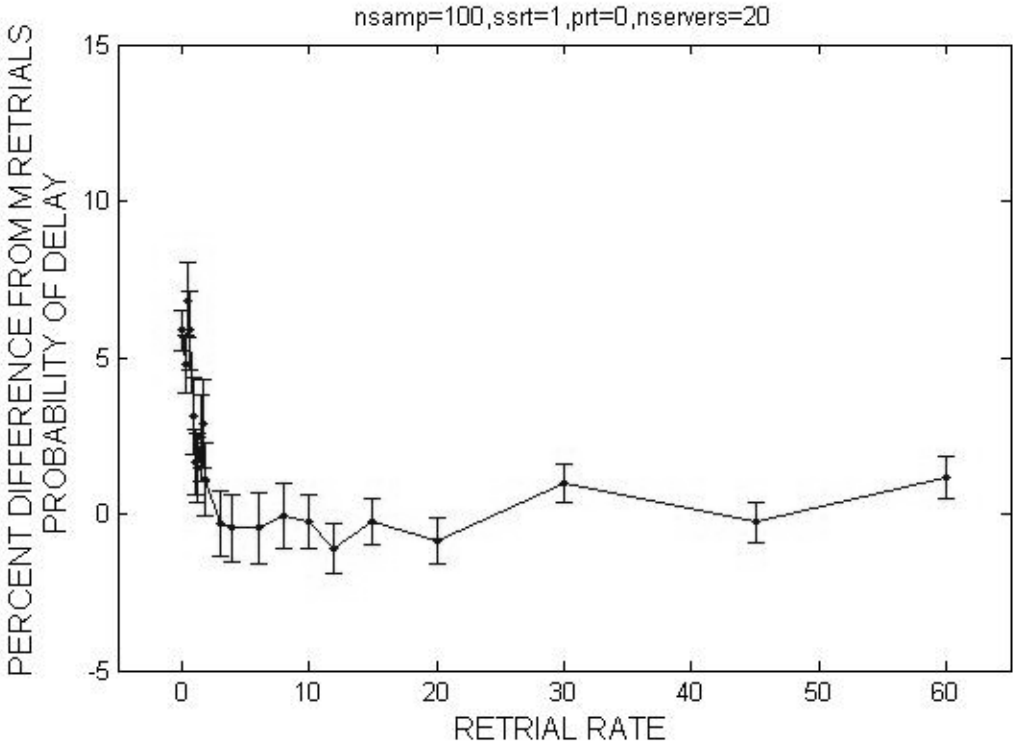

nsamp=100,ssrt=1,prt=0,nservers=42
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

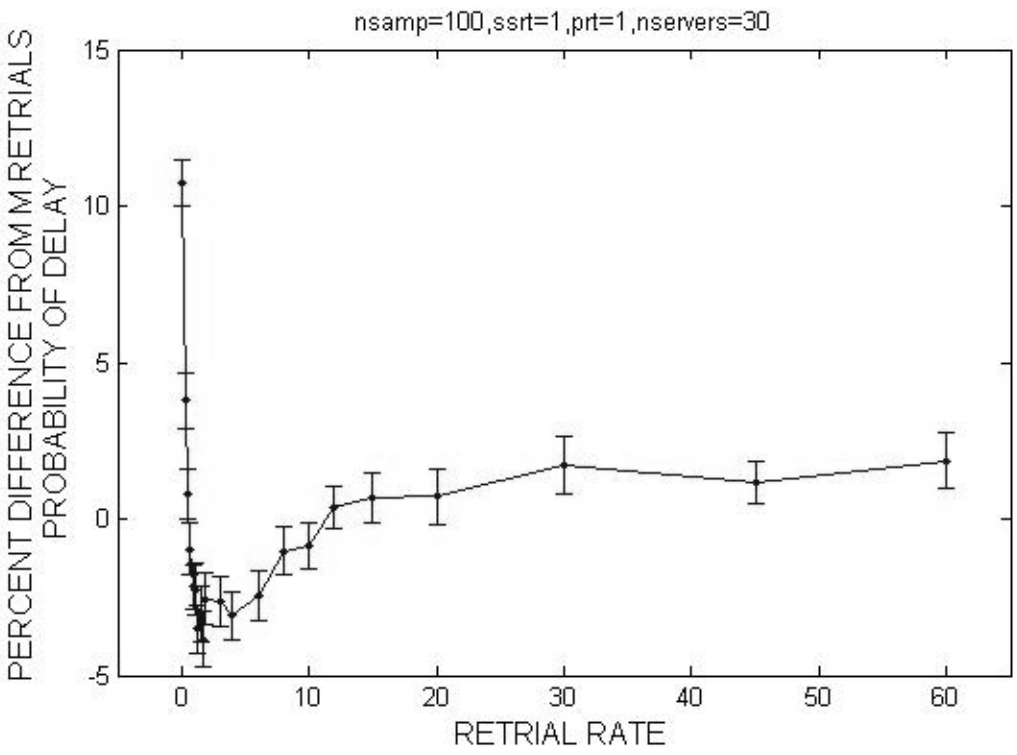

nsamp=100,ssrt=0,prt=1,nservers=42
PR(DELAY) FOR OUTSIDE ARRIVAL
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
RETRIAL RATE

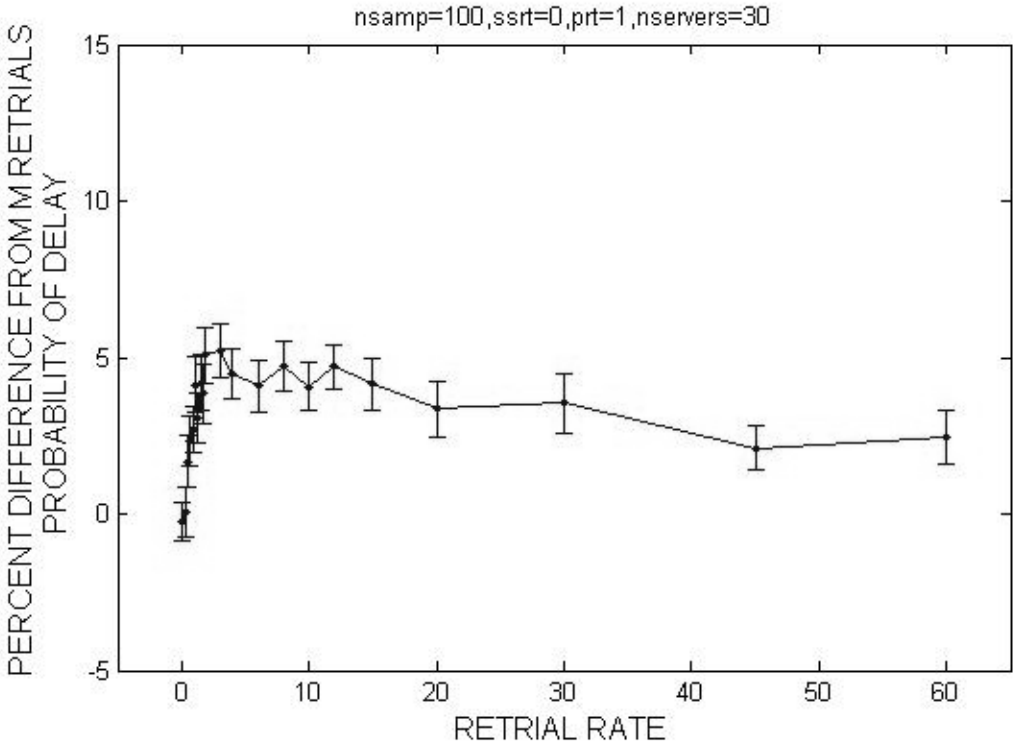

nsamp=100, nservers=12
PERCENT DIFFERENCE FROM M RETRIALS
PR(DELAY) FOR OUTSIDE ARRIVAL
RETRIAL RATE
Deterministic
PRT
SSRT
IID
8
6
4
2
0
-2
-4
0
10
20
30
40
50
60

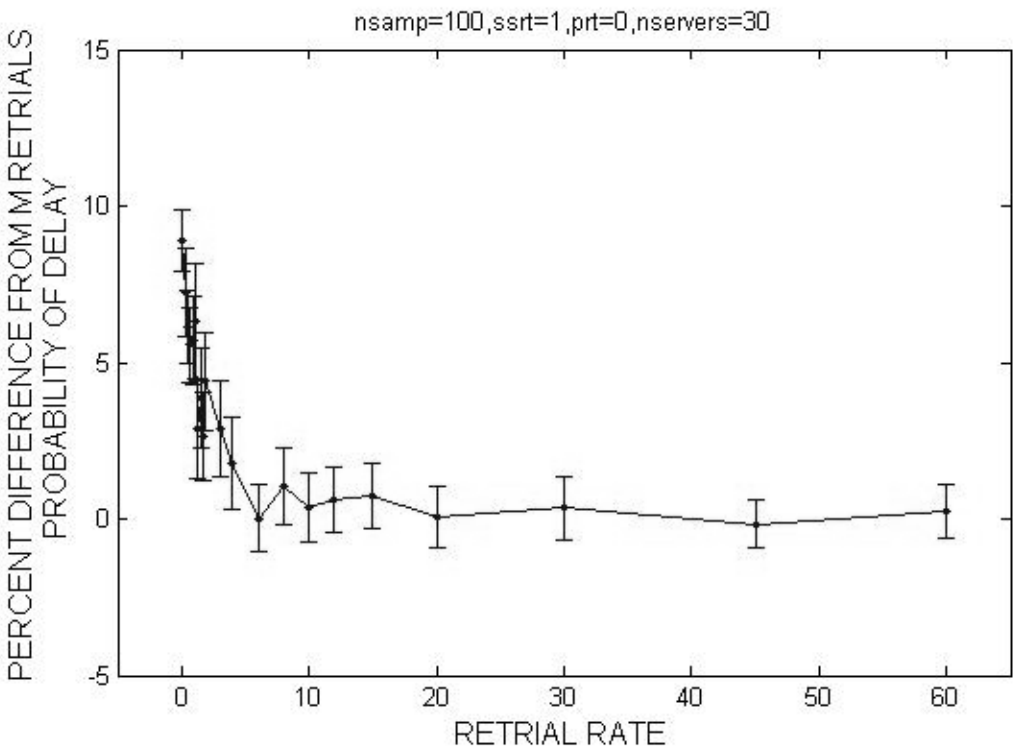

nsamp=100,ssrt=1,prt=1,nservers=20
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
RETRIAL RATE
15
10
5
0
-5
0
10
20
30
40
50
60

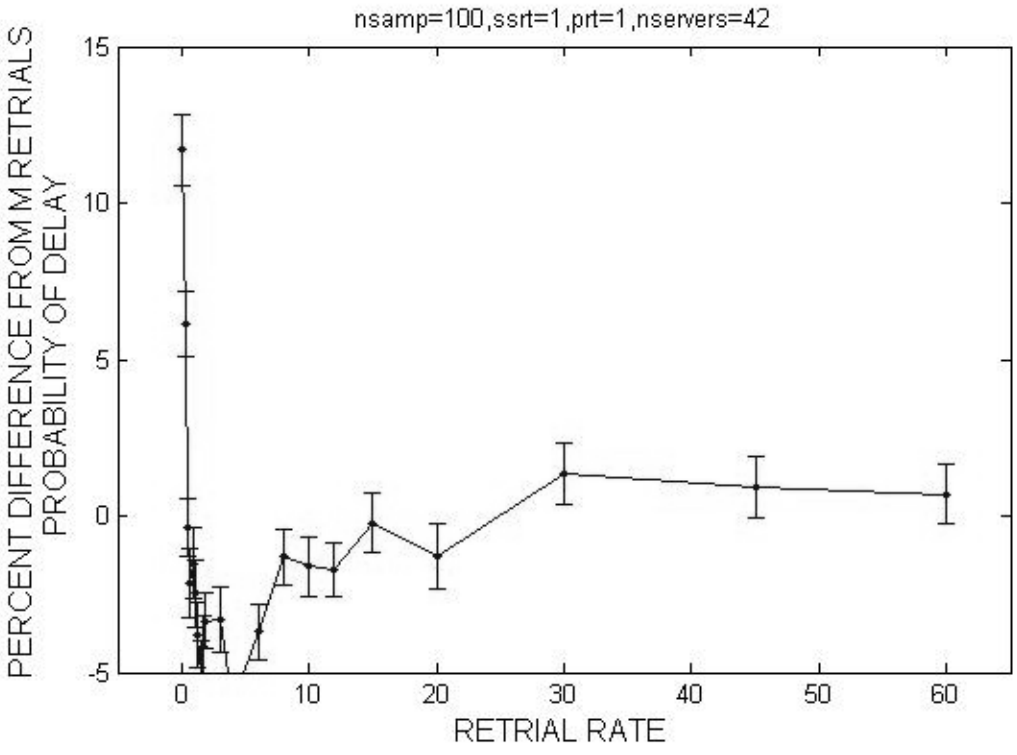

nsamp=100,ssrt=0,prt=1,nservers=20
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
15
10
5
0
-5
0
10
20
30
40
50
60
RETRIAL RATE

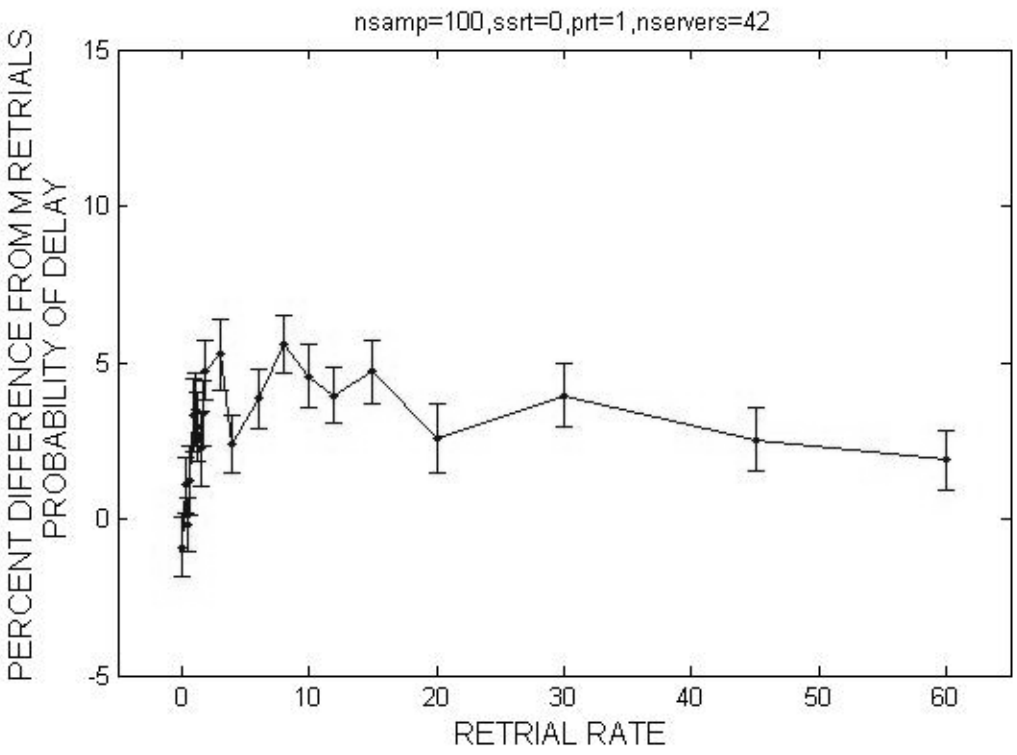

nsamp=100,ssrt=1,prt=0,nservers=20
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
15
10
5
0
-5
0
10
20
30
40
50
60
RETRIAL RATE

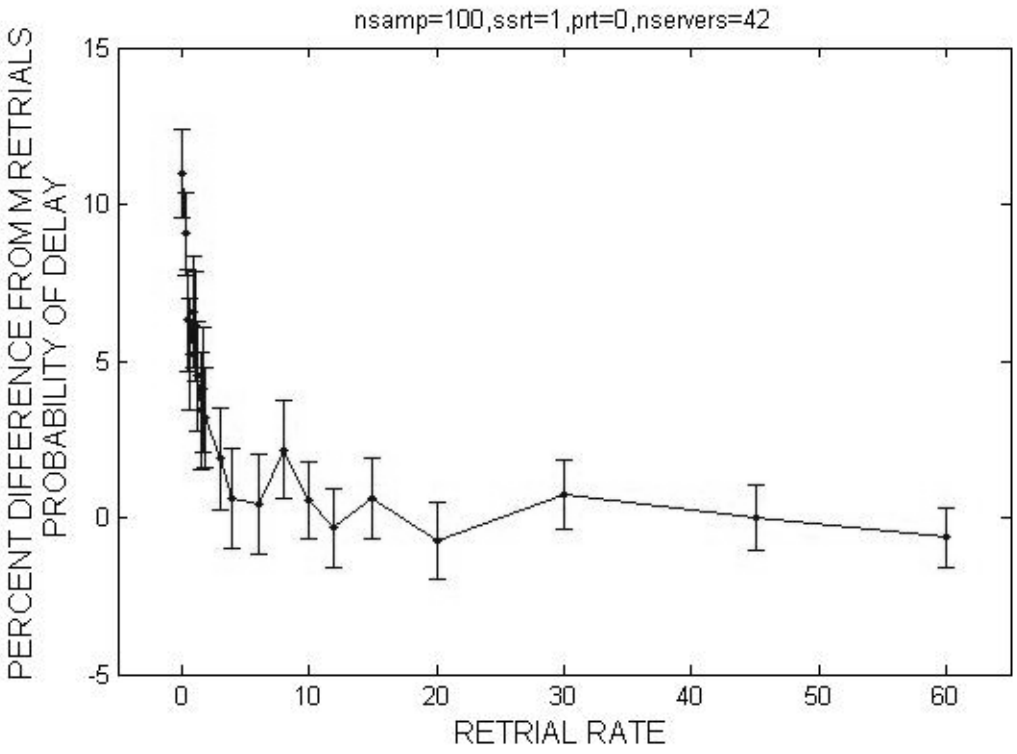

nsamp=100,ssrt=1,prt=1,nservers=30
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
RETRIAL RATE
15
10
5
0
-5
0
10
20
30
40
50
60


nsamp=100,ssrt=0,prt=1,nservers=30
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
RETRIAL RATE
15
10
5
0
-5
0
10
20
30
40
50
60


nsamp=100,ssrt=1,prt=0,nservers=30
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
15
10
5
0
-5
0
10
20
30
40
50
60
RETRIAL RATE


nsamp=100,ssrt=1,prt=1,nservers=42
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
15
10
5
0
-5
0
10
20
30
40
50
60
RETRIAL RATE


nsamp=100,ssrt=0,prt=1,nservers=42
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
15
10
5
0
-5
0
10
20
30
40
50
60
RETRIAL RATE


nsamp=100,ssrt=1,prt=0,nservers=42
PERCENT DIFFERENCE FROM M RETRIALS
PROBABILITY OF DELAY
15
10
5
0
-5
0
10
20
30
40
50
60
RETRIAL RATE