"What did I learn in Performance Analysis last year?": Teaching Queuing Theory for Long-term Retention

Varsha Apte
Computer Science and Engineering Department
Powai, Mumbai 400 076, India
varsha@cse.iitb.ac.in

ABSTRACT

This paper presents experiences over thirteen years of teaching a queuing systems based performance analysis course. We discuss how a 'mathematics first' approach resulted in students not retaining the intuitive concepts of queueing theory, which prompted us to redesign a course which would emphasize the 'common sense' principles of queuing theory as long-term takeaways. We present a sequence of syllabus topics that starts with developing and arriving at a host of queuing systems based insights and 'formulae' without going into the mathematics at all. Our key insight is that in practice, only *asymptotic values* - at both low and high load - are critical to (a) understand capacities of systems being studied and (2) basic sanity checking of performance measurement experiments. We also present two assignments (one measurement, and one simulation) that we now give, that help in reinforcing the practical applicability of queuing systems to modern server systems. While we do not have formal studies, anecdotally, we have reason to believe that this re-design has helped students retain for the long term, the most essential results of queuing systems, even if they do not study this subject further.

CCS CONCEPTS

• **General and reference** → **Measurement**; **Performance**; • **Mathematics of computing** → **Queueing theory**; • **Social and professional topics** → **Model curricula**; • **Applied computing** → **Education**.

ACM Reference Format:
Varsha Apte. 2019. "What did I learn in Performance Analysis last year?": Teaching Queuing Theory for Long-term Retention. In *Tenth ACM/SPEC International Conference on Performance Engineering Companion (ICPE '19 Companion), April 7–11, 2019, Mumbai, India.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3302541.3311526

ICPE '19 Companion, April 7–11, 2019, Mumbai, India

ACM ISBN 978-1-4503-6286-3/19/04…$15.00
https://doi.org/10.1145/3302541.3311526

CS 681 (Performance Analysis of Computer Systems and Networks) Syllabus
Introduction to performance measures; performance measurement; analytical modeling for performance; use of basic probability for performance evaluation; Markov chains; Markovian queues; non-Markovian queues; queueing networks; design, implementation and analysis of discrete event simulation models (confidence intervals, discarding transients); Applications to: operating systems (paging, scheduling, multi-threading); networks: Web servers, TCP/IP models, Wireless LAN, power management, virtualization.

Figure 1: CS 681 Syllabus

1 INTRODUCTION

A graduate course by the name of "Performance Analysis of Computer Systems and Networks" and course code 'CS 681' has been taught by this author for about fifteen years in the Department of Computer Science and Engineering, IIT Bombay. The official syllabus of the course is shown in Figure 1.

The course originally followed a conventional approach of starting with teaching the basics of probability distributions, the usual properties and metrics derived from them, the mathematics related to functions of random variables, culminating with stochastic processes. Then followed the use of Markov chains to model queuing systems. The course has always included a hands-on project component of developing a discrete event simulation model of a system, and applying sound statistics to analyze the results of the simulation. This allowed students to compare the simulation modeling method with analytical modeling.

At the Spring 2015 offering of this course we made a major change in the teaching approach of the course. As we taught the course, a fresh set of notes were written [2] where this approach can be found in further detail. The main motivation behind this change was an observation over the years that students were not retaining the basic queuing systems 'intuition' that this course should have fostered in their mind. Graduate students in India, from institutes other than IITs generally do not have a particularly rigorous mathematical preparation, and thus the students got buried under the mathematics, and missed the opportunity to take away basic queuing principles that are simply 'common sense' and are mathematically trivial. To address this problem, we completely turned the sequencing around to introduce basic

For each topic, examples of application of the theory to systems and networking were continuously provided. From networking: Aloha, TCP, IEEE 802.11, Ethernet, TDMA/FDMA. From Systems: Multithreaded servers such as Web servers, multiple Server back-ends, etc, power-managed systems, virtualization, mobile phones.

- Introduction to resources of contention, parameters and metrics of performance.
- Basic probability review, conditional probability, Law of total probability
- Random variables, expectation, variance, moments
- Common distributions - Special properties of Poisson, Exponential distributions (memorylessness)
- Functions of random variables: order statistics, Transform methods, Random Sums
- Distrete Event Simulation and its sound statistical analysis
- Stochastic Processes
- Continuous Time Markov Chains
- Markovian Queuing Systems and Queuing Networks(Open). Basic elements of an open queueing system, Kendall notation, operational laws, Litte's Law, Jacksonian Open Queueing network and its analysis.
- Closed Queuing Network: CTMC based analysis (example only). Mean Value Analysis
- Discrete Time Markov Chains and $M/G/1$ queue

Figure 2: Original Sequence of Syllabus Topics

principles of queuing theory before introducing probability and stochastic processes, and introduced a 'discovery-based' approach for students to better understand queuing theory.

The rest of the paper describes the problems faced in the original teaching sequence, the various teaching strategies used to address them and concludes with a discussion of whether our course re-design has helped students retain the most practically applicable tools of queuing theory.

2 CHALLENGES WITH A MATHEMATICS - FIRST APPROACH

For the first many years, we taught this course with a belief that introducing probability, random variables, and stochastic processes was a pre-requisite to a solid understanding of queuing systems, which is the approach taken in most textbooks [1, 6]. The sequencing of the syllabus earlier is shown in Figure 2. In the second half of the course, after students were introduced to discrete event simulation, students also did a self-proposed simulation analysis project. Students would submit a proposal of a system (based on an existing paper) to analyze, ask some design questions around it, and answer those using a statistically sound simulation. They had to code the simulator from scratch, so that they truly understood the workings of a simulator.

Over the years, this course started looking very challenging to students. Graduate students taking this class in IIT Bombay are usually from Universities lacking a rigourous curriculum in Engineering Mathematics. Thus, the probability and random variables background had to be taught at introductory level. This part, starting from basics to functions of random variables, which then can be understood fully only by learning transform methods, forms too large a syllabus component to be done as a *pre-requisite* for the rest of the course on queueing systems. With a two-weeks budget, it still felt rushed to a majority of graduate students.

In this early form, queuing systems were taught as 'applications' of stochastic processes - Continuous Time Markov Chains (CTMCs) for $M/M/c/k$ queues, and Discrete Time Markov Chains (DTMC) for the $M/G/1$ queue. This had another unintended impact, which is that inferences one can draw about a queueing system, without resorting to modeling it using a Markov Chain, were not obvious to a student. E.g. almost all low-load and high-load asymptotes of performance measures derived from queueing systems can be reasoned about without using Markov chains - this was not obvious to students.

In the recent years, the discrete event simulation project also resulted in numerous complaints from students, and an increasing display of complete lack of comprehension of statistical concepts taught in class such as confidence intervals.

Lastly, this sequence left very little time to truly understand DTMCs - in the final exams, most students routinely failed to answer the one question on DTMC modeling. Students also found Closed Queuing Network modeling very difficult to understand, as discussion on MVA was done at the very end, when students were already saturated.

At the end of the course, the students were left with no clear takeaways of what they had learnt which they could *apply* easily in their day-to-day systems and networking R&D work.

There are some alternative books that employ a significantly less probability-and-stochastic-processes approach to discussing queueing systems, e.g. Menasce's book [5] offers a fairly practical discussion on E-business application analysis. However, this book is somewhat specific to computer applications, and capacity planning - whereas we wanted to continue to cover methods applicable to both networks and systems, and thus focus on the formulations and theory, rather than the application area.

Mor Harchol-Balter's book [4] is closest to what we wanted. In her book, she infuses the theory coverage with substantial intuition; the book starts out with design questions that force the reader to think intuitively. Nonetheless, her book also follows the approach of covering probability pre-requisites before presenting queuing theory, and is in fact mathematically fairly demanding.

Table 1: Low and High Load asymptotes of Various Metrics of Queuing Systems: Arrival rate is λ, service time is τ

	$G/G/1$		$G/G/c$		$G/G/1/K$		$G/G/c/K$	
Metric	$\lambda \to 0$	$\lambda \to \infty$	$\lambda \to 0$	$\lambda \to \infty$	$\lambda \to 0$	$\lambda \to \infty$	$\lambda \to 0$	$\lambda \to \infty$
Throughput	0	μ	0	c μ	0	μ	0	c μ
Utilization	0	1	0	1	0	1	0	1
Queue Length	0	∞	0	∞	0	K	0	K
Number in System	0	∞	0	∞	0	$K+1$	0	$K+c$
Waiting Time	0	∞	0	∞	0	$K\tau$	0	$\frac{K}{c}\tau$
Response Time	0	∞	0	∞	0	$(K+1)\tau$	0	$(\frac{K}{c}+1)\tau$

3 A COMMON - SENSE - FIRST APPROACH TO TEACHING QUEUING SYSTEMS

After our survey of existing books, we concluded that we needed to propose our own sequence and approach in which to introduce and develop queueing systems theory and its applications. The revised sequencing of the syllabus that we follow now since 2015, is shown in Figure 3. As can be seen, we can cover a lot of ground in Queueing Systems and their practical use without going into much mathematics.

The main insight here is that one can engage students in thinking about *low load asymptotes* and *high load asymptotes*, of queue performance metrics, or their lower or upper bounds, in a completely intuitive manner. Table 1 shows the table of asymptotes we get students to arrive at by themselves by facilitating intuitive brainstorming in class.

For example, for the $G/G/c/K$ queue, it is easy to get students to think through the low load asymptote of throughput (0) and the high load asymptote of throughput $c\mu$. Similarly, the load load and high load of queue lengths are trivial: 0 and K. To reason about, say mean response time (of accepted requests), strictly mathematically speaking, students should have a good understanding of memoryless properties and conditional distribution of queue length as seen at arrival epochs. They should also understand that in case of limited buffer queues, when we talk about response time, we must condition it on not having an "infinite" value. We should also be first introducing defective distributions. But in the approach of this course, we choose to not follow this mathematically rigourous path. When only talking about conditions where $\lambda \to \infty$ it is not hard to take the student through the following thought process

- As load grows, the likelihood that a request that is not dropped, joins in the queue at the last position grows.
- The mean remaining time of the requests in service may vary based on when this request has joined, but an upper bound is τ.
- A request will move one position up in the queue when any request from the queue leaves, an upper bound for the expected time for this with one server would have been τ; intuition says that with c servers it should be $\frac{\tau}{c}$. Since buffer length is K and then there is the request's own service time, the formula follows.

For each topic, examples of application of the theory to systems and networking were continuously provided. From networking: Aloha, TCP, IEEE 802.11, Ethernet, TDMA/FDMA. From Systems: Multithreaded servers such as Web servers, multiple Server back-ends, etc, power-managed systems, virtualization, mobile phones.

- Introduction to resources of contention, parameters and metrics of performance.
- Introduction to queuing systems (open): basic elements of an open queueing system, Kendall notation, operational laws, low load and high load asymptotes of queue performance metrics, Litte's Law. Brief intuitive mention of Exponential distribution and memoryless property here, and Poisson distribution and PASTA.
- Closed queuing system: operational and low/high load asymptotic analysis of single server multiple users closed system, closed tandem queuing networks, Jacksonian closed queuing networks: Asymptotic analysis and MVA.
- Motivate that now for further analysis, formal probabilistc methods will be required
- Basic probability review, conditional probability, Law of total probability
- Random variables, expectation, variance, moments
- Common distributions - Special properties of Poisson, Exponential distributions (memorylessness)
- Distrete Event Simulation and its sound statistical analysis
- Functions of random variables: order statistics, Random Sums (if time permits)
- Stochastic Processes
- Continuous Time Markov Chains: examples of modeling systems which cannot be formulated as queueing systems
- Markovian Queuing Systems: Open and Closed. The coverage here focusses now only on filling the "holes" (i.e. values of metrics between low and high load asymptotes).
- Discrete Time Markov Chains and $M/G/1$ queue

Figure 3: Revised Sequence of Syllabus Topics Followed

This puts students in a very good position, where we can ask them to draw graphs where they can draw lower and higher asymptotic values of the plots, of all the standard queueing system metrics. After this much has been absorbed well, only then we motivate the need to study Markov chains,

so that the exact values of the 'middle' portion of the graphs can be calculated.

We also introduce *closed systems* very early. This allows us to then quickly give a *performance measurement* assignment in which we ask students to measure the performance of a simple Web server running a CPU bound script by generating both 'open load' and 'closed load' on it (Figure 7 and 8). Students are supposed to apply simple operational laws, and the derived asymptotes to carry out a 'sanity check' on the results. We have found during our vivas that most students actually do not cross check their results by using simple queuing systems heuristics and it is only during vivas that we engage them in such thinking. In many submissions, we are able to point to the students that the numbers are failing very basic queueing theory sanity checks, which means something is wrong with their experiment.

Based on this assignment, students submit a report which includes graphs such as the one shown in Figure 5, which is the Response Time (R) vs Arrival rate (λ) graph. Based on this graph, we ask students questions such as:

- Can you estimate the high load asymptotic value of the Throughput? (A good estimate should be $\frac{c}{R_{min}}$ where c is the number of cores in the server.)
- Is the knee of the response time curve at the expected point? (Should be roughly near the $\frac{c}{R_{min}}$. In some submissions it is not, and the students are asked to explain why. Often this led to realizing of some error in the experiment.)
- What is a good estimate of the service time of the server? (R_{min}). How should this relate to the slope of the Utilization plot? (Slope should be roughly $\frac{R_{min}}{c}$).

Thus we drive home the point that just from, say the Response Time graph, one should be able to have a good idea of what the throughput and utilization graphs should look like and employ these 'queueing theory heuristics' as a sanity check on the graphs.

Students also submit the response time vs number of users graph for closed system, and are asked to relate the slope of the curve to the service rate of the server system, explain the low load and high load asymptotes, and verify how good Kleinrock's Saturation heuristic was for their experiment.

All of the above is done without any review of probability, stochastic processes or Markov chains. Note that this also includes teaching Mean Value Analysis of Closed Product Form Queueing Networks. This again means that concepts such as the Sevcik-Mitrani Arrival Theorem has to again be discussed 'intuitively', and various probability distribution assumptions stated clearly. However, in our experience, this is do-able and again, worthwhile to be done before diving into stochastic processes, rather than after. Once stochastic processes (CTMCs and DTMCs) start, students mental stamina is spent mostly in dealing with the newness of those concepts. In contrast, MVA is actually quite palatable. Once we teach MVA, we also carry out an in-class laptop based exercise where students code the MVA iteration for an example Closed QN in a spreadsheet. Here again, we engage students intensely in predicting and explaining the *asymptotes* for all the metrics.

In this version of the course, we also changed the nature of the Discrete Event Simulation assignment (Figure 4). Instead of varied self-proposed systems being evaluated (which had a very successful run earlier), we now give an assignment where we essentially again analyze a Web server, this time using simulation instead of measurement, and encourage students to ask 'what-if' questions which would be difficult in measurement as well as in an analytical model. For example see Figure 6 where the students compared throughput with increasing quantum size, and then tried to explain the trend observed. Furthemore, for extra credit we encourage students to model the same Web Server, using MVA, and compare the results with both measured, and simulated. We have had students do this for extra credit and feeling very satisfied when the results approximately matched, and were also able to reflect on what would be the reasons behind the divergences between the values produced by the three approaches.

Finally, we now do not teach transform methods at all, and thus do not study topics based on distributions of sums of random variables - which is where transform methods would be needed. Even in case of $M/G/1$ queue we follow the visual method based on remaining service time (from the book by Bertsekas and Gallagher [3]) and avoid teaching and using transform methods completely. This reduces the mathematical burden on students greatly.

We have not done formal studies regarding whether this improved the intuitive understanding of queuing systems by students. But 'anecdotally' we do find better answers in the theory quizes and finals, after such hands-on experiments relating measurement to queueing theory-based predictions, is done, and finally also related to simulation modeling.

4 DISCUSSION AND FUTURE OUTLOOK

In this paper, we presented the gist of the changes we had to make, to ensure that the most practically applicable methods of queueing theory are taught early, and reinforced throughout the course through hands on load testing and measurement assignments, and a discrete event simulation assignment. We used the example of a multi-threaded Web server, with a CPU-bound script, running on a multi-core machine, on a high-speed LAN, to keep things simple and comparable to theoretical queuing systems. This allows students to appreciate the predictive power of queueing system models, and still understand the assumptions that are required to keep the mathematics tractable.

Our main contribution in this course re-design is to have gleaned results from queuing systems that do not require a rigorous stochastic process background to understand, and bring them to the early part of the course. Most importantly, this included low and high load asymptotes of various metrics. Our claim is that practically, performance engineers are most interested only in these asymptotes (especially at high

Simulation Modeling of a Web Application

Overall Goal of the Assignment

In this assignment you will continue the study of the performance of a Web Application, this time through a discrete event simulation model. You will take advantage of the power of simulation analysis and model behaviours that are difficult to capture using theoretical queuing systems and Markov chain models. You will write a program, and then run it to ask various interesting questions. For all metrics you will perform proper statistical soundness analysis.

Detailed Specifications

Write a simulation program in any general purpose, object-oriented programming language (C++, Java or python)

System Characteristics

Your system should have the characteristics of the Web server environment that you measured. The following is recommended, you can add/modify.

- Multi-core server machine
- Assume thread-to-core affinity. Once a thread is "assigned" to a core, it will remain there
- Multi-threaded Web server
- Thread-per task model - until max is reached, after which queuing for threads starts.
- Round-robin scheduling, with context-switching overhead
- Request time-outs
- Users are in a standard closed loop - issue request, wait for response, think then issue request again.
- Think time should have a mode not close to zero - do not assume exponential distribution for think time.
- Have options of various request execution time distributions, such as - constant, uniform, exponential.
- Request timeout should also have a well-chosen distribution (Have some sort of a minimum, and then a variable component)

Performance Metrics

Metrics/graphs will be similar to what you measured for the earlier assignment .

- Average Response Time vs number of users
  Generate confidence intervals for this metric. Point estimates (averages of estimates from independent runs) are ok for the remaining metrics
- Throughput, Goodput, Badput vs Number of users
  - Badput = rate of completion of requests which were already timed out.
  - Goodput = rate of completion of requests that were not timed out
  - Throughput = goodput + badput
- Request Drop rate vs number of users
- Average core utilization vs Number of users
- Some additional graphs representing your own curiosity regarding system performance vs some system parameter

Ensure that the transient is determined and discarded in each of your simulation runs. You can do this informally, or follow Welchs procedure.

Figure 4: Discrete Event Simulation Assignment for Reinforcing Queuing Theory Principles

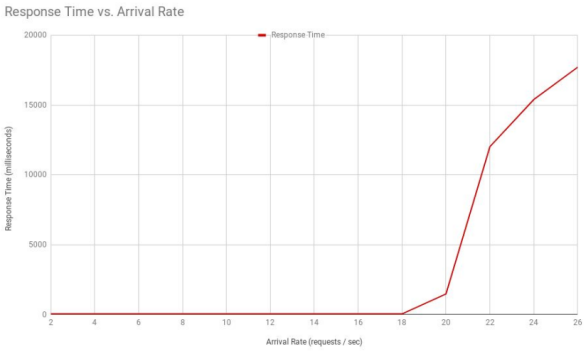
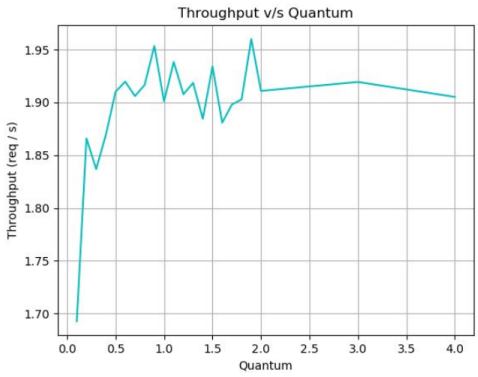


Figure 5: Typical Response time vs Arrival Rate graph submitted as part of assignment report, based on which many queueing theory based deductive questions can be asked.



Observations

- When quantum size is very low, there are more context switches therefore less throughput
- When quantum size is very high, there are very less context switches. So the throughput saturates as quantum is much higher than the service time of the request

Figure 6: A "what-if" analysis graph submitted by students using Discrete Event Simulation for studying performance of Web Servers

load), and an intuition to quickly predict and calculate these asymptotes is critical for a systems researcher.

It is true that discussing some intuitive results of queuing systems without using the rigour of mathematics ends up being somewhat 'hand-wavy' and certainly is often approximate. Nonetheless this is a trade-off that, in our assessment, was necessary to engage student attention and increase the chance of developing the intuition required if they were to continue systems R&D work. Unfortunately, we have not carried out formal pedagogical studies to quantify the impact of this change of teaching approach. Anecdotally, we feel that students are better retaining some take-aways for

Performance Measurement of a Web Application

Overall Goal of the Assignment The high level goal of the assignment is for learners to "experience" and understand the performance of an actual queuing system first-hand, by direct measurement of metrics, and experimenting with how these metrics change with various parameters. The example we will use is of a Web Server, which was discussed in class as a typical queuing system. The resources a Web server uses are socket connections, threads, CPU, network, disk, memory and so on. Each resource forms a queueing system of its own and impacts overall performance. You will try your best to understand these impacts, and relate what you are learning in class to what you observe in the experiments.

Detailed Specification
For this assignment you will need TWO machines. One one machine, you must install a Web server (e.g. apache, or any other web server of your choice). You will also be given a php script to run on this Web server, so you must load the PHP module and make PHP work with this Web server. On the other machine, you will install two Web load generators. Load generators are programs which emulate users of a Web site by sending certain specified URLs to the Web server, reading the response, and as a result noting down performance metrics such as response time, throughput, number of requests lost, timed out, etc. The two load generators to be installed are: httperf, and Tsung. Httperf is capable of generating an "open" load - i.e. open arrivals specified by an arrival rate, while Tsung generates a "closed load" - i.e. it creates multiple virtual users who are in a request-response loop with the server. Both these tools will take specifications in their own formats, of which URLs are to be sent by the tool, and the load quantifier. For httperf, you have to specify an arrival rate, and for Tsung, a number of users and think time. In both cases, a load test duration is specified. To get sensible results, you must run a load test at each "load level" for at least 5 minutes. Each of these tools will output "client-side" performance metrics such as response time, throughput, requests refused, timed out, etc.
For this assignment, you are also required to measure some "server-side" performance metrics. Such metrics can be measured using Linux utilities such as top and vmstat while the load generation is going on.
The Load Test
Your "loadtest" will generally proceed as follows:
Start the Web server. ENSURE THAT NO OTHER PROCESS IS RUNNING ON THE SAME MACHINE THAT MAY AFFECT PERFORMANCE.
Start the load (e.g. httperf on the client machine) - here also, it is best that the client is not running other major processes. Run the load for a few minutes. For this assignment I suggest setting the timeout to something very large (we won't measure timeouts).
As soon as you start load, start server side performance measurement (top, or vmstat that writes out measurement snapshot periodically into a file). Wait for a few minutes (load generation should be going on, 4-5 mins should be enough) and then stop the server side performance measurement
Stop the load generation.
These steps are important because you must measure the server only when a steady load of the rate that you specified is actually coming to the server. If you start measurements before load starts, then you may get nearly zero CPU utilization. The same problem will happen if you continue taking measurements after load generation has stopped. So it is critical that the average CPU utilization is calculated only for the phase when the server is busy serving requests at the rate offered to it.
For generating performance curves, you must repeat the load test at various load levels.
Server side performance measurement This is essentially just for getting snapshots of CPU utilization. You should take, say, 10-second snapshots of this metric and then average the snapshots. Do NOT take "visual" snapshots. Write out the output of top or vmstat to a file and then take the average (you may need to use your scripting skills for calculating this average - e.g. awk or python).
You must ensure that your server side configuration is sensible. Ensure that Apache has enough threads so that threads are never the bottleneck. It will be easiest if you run this assignment on a single-core server. If you have a multi-core server, ensure again that number of Web server threads is enough, and remember this fact when you make plots and compare with theory (multiple server queueing system).
. . . (Continued)

Figure 7: Performance Measurement Assignment for Reinforcing Queuing Theory Principles

the long-term, especially if they take up jobs in "systems" companies.

Going forward, the main evolution of Performance Modeling courses will need to be in the direction of how these approaches fit in with new machine learning based empirical models. Students are increasingly resistant to trying to understand a system deeply, which is a pre-requisite for modeling it well, since the impression is that system behaviour need not be 'understood' to model it - modern statistical models can 'learn' anything. Thus, future courses like this one, will need to trade off the teaching of some conventional topics in stochastic models, against covering statistical learning models.

REFERENCES

[1] H. Akimaru and K. Kawashima. *Teletraffic: theory and applications*. Telecommunication networks and computer systems.

...Performance Measurement of a Web Application (Contd)

Plots of performance metrics We will study the behaviour of performance metrics by mainly varying the request arrival rate for open load, and the number of users, for closed load.
For generating sensible plots, you should generate "points" corresponding to a very low CPU utilization (5%) and go smoothly up to 100% system utilization. At least 10 points evenly spaced points should be generated. This means you have to do at least ten "runs" with varying load (request arrival rate or number of users). Ensure that the points are evenly spaced out - one hint is that your throughput curve should be such that it increases and then you can see it flattening out (or starts dropping). Most of the graph should have points before it flattens out, but there should be enough points for us to see throughput flattening/dropping.
Open Load
Collect data to plot the following curves. In each of the following curves the metric (the first quantity) should be on Y and the second quantity should be on X
• Response time vs request arrival rate • Throughput vs request arrival rate • Server CPU utilization vs request arrival rate • Fraction of requests dropped (basically, unsuccessful) vs request arrival rate
Based on the plots, make the following observations and calculations

- For each plot comment on its "nature" (Increasing? Decreasing? Linear? Sub-linear? Exponential? What is the min? Is it flattening out? Increasing then dropping? Etc...).
- Explain intuitively why you see this type of plot (E.g. why is it increasing?). The explanation in most cases will be obvious and can be brief. In some cases if you have no explanation simply state so.
- What is the maximum load in terms of request rate supported by this server?
- Is your utilization graph roughly linear? Use it (with Utilization Law) to estimate the service time of the request on the Web server CPU. Remember to account for multiple CPUs.
- Does your response time vs load curve have a shape in which there is graceful increase initially, and then a sharp increase? At what point approximately does this sharp increase occur? Find this point in terms of request rate and utilization.
- Use Little's Law to estimate the average number of requests at the server. (For bonus points, explore whether there is a way to verify this number by direct measurement)

Figure 8: ...(Contd)Performance Measurement Assignment for Reinforcing Queuing Theory Principles

Springer-Verlag, 1993.
[2] V. Apte. Performance analysis of computer systems and networks. Lecture Notes for CS 681, CSE Department, IIT Bombay.
[3] D. Bertsekas and R. Gallager. *Data Networks (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.
[4] M. Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, New York, NY, USA, 1st edition, 2013.
[5] D. A. Menasce, V. A. Almeida, L. W. Dowdy, and L. D owdy. *Performance by design: computer capacity planning by example*. Prentice Hall Professional, 2004.
[6] K. S. Trivedi. *Probability & Statistics with reliability, queuing and computer scienc e applications*. John Wiley & Sons, 2008.