Performance Modeling of a Power Management/Control System

Reda Ammar, Howard Sholl, and Ahmed Mohamed
University of Connecticut
Computer Science and Engineering Department
191 Auditorium Road, Storrs, CT 06269, USA
Tel. (860) 486-5963
Fax. (860) 486-1273

Abstract: *This paper develops a performance model for a class of soft real time systems. The incentive came from a specific commercial system; however the resulting model can have other applications. The model incorporates a set of properties that are not all common in prior models such as a queueing network with a polled serving pattern, programmable priority levels, multiple message classes, network-load-dependent multiple service, and a subset of queues with direct feedback. The analytical model can estimate performance parameters for both the overall queuing network and its specific queues. The model can be used to aid design decisions at an early stage, and can be used for later system calibration and diagnosis. Both the performance model and a sample application are described.*

Keywords: *Queueing Networks, Polling Systems, Priority Queues and Performance Modeling*

Received: September 21, 2003 | ***Revised:*** May 03, 2004 | ***Accepted:*** April 30, 2004

.

1. Introduction and Background

It is important to control the design structure and operating parameters of the system such that its response time for queries and problem reports is within acceptable limits. The queuing network involves a prioritized combination of flow-through query requests, and circulating requests (circulating requests are re-entered into their specific queues upon the initiation of service). The model required for flow analysis of the queuing network was unusual because of the combination of system properties: multiple priorities; re-circulating message subsets; programmable cyclic queue service polling pattern (called "program" below) to implement the priorities; multiple, queuing-network-load-dependent, class service.

Below we describe the development of a flow analysis model and solution for this application. The model developed incorporated all of these behavioral properties, allowing estimated execution time performance to occur. The solution can assist the designers in assuring that the system will perform adequately.

Background

Basic queuing analysis has been developed over many years, and is described in many textbooks [e.g., 1,2,5,7,15,17]. Computer system modeling can often be characterized/approximated as a product form queuing network [4,5,8,9,10]. Reference [11] contains a survey of this field. In our case, however, the cyclic service program, as a means to implement priorities, is effectively a polling system. Queuing analysis of polling systems has also been widely studied and reported [6,12,13,16]. Reference [14] contains a survey of earlier work in this area. Our application can fall into the area of "limited service, infinite queue polling systems," [3]. However, unlike this prior work, our requirements include a network-load-dependent, and conditional multiple service. We have not found earlier work, which contains the set of properties inherent in this application.

2. System description

The application system is a software-based electrical power monitoring/control system. The design, based on the client-server model, consists of the following components (mostly software).

- Power Metering and Protection Devices, which report data/statistics about, and control, the power at selected monitoring/control points in the system.
- The Network layer, which connects the devices to a single central system.
- System Server, which sets up and administrates a specified system.
- Device Objects, which translate user and system requests into a specific device requests.
- Protocol Objects, which enqueue and service the user and system requests to devices.

The Protocol Objects implement multi-level prioritization schemes. Every request made to a Protocol Object carries two types of priorities: the primary one is the priority of the requested data type,

and the secondary one is the priority of requested device object. To accommodate the prioritization demands, the Protocol Objects each implement two types of queues as shown in Figure 1. One type is used for one-time requests and the other type for circulating requests (reentered in the queue after service). Each queue set consists of *G* groups of queues, in which a separate group is used for each primary priority level. Each group consists of *R* request queue pairs, where a separate pair is used for each secondary priority level. Service priority is achieved through a program cycle, which specifies an order of queue service and allows individual queues to be serviced a multiple number of times/places in a cycle. One member of the pair, which is serviced before the other, handles one-time *requests* for a device. The second member of the pair is used only for *circulating requests*.

Figure 1 shows a sample protocol queue set. There are three sets of three queue pairs (G = 3, R = 3). Requested devices are activated through a Port/Device server object. Device activations are made in succession (by message) as long as the next requested device is idle, without waiting for device completion. This provides for conditional *multiple server* behavior. A typical system can have a multiplicity of protocol queue sets, each with attached Port/Device server objects and a unique set of devices. Communications between Port/Server objects and devices varies with each application, and could be independent or shared (e.g., Ethernet).

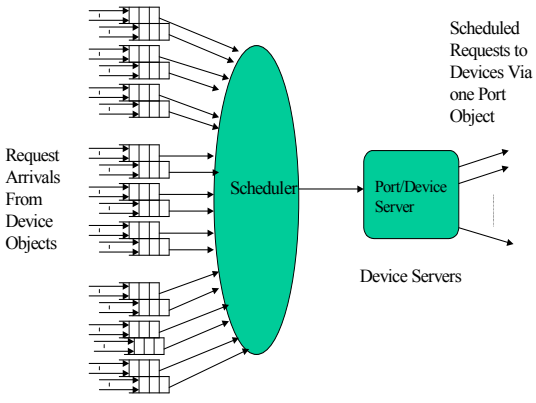

Figure 1

3. Description of System Model

3.1 Port/Device Server Analysis

Every T time units, a thread is activated, which scans through the multiplicity of Ports in round robin fashion and initiates all the next programmed priority requests for which the requested devices are successively available (idle). There is no implicit waiting for a device to respond. Thus the Ports act as multiserver stages, but have no internal queuing, forcing all request waiting to occur within the Protocol Queue set. A Port is connected to a unique subset of the overall Device set, and all requests for a specific device are routed by the system to its unique Protocol Queue set and Port.
The Port Object is modeled as a unity length buffer to hold the message for transmission to a message-specified device. See Figure 2. The total input Read/Write request rate to each Port Object buffer must match the input rate to the Protocol Object queues.

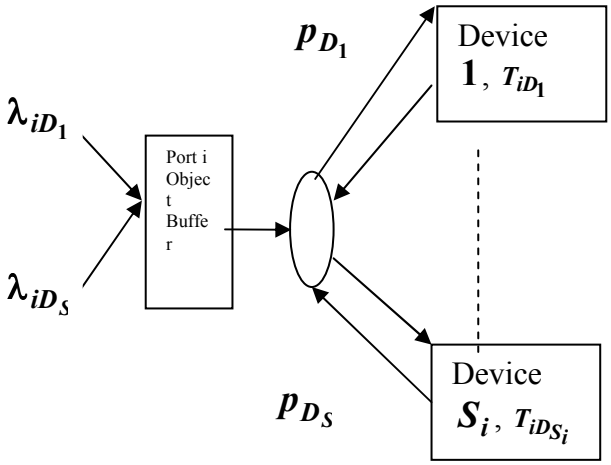

Figure 2

In the above figure, we associate D_i devices with Port ***i***, and assume that we have lumped all arrival rates to each of these devices. Thus λ_{ij} contains the entire arrival rate from Protocol Object i to Device j which is connected to Port i. Each such input rate consists of two major components:

$$\lambda_{iDj} = \lambda_{iADj} + \lambda_{iBDj} \ ,$$

where the ***A*** component represents the effective arrival rate coming from the circulating Request Queues; and the ***B*** component represents the total arrival rate coming from the Read/Write Request Queues, all from Protocol Queue set ***i***. λ_{iBD_j} is defined a-priori, and

$$\lambda_{iAD_j} = \frac{c_{ij}(no.of.active.A.queues.for.D_j)}{T_{cycle_i}}$$

which cannot be calculated numerically until the program cycle time, T_{cycle_i}, is determined for protocol Queue set ***i***. Since the cycle time is a function of the queuing network loading, and the λ_{iAD_j} depends upon

the cycle time, the effective service time has *load-dependence*. Also, c_{ij} is the number of programmed services for queue ij in one cycle.

Let the total input request rate be $\lambda_{T_i} = \sum_{j=1}^{S_i} \lambda_{iDj}$.

The probability of each device j service request is $p_{Dj} = \frac{\lambda_{iDj}}{\lambda_{T_i}}$, since all devices may be simultaneously busy. The behavioral model assumed is that if the next request in the priority program requests a device which is busy, the worker thread moves to the next Protocol Queue set and related Port in a round robin fashion. Each device has a utilization, ρ_{iD_j} (the jth device connected to Port i).

$\rho_{iD_j} = \lambda_{iD_j} T_{iD_j}$,

where T_{iD_j} is assumed to be deterministic.

Since the devices represent parallel servers, albeit for specific subsets of requests, there are instances when a specific request must be delayed. Since multiple queues in the Protocol queue set may be delivering requests to the same device, we do not know what residual time yet exists for the prior, but still active, device.

There is no effective way of determining this. In these cases we make the "memoryless" assumption, which is that the entire device time will yet occur. While this suggests that the device times have an exponential behavior/distribution, it also makes a "worst-case" assumption.

Using the above ideas we can now estimate the overall average Port/Device_set service time, $E(T_{iD})$. We note that when the requested device is available, the service time contribution (with respect to the Protocol Queue) is null, since the thread can immediately go to the next request, which may need an available device. Thus any effective delay/service time consists of waiting for a busy device to complete. Therefore we can average over just those possible states in which the devices are busy.

$$E(T_{iD}) = \sum_{j=1}^{S_i} p_{iD_j} T_{iD_j} \rho_{iD_j} ,$$

where T_{iD_j} is the expected time for device function D_j.

Essentially, this equation says that the only times that a service delay exists are when a device is requested (p_{iD_j}) and that device is already busy (ρ_{iD_j}).

Finally, the utilization of the Device Set (Port object), $\rho_{iD} = \lambda_{T_i} E(T_{iD})$.

3.2 Switched Behavior

The thread which activates Protocol Queue sets and related Ports is activated every T_A time units. Since this time interval is short relative to the typical device service times, it is likely that for each activation only a subset of Ports will have become idle. There is some wasted time in service when a device has become available, but the port must yet wait for a thread activation.

Here we estimate this performance effect and apply it uniformly to all queues in the queue set. Consider Protocol Queue Set i. We can estimate the expected the Port/Device_Set i service time, $E(T_{iD})$.

We notice that a device is activated initially, and then the queue_set/port is iteratively checked every T_A time units. Eventually, the port/device becomes available and is re-activated. In this cycle of re-activation, we note that there is a busy period, representing one service, followed by an idle period. We now analyze the average behavior of this effect.

$\left\lfloor \frac{E(T_{iD})}{T_A} \right\rfloor = n_i$, the expected number of times that T_A can fully occur in $E(T_{iD})$.

Define $f_i = \frac{E(T_{iD})}{E(T_{iD}) + E(T_{idle})}$ as the fraction of time the queue service is busy, and

$1 - f_i$ as the fraction of time the queue service is idle.

We note that $E(T_{idle}) = (n_i + 1)T_A - E(T_{iD})$, from the above sketch.

Thus, $f_i = \frac{E(T_{iD})}{E(T_{iD}) + (n_i + 1)T_A - E(T_{iD})} = \frac{E(T_{iD})}{(n_i + 1)T_A}$

And the resulting effective service time,

$$E'(T_{iD}) = \frac{E(T_{iD})}{f_i} = T_A(n_i + 1) .$$

3.3 Protocol Queue Analysis

The overall model is shown in Figure 4. A continuous flow of requests arrives from the device objects, and arrives at a set of assigned cyclic, priority Protocol_Queue sets. These are each served via a pre-defined cycle of accesses. The response data is placed on a response queue, which is served by another worker thread.

Basic Queuing Cycle:

For each Protocol Queue Set i, there are $2GR$ queue pairs, grouped as G sets of R pairs each: Q_{jk}, where j,k represent the group and subgroup indices for each pair, $1 \le j \le G, 1 \le k \le R$.

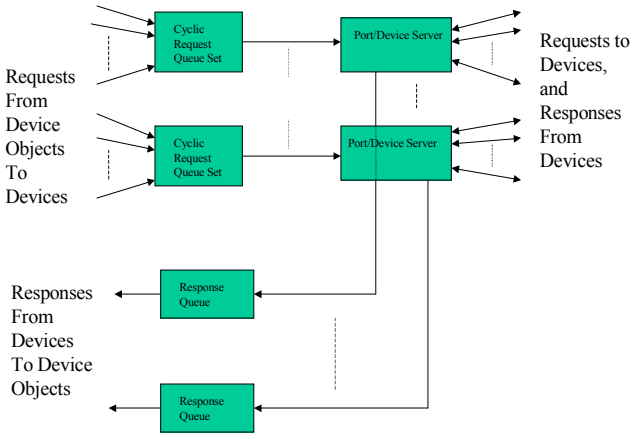

Figure 3

We further identify the members of a pair as Q_{jkB} for the immediate requests and $\boldsymbol{Q_{jkA}}$ for the cyclic requests. Serving within a queue-pair occurs alternately (e.g., for each service of queue A there is a service of queue B). Each Subgroup has a *polling cycle*, $j_1, j_{2,\ldots,} j_R$ (sequence of individual queue services) which describes the polled cyclic order of activations/service within a Group. Similarly, each Group also has a *polling cycle.* For example, for $G = R = 3$, a Group polling cycle of (112 112 113) and a Subgroup polling cycle of (112 112 113) result in a total cycle of 100 queue pair visits (or 200 queue visits.

Cyclic Service Behavior, on a per-queue basis:

In this section, we determine how the cyclic behavior influences a specific queue service time. We will assume that we are referring to Queue Set i, and will drop the i subscript for notational simplicity. Let:

$E(T_{jkD})'$ = the expected service time over the various request/data types for Q_{jk} (e,g, in Queue Set $\boldsymbol{i}$), *given that the queue is program-selected.*

where $E(T_{jkD})' = \dfrac{E(T_{jkD})}{f_i}$,

f_i = the fraction of time the thread is busy (see Section 2),

$E(T_{jkD})$ = the expected service time for Q_{jk}, *if the queue were continuously active, and also program-selected.*

We now need to determine the expected service time under these circumstances, and also in the context of program selection ($E(T_{jkD})''$).We first will estimate the time over the specific *request probabilities*, and their *device utilizations* for the referenced queues

$E(T_{jkD}) \approx t_S + \sum_{r=1}^{M_{jk}} p_{D_r} \rho_{D_r} T_{D_r}$, where

M_{jk} = the quantity of active request (device) types in the queue.

$$p_{D_r} = \frac{\lambda_{D_r}}{\lambda_{T_{jk}}}$$

$$p_{D_r} = \frac{m_r}{M_{jk}}$$

m_r = the number of device r requests on the circulating queue

$\lambda_{T_{jk}} = \sum_{r=1}^{M_{jk}} \lambda_{D_r}$, the total input arrival rate of read/write requests

ρ_{D_r} = the utilization of the requested device r.

T_{D_r} = the time cost of the device $\boldsymbol{r}$ for the requested function.

t_S = queue service software time cost

The logic for this approach is that a request at the head of Q_{jk} would need to wait for a *same* device to become available as that needed for the request. Thus we can average over just the devices represented by the specific set of requests at Q_{jk}, again invoking the "memoryless" approximation.

The complication is now that each queue may be delayed from being re-serviced by the polling program. Therefore we must consider the behavior of the re-servicing cycle in order to determine the actual re-service time. This requires us to examine all possible re-visit sequences before we can determine the actual queue cyclic service time.

Let Δ_{jk} = the expected inter-service delay, given the condition that Q_{jk} has just been serviced.

This means that $E(T_{jkD})'' = \Delta_{jk}$, where $E(T_{jkD})''$ = the actual expected service time under cyclic conditions. The entirety of the time until the next service depends

upon how many device waits occur in the intervening program steps.

$\rho_{jk}" = \lambda_{T_{jk}} E(T_{jkD})" = \lambda_{T_{jk}} \Delta_{jk}$, for a non-circulating queue.

Note that for a circulating queue, the utilization is either 1, or 0, depending on whether it has any customers. Thus we do not need to do this type of calculation for a circulating queue. However, we *do* need to estimate the time delay caused by circulating queue service. We note that this equation shows that each stand-alone utilization for a read/write queue will increase incrementally because of the polling process, as we would expect. To determine the inter-service delay, Δ_{jk}, we need to average over all possible poll contexts in which Q_{jk} executes.

$\Delta_{jk} = \frac{1}{C_{jk}} \sum_{n=1}^{C_{jk}} \Delta_{jk_n}$, where *n* denotes each successive member of the set of cycle positions for the queue, and C_{jk} is the number of poll visits to Q_{jk}.. This expression simply averages the interposition program delay over the set of cyclic positions. We note that:

$$\Delta_{jk_n} = \sum_{m=1}^{R_n} \rho_m" E(T_{mD})',$$

where the summation is over the interval between position *n* and the next sequential position in the program cycle, when the same queue is re-serviced. R_n represents the quantity of sequential queue positions in the intervening interval queue positions for occurrence n, and m represents the identity of the intervening queue positions

For each position there is a specific interval service sequence of length R_n, for which the time interval, Δ_{jk_n} must be specified as shown above. Therefore, we must individually specify the appropriate $\sum_{m-1}^{R_n} \rho_m" E(T_{jkD})'$ for each of these cycles.

By algebraic manipulation:

$$T_{cycle} = C_{jk} E(T_{jkD})" = C_{jk} \Delta_{jk} = \sum_{m=1}^{C_{jk}} \Delta_{jk_m} = \sum_{m=1}^{C_{jk}} \sum_{n=1}^{R_m} \rho_m" E(T_{nD})'$$

Now by separating the circulating queue terms from the non-circulating queue terms, and by substituting for $\rho_m"$ ($\rho_m" = \lambda_{T_m} E(T_{mD})" = \frac{\lambda_{T_m} T_{cycle}}{C_m}$), we can write:

$$T_{cycle} = \sum_{m=1}^{C_{jk}} \left\{ \sum_{n \in A} \frac{\lambda_{T_m} T_{cycle} E(T_{mD})'}{C_m} + \sum_{n \in B} E(T_{md})' \right\}$$

$$T_{cycle} = T_{cycle} \sum_{m=1}^{C_{jk}} \sum_{n \in A} \frac{\lambda_{T_m} E(T_{mD})'}{C_m} + \sum_{m=1}^{C_{jk}} \sum_{n \in B} E(T_{mD})'$$

$$T_{cycle} = \frac{\sum_{m=1}^{C_{jk}} \sum_{n \in B} E(T_{mD})'}{1 - \sum_{m=1}^{C_{jk}} \sum_{n \in A} \frac{\lambda_{T_m} E(T_{mD})'}{C_m}}.$$

This final expression can now be used to estimate the cycle time.

We have now shown how all the variables in the system are interrelated. Finally, to get a numerical solution, one can initially estimate a numerical value for T_{cycle}, and then progress through the above solution stages, as described. Upon completion, one obtains another T_{cycle} value, which can be used as the basis for another round of calculation. When the starting and final values for T_{cycle} are near-equivalent, the solution has stabilized, and the system evaluation can be done:

Given this solution, we can calculate, for each queue and for the overall system, a set of properties (e.g.,):

- The queue utilization, $\rho_{jk}"$, $\rho_{jk}" = \frac{\lambda_{T_{jk}} T_{cycle}}{C_{jk}}$
- The cyclic service time, $E(T_{jkD})"$
- The estimated queue delay (M/M/1 estimate), $T_{delay_{jk}} = \frac{E(T_{jkD})"}{1 - \rho_{jk}"}$
- The average number of requests waiting, $N_{jk} = \lambda_{jk} T_{delay_{jk}} - \rho_{jk}"$
- Average Overall Requests Waiting, $\sum_{j=1}^{3} \sum_{k=1}^{3} N_{jk}$ for a immediate queue

Also, we can check the status of each queue for stability (since it is possible in this type of system to inadvertently create instability for one of the queues). For each Q_{jk}, $\rho_{ij}" < 1$.

For each Protocol Queue set and related Port, there is a **Response Queue**, whose function is to hold the response data which has been received from devices in response to earlier requests. The behavior of the response queue can be handled by conventional single queue analysis, and is not further considered here.

4. Example

System Description: Three circulating queues, two immediate queues and four devices.
$T_{D1} = 1.4$, $T_{D2} = 0.9$, $T_{D3} = 1.5$, $T_{D4} = 1$.
For immediate queues:
$\lambda_{1D1} = 0.15$, $\lambda_{1D2} = 0$, $\lambda_{1D3} = 0$, $\lambda_{1D4} = 0.03$
$\lambda_{2D1} = 0$, $\lambda_{2D2} = 0.05$, $\lambda_{2D3} = 0$, $\lambda_{2D4} = 0.01$
For circulating queues:
Q1: requests to devices 1, 2 and 3.
Q2: requests to devices 1 and 3.
Q3: requests to devices 2, 3 and 4.
Number of times queues get service per cycle:
Q1: $C_1 = 6$, Q2: $C_2 = 3$, Q3: $C_3 = 1$. $T_A = 0.065$.

Estimating the cycle time
First assume the devices utilizations is 0.5 for all devices. Then for each queue calculate the expected service time at the port using the equation: $E(t_{iD}) = \sum p_D * \rho_D * T_D$.
Circulating queues
$E(t_{11}) = 0.33 * 0.5 * 1.4 + 0.33 * 0.5 * 0.9 + 0.33 * 0.5 * 1.5 = 0.633$
$E(t_{12}) = 0.5 * 0.5 * 1.4 + 0.5 * 0.5 * 1 = 0.6$
$E(t_{13}) = 0.33 * 0.5 * 0.9 + 0.33 * 0.5 * 1.5 + 0.33 * 0.5 * 1 = 0.5667$
Immediate queues
$E(t_{21}) = 0.8333 * 0.5 * 1.4 + 0.1667 * 0.5 * 1 = 0.6666$
$E(t_{22}) = 0.8333 * 0.5 * 0.9 + 0.1667 * 0.5 * 1 = 0.4583$
For each queue find how many times that T_A can fully occur in E(t). $n = E(t) / T_A$
$N_{11} = 9$, $N_{12} = 9$, $N_{13} = 8$
$N_{21} = 10$, $N_{22} = 7$
For each queue calculate the new expected service $E(t)` = T_A * (n + 1)$.
$E(t_{11})' = 0.65$, $E(t_{12})' = 0.65$, $E(t_{13})' = 0.585$
$E(t_{21})' = 0.715$, $E(t_{22})' = 0.52$

Calculate the value for $T_{cycle} = \frac{\sum_{i=1}^{m} Ci * E(Ti)'}{1 - \sum_{j=1}^{k} \lambda_{Tj} E(T_j)'}$, m is the number of active circulating queues and k is the number of active immediate queues.

$$T_{cycle} = \frac{6 * 0.65 + 3 * 0.65 + 1 * 0.585}{1 - (0.18 * 0.715 + 0.06 * 0.52)} = 7.6598$$

For each queue calculate the cyclic service time
$E(T_{ij})'' = T_{cycle} / C_i$.
$E(T_{11})'' = 7.6598 / 6 = 1.2766$, $\lambda_{11} = 1/ E(T_{11})'' = 0.78331$
$E(T_{12})'' = 7.6598 / 3 = 2.5533$, $\lambda_{12} = 1/ E(T_{11})'' = 0.39166$
$E(T_{13})'' = 7.6598 / 1 = 7.6598$, $\lambda_{13} = 1/ E(T_{11})'' = 0.13055$
$E(T_{21})'' = 7.6598 / 6 = 1.2766$,
$E(T_{22})'' = 7.6598 / 3 = 2.5533$
Calculate the new devices utilization.
$\lambda_{D1} = 0.33 * 0.78331 + 0.5 * 0.39166 + 0.15 = 0.60693$
$\rho_{D1} = T_{D1} * \lambda_{D1} = 1.4 * 0.60693 = 0.8497$
$\lambda_{D2} = 0.33 * 0.78331 + 0.33 * 0.139055 + 0.05 = 0.3546$
$\rho_{D2} = T_{D2} * \lambda_{D2} = 0.9 * 0.3546 = 0.3192$
$\lambda_{D3} = 0.33 * 0.78331 + 0.33 * 0.139055 = 0.3046$
$\rho_{D3} = T_{D3} * \lambda_{D3} = 1.5 * 0.3546 = 0.4564$
$\lambda_{D4} = 0.5 * 0.39165 + 0.33 * 0.139055 + 0.04 = 0.2793$
$\rho_{D3} = T_{D3} * \lambda_{D3} = 1 * 0.2793 = 0.2793$
Using the new devices utilization we'll repeat the same above steps again. It'll take four iterations to finally have the final solution. $T_{cycle} = 8.4783$.

5. Conclusion

In this paper we develop a performance model for a soft real time application. The model incorporates a number of properties that not all common in other available models such as a complex queueing system, multiple priority levels, different types of requests, load-dependent conditional multiple service, and a polled service pattern. Our analytical model, which requires an iterative numerical solution, can estimate performance parameters for both the system and specific queues. For the system we can estimate the system utilization, the average cycle time and the average number of requests in the system. Also, for each queue we can estimate the cyclic utilization, the average cyclic response time and the average number of requests. In this manner, the model can predict the bottlenecks in the system and help users investigate and avoid unstable running modes. Both the performance model and a sample application are described.

References

1. F. Baccelli and P. Bremaud, "*Elements of Queueing Theory*", Springer-Verlag,, 2002, ISBN 3-540-66088-7.
2. L. Breuer, " *From Markov Jump Processes to Spatial Queues*", Kluwer, 2003.
3. Gianini, Mansfield, "*Cyclic Multiqueue Systems With Two Priority Classes and Exhaustive Service* in *Data Communicatiuon Systems and Their Performance"* Elsevier North- Holland Amsterdam, pp 511-526, and *Perform Eval* 8, 2 (Apr.) , pp93-115
4. K. Chandy, C. Sauer, "Computational Algorithms for Product Form Queueing Networks", *ACM Journal of Communication*, Vol 123, No. 10, pp:573-583, 1980.
5. X. Chao, M. Miyazawa and M. Pinedo, "*Queueing Networks: Customers, Signals and*

Product Form Solution", Wiley, 1999, ISBN 0-471-98309-8.

6. W. Groenerdijk and H. Levy, "Performance Analysis of Transaction Driven Computer Systems Via Queueing Analysis of Polling Models", *IEEE Transaction on Computers*, Vol. 41, No. 4, pp: 455-466, April 1992.
7. E. Lazowska, G. Graham, J. Zahorjan, "*Quantitative System Performance*",Prentice-Hall, 1984.
8. A. Mohamed, L. Lipsky and R. Ammar, "Performance Modeling of parallel and distributed Systems with Finite workloads", To appear in the *International Journal of Performance Evaluation*, 2004.
9. A. Mohamed, L. Lipsky and R. Ammar, "Transient Model for Jackson Networks and its Approximation", *7th International Conference on Principles of Distributed Systems*, Dec. 2003.
10. A. Mohamed, L. Lipsky and R. Ammar, "Performance Model for a Cluster of Workstations." *The 4th International Conference on Communications in Computing (CIC 2003*). Las Vegas, NV, Jun. 2003.[11] Nelson, Randolph, "The Mathematics of Product Form Queuing Networks," *ACM Computing Surveys*, September, 1993.
12. E. Silva, H. Gail and R Muntz, "Polling Systems with Server Timeouts", *IEEE/ACM Transaction of Networking*, Vol 3, No. 5, pp: 50-575- 1995.
13. P. Tran-Gia, "Analysis of Polling Systems with General Input Process and Finite Capacity", *IEEE Transaction on Communication*, Vol. 40, No. 2, pp: 337-344, 1992.
14. Takagi, Hideaki, "Queuing Analysis of Polling Models*," ACM Computing Surveys*, March, 1988.
15. K. Trivedi, "*Probability & Statistics with Reliability, Queuing and Computer Science Applications*", Prentice-Hall, 1982.
16. M. Reiman and L Wein, "Heavy Traffic Analysis of Polling Systems in Tandem", *Operational Research*, 1998.
17. P. Robert, " *Stochastic Networks and Queues*", Springer-Verlag, 2003, ISBN 3-540-00657-5.

Reda A. Ammar received the B.S. degree in 1973 in Electrical Engineering from Cairo University, Egypt, the B.S. degree in mathematics in 1975 from Ain-Shames University, Egypt, and the M.S. and Ph.D. degrees in 1981 and 1983 from the University of Connecticut, Storrs.

Currently he is a professor and the department head of the Computer Science and Engineering department, University of Connecticut. His primary research interests are in the area of software performance engineering, real-time systems, and distributed and parallel computing. His publication record exceeds 185 journal and conference papers. Dr. Ammar served many conferences as a chair and a member of the steering committees. He served the International Society of Computers and Their Applications as the vice president, the conference coordinator and a member of the board of directors, and served the IEEE as the chairman of the IEEE technical committee on Simulation. He is currently the editor-in-chief of the International Journal on Computers and Their Applications.

Howard Sholl is an Emeritus Professor of Computer Science and Engineering at the University of Connecticut, where ha has been on the faculties of Computer Science and Engineering, and (earlier) Electrical Engineering, for 38 years.

He also, served as a director of the Taylor L. Booth Center for Computer Applications and Research for 11 years. He is the founding Editor for the International Journal for Computers and Their Applications (IJCA), and served as its Editor-in-Chief for approximately eight years. He is currently serving as the President of Society for Computers and Their Applications (ISCA). He has been a Fulbright senior research professor at the Technical University in Munich, and a Levelhulme professor at the University of Edinburgh. His primary research interests are distributed /parallel computing, real-time systems and performance analysis.

Ahmed Mohamed, is currently a Ph.D. candidate at the Computer Science and Engineering Department, University of Connecticut.

He received the B.Sc. degree in Electrical and Computer Engineering from Assiut university, Egypt, in 1994, and the M.Sc. degree in Computer Science and Engineering from University of Connecticut, USA in 2001. He is a member of Society for Computers and Their Applications (ISCA). His research interests include, performance modeling, queueing analysis, distributed/parallel computing and reliability analysis.