Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

SIMULATING THE PERFORMANCE OF A CLASS-BASED WEIGHTED FAIR QUEUEING SYSTEM

Martin J. Fischer, PhD

3150 Fairview Park Drive, South
Noblis, Inc.
Falls Church, VA 22042-4519, USA

Denise M. Bevilacqua Masi, PhD

3150 Fairview Park Drive, South
Noblis, Inc.
Falls Church, VA 22042-4519, USA

John F. Shortle, PhD

4400 University Drive
George Mason University
Fairfax, VA 22030-4444, USA

ABSTRACT

Class Based Weighted Fair Queueing (CBWFQ) is a very important router discipline that allows different types of Internet Protocol (IP) traffic like voice, video, and best effort data to receive the required quality of service measures they individually need. CBWFQ dynamically allocates the available bandwidth to each traffic class based on the class's weight. This discipline is playing a vital role as IP brings these traffic classes together in a truly converged network. Under stress and in extreme emergencies, it is critical to be able to determine how the CBWFQ discipline will perform. In this paper, we present and discuss the critical role simulation has played in our development of performance analysis tools for the CBWFQ discipline.

1 INTRODUCTION

Industry is moving away from circuit-switched technology to Internet Protocol (IP) technology for telecommunications applications, including voice traffic. When IP networks become heavily loaded, packets get dropped and other quality of service (QoS) measures like packet latency, loss, and jitter are significantly degraded. At some point, the QoS for voice and video packets becomes poor enough that their communication is lost. As Voice over IP (VoIP) and video become more prevalent in these networks, these issues become increasingly important—particularly, since there is no dedicated communication path for a voice or video call.

The Class Based Weighted Fair Queueing (CBWFQ) discipline is being used frequently within the Internet with these potential QoS problems for the multiple traffic classes (see Cisco Systems Quality of Service Solutions Configuration Guide, and Semeria (2001)). Under CBWFQ, the bandwidth is shared in accordance with the weights assigned to the CBWFQ traffic classes. These weights are designed to ensure the traffic class gets a certain portion of the available bandwidth. CBWFQ extends weighted fair queueing (WFQ) by assigning weights to classes of traffic rather than individual flows of traffic identified by origination/destination pairs.

The CBWFQ discipline is very important to understand as voice, data, and video converge and compete for use of the same shared transmission resources. Each of these traffic types has its own required QoS measures to be met; CBWFQ is one discipline that may be able to meet these QoS measures. There are several methods that can be used to evaluate the performance of such a system. These range from a direct simulation of the system to the development of the system equations and their solution via numerical or analytical methods. In this paper, we discuss the crucial role simulation has played in our development of performance analysis tools for CBWFQ.

In order to better present this, we consider a simple CBWFQ base system. This system has two traffic classes with Poisson packet arrivals for each class and exponential packet lengths and finite buffers for each class (see Section 2). There we define the parameters for this system and develop the steady state equations for the state probabilities. The simulation model we developed is presented and discussed in Section 3. Other methods we used in the development of performance tools are presented in Section 4. In both Sections 3 and 4, illustrative examples are used to solidify the points we are making. Section 5 contains some closing remarks.

2 BASE SYSTEM DESCRIPTION

The base CBWFQ system has two classes of packets. We assume that the arrival process for each class is an independent Poisson process and that the service times are exponentially distributed. In addition, we assume each class has its own buffer.

In the CBWFQ system, there is no service interruption of packets by another. Upon service completion of a packet, the next type of packet to be served is randomly chosen based on the class weights. Specifically, in the two-class system, when both classes are present, a class 1 packet is chosen with probability α and a class 2 packet is chosen with probability $1-\alpha$. If only one class of packets is present, then that class is served. Within a class, service is on a First Come First Served (FCFS) basis. At the boundaries ($\alpha = 0$ or 1), this system is the standard non-preemptive queueing system that has been extensively studied in the past (see Morse (1958), Cohen (1969), Miller (1981), and Gross and Harris (1998)).

Fundamentally, the model described in this paper is a *random polling model*. After serving one queue, the next queue to serve is randomly chosen according to specified weights. There is a large body of research on polling models in general. However, most of the literature deals with *cyclic* polling models in which the server visits each queue in ordered succession, rather than in a random fashion. For example, a standard reference on polling models is the book by Takagi (1986) as well as a subsequent survey paper (Takagi 1988). Both of these works deal primarily with cyclic polling models.

Two papers that deal with random polling models and that are closely related to this paper are Kleinrock and Levy (1988) and Lee (1997). Kleinrock and Levy (1988) consider a discrete-time random polling model under several different service disciplines. For the limited-service discipline, where the server serves at most one customer at a time before switching queues, results are provided only for a completely symmetric system in which all queues have identical statistical properties. No results are given for the asymmetric case. For some other service disciplines – for example, where the server continues to serve customers until the queue is empty – they derive a set of m^2 linear equations that can be solved to obtain expected waits. Lee (1997) provides exact results for the asymmetric random polling model. However, we have not been able to match our simulation results with his analytical results.

Finally, a related concept is processor sharing in which customers from different queues are served simultaneously. For this system, the queues are served individually, one customer at a time. If we regard the customers as packets and we set the maximum packet size to be small, then the model in this paper can be approximated by appropriate processor sharing models—for example, the generalized processor sharing and its packet-by-packet variants (e.g., Parekh and Gallager 1993).

We define the following parameters and random variables.

- λ_i is the packet per second arrival rate for class i (i = 1,2).
- $1/\mu_i$ is the average time to serve a class i packet.
- α is the probability a class 1 packet is selected for service when both classes are waiting.
- Q_i is the random number of class i packets in the system.
- Y is a random variable identifying the class of packet in service (Y = 1,2) or that the system is empty (Y = 0).
- $P_{i,j,k} = \Pr\{Q_1 = i, Q_2 = j, Y = k\}$ in steady state (i = 0,1,2,…, j = 0,1,2,3…, Y = 0,1,2).
- K_i is the maximum number of class i packets allowed in system
- $\rho_i = \lambda_i / \mu_i$ is the class i load. For the infinite buffer case, we assume that $\rho = \rho_1 + \rho_2$ is less than one.

For a discussion of the appropriateness of these arrival and packet size distributions assumptions, see Cao (2002), Fischer (2005), Newman (2001), and Thompson (1997). The assumption of Poisson packet arrivals and exponential service times are the most lenient assumptions one could make and get analytic results. As we will see, even making these assumptions does not allow a simple analytic solution except for boundary cases and we are forced to go to simulation or numerical methods.

The steady state birth and death equations for this system for the infinite buffer ($K_1 = K_2 = \infty$) case are:

$$\begin{aligned}(\lambda_1+\lambda_2)P_{0,0,0} &= \mu_1 P_{1,0,1} + \mu_2 P_{0,1,2}\\ (\lambda_1+\lambda_2+\mu_2)P_{0,1,2} &= \lambda_2 P_{0,0,0} + \mu_1 P_{1,1,1} + \mu_2 P_{0,2,2}\\ (\lambda_1+\lambda_2+\mu_1)P_{1,0,1} &= \lambda_1 P_{0,0,0} + \mu_1 P_{2,0,1} + \mu_2 P_{1,1,2}\end{aligned} \tag{1}$$

and for i = 1,2 … and j = 1,2, …

$$(\lambda_1+\lambda_2+\mu_1)P_{i,j,1} = \lambda_1 P_{i-1,j,1} + \lambda_2 P_{i,j-1,1} + \alpha\,\mu_1\, P_{i+1,j,1} + \alpha\,\mu_2\, P_{i,j+1,2} \tag{2}$$

and

$$(\lambda_1+\lambda_2+\mu_2)P_{i,j,2} = \lambda_1 P_{i-1,j,2} + \lambda_2 P_{i,j-1,2} + (1-\alpha)\mu_1 P_{i+1,j,1} + (1-\alpha)\mu_2 P_{i,j+1,2} \tag{3}$$

When i = 1 and j ≥ 1 (the first expression in the right hand side of equation (2)), we have $P_{0,j,1} = 0$, and, similarly in equation (3), if j = 1 and i ≥ 1 (the second expression on the right hand side of equation (3)), we have $P_{i,0,2} = 0$.

$$\begin{aligned}&(\lambda_1+\lambda_2+\mu_2)P_{0,j,2} = \lambda_2 P_{0,j-1,2} + \mu_1 P_{1,j,1} + \mu_2 P_{0,j+1,2}\\ &\textit{and}\\ &P_{0,j,1} = 0\end{aligned} \tag{4}$$

For j = 0 and I = 1,2…

$$\begin{aligned}&(\lambda_1+\lambda_2+\mu_1)P_{i,0,1} = \lambda_1 P_{i-1,0,1} + \mu_1 P_{i+1,0,1} + \mu_2 P_{i,1,2}\\ &\textit{and}\\ &P_{i,0,2} = 0\end{aligned} \tag{5}$$

For these equations to have a solution we must have $\rho < 1$.

In general, we know of no solution to this system; although some results exist for special cases in steady state. For the case of $\alpha = 0$ or 1, we have a non-preemptive priority queueing system and Miller (1981) gives an expression for the probability of the number of high priority packets in the system. If $\alpha = 1$, then the class 1 packets have queue priority and $\Pr\{Q_1=n\}=P_n^1$ is given by

$$\Pr(Q_1 = n) = (1-\rho)\left(\frac{\lambda_1}{\mu_1}\right)^n + \frac{\lambda_2}{\lambda_1+\mu_2-\mu_1}\left[\left(\frac{\lambda_1}{\mu_1}\right)^n - \frac{\mu_1\lambda_1^n}{(\lambda_1+\mu_2)^{n+1}}\right] \quad (6)$$

When $\alpha = 0$, then the class 2 packets have priority and $\Pr\{Q_2=n\}$ is given by equation (6) with a suitable changes in the subscripts (this result only holds when $\mu 1 = \mu 2$). Also in steady state, for any α when the service rates of each class are equal then the probability of the total number of packets ($=Q_1+Q_2$) is given by standard M/M/1 results; that is

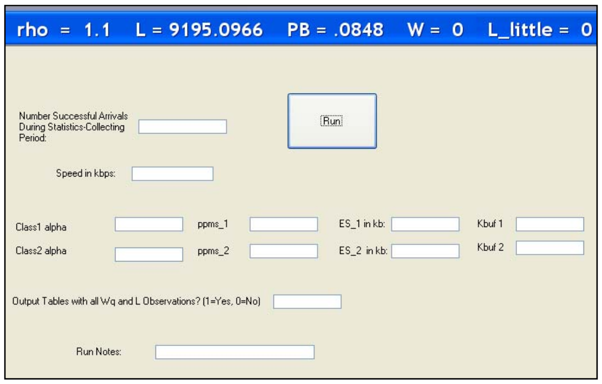
$$P_n = P\{(Q_1+Q_2)=n\} = (1-\rho)\rho^n, \textit{where } \rho = \rho_1+\rho_2 . \quad (7)$$

This result stems from the fact that it does not matter which customer is in service, the departure rate is the same. Also the server always works when customers are present and no work is lost.

3 SIMULATION MODEL

We were first interested in developing a simulation model for a Low Latency Queueing (LLQ) system (see Masi (2007a, 2007b). That system is composed of a Priority Queueing (PQ) module and a CBWFQ module. Because we understand the interaction between the priority queues and the class based weighted fair queues, and how to model that interaction, our focus in this modeling is within the CBWFQ portion of the router, which is not understood. We considered developing our own custom simulation versus using a simulation package such as OPNET Modeler. OPNET Modeler is a popular discrete-event simulation package which is designed specifically to model telecommunications networks. We decided to develop our own custom simulator of the CBWFQ system for several reasons, one of which was that we wanted to have a flexible simulator that could be modified to examine future system configurations. OPNET Modeler has limited ability to model router configurations that do not exist in current equipment without custom programming. In addition, details on the precise specifications of CBWFQ scheduling (both within OPNET and in real routers) were not readily available, and we wanted to have control over and knowledge of the precise mechanics of the CBWFQ algorithm in the simulation. Also, we anticipated that run times in OPNET Modeler would be longer than in our custom simulator, which has much less overhead. Later comparisons (see Masi et al. (2007a, 2007b)) showed that our custom simulator run times for a router using CBWFQ were about one tenth that of the OPNET run times.

We developed a simulation of the CBWFQ system in Visual Basic for Applications (VBA). The interface is shown in Figure 1. Development time for this model was relatively short, taking several days. Background research on the precise rules used by class based weighted fair queues to select which class of packet to transmit when the router is available was required and is described in Masi et al. (2007a, 2007b). The original simulator allowed three classes of traffic, but for this paper we use the two-class version, each having Poisson arrivals. Our original simulation allowed for a packet service time distribution that was Internet specific; three packet sizes are dominant in IP traffic (Thompson et al. (1997)). However, for this paper we modified the simulation to use exponential service times to enable comparisons with analytic models. The simulation computes several measures of effectiveness: latency (mean queue wait), mean number of packets in the system, and packet loss. The simulation can also output individual packet queue waiting times and number of packets at each departure point, enabling the entire distributions to be obtained if desired.



```
rho  =  1.1    L = 9195.0966    PB = .0848    W =  0    L_little =  0
```

Number Successful Arrivals During Statistics-Collecting Period:
Run
Speed in kbps:
Class1 alpha
ppms_1
ES_1 in kb:
Kbuf 1
Class2 alpha
ppms_2
ES_2 in kb:
Kbuf 2
Output Tables with all Wq and L Observations? (1=Yes, 0=No)
Run Notes:

Figure 1: VBA CBWFQ Simulator

We now consider some illustrative examples using the simulation model. Our input parameters are as follows. The class 1 packets have a mean packet size 1.6 kb, and the class 2 packets have a mean packet size of 5.57 kb. We assume the line speed is a T1 (1536 kbps). The code is written in VBA and the runs were made on a 2.4 GHz laptop. We ran the simulation for 3,000,000 packet arrivals. For these examples $\alpha = 0.77$, $\lambda_1 = 425$ pps and $\lambda_2 = 130$ pps which result in $\rho_1 = 0.443$ and $\rho_2 = 0.471$ or a total load of 0.914. In Figures 2, 3, and 4, we see that as the buffer size increases, the packet loss for each class is going to zero, and the expected number of each type of packet and the buffer latency by class are converging to the infinite buffer cases. These parameter values will also be used in the next section, where we show the results in tabular format in a comparison with other analysis methods.

Figure 5 shows the simulation run time. Each run took about 29 seconds to simulate 3 million packet arrivals. The run time did not vary much with the size of the buffers.

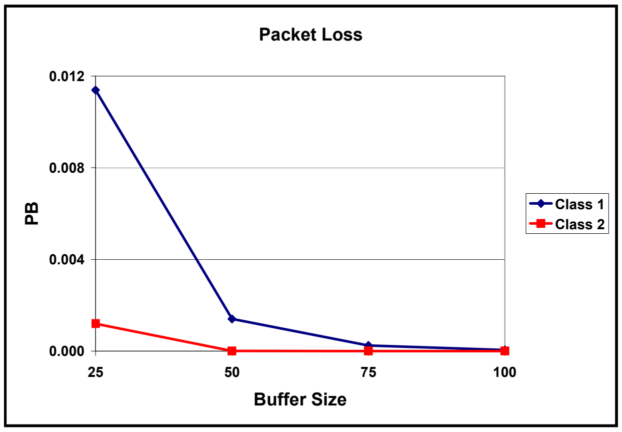


Figure 2: Simulation Packet Loss

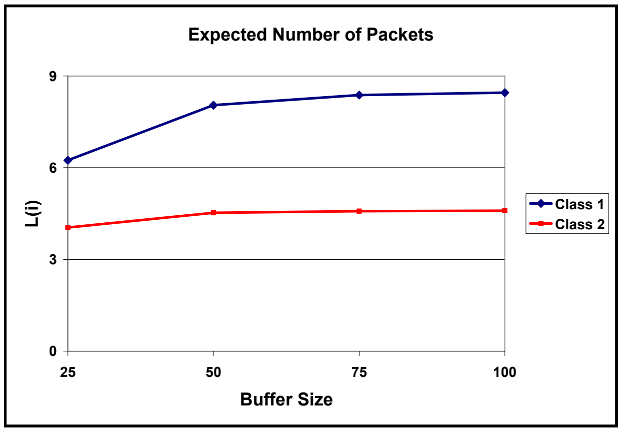


Figure 3: Simulation Expected Number of Packets in System

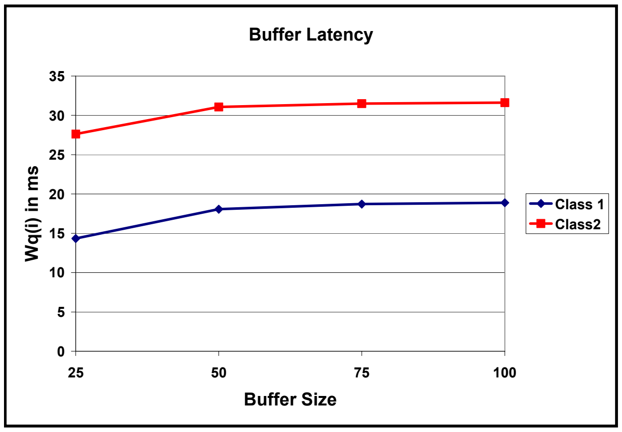


Figure 4: Simulation Buffer Latency

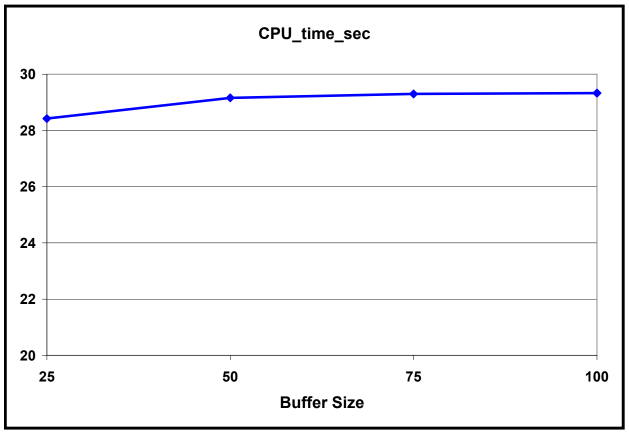


Figure 5: Simulation Run Time in Seconds

4 COMPARISON WITH OTHER METHODS

We also investigated other methods such as solving the infinite buffer steady state equations analytically using the standard generating function approach (see Gross and Harris (1998)) or solving the problem numerically.

In trying to solve the problem analytically, we met with extreme difficulty. We began with the simplest assumptions: two service classes, exponential service times, Poisson arrivals, and infinite buffers. The generating function approach has been successfully applied to analytically solve similar queues, such as the priority queue (e.g., Gross and Harris (1998) Section 3.4.1, or Morse (1958) Chapter 9), so we followed the approaches given in these references. The two-class priority queue is a special case of the two-class CBWFQ when the weights of one class or the other is zero.

For the two-class priority queue, the approach defines a two-dimensional generating function $H(z_1, z_2)$. By appropriate summation of the state equations, $H(z_1, z_2)$ can be obtained as a function of known parameters and an unknown one-dimensional generating function $P_{02}(z_2)$. Gross and Harris (1998) point out (for equal service rates) that the complete form for $P_{02}(z_2)$ is not needed to evaluate the partial derivative of $H(z_1, z_2)$ with respect to z_1—only the value $P_{02}(1)$ is needed and this can be easily obtained. Thus, it turns out that all standard first-moment performance measures for the priority queue can be obtained even without obtaining the exact functional form for $H(z_1, z_2)$.

However, for the CBWFQ, the analogous form of $H(z_1, z_2)$ involves *two* unknown generating functions, which we may call $P_{02}(z_2)$ and $Q_{01}(z_1)$. It turns out that taking the partial derivative of $H(z_1, z_2)$ with respect to either z_1 or z_2 requires knowing the complete form of at least one of these two functions. Thus, the mean-value performance measures cannot be obtained without knowing the complete form of these two functions.

Morse (1958) provides some guidance here. For the priority queue, he shows that the function $P_{02}(z_2)$ can be obtained by solving a first-order homogenous difference equation (easy) and then solving a second-order non-homogenous difference equation. The method works because the two difference equations can be decoupled, so one equation can be solved first and then used as an input into the second. However, for the CBWFQ queue, the result is a coupled system of two second-order non-homogenous difference equations, with driving terms involving unknown probabilities from the original queue. We did not make any serious attempts to solve this system.

Even if solvable, the approach has limitations in being restricted to the assumptions of two-service classes, exponential service times, infinite buffers, and load less than 1. All of these assumptions can be easily generalized using simulation.

Our next step was to solve the equations numerically. Some candidate approaches were the Matrix Geometric Method (see Miller (1981)), or the method presented by Cidon (1990); but they involved significant analytic set-up time to put the equation in a usable form. As an alternative, we used an iterative method to solve the steady-state balance. In this approach, we plug an initial guess for the $P_{i,j,k}$ values into the right-hand side of equations (1) – (5) to generate the next iteration values of $P_{i,j,k}$. As the process is repeated, the $P_{i,j,k}$ value converge to the steady-state probabilities. As an initial starting value of $P_{i,j,k}$ we set $P_{1,0,1}=1$.

Finally, we considered another numerical method. This numerical method also yields the transient probabilities. This method solves the system of difference differential equations for $P_{i,j,k}(t)$, the conditional probability at time t there are i class 1 and j class 2 packets in the system, and class k (=1,2) is in service, given the system state at 0. These equations can be easily developed using standard birth and death arguments. We use the method described in Knepley and Fischer (1977) to solve these equations. The code for each of these methods was written in VBA and all runs were made on the same computer.

Table 1 compares the QoS results when using the simulator and the iterative and Knepley and Fischer method to solve equations (1) through (5). We see that for all practical uses the results are the same. The buffer latency (Wq_i) are given in ms. When using the Knepley and Fischer method, the integration constant *h* has to be selected. For the parameters used in these examples, h = 0.001 was used. The steady state convergence criterion for the iterative and Knepley and Fischer methods was that the expected number of packets in the system at one iteration was within 0.0001 of the previous iteration. We see all three methods give the same results for decision making purposes.

Figure 6 compares the run times of the three methods. We see the run times for the iterative and Knepley and Fischer methods are dependent on the buffer size, because at each iteration the complete probability distribution of the state space is determined. As the buffer size increases, the state space increases and the number of calculations increases as well. The simulation run time does not depend on the buffer size. The Knepley and Fischer method is much more sensitive to the buffer size as we had to set h = 0.001 for this problem, because of the values of the input parameters, λ_i and μ_i.

The major advantage of the Knepley and Fischer method is that it generates the transient results, while the iterative method does not. The simulation method could be used to do that, but it would require significantly more runs. One option of the simulator is to output the buffer delays and the number of packets in the system at each departure point. Here we are only interested in expected value results in steady state. But if expected transient results were required via the simulator; then multiple runs of the simulator would have to be made.

Also if steady state probability distributions of the number of packets in the system are needed, the iterative and Knepley and Fischer methods give those results directly with no additional computational work. They can also be obtained from the simulator, but the run time is significantly increased because the associated table has to be output. The increase in run time to output the tables is a factor of 10. So the simulation run times become 290 seconds as opposed to 29 seconds. In addition, the output tables then have to be post-processed to generate the desired probability distributions. The simulation run time does not depend of the buffer size.

A significant advantage of the simulation model is that it outputs the actual waiting time for each packet by type. We got the mean delay for the iterative and Knepley and Fischer methods by using Little's formula; higher moments can be obtained via the higher moments of Little's formula (Gross and Harris (1998)), but the total probability distribution of the queue wait is not available from those methods.

We close this section with a brief look at the accuracy of the methods. In order to do this analysis, we ran the iterative method for the 100 buffer case for 15,000 iterations—a graph of the convergence profile and run time is given in Figure 7. It took approximately 160 seconds for this run and the resulting number of expected packets (both classes) in the system was 13.0292. We use this value as the true steady state value. Table 2 presents the percentage difference of the three methods with this value, using the data from Table 1.

Three points are immediate from Table 2. First, the results presented in Table 1 are very close to the steady state values, less than 1 percent difference. Second, the simulation method gets closer to the steady state value as compared to the other two methods based on computer time. Even though the Knepley and Fischer method was closer to the steady state value, it took 180 seconds of computer

Table 1: QoS Comparison of Simulation, Iterative, and Knepley and Fischer

	Simulation		***Iterative***		***Knepley and Fischer***	
Buffer Size	***PB_1***	***PB_2***	***PB_1***	***PB_2***	***PB_1***	***PB_2***
25	0.011395	0.001196	0.011339	0.001012	0.011658	0.001450
50	0.001401	0.000004	0.001290	0.000019	0.001309	0.000029
75	0.000243	0.000000	0.000174	0.000000	0.000180	0.000001
100	0.000047	0.000000	0.000024	0.000000	0.000027	0.000000
Buffer Size	***L_1***	***L_2***	***L_1***	***L_2***	***L_1***	***L_2***
25	6.24	4.05	6.43	3.97	6.48	4.08
50	8.04	4.43	8.05	4.41	8.08	4.47
75	8.38	4.58	8.40	4.46	8.44	4.51
100	8.45	4.60	8.47	4.47	8.51	4.52
Buffer Size	***Wq_1(ms)***	***Wq_2(ms)***	***Wq_1(ms)***	***Wq_2(ms)***	***Wq_1(ms)***	***Wq_2(ms)***
25	14.33	27.62	14.27	26.96	14.27	26.96
50	18.06	31.07	17.92	30.30	17.92	30.30
75	18.73	31.49	18.72	30.69	18.72	30.69
100	18.88	31.61	18.88	30.76	18.88	30.76

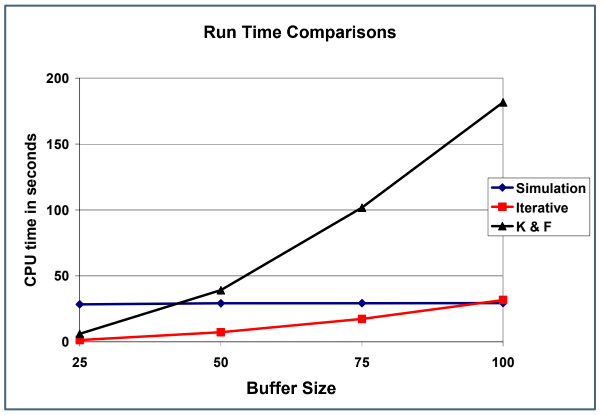


Figure 6: Run Time Comparisons of Methods

time where as the simulation only took 29 seconds. Finally, the Knepley and Fischer method is better than the iterative method on a per-iteration basis, because it builds in more information. Its recursive calculation uses data from the past two iterations. The iteration method took a little under 160 seconds for the 15,000 iteration, the Knepley and Fischer method got there in fewer iterations, but the same length of time.

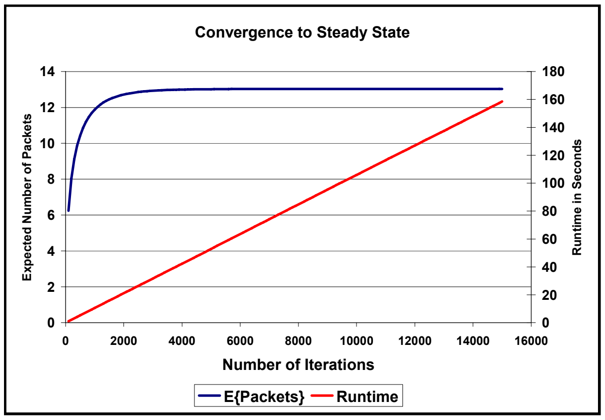


Figure 7: Convergence to Steady State of Iterative Method

Table 2: Percent Difference of Methods and Steady State

Method	***Value***	***Percent Difference***
Iterative	12.9362	0.0071
Knepley and Fischer	13.0291	0.0000
Simulation	13.0483	0.0015

5 CONCLUDING REMARKS

In this paper we have shown the critical role simulation has played in our development of performance models for a very important IP router discipline, CBWFQ. At the start of this development, no performance analysis tools were available. We were tasked with the development of tools and found simulation played a crucial role in our development efforts.

In this paper simulation was presented and compared to two numerical methods. In order to make the numerical methods tractable, we had to restrict the comparison to only two classes of packets with Poisson arrivals and exponentially distributed packet lengths. Those simplifying assumptions allowed us to generate the balance equations for the system, but not an analytic solution. When more realistic assumptions about the packet arrival processes and packet length distributions are used, it appears that simulation will be the only tool that will yield practical results. We also suggest that researchers develop custom simulators for an analysis of router queueing disciplines like CBWFQ, rather than relying on commercially-available simulators, as they have a substantial amount of run time overhead and may not have the flexibility of use and understanding.

ACKNOWLEDGMENTS

This work was partially funded by the National Communications System under Contract Number NBCH-D-02-0039 and Noblis' Center for Network-Based Systems, which is a research collaboration of Noblis and George Mason University

REFERENCES

Altman, E., K. Avrachenkov, and U. Ayesta. 2006. A Survey on Discriminatory Processor Sharing. Queueing Systems 53: 53-63.

Cao, J., W. S. Cleveland, D. Lin, and D. X. Sun. 2002. Internet Traffic Tends Toward Poisson and Independent as the Load Increases. Nonlinear Estimation and Classification. Editors: C. Holmes, D. Denison, M. Hansen, B. Yu, and B. Mallick, Springer, New York, NY. http://cm.bellabs.com/who/dong/papers/lrd2poisson.pdf.

Cidon, I. and M. Sidi. 1990. Recursive Computation of Steady-State Probabilities in Priority Queueing Systems. Operations Research Letters, Vol. 9, No. 4, pp. 249-256.

Cisco Systems, IOS Quality of Service Solutions Configuration Guide, Release 12.2 (Congestion Management Overview). http://www.cisco.com/en/US/proucts/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800b75a9.html.

Cohen, J. W. 1969. The Single Server Queue. North-Holland Publishing Company, New York, NY.

Fischer, M. J., and D. M. B. Masi. 2005. Voice Packet Arrival Models and Their Effect on Packet Performance. Applied Telecommunications Symposium, San Diego, CA.

Gross, D. and C. M. Harris. 1998. Fundamentals of Queueing Theory. Third Edition, John Wiley, New York, NY.

Kleinrock, L. and H. Levy. 1988. The Analysis of Random Polling Systems. Operations Research 36 (5): 716-732.

Knepley, J. E. and M. J. Fischer. 1977. A Numerical Solution for some Computational Problems Occurring in Queueing Theory. TIMS Studies in the Management Sciences, Vol. 7: 271-285.

Lee, T. 1997. A Closed Form Solution for the Asymmetric Random Polling System with Correlated Levy Input Process. Mathematics of Operations Research 22 (2): 432-457.

Masi, D. M. B., M. J. Fischer, and D. A. Garbin. 2007a. Modeling the Performance of Class Based Weighted Fair Queueing with OPNET and Custom Simulators. OPNETWORK Conference, Washington, DC.

Masi, D. M. B., M. J. Fischer, and D. A. Garbin. 2007b. Modeling the Performance of Low Latency Queueing for Emergency Telecommunications. Winter Simulation Conference, Washington, DC.

Miller, D. R. 1981. Computation of Steady-State Probabilities for M/M/1 Priority Queues. Operations Research 28 (5).

Morse, P. M. 1958. Queue, Inventories and Maintenance. Wiley, New York, NY.

Newman, D., G. Chagnot, and J. Perser. 2001. Networking the Telecom Industry, Light Reading. Detailed Methodology, Section 3–Test Procedures. www.lightreading.com/document.asp?doc_id=3972.

Parekh, A. K. and R. G. Gallager. 1993. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-node Case." IEEE/ACM Transactions on Networking 1 (3): 344-357.

Semeria, C. 2001. "Supporting Differentiated Service Class: Queue Scheduling Disciplines." Juniper White Paper. http://www.juniper.net/soltions/ literture/white_papers/200020.pdf.

Takagi, H. 1986. "Analysis of Polling Systems." The MIT Press, Cambridge, MA.

Takagi, H. 1988. Queueing Analysis of Polling Models." ACM Computing Surveys 20 (1): 5-28.

Thompson, K., G. J. Miller, and R. Wilder. 1997. Wide Area Internet Traffic Patterns and Characteristics. IEEE Network.

AUTHOR BIOGRAPHIES

MARTIN J. FISCHER is a senior fellow at Noblis where his experience includes network design and performance analysis in telecommunications. He has published more than 50 articles in refereed journals. He received his doctorate degree in operations research from Southern Methodist University.

DENISE M. BEVILACQUA MASI is a senior principal engineer at Noblis where her experience and research interests include queueing theory and simulation applied to telecommunications networks. She received her doctorate degree in information technology and engineering at George Mason University.

JOHN F. SHORTLE is an associate professor of systems engineering and operations research at George Mason University (GMU). He is a member of the Center for Air Transportation Systems Research at GMU and a member of the Center for Network-Based Systems, a collaborative initiative between Noblis and GMU. His experience includes developing stochastic, queueing, and simulation models to optimize networks and operations. His research interests include simulation and queueing applications in telecommunications and air transportation. He received his doctorate degree in operations research from UC Berkeley.