The Octave queueing Package*

Moreno Marzolla

Dipartimento di Informatica–Scienza e Ingegneria (DISI), Università di Bologna,
Mura Anteo Zamboni 7, I-40127 Bologna, Italy
moreno.marzolla@unibo.it

Abstract. Queueing Networks are a widely used performance modeling tool that has been successfully applied to evaluate many kind of systems. In this paper we describe the `queueing` package, a collection of numerical solution algorithms for Queueing Networks and Markov chains written in GNU Octave (an interpreted language for numerical computations). The `queueing` package allows users to compute steady-state performance measures for product-form and some types of non product-form Queueing Networks. Additionally, the package provides functions to analyze single station queueing systems and Markov chains. Therefore, the `queueing` package can be used for reliability analysis, capacity planning and general systems modeling and evaluation.

1 Introduction

Queueing Networks (QNs) are a powerful modeling notation that can be used for capacity planning, bottleneck analysis and performance evaluation of many kinds of systems. In its basic form, a QN consists of K service centers, each containing one or more servers sharing a common queue. Requests circulate through the system, joining the queues from which they are extracted according to a queueing policy, e.g. First-Come First-Served (FCFS), to receive service from one of the associated servers. After service completion, a request may join another queue or, for open queueing networks, leave the system. Many extensions to this basic model have been proposed in the literature (e.g., networks with multiple request classes, jobs with priorities, passive queues, finite capacity regions and so on).

Despite the vast literature on numerical solution techniques for QN models (see [1] and references therein), there is a shortage of software packages for QN analysis[1]. To this aim, we developed the `queueing` package for GNU Octave [4], an interpreted language for numerical computations. The `queueing` package implements numerical algorithms for stationary analysis of product-form queueing models; open, closed and mixed networks with different classes of customers are supported. Furthermore, the package allows transient and steady-state analysis

* This is an author-generated version of a paper published by Springer. The final version and citation information are available from SpringerLink at `http://dx.doi.org/10.1007/978-3-319-10696-0_14`

[1] A list of tools is available at `http://web2.uwindsor.ca/math/hlynka/qsoft.html`, although many links are broken as packages disappear

of discrete and continuous-time Markov chains (e.g., state occupancy probabilities, first passage times, mean time to absorption).

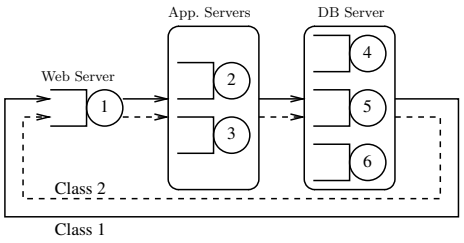
2 Design Principles

`queueing` is a collection of Octave functions for computing various transient and steady-state performance measures of queueing models and Markov chains. The Octave interactive environment provides the glue which allows complex models to be built and evaluated programmatically. This can be useful, e.g., to do parametric evaluation of complex models, or to perform ad-hoc analysis not already covered by one of the functions provided. While this allows the greater degree of flexibility, it imposes a steep learning curve.

The following usage scenarios for `queueing` can be identified: (*i*) **Incremental model development**: the `queueing` package and GNU Octave can be used for rapid prototyping and iterative refinement of QN models. (*ii*) **Modeling environment**: large and complex performance studies can be done quickly, since models involving repetitive or embedded structure can be easily defined. (*iii*) **Queueing Network research**: new algorithms can be programmed and tested against existing ones. The Octave language is well suited for implementing numerical algorithms which operate on arrays or matrices; QN algorithms fall in this category. (*iv*) **Reference implementations**: as observed in [2], some large research communities (e.g., linear algebra and parallel computing) have a long history of sharing implementations of standard algorithms. The `queueing` package aims at providing reference implementations of core QN algorithms. (*v*) **Teaching**: `queueing` is being used in some Universities to teach performance modeling courses. Since the package implements many textbook QN algorithms, students can immediately put those algorithms at work to solve practical problems, encouraging "learning by doing".

Special care has been put to make `queueing` a useful tool for research, education of practical use. The documentation of each function can be accessed using the `help()` Octave command (e.g., `help(ctmc)` prints the usage documentation of the `ctmc()` function, that computes the transient or stationary probability of a continuous-time Markov chain). Usage demos are available as well, and can be accessed using the `demo()` command, e.g., `demo("ctmc")` displays and executes all demo blocks for the `ctmc()` function.

One important issue of numerical software is to make sure that the computed results can be relied upon. Most of the functions included in the `queueing` package embed unit tests as specially-formatted comments inside the source code. These tests can be executed automatically to check the results against known values. When reference results are not available, cross-validation may be possible by executing two different functions on the same model and comparing the results. For example, the same closed network can be analyzed by Mean Value Analysis (MVA), or using the convolution algorithm. Finally, results can be compared with those produced by different tools.



#	Name	Demands Class 1	Class 2
1	Web Server	12	2
2	App. Server 1	14	20
3	App. Server 2	23	14
4	DB Server 1	20	90
5	DB Server 2	80	30
6	DB Server 3	31	33

Fig. 1: Three-tier enterprise system model from [3]

3 Usage Example

The model depicted in Figure 1, taken from [3], shows a three-tier enterprise system with $K = 6$ service centers. The first tier contains the *Web server* (node 1), which is responsible for generating Web pages and transmitting them to clients. The application logic is implemented by nodes 2 and 3, and the storage tier is made of nodes 4–6. The system is subject to two workload classes, both represented as closed populations of N_1 and N_2 requests, respectively. Let $D_{c,k}$ denote the service demand of class c requests at center k. We use the parameter values given in [3] and reported on Figure 1.

We set the total number of requests $N = N_1 + N_2$ to 100, and we study how different population mixes (N_1, N_2) affect the system throughput and response time. Let $\beta_1 \in (0, 1)$ denote the fraction of class 1 requests: $N_1 = \beta_1 N$, $N_2 = (1 - \beta_1)N$. The following Octave code defines the model for $\beta_1 = 0.1$:

```
N = 100; beta1 = 0.1;
S = [12 14 23 20 80 31; 2 20 14 90 30 33 ];
V = ones(size(S));
pop = [fix(beta1*N) N-fix(beta1*N)];
[U R Q X] = qncmmva(pop, S, V);
```

The `qncmmva(pop, S, V)` function uses the multiclass MVA algorithm to compute per-class utilizations $U_{c,k}$, response times $R_{c,k}$, mean queue lengths $Q_{c,k}$ and throughputs $X_{c,k}$ at each service center k, given a population vector `pop`, mean service times `S` and visit ratios `V`. Since we are given the service demands $D_{c,k} = S_{c,k}V_{c,k}$, but function `qncmmva()` requires separate service times and visit ratios, we set the service times equal to the demands, and all visit ratios equal to one. Overall class and system throughputs and response times can be computed as [5]:

```
X1 = X(1,1) / V(1,1);  X2 = X(2,1) / V(2,1);
XX = X1 + X2;              # system throughput
R1 = dot(R(1,:), V(1,:));  R2 = dot(R(2,:), V(2,:));
RR = N / XX;               # system resp. time
```

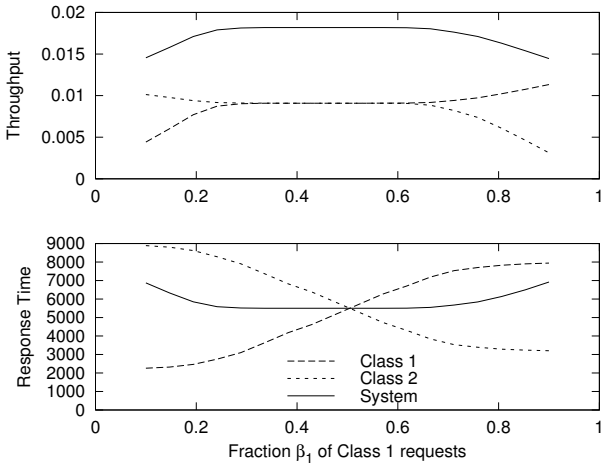


Fig. 2: Throughput and Response Times as a function of the population mix β

For $\beta_1 = 0.1$ we get `X1` = 0.0044219, `X2` = 0.010128, `XX` = 0.014550, `R1` = 2261.5, `R2` = 8885.9, `RR` = 6872.7. We can iterate the computations above for various values of β_1 to obtain the results shown in Figure 2.

4 Conclusions

In this paper we presented the `queueing` package for GNU Octave. The `queueing` package is available from `http://octave.sourceforge.net/` and can be freely used under the terms of the GNU General Public License (GPL) version 3.

References

1. G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley, 1998.
2. G. Casale, M. Gribaudo, and G. Serazzi. Tools for performance evaluation of computer systems: Historical evolution and perspectives. In *Performance Evaluation of Computer and Communication Systems. Milestones and Future Challenges. IFIP WG 8.3/7.3 International Workshop, PERFORM 2010*, volume 6821 of *LNCS*, pages 24–37. 2011.
3. G. Casale and G. Serazzi. Quantitative system evaluation with Java modeling tools. In *Proc. second joint WOSP/SIPEW international conference on Performance engineering*, ICPE '11, pages 449–454, New York, NY, USA, 2011. ACM.
4. J. W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002.
5. E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice Hall, 1984.