High-level Approach to Modeling of Observed System Behavior

Thomas Begin[*] thomas.begin@lip6.fr

Alexandre Brandwajn[+] alexb@soe.ucsc.edu

Bruno Baynat[*] bruno.baynat@lip6.fr

Bernd E. Wolfinger[#] wolfinger@informatik.uni-hamburg.de

Serge Fdida[*] serge.fdida@lip6.fr

[*]Université Pierre et Marie Curie-Paris6, UMR (CNRS) LIP6, Paris, France

[+]University of California Santa Cruz, Baskin School of Engineering, USA

[#]Universitaet Hamburg, Dept. Informatik, Germany

ABSTRACT

Current computer systems and communication networks tend to be highly complex, and they typically hide their internal structure from their users. Thus, for selected aspects of capacity planning, overload control and related applications, it is useful to have a method allowing one to find good and relatively simple approximations for the observed system behavior with little knowledge of the system internal structure or operation. This paper investigates one such approach where we attempt to represent the observed behavior by adequately selecting the parameters of a set of queueing models. We identify a limited number of queueing models that we use as Building Blocks in our procedure. The selected Building Blocks allow us to accurately approximate the measured behavior of a range of different systems. The models used include a small selection of elementary queueing models, as well as original models of system congestion and saturation. We propose an approach for selecting suitable Building Blocks, as well as for their calibration. In particular, we automate the calibration step and make it systematic through the use of a Derivative Free Optimization technique. We are able to successfully validate our methodology for a number of case studies. Finally, we discuss the potential and the limitations of the proposed approach.

Keywords

High-Level Modeling, Constructive Modeling, Computer and Communication Systems, Model Calibration, Building Blocks, Performance, Measurements

1 Introduction

1.1 Motivation

Analytic performance modeling of computer and communication systems has numerous applications throughout the life-cycle of such systems, from their design phase to the actual configuration, tuning and capacity planning. It has been successfully applied over a large spectrum of computer systems and communication networks, including centralized and distributed systems [17]. A commonly used method, which we refer to as the constructive approach, is to attempt to reproduce in the mathematical model essential aspects of the system structure and operation. The mathematical model may be based on queueing, fluid, graph or some other theory. This constructive approach has its limits. First, important aspects of large and heterogeneous computer or communication systems, such as modern I/O controllers, public mobile networks, or Internet Service Provider networks, may be largely unknown. These unknown aspects contribute to the difficulty of building successful constructive models of such systems. Second, in order to identify critical performance factors in systems with many complex internal interactions, the performance analyst would need extensive knowledge and expertise that may simply not be available. In particular, when dealing with such complex systems it is critical to correctly identify key system components and features lest the resulting models become unrealistic or intractable in their complexity. These difficulties motivate in part our approach.

In our high-level modeling, we don't necessarily seek to "mimic" the structure of the system under study. Rather, we focus on the observable interaction of the system with the outside world as represented by measurements or performance indices, and attempt to infer a possible high-level model structure from it. More specifically, we assume that we have at our disposal some number of measurement points for the system under study, and we seek a relatively simple model capable of adequately reproducing the observed system behavior. In doing so, we forego the detailed representation of the system in favor of the possibility that a relatively simple model, not necessarily related to the apparent structure of the system, might be able to capture the behavior of the system under consideration. An obvious justification for our approach is that, even in a complex system, it is possible that a small number of components, or a single component, may be the critical bottleneck, effectively driving the system behavior. This idea is by no means novel, and has been frequently employed in the past, e.g. in the case of an Internet path [32, 2], disk arrays [34] or a Web server [10].

Our approach has several objectives. First, it may help discover properties of the system not immediately apparent from its structure. This may include both the fact that a simple model can represent the system performance or, on the contrary, that no simple model (among the ones examined) will be able to adequately represent the system. As such, our approach can be viewed as helpful for and complementary to constructive system modeling. Second, when a sufficient number of measurement points are available to find an adequate model, our approach may provide the performance analyst with a ready-to-use model to generate reliable forecasts for system performance at other workload levels, without the expense and the effort of obtaining additional measurements. Finally, for a subsystem embedded in a larger system (with the obvious proviso that measurements be available for the subsystem), our approach may be able to provide a model of the subsystem that can then be incorporated in the overall system model. This latter application has a clear connection with decomposition methods [7].

The advantage of the proposed approach is that it requires a priori little information about the system. Our contribution is twofold. We automate the process of model selection and make it systematic by embedding it into a software tool with an optimization method. As a result, the approach requires no special modeling or queueing theory expertise from the end user. Our second contribution is to define, in addition to selected standard queueing models, original "Building Blocks" that allow us to represent a wide range of behaviors. By design, our modular approach lends itself to extensions and makes it easy to add new Building Blocks if required.

Our approach has obvious limitations. The system considered (or its detailed constructive simulation model) must exist, and we must have a sufficient number of measurement points. In general, the model produced by our approach has little predictive power if the underlying system or the basic nature of the workload (other than its intensity) changes. Additionally, there is no guarantee that the approach will find an adequate model, and, since we only consider a limited selection of models, a failure of the approach does not necessarily imply that there is no adequate simple model for the given system. However, as noted before, the modularity of our approach enables the addition of an appropriate "Building Block" that may improve the chances of success of the method.

1.2 State of the art survey

A significant number of existing computer or communication systems have been extensively measured. The measurements obtained permit, among other things, to investigate the system by observing its behavior. This investigation can be done either directly from the measurements or indirectly using the measurement results in a model of the system. A good example of the "all-measurement schemes" in computer networks is given by Dovrolis et al. [15], which presents a method to estimate a path capacity (bottleneck bandwidth) using probing packets. The specificity of the underlying logic may not allow one to generalize this approach to other cases. On the other hand, the less direct use of measurements is typically associated with the constructive modeling approach. Here, measurements are mainly used to determine the value for parameters of a pre-designed constructive model (see [17]). For instance, Liu et al. [26] present an efficient but model specific way to adjust the parameters of their constructive model using measurements data. Crovella and Krishnamurthy [14], as well as Paxson [28] give additional useful information regarding network measurements.

Our approach lies somewhat in between these two schemes. It is high level and driven by measurements (cf. [4]]). The fundamental assumption underlying our methodology is that relatively simple models may be able to approximate sufficiently accurately certain, possibly complex, system behaviors. Some of the simple models

(Building Blocks) considered are clearly motivated by a constructive modeling approach. In Section 2.4 we show a few examples of more or less complex systems whose behavior can be reproduced using a simple Building Block. Examples abound in the literature. A case in point is the early model developed by Scherr [33] who used a simple machine repairman model to represent the performance of a time-sharing system. In fact, Scherr's model can be viewed as a precursor of our high-level approach. In the case of disks arrays, Varki et al. [34] recognize the key system components and present an accurate model sufficiently simple to be tractable. In the case of servers, Cao et al. [10] show that the performance of a Web server may be well approximated using a simple queueing model.

As mentioned in the introduction, while in the constructive approach models tend to mimic the structure of the actual system [32, 2, 34, 10], in our high-level approach the choice of the model is determined by the available measurement data and the set of Building Blocks considered. It is important to stress that our approach is not a replacement for but a complement and possible aid to constructive modeling.

1.3 Structure

The paper is structured as follows. Section 2 describes the general framework in which we cast our approach. We present the selected models (Building Blocks), as well as our general approach to determining the best set of model parameters and assessing the goodness of fit for a given Building Block. Section 3 presents several examples of application of our tool. All case studies in our paper use measurement data from real-life systems. Finally, Section 4 summarizes the main contributions of our approach, as well as its limitations, and outlines possible future extensions of our work.

2 General framework

2.1 Terminology

Systems considered in our study may represent a whole computer or communication system, or specific components such as processors, a disk array, an Ethernet network or a WLAN, etc. We use the term "requests" to refer to the individual entities that are treated by the system, such as packets or frames in the case of networks, I/O requests in the case of storage systems, HTTP requests in the case of web servers, etc. The offered "load" (workload) includes all the requests that are submitted to the system for treatment. In our view, the system performance changes in response to the workload, and these changes are reflected in the corresponding measurements. Finally, the "environment" represents all the sources that generate the system workload. Environment can consist of human users, system or application processes, threads, etc. In an open system, the environment is viewed as independent of the system, while in a closed system, the environment and the system are in a feedback loop, i.e., the behavior of the system influences the environment.

Formally, we view the environment as submitting requests to the system through a well-defined interface. Then, the workload relative to some observation period is the sequence of requests submitted to the system during the observation period (cf. [36] and [12]).

2.2 Measurements of the observed system's behavior

In its present form, our approach relies on the availability of measurements of specific system performance parameters. These parameters may include quantities that we view as system inputs (e.g. average number of requests submitted per time unit) and quantities that we view as outputs (e.g. response time). This is illustrated in Figure 1. Typically, input measurements pertain to the workload. Thus, they may include data pertaining to the request arrival process, such as the average rate of request arrivals denoted by λ, higher moments of the inter-arrival time, etc., or other attributes related to the incoming requests such as the distribution of their size. In some cases, input measurements may simply not be available. Output measurements may include quantities such as the attained throughput of requests processed by the system per time unit, as well as measures of internal system congestion such as the number of requests inside the system. Typical measured performance parameters include throughput, loss probability, average response time and queue length, denoted by $\overline{X}_{mes}$, $\overline{L}_{mes}$, $\overline{R}_{mes}$, and $\overline{Q}_{mes}$, respectively. The throughput $\overline{X}_{mes}$ represents the average number of requests that leave the system per unit time (this quantity may differ from the offered workload if the system is subject to losses). $\overline{L}_{mes}$ gives the probability that an arriving request is rejected, i.e., denied entry to the system. $\overline{R}_{mes}$ defines the average sojourn time (waiting for and receiving service) experienced by a request inside the system. Finally, $\overline{Q}_{mes}$ represents the average number of requests in the system. Note that, by Little's law [23], $\overline{Q}_{mes} = \overline{X}_{mes}\overline{R}_{mes}$ so that it suffices to measure any two of these three quantities.

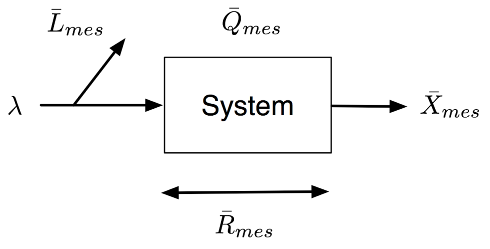

Figure 1: Performance parameters.

In computer networks, typical performance parameters are the throughput at an interface, the time spent by packets inside the network and the packet loss ratio. In disk arrays, performance parameters may represent the I/O response time, I/O request throughput, device utilization, etc.

Each measurement point corresponds to a set of performance parameters that have been measured at a particular state of the load, and may in general also include input parameters such as the corresponding offered load. Clearly, a measurement point can represent a single measurement instance or be obtained by averaging several measurement instances corresponding to the same load level. A total of *n* measurement points for the same system constitutes a set of measurements. Generally, the measurement data set reflects how the system reacts to a varying environment.

Note that in our high-level modeling approach, the measurement set must include measurement points for different load levels. Hence, methods that seek to fit a model of a discrete or continuous distribution to a sample of measurements for a single level of system load are clearly unsuitable for our approach [6].

2.3 A quick overview of the approach

Although our approach is more general, in this paper, we restrict the presentation to the case where the average sojourn time in the system $\overline{R}$ is expressed as a function of the average throughput out the system $\overline{X}$. We thus have at our disposal measurement points $(\overline{X}_{mes}, \overline{R}_{mes})$, each of them corresponding to a different load (as explained in Section 2.2), and we aim to find a model that closely reproduces the observed system behavior.

The starting point of the proposed high-level modeling approach consists in defining a pool of “Building Blocks”. The Building Blocks are generic models, i.e. models in which certain parameters are unspecified, which may come from various domains such as queueing or fluid theory. It is clearly important to select and define Building Blocks whose spectrum covers a wide range of system behavior. In this paper, we use a set of Building Blocks including a limited selection of elementary queueing models (e.g. M/M/C, M/M/C/K, M/G/1, M/G/1/K [9], M/G/C approximation [22, 5]), as well as original models of system congestion and saturation. Thus, we define a new Building Block, referred to as the “Embedded Queue block”, in which the service time at the “main” server includes the expected waiting time at a “secondary” queue. We also propose another “non-standard” Building Block, referred to as the M/M/C_v block, defined as a multi-server queue in which the number of servers C_v is allowed to change as the resource utilization changes. This type of behavior is abstracted from systems that make additional resources available to meet increased demand. These two less standard models are discussed in more detail in Sections 2.4 and 3.7, respectively.

To represent the fact that, in some systems, the response time comprises a fixed overhead as an additive load-independent component, we expand our Building Blocks to include a fixed “Offset” value. Note that this Offset does not affect the congestion at the server. The response time in our Building Blocks is simply the sum of the Offset value and the response time at the server. This Offset quantity can be viewed as constant overhead in the response time. For disk controllers, it may represent the time required to examine an I/O command before queueing or serving it. For an application server, it might include the predetermined time for a request to travel from the client issuing the request to the server, provided that this time does not increase when the intensity of the workload does. For data transmissions, the Offset value could reflect the signal propagation delay between communicating nodes. This constant Offset value is denoted by *Ofst* in all figures and formulas.

In the proposed approach, we consider the Building Blocks one by one, and calibrate each of them independently. By calibrating a given block we mean selecting a set of values for its parameters so that the performance of the model matches as closely as possible system measurements. To this end, we define (cf. Section 2.5) an error criterion that quantifies how far a given model is from the measurements, as well as a technique to perform the calibration search (cf. Section 2.6). We use the same error criterion to rank the Building Blocks candidates and to determine the “laureate model”, i.e. the Building Block with the best calibration.

A tempting idea might be, instead of dealing with our Building Blocks based on queueing models, to simply try and use some sort of interpolation method between the measurement points. In our experience, such an approach is less likely to succeed. Indeed, our queueing-theory-based Building Blocks involve service times, as well as congestion and queue build up, much like what is happening in the actual system. Thus, there is a high likelihood that at least one of the Blocks will exhibit behavior fundamentally similar in nature to the one observed in the measurement data. On the other hand, as illustrated in Figure 2, a simple interpolation based on mathematical functions such as polynomial, Lagrange or Bessel functions, while certainly capable of connecting the measurement points, might exhibit inadequate behavior outside of the latter (e.g. negative or otherwise infeasible values, oscillations, etc., might occur). Thus, we believe that simple curve fitting is likely to be of lesser value in understanding and predicting the system behavior than our approach based on models of congestion.

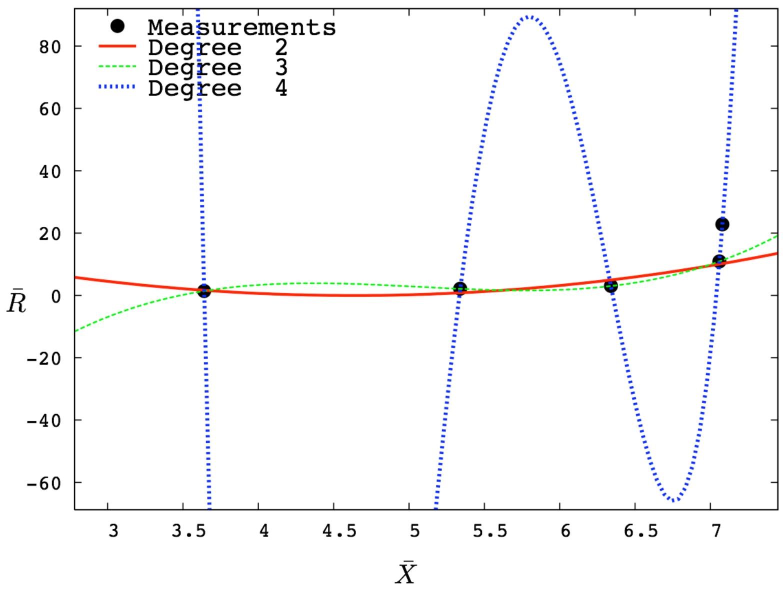

Figure 2: Inadequate results from polynomial interpolations.

2.4 Simple and not so simple models

As discussed in Section 1, one of the premises of our approach is that a complex system may exhibit behavior that can be reproduced by a relatively simple queueing model. The first example we consider is an M/G/1 queue with preemptive-resume priority discipline and three priority levels where level 1 has the highest and level 3 the lowest priority. We denote by λ_i the rate of arrivals to priority level *i*, by $1/\mu_i$ the mean and by γ_i the coefficient of variation of the service time for level *i*. We first look at the mean response time of the lowest priority level for a number of values of λ_3 with the workload of higher priority levels kept constant. The well-known solution of the M/G/1 priority queue (e.g. [1]) gives the performance data represented in Figure 3. While the analytical formula used to generate this curve is manageable, it may not be obvious that the mean response time for the selected priority level is in fact that of a simple M/G/1 queue with a different (higher) coefficient of variation and an Offset as shown in Figure 3. It is interesting to note that, for the parameter values used in this example, a simple M/M/1 queue (as proposed in some approximations), even with an Offset, cannot adequately represent the behavior at lower priority levels (cf. [20]). Note that the search for the appropriate parameter values for this example was performed using the automatic calibration method described in Section 2.6.

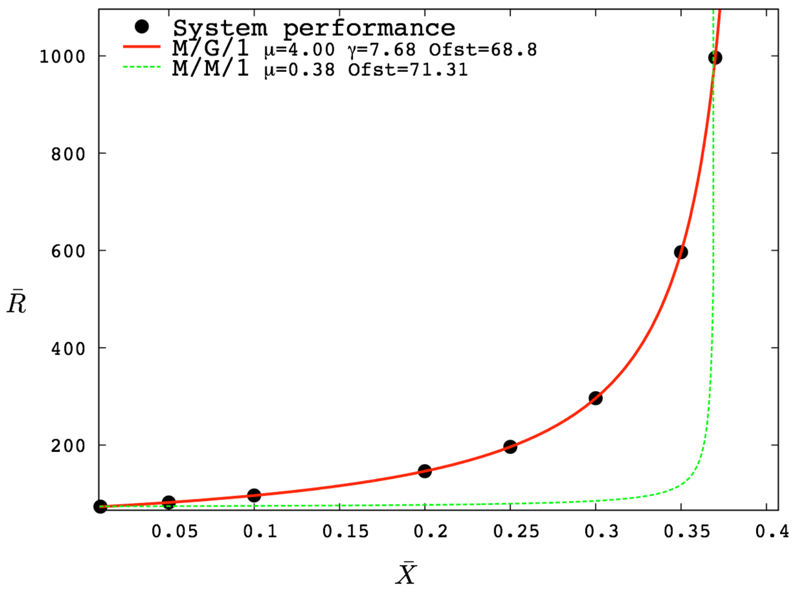

Figure 3: Behavior of the lowest priority class in an M/G/1 priority system with $\{\mu_1 = 0.1, \gamma_1 = 2\}$, $\{\mu_2 = 0.5, \gamma_2 = 2\}$ and $\{\mu_3 = 1, \gamma_3 = 2\}$with higher class workload kept constant.

In our second example, we consider an open tandem network of queues with limited buffer capacities and communication type of blocking, i.e., service at a node cannot start unless there is room in the downstream buffer. This network is represented in Figure 4. We assume that arrivals to the system come from a Poisson source with rate λ, the service times at each node are exponentially distributed, and we let μ_i be the corresponding service rate for node *i*. We denote by B_i the total capacity at node *i*. We select small buffer capacities, as well as service and arrival rates such that there is a significant amount of blocking in the network. Under these conditions, the performance of this tandem network can be expected to be far from that of a Jacksonian queueing system [18]. We use the approximation method of Brandwajn and Jow [8] to solve this tandem network of queues with a total of five nodes. Figure 5 shows the total sojourn time in such a network for a number of values of attained system throughput. Clearly, a queueing system with infinite buffer space does not allow us to model the rate of loss that requests can experience at the first node. It is interesting, however, to note that, as illustrated in Figure 5, an M/G/1 queue (with an Offset) can adequately reproduce the mean total sojourn time in such a system.

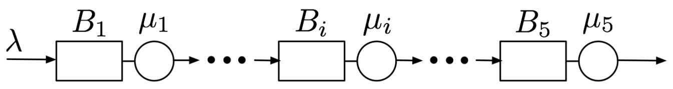

Figure 4: Open tandem network with finite buffer capacities.

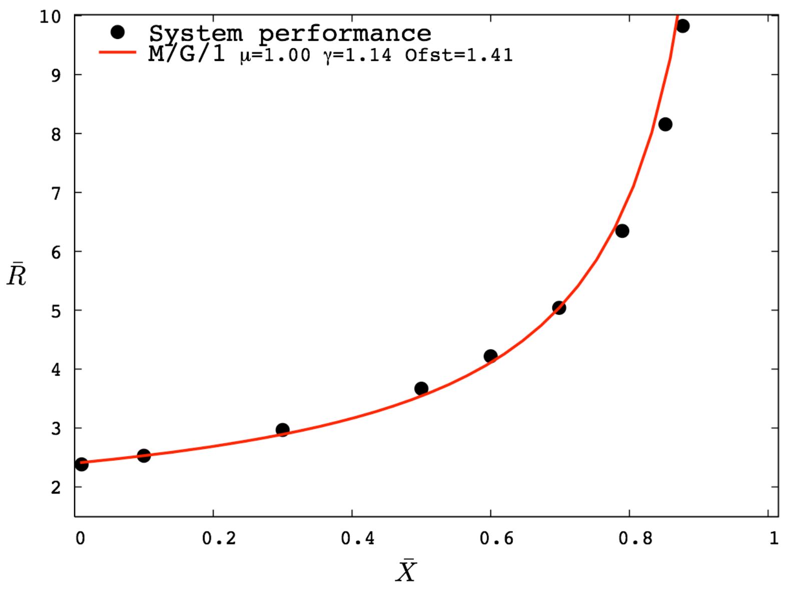

Figure 5: Behavior of an open tandem network with $\{B_1 = 5, \mu_1 = 5\}$, $\{B_2 = 3, \mu_2 = 3\}$, $\{B_3 = 3, \mu_3 = 3\}$, $\{B_4 = 2, \mu_4 = 1\}$ and $\{B_5 = 3, \mu_5 = 2\}$.

We now turn back to the first example, i.e. the M/G/1 queue with 3 priority levels studied previously. We look again at the performance of the lowest priority level, but this time we increase proportionately the arrival rate to all priority levels. Figure 6 shows the expected time in system for level 3 requests. Here, no simple M/G/1 queue can match the observed response times since the delays impacting the lowest priority level increase as its own rate of arrivals λ_3 increases. Analogous apparent elongation of the service time can in fact be observed in a number of other systems. This leads us to define a new Building Block in which the service time at the "main" server includes the expected waiting time at a "secondary" queue. As mentioned before, we refer to this queueing model as the "Embedded Queue block". The main server in our embedded block is an M/G/1 server whose service time is a sum of two random variables: the base service time T_b and the service time elongation T_e. The latter is obtained as the queueing time in the secondary server, which is an M/G/C queue. The arrival rate to the secondary queue is some multiple of the arrival rate at the main server. Denoting by λ_m the rate of request arrivals to the main server, and by λ_s the arrival rate at the secondary server, we have $\lambda_s = \eta\lambda_m$ where η is a parameter of our embedded block. Note that we use the M/G/1 queue (with mean service time $1/\mu_e$ and coefficient of variation of the service time γ_e) as the secondary server in the simple form of the Embedded Queue block. Notice that such a simple Embedded Queue block produces a good match for the performance at lowest priority level in our example.

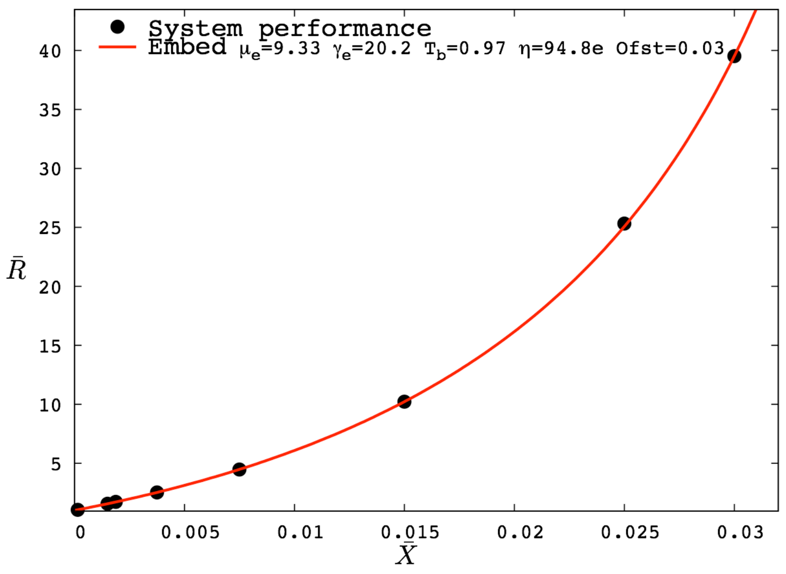

Figure 6: Behavior of the lowest priority class in an M/G/1 system with $\{\mu_1 = 0.1, \gamma_1 = 2\}$, $\{\mu_2 = 0.5, \gamma_2 = 2\}$ and $\{\mu_3 = 1, \gamma_3 = 2\}$ with variations of higher class workloads.

In Figure 7, we look at the expected sojourn time at the middle node in our tandem network of five nodes (second example). Perhaps not unexpectedly, we find that no simple M/M/C/K or M/G/1/K queue correctly reproduces the behavior of the node considered. We note, however, that an Embedded Queue block allows us to match the behavior of the middle node in our five-node tandem network.

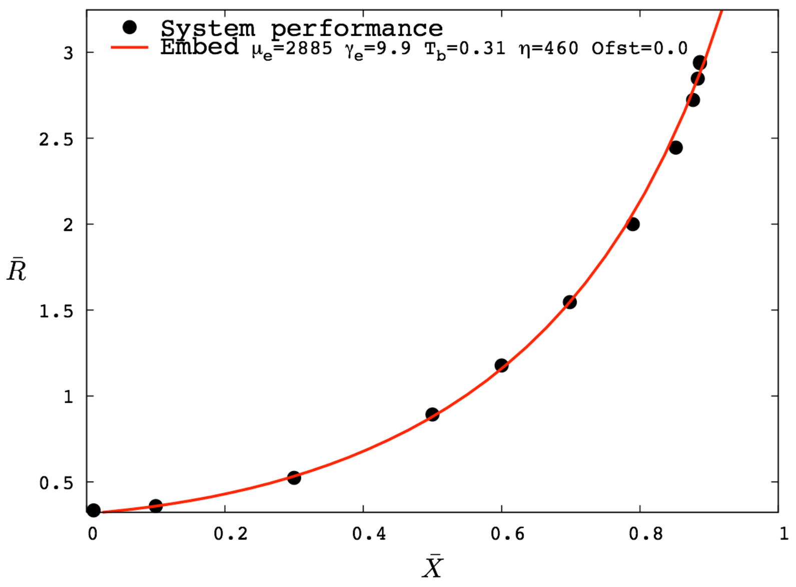

Figure 7: Behavior of the middle node in a tandem network $\{B_1 = 5, \mu_1 = 5\}$, $\{B_2 = 3, \mu_2 = 3\}$, $\{B_3 = 3, \mu_3 = 3\}$, $\{B_4 = 2, \mu_4 = 1\}$ and $\{B_5 = 3, \mu_5 = 2\}$.

As a final point, note that even though the M/G/1 block is a good representation for the lowest priority level in an M/G/1 priority queue (as depicted in Figure 3), in our approach, we are unable to readily derive the values of its parameters directly from the underlying model, as would be the case in constructive modeling. This limits the predictive power of our approach when parameters other than the attained throughput change. However, the mere fact that an M/G/1 queue (in this example) is a good fit, and the M/M/1 is not, may be valuable in the search for a simple constructive model. Incidentally, several authors [25, 24] have contemplated the use of the M/G/1 queue as the basic block in modeling general queueing networks.

2.5 Error criterion

As mentioned before, we need a way to measure the goodness of fit of a given model versus the measurement set. This is the role of the error criterion, referred to as ϕ. The goal of the function ϕ is to provide a convenient way to compare fairly any models. There are many ways to define such a function. Perhaps the most intuitive definition for ϕ is to sum the deviations between the system measured sojourn times and those obtained from the model at the same throughput values. This "stick based" error criterion is illustrated in Figure 8. We use the subscript *th* to denote values obtained from a model. Thus, let $\bar{R}_{mes,i}$, $i = 1,...,n$ be the measured mean response time values, and $\bar{R}_{th,i}$, $i = 1,...,n$, the corresponding mean response times obtained from a model. Then, ϕ can be formally expressed as:

$$\phi = \sum_{i=1}^{n} \left|\bar{R}_{th,i} - \bar{R}_{mes,i}\right| \tag{1}$$

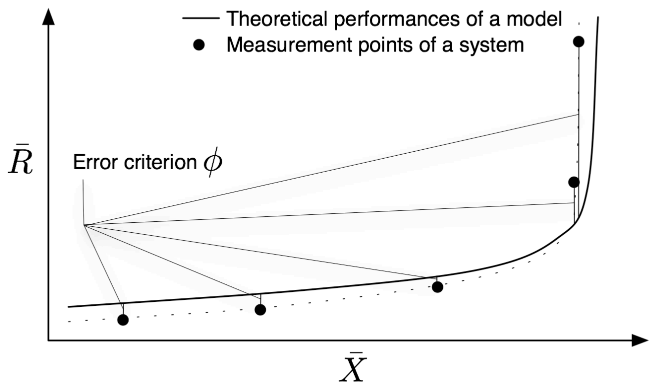
Theoretical performances of a model
● Measurement points of a system
$\bar{R}$
Error criterion ϕ
$\bar{X}$

Figure 8: Stick based error criterion.

Some adjustments to the definition of ϕ are possible. It may be desirable to take into account absolute and relative components for deviations, for example as a linear combination of both (absolute with a factor α and relative with a factor $1-\alpha$). Depending on the measurements and the intended domain of application of the resulting model, it may be useful to incorporate in ϕ weight factors w_i associated with each measurement point. These factors make it possible to emphasize selected measurement according to how much the analyst trusts them or which ranges of values are of interest in a particular study. Incorporating these changes, the definition of the error criterion ϕ becomes:

$$\phi = \alpha \frac{1}{\hat{R}_{mes}} \sum_{i=1}^{n} w_i \left|\bar{R}_{th,i} - \bar{R}_{mes,i}\right| + (1-\alpha) \sum_{i=1}^{n} w_i \left|1 - \frac{\bar{R}_{th,i}}{\bar{R}_{mes,i}}\right| \tag{2}$$

where $\hat{R}_{mes} = \frac{1}{n} \sum_{i=1}^{n} \bar{R}_{mes,i}$ is the arithmetic average of the measured response times.

In all numerical examples in the paper, we let $\alpha = 0.5$ and we use equal weights w_i. We make this choice because we trust all measurement points equally, and we have no reason to emphasize a particular operating region of the systems considered.

Note that, if α is set to 1 (i.e. only the absolute component is taken into account) and all w_i are equal, the definition of ϕ given in (2) is similar to the so-called "efficiency coefficient" defined by Nash and Sutcliffe [27] for adjusting hydrological models. Note also that, if confidence intervals are known for each measurement point, one can modify the definition of the error function in order to take them into account.

Despite their simplicity and ease of use, the definitions of ϕ given in (1) and (2), as well as the "efficiency coefficient" can lead to problems in some cases. As an example, in the vicinity of server saturation for open queueing models, these criteria can be exceedingly sensitive. This is particularly true when the saturation is steep, resulting in a misleading assessment of the accuracy of the model compared with measurement data. As an illustration, the model depicted in Figure 8, which is visually close to all the measurement points, will be ranked low because of the last point that generates a very large error. A promising approach seems to be to define the error criterion based on the estimation of the area between the curve formed by the measurement points and the one derived from the model. This area-based criterion has the advantage to be well defined when the last measurement points are beyond the maximum achievable throughput of the model (as illustrated in Figure 9). To calculate this area, we associate with each measurement point the closest point on the model curve. The segments connecting consecutive measurement points with their associated (coupled) points on the model curve define quadrilaterals. We then compute the error as the sum of the areas of these resulting quadrilaterals. This scheme is illustrated in Figure 9 and will be investigated in more detail in future work. Note that, in most examples shown in this paper, using the area-based criterion instead of that given by (2), has limited impact on the results (i.e. we obtain almost exactly the same laureate model), but it may enhance the search procedure (cf. Section 2.6).

For readers with interest in optimization techniques, the next section is devoted to details of the procedure we use to search for the best building block.

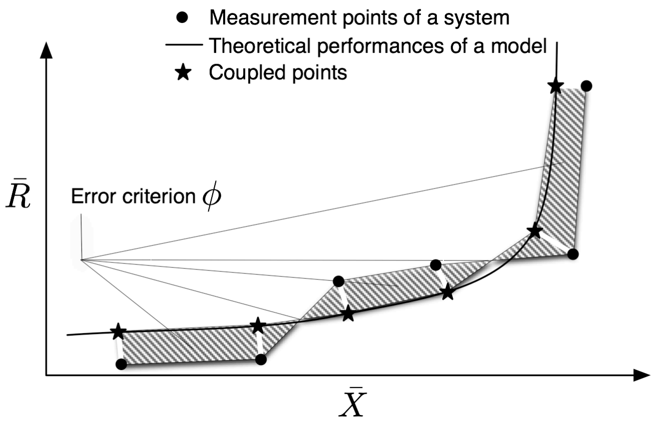
● Measurement points of a system
—— Theoretical performances of a model
★ Coupled points
$\bar{R}$
Error criterion ϕ
$\bar{X}$

Figure 9: Area-based error criterion.

2.6 A DFO technique to search the laureate model

Recall that the laureate model produced by our high-level approach is obtained from the combination of a particular Building Block and a corresponding set of parameters values. Thus, the search for the laureate is twofold. First, for each Building Block, it aims at finding a set of values for the model parameters that minimizes the selected error criterion ϕ (error criteria have been discussed in Section 2.5). We refer to this step as the "calibration process", also known as the "adjustment" of the blocks. Second, it consists in stating some rules to decide which model, among the Building Blocks with the best set of parameters, is elected, if any, as the laureate model for the observed system behavior.

We formalize the calibration process for a Building Block as a numerical optimization problem in which the objective function to minimize is simply the error criterion ϕ. The Building Block parameters are the variables of the objective function ϕ. For a given Building Block with n parameters, each possible combination of these parameters induces a value for ϕ and, taken all together, the Building Block parameters define a hypersurface of dimension n in a space of dimension n+1. Figure 10 shows an example of such a hypersurface in case of a Building Block with two parameters. Clearly, by minimizing ϕ, we get the coordinates of the optimum (minimum) of the hypersurface, which corresponds precisely to the best calibration of the parameters for the given Building Block.

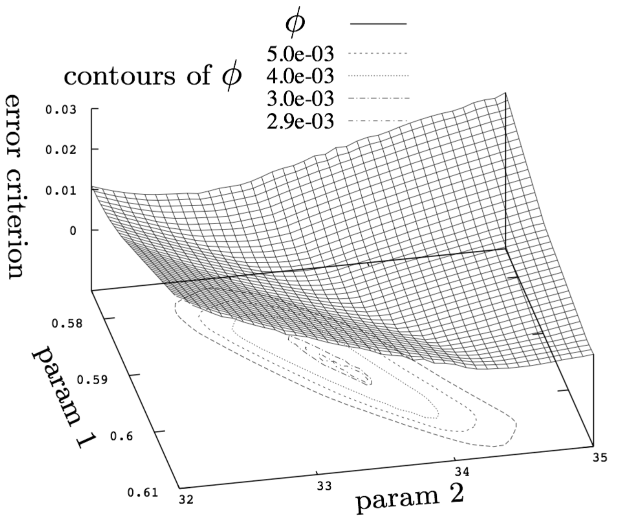

Figure 10: An example of ϕ for a Building Block with two parameters.

At this point, it is worthwhile stating some general remarks regarding the calibration search and its objective function ϕ. First, we have to find an efficient way to perform this optimization process since it must be repeated for each Building Block. Second, this optimization problem has to be managed taking into account two intrinsic difficulties. The first one pertains to the evaluation of the objective function ϕ, which may be computationally difficult (see Section 2.5). Indeed, to evaluate the objective function we need not only to evaluate the performance parameters of the model for each measurement point, but also, for each of them, to infer the input workload level that corresponds to the measured attained throughput. A second difficulty comes from the derivatives of ϕ that may not be computable. In fact, for some Building Blocks, there are no closed-form expressions that relate directly the measured performance parameters (viz. $\overline{R}$ as a function of $\overline{X}$ considering the formalism used in this paper). As an example, for M/M/C/K and M/G/1/K [9] queues, $\overline{X}$ and $\overline{R}$ are both function of the input workload, and are thus only implicitly linked through this hidden variable.

We have initially considered several different approaches to handle this optimization problem. First, we looked at the exhaustive search in the parameter space. The inherent exponential complexity of this systematic approach makes it unscalable and thus intractable as soon as a Building Block has more than three parameters. In the case of a block with four parameters, such a search typically requires more than 10^6 evaluations of ϕ. In fact, such an exhaustive search is inappropriate for calibrating virtually any of the Building Blocks, including blocks with only two or three parameters, since it requires the discretization of continuous parameters, which can have a severe impact on the robustness of the search.

Next, as mentioned before, computing the derivatives of ϕ, if at all possible, requires specific analytic development for each Building Block. Since we want our methodology to be as generic and as "extensible" as possible, the inclusion of a new Building Block must remain an easy task. This argues against derivative- based techniques.

Then, we looked at iterative "descent techniques". Since ϕ is in general not differentiable, we did not consider algorithms based on numerical approximations of gradients, and decided to focus on "Derivate Free Optimization" (DFO) methods [30, 13]. DFO methods are purpose-built for optimization problems in which the objective function is not easy to compute and has no particular simplifying properties. In such cases, DFO methods have been shown to frequently outperform techniques based on derivatives or on approximations of gradients [29]. The DFO methods make no assumptions on the objective function and thus simplify the inclusion of new Building Blocks. For these reasons, we decided to use a DFO method to perform the calibration process of our automatic approach.

As many numerical methods, DFO techniques require that all the parameters of the objective function be continuous, whereas some of the parameters of our Building Blocks are discrete (e.g. the number C of servers in an M/M/C or the size K of the buffer of an M/M/1/K). To treat all Building Block parameters as continuous, we define "intermediate models" in which discrete parameters are replaced by their corresponding continuous extensions. These intermediate models coincide with "standard" models when their "artificial" extension parameters have integer values. This step consists actually in relaxing the discrete constraints on the parameters.

As an example, we define an M/M/C block where *C* is real, e.g. $C = 2.87$, just by redefining the departure rates of the associated birth and death process as follows: from state 1 to state 0, the rate of service is μ, from state 2 to state 1, the rate is 2μ, and from any state $n > 2$ to state $n-1$, the rate is 2.87μ. Obviously, this intermediate M/M/C block coincides with the traditional queue when *C* has an integer value. Sometimes the intermediate models require a more involved derivation. As an illustration, for defining an M/M/1/K block with a real value of *K*, e.g. $K = 7.35$, one must rearrange the associated Markov chain by adding a new state K+1 and adjusting the arrival rate between state K and state K+1 proportional to $K - \lfloor K \rfloor$, e.g. 0.35λ. Note that these intermediate models significantly enhance the spectrum of behaviors covered by the Building Blocks, and can be used "as is" with their continuous parameters as laureate models. On the other hand, if only "classical" models (with discrete values for the appropriate parameters) are required as laureate models, we can evaluate all the models obtained from the intermediate laureate model, rounding the artificial continuous parameters to integers, and keep the best one (i.e. the one that results in the smallest error). In certain cases, it may be preferable to perform a further optimization search starting from each rounded (ceiled or floored integer) set of parameters that are kept constant, and run the DFO technique on the remaining continuous parameters. It is worthwhile noting that these further searches imply only a small additional complexity as the number of parameters of the search is smaller and the starting point has a good chance to be close to the optimum.

DFO methods proceed by neighborhood search. At any given iteration *i*, the DFO technique starts from a current solution x_i, which corresponds to a set of parameters for the considered Building Block, and tries to compute a new solution x_{i+1} with a lower level of ϕ. There are many possible approaches for developing DFO methods. These approaches differ mainly by their strategy to find a better solution at each step. Overall, the successive x_i points in the iterative search give a path towards a minimum of the hypersurface. Figure 11 shows an illustration of such a path for a given Building Block with two parameters. Clearly, like in many other optimization methods, the attained minimum might not be global.

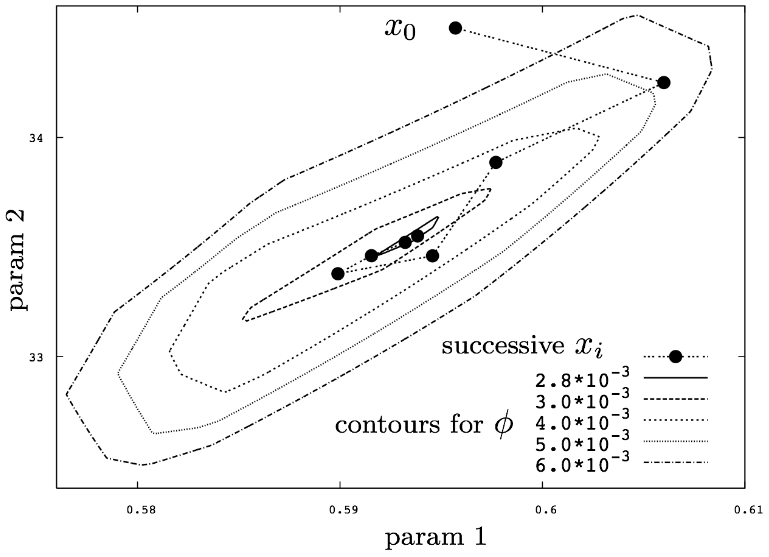

Figure 11: An example of the iterative descent through a DFO method.

To speed up the iterative search process, we enhance the approach by using a local "substitution model". At a given step *i* of the search process, we approximate the neighborhood of the current solution x_i with a substitution model that is a local mathematical approximation of the real hypersurface defined by ϕ. More specifically, our DFO method is based on a local quadratic approximation (that is asymptotically true for any "regular surface"). The use of a simple mathematical approximation enables us to drastically speed up the convergence towards the minimum, as it not only gives information about the direction of the minimum, but also about the distance to reach it. Of course this information is only an approximation, but as the process gets closer to the solution, the substitution model approximates better the surface and drives us faster to the minimum.

Since ϕ is in general not convex, a diversification phase is driven to avoid being trapped in a local optimum. This phase is based on multi-starting point runs. The search has also been strengthened by the inclusion of bounds and constraints on parameters that are derived from elementary properties on the Building Blocks. For instance, in case of an M/M/C/K block, the average sojourn time is necessarily between $1/\mu + Ofst$ and $K/C\mu + Ofst$. In order to enforce the compatibility between the measurements data set and the behavior of the model, we thus have to restrict the search space of the parameters to match these constraints.

Additional details of our DFO algorithm, as well as an analysis of its performance will be the object of a forthcoming paper. Preliminary experiments indicate that the proposed search method tends to be robust and very fast for Building Blocks with a limited number of parameters (say, up to 5 or 6), which is actually the case for all the Building Blocks we have tested so far (the calibration of any Building Block usually requires no more than 10^4 evaluations of ϕ and most often even considerably less). The intrinsic computational complexity of our method tends to grow slightly faster than the square of the number of parameters. Note that in our comparisons of the proposed approach to general DFO solvers (Solvopt [19], Appspack [16]) our method seems to be significantly more robust at the expense of a limited overhead in its search time.

Once the calibration phase has been performed for each Building Block, we must choose which parameterized Building Block is elected, if any, as the laureate model. In the current form of our method, the same estimator of goodness of fit, namely ϕ, drives this decision. For each Building Block, the DFO search ends up with a given calibration and a corresponding value of the error ϕ. We use these final error values to determine a ranked list of Building Blocks, and the first in line is the laureate model. In practice, when several Building Blocks in our ranked list exhibit comparable levels of error, as suggested by the principle of parsimony [6], one should give preference to the simplest model (in terms of the number of parameters and computational complexity) instead of the laureate model.

Finally, it is interesting to consider the impact of different error criteria during the calibration of the Building Blocks. First, the calibration found by our DFO method may differ according to the error criterion used since the "best" set of parameters values for a given error criterion is not necessarily the best set for another error criterion. In most cases, the final results of the method, however, as mentioned before, seem to be remarkably stable. Second, perhaps more interestingly, we have noticed during our experiments that the more relevant the error criterion is, the easier the calibration process is. Conversely, the choice of a less appropriate error criterion tends to increase, among other undesirable things, the number of local optima in the hypersurface considered, making the optimization search harder. For that reason, in certain cases, choosing a more relevant but computationally costlier error criterion (e.g. the area based error criterion mentioned in Section 2.5), enhances significantly the robustness, as well as the overall computational efficiency of the calibration search.

2.7 Requirements for the methodology

Measurements represent a key component for our approach. To be of use, the sets of measurements must satisfy certain common sense conditions.

First, the different measurement points from a particular set must come from the same system, and correspond to varying load levels. As a result, it makes sense to require that key parameters of the system stay identical for every measurement point or vary in a "non-random" way as a function of the workload.

Second, in our view, the system resources can be shared by two types of request streams: the one directly captured in the available measurements (captured traffic) and the "uncaptured" or "background traffic". In a large computer network, a significant part of the traffic may be processed without being directly captured by measurements. Since the background traffic competes for shared system resources, the common sense condition discussed above requires that the background traffic be either negligible, constant, or in a clear relationship to the captured (measured) traffic for all measurement points.

Third, the available measurement data must adequately capture the salient features of system behavior in the range of interest. Clearly, for instance, if the system response exhibits an inflection point and this inflection point is not present in the measurement data, there is little chance that the model proposed by our approach will correctly reproduce such a behavior.

The actual number of data points needed to obtain reasonable results depends on the nature of the system behavior and how well the available measurement points "cover" the domain of interest. In practice, for many systems 4 to 7 points tend to be sufficient. In some cases, more points may be needed to capture the system behavior.

3 Case Studies

3.1 Preliminaries

The proposed approach aims at finding a model whose performance parameters match as closely as possible those known from system measurements. Recall that we refer to this best match (in terms of the error function described in Section 2.5) as the laureate model. As discussed before, this laureate model is chosen from a set of pre-defined more or less simple Building Blocks. We explore the Building Blocks one by one, and we attempt to calibrate each block considered. Recall that by calibrating a Building Block we mean selecting a set of values for the parameters of the given block to minimize the error function.

In addition to simply matching the data points in the measurement set, we want the laureate model to be able to correctly predict the performance of the system within some reasonable domain. Therefore, in the case studies that follow we deliberately remove one or more data points from the measurement sets. Having found the laureate model for a given data set, we then test the ability of this model to predict the system performance at the removed data points. In actual application of our method one would of course tend to use all available measurement points. Our data sets come from actual real-life systems and include wireless networks, I/O controllers, an Ethernet network, as well as a multiprocessor system.

As discussed earlier (see Section 2.6), we calibrate the Building Blocks using a numerical method. Thereby, the models produced by the proposed approach may have continuous values for traditionally integral parameters.

3.2 Broadband Wireless Network

We start by considering the high-speed wireless network for which Quintero et al. give a set of performance measurements in [31]. The measurement points, shown in Figure 12, relate packet throughput to queueing delays experienced by packets in high load scenarios. As mentioned before, we remove some number of measurement points from the measurement set during the search for the laureate model. It is apparent from the shape of the delay time curve in Figure 12 that, in this case, the point for the highest load level is likely to be most difficult to reproduce accurately. We elect to remove precisely this point from the measurement set in order to test the predictive capabilities of the laureate model.

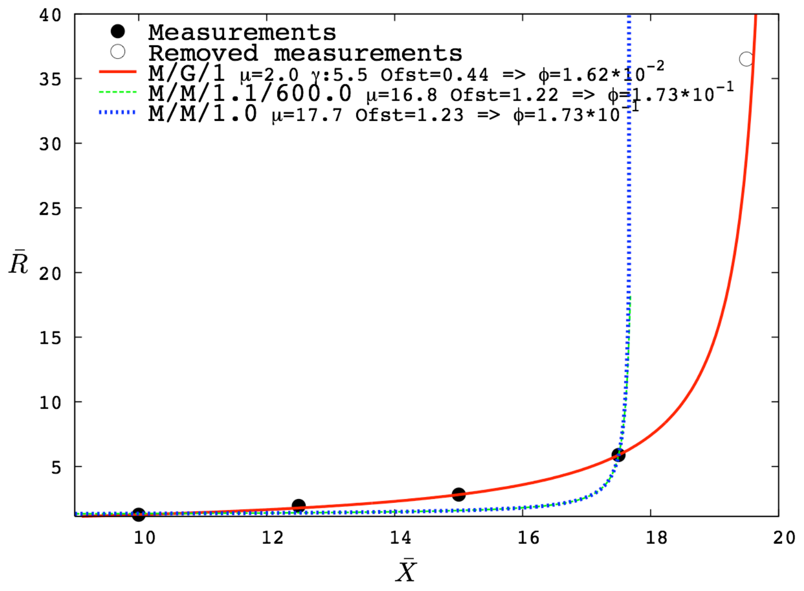

Figure 12: Broadband Wireless Network.

Figure 12 shows that a simple M/G/1 queue with adequate parameters determined by our approach, viz. $\mu = 20.03$, $\gamma = 5.5$ and $Ofst = 0.44$, closely approximates the observed performance for this system. We have also represented in Figure 12 the results of the "best" (in terms of smallest error) M/M/C and M/M/C/K queues (these two curves are so close that they are difficult to tell apart). We notice that neither of these two queueing models is able to correctly reproduce the measured system behavior.

The laureate M/G/1 model provides also a reasonable prediction for the removed point. When comparing the expected sojourn times for the throughput level of the removed point, we observe a relative error of 15%, while a comparison of expected throughputs at the same mean sojourn time for the removed point yields a relative difference of less than 1%. Given the steep slope of the performance curve in the vicinity of the removed point, we view the attained accuracy as more than reasonable. Not surprisingly, we note that the performance predictions of the M/M/C and M/M/C/K Building Blocks are poor. If we remove other, randomly selected points, and repeat the calibration procedure, we find that the laureate M/G/1 model yields predictions whose relative errors are all below 5%.

3.3 Disk controllers

In this Section, we investigate several sets of measurements obtained for disk controllers in mainframe environments. The measurement points give the expected I/O response time as a function of the attained I/O throughput (in requests per time unit).

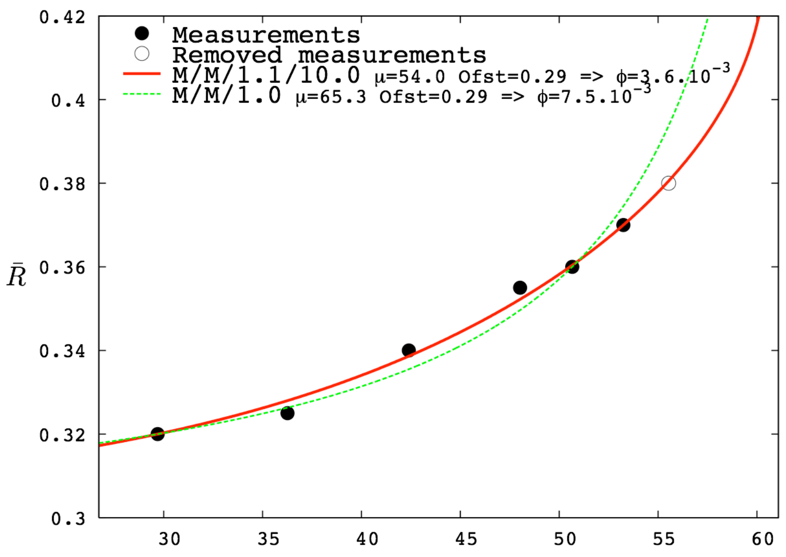

Figure 13: Disk controllers - Set 1.

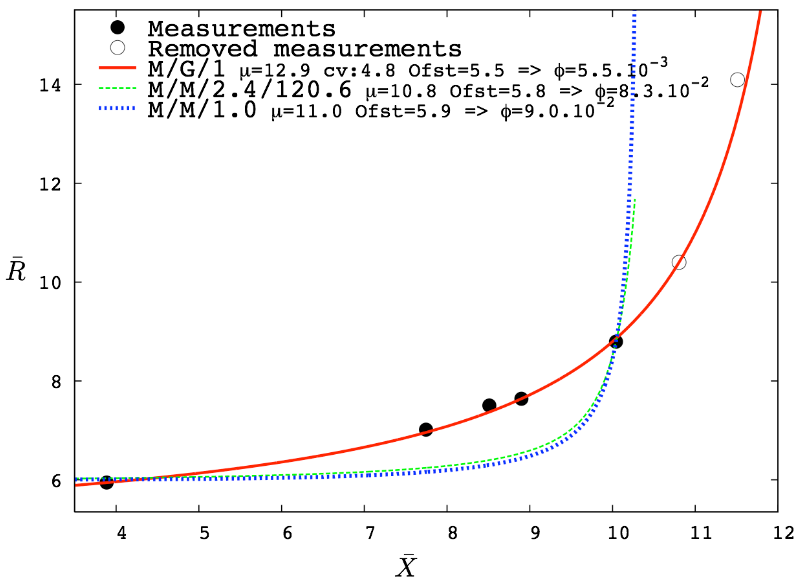

Figure 14: Disk controllers - Set 2.

Figures 13 and 14 illustrate the results obtained for measurement Set 1 and 2, respectively. We observe in Figure 13 that the “best” model - among the Building Blocks considered - in this case is the M/M/C/K queue, while for the measurement set of Figure 14 it is the M/G/1 queue. In both cases, the laureate models not only closely reproduce the data points used in the calibration process, but are also able to adequately predict the performance for the removed data points. The relative prediction errors for the removed data points are below 1% both for the expected throughput and the mean I/O time in Figure 13, and below 5% in Figure 14. We note that removing different points during the calibration process leads to similarly successful predictions by the laureate models.

It may be of interest in building a constructive model of I/O performance for the controller considered that for Set 2 neither the M/M/1 (M/M/C) nor the M/M/1/K (M/M/C/K) models appear adequate. Our results show that even with the best possible combination of parameters these two Building Blocks fall short from matching the observed performance.

In addition, as mentioned before, the resulting laureate model may be of help in capacity planning. For example, the analyst can use the laureate model to forecast the system behavior in face of projected growth in the I/O workload. Consider for instance, measurement set 1 (depicted in Figure 13), with the help of the laureate model, we forecast that in the vicinity of the last removed measurement point (I/O rate of around 55.5), an increase in the I/O rate of, say, 5% will only cause a 5% degradation in the system average response time. By contrast, in case of measurement set 2 (depicted in Figure 14) an increase of 5% in the I/O rate in the vicinity of the last removed point (I/O rate 11.5) will result in a 37% surge of the mean response time.

3.4 Wireless network (IEEE 802.11)

We now consider the wireless network (conforming to IEEE 802.11) described and measured by Chandran-Wadia et al. [11]. The available WLAN capacity in this network is 2 Mbps, and its behavior has been characterized for two different packet lengths: 500 and 1500 bytes. Measurement points give the expected sojourn time of a packet inside the WLAN and the corresponding mean throughput at the WLAN. We have two sets of measurements, one for each packet length.

As one could expect, the behavior of the WLAN considered depends on the packet size. Since the service time of a packet in this network (like in many other communication and computer systems) includes a fixed incompressible overhead, using shorter packets reduces the transfer time of a packet but also the achievable network throughput. This tradeoff between minimizing delay versus maximizing network throughput has been extensively studied.

To represent the effect of the size of packets on the WLAN behavior, we assume that the intrinsic service time in our Building Blocks $1/\mu$ can be expressed as

$$\frac{1}{\mu} = S_0 + \frac{U}{cap} \tag{3}$$

where U is the length of a packet in bits (units of work considered), S_0 is the fixed overhead expressed as a time, and cap denotes the processing capacity of the server in terms of units of work per time unit. A similar representation of the service time may be of interest in other applications such as I/O subsystems, virtual memory, file systems, etc. In our case, two parameters, S_0 and $capa$, are required to define the service-time for a given packet size. Clearly, the use of a formula like (3) implies some knowledge of the system and a bit of constructive modeling.

Figure 15 illustrates the results obtained for this wireless network with 500 and 1500 byte blocks. Note that the throughput in Figure 15 is expressed in requests per time unit and not in bits or bytes per time unit.

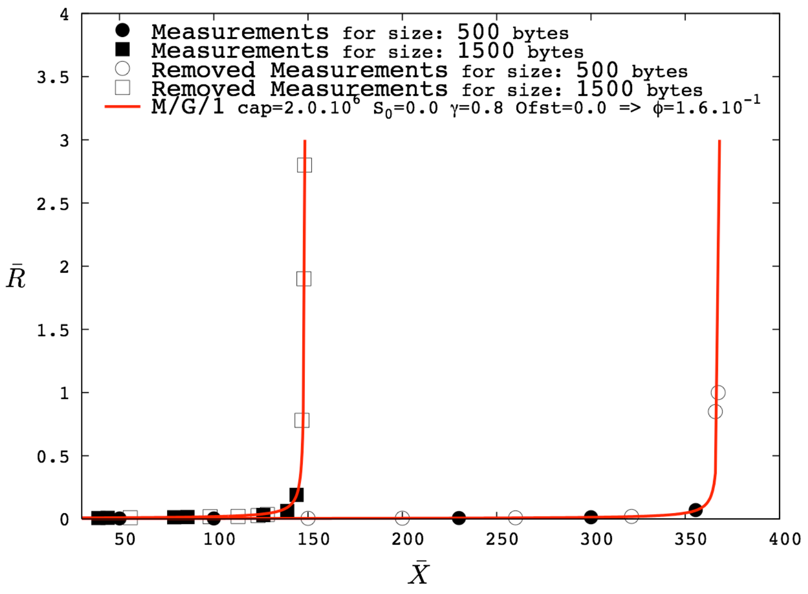

Figure 15: Wireless network

We observe that an M/G/1 queue (with $capa = 2.0 \times 10^{+6}$, $S_0 = 7.0 \times 10^{-4}$, and $\gamma = 8.4 \times 10^{-1}$) adequately reproduces the WLAN behavior for both packet sizes. As before, we tested the predictive capability of the laureate model by removing some number of measurement points from the search and calibration process. As shown in Figure 15, the laureate M/G/1 yields accurate predictions for the removed data points. It is worthwhile noting that removed points in this case represent over 50% of the available data points, and they include data points at higher workload levels for which accurate predictions can be expected to be more difficult.

3.5 Ethernet Network

In this example, we consider the Ethernet network with a nominal rated capacity of 10 Mbps described and measured by Wang and Keshav [35]. The performance of this network has been measured for three packet sizes: 64, 512 and 1500 bytes. Thus we have three measurement sets, one per packet length. The data points in each set give the expected sojourn time and the corresponding average packet throughput in the network. As we did in Section 3.4, to account for the dependency between the Ethernet behavior and the packet size, we assume that the service time requirement can be expressed using formula (3).

Objectives of this case study are twofold. First, we show that a simple model can adequately represent the performance of this Ethernet system. Second, we show that, if we use only two of the three measurement sets, our laureate model is able to correctly predict the performance of this network for the third packet size, not used to calibrate the laureate model.

Figure 16 illustrate the first objective for the Ethernet network considered. We observe that a simple M/G/1 (with $cap = 9.7 \times 10^{+6}$, $S_0 = 1.4 \times 10^{-4}$ and $\gamma = 6.0 \times 10^{-1}$) is well suited to reproduce the measured system behavior for different throughputs and packet sizes.

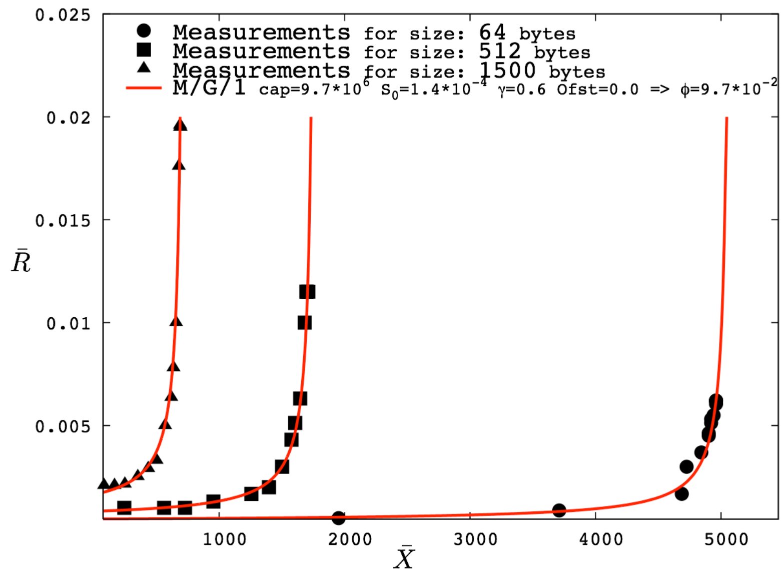

Figure 16: Ethernet network.

To illustrate our second objective, we remove one of the data sets (corresponding to one of the packet sizes) from the model search and calibration procedure. As an example, we remove the measurement set for 64 byte packets, and we search for the "best" model. We find that the resulting laureate model is very close to the one obtained previously (viz. an M/G/1 queue with $cap = 9.8 \times 10^{+6}$, $S_0 = 1.4 \times 10^{-4}$ and $\gamma = 5.3 \times 10^{-1}$). Therefore, it is not surprising that the laureate model correctly predicts the performance of the network for the "missing" packet size of 64 bytes. Similar experiments where we remove the measurement set for 512 and 1500 bytes, respectively, yield the same result. This is illustrated in Figures 17 (a) - (c).

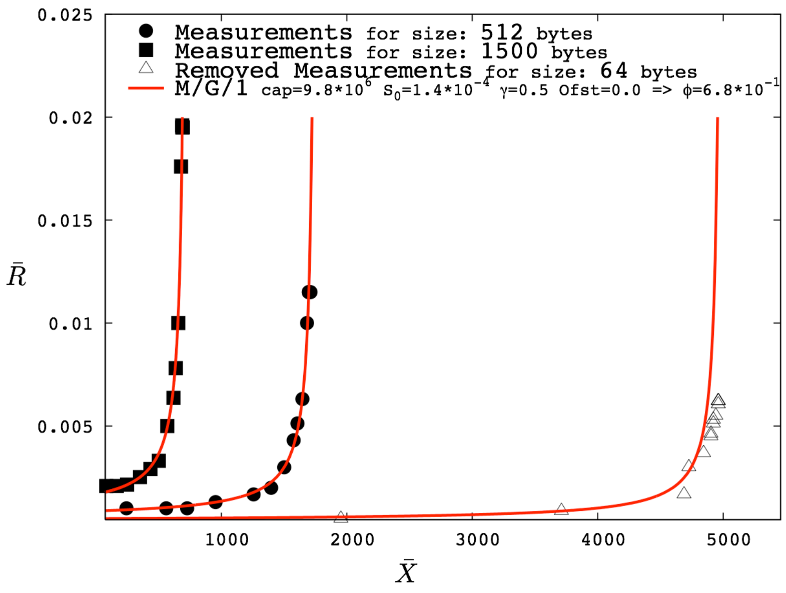

Figure 17 (a): Ethernet network – only sets for 512 and 1500 bytes packets are used for calibration.

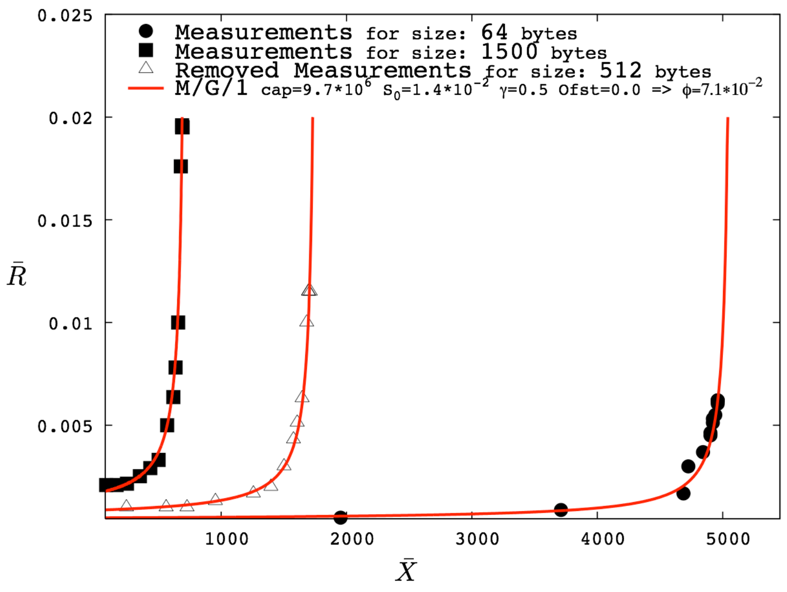

Figure 17 (b): Ethernet network - only sets for 64 and 1500 bytes packets are used for calibration.

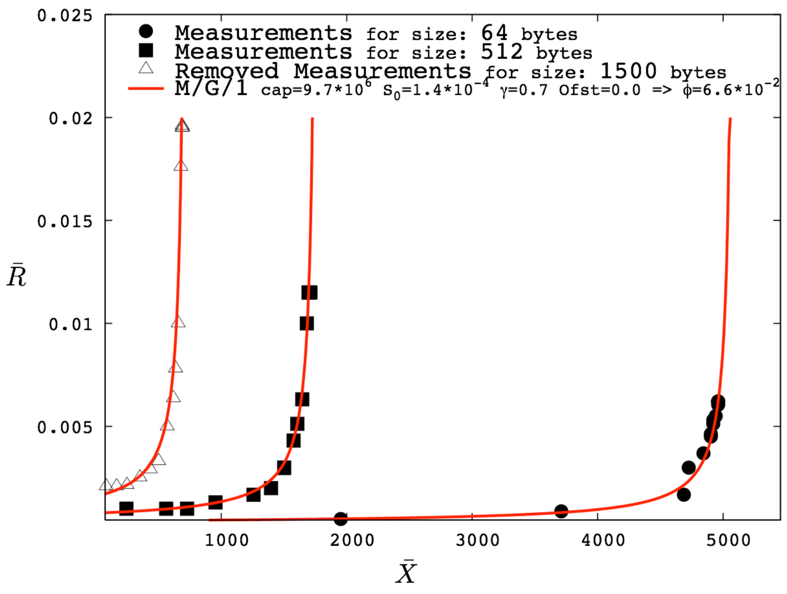

Figure 17 (c): Ethernet network - only sets for 64 and 512 bytes packets are used for calibration.

It is important to note that, in general, the execution times for our approach are quite short (typically they fluctuate between a few seconds and one minute on a medium-speed processor). Thus, to the extent that the network can be adequately represented as one of the Building Blocks, and we have measurement sets for two different packet sizes, the laureate model provides a convenient way to approximately determine the optimal packet size in a specific application.

Clearly, in some systems, the real dependence of the service time on the request size may be more complex than the one given in formula (3). The successful prediction of performance for two different packet sizes in Section 3.4, as well as the results presented here, suggest that, at least for these two types of system, formula (3) is adequate.

3.6 More I/O performance data

In this section, we look at four additional measurement sets obtained from mainframe I/O controllers for distinctly different workloads. As in Section 3.3, the measurement points give the expected I/O sojourn time as a function of the attained I/O throughput. Figure 18 illustrates the results obtained for these four measurement sets. Consider first the measurement points represented in Figure 18 (a). Although at first sight the measured I/O response time curve looks deceptively simple, a closer inspection reveals a behavior pattern that cannot be reproduced by any of the simple Building Blocks such as the M/M/C/K, M/G/C or the M/G/1/K models. Indeed, the expected response time starts out at a value that is much too low to lead to saturation at the throughput value indicated by the measurements. A possible explanation is that the mean service time itself increases as the attained I/O throughput increases. We observe that our Embedded Queue block, defined in Section 2.3, nicely reproduces the observed performance for this measurement set. Figures 18 (b) and (c) show similar results for other two sets of data.

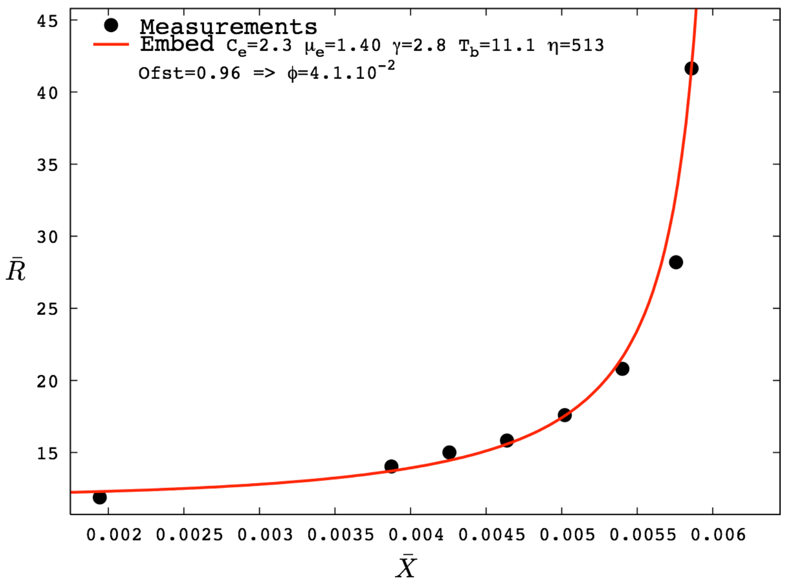

Figure 18 (a): Performance for an I/0 controller.

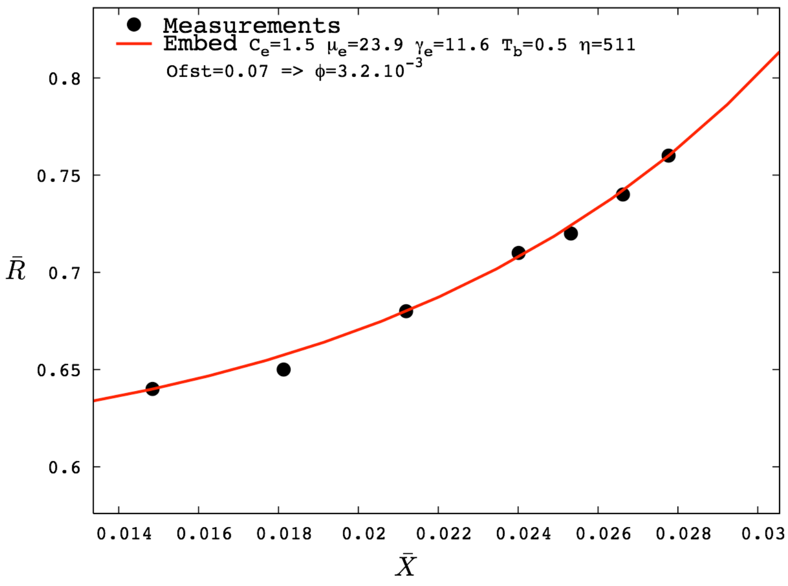

Figure 18 (b): Performance for an I/0 controller.

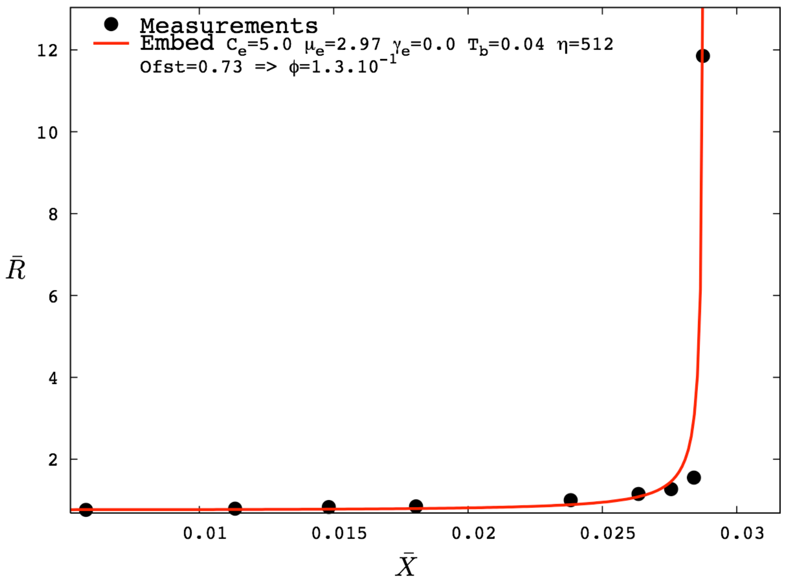

Figure 18 (c): Performance for an I/0 controller.

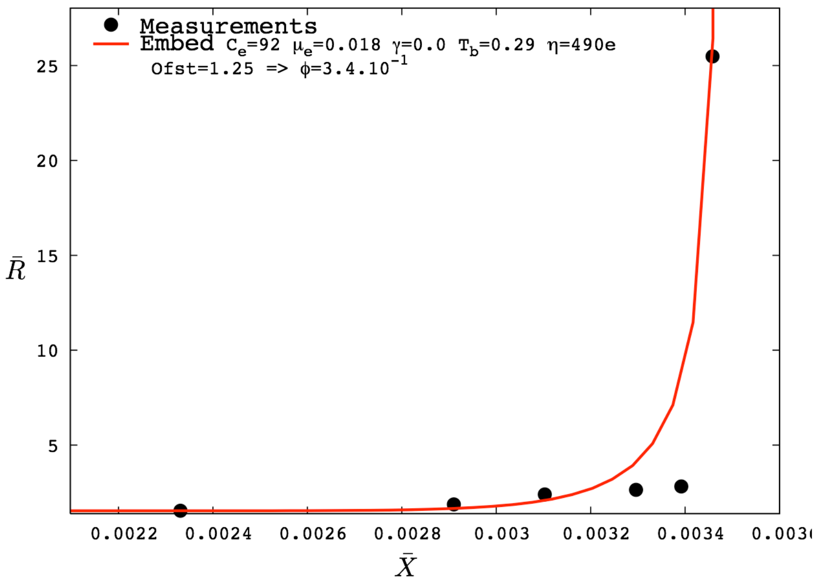

Figure 18 (d): Performance for an I/0 controller.

The measurement set represented in Figure 18(d) exhibits what appears to be an extremely steep saturation pattern. Clearly, no simple Building Block among the ones considered comes even close to matching the measured response times. The Embedded Queue yields a reasonable albeit far from perfect representation. It is possible that the points for the highest throughput are not accurate in this case or that the system cannot be viewed as having attained a steady state. It is also possible that a different type of modeling block is needed to deal more accurately with this type of behavior.

Overall, we conclude that the Embedded Queue block is a good representation for what appears to be a load-dependent service time elongation.

3.7 Multiprocessor system

We now turn our attention to an operating system in which the number of physical processors available to processes varies dynamically according to the system load level. The data points in the measurement sets in this case include the expected process response time for a given attained process throughput. Figure 19 illustrates the measurement data for just such a system. We observe that the response time exhibits a “discontinuity” in that it starts by increasing, and then decreases before increasing again as the attained throughput increases. Clearly, a "classical" queueing model such as the simple M/M/C, M/M/C/K or M/G/C queues cannot exhibit this sort of behavior. Our M/M/C_v Building Block (which corresponds to an M/M/C queue in which the number of servers changes if the load exceeds a certain threshold) was designed with this type of system in mind. As shown in Figure 19, it fares well in reproducing the observed system performance.

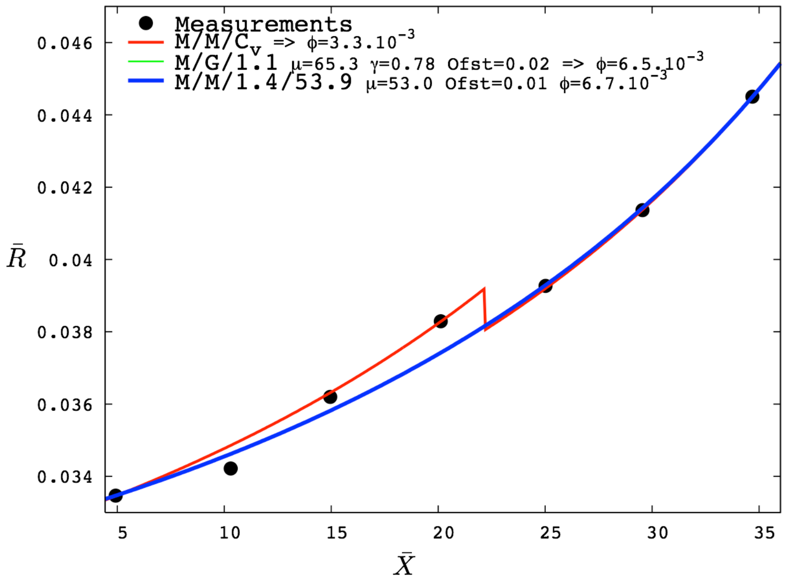

Figure 19: Performance for a multiprocessor system.

4 Conclusions

We have presented a high-level modeling approach based on measurement data. Unlike in constructive modeling, we don't seek to represent "explicitly" the structure of the system being studied. We focus on the measurement results, and attempt to discover a more or less elementary model that might correctly reproduce the observed behavior. We identify a few obvious classical queueing models as possible Building Blocks for our approach. We also define a couple of new, not-so-classical Building Blocks: a model in which the service time increases in response to congestion at a hidden (embedded) resource, as well as a model of a system in which the number of servers varies accordingly to the level of load in the system. Using several sets of measurements from real computer and communication systems, we have shown that our Building Blocks are not only able to reproduce the observed system behavior, but have also some predictive power.

We embed the search for a best fitting model in an efficient Derivative Free Optimization (DFO) procedure. We discuss in some detail the reasons for this choice, focussing on the steps involved in a successful search for a suitably calibrated model. We highlight the main pitfalls that may be specific to our domain such as the choice of a well-suited error criterion. In particular, we point out that the error function given by (2) has a tendency to assign an undue importance to differences between models in the vicinity of server saturation. The speed and the efficiency of this approach allow us to automate the search for the best fitting Building Block. Note that, owing to the use of DFO methods, our approach is not limited to the particular performance measures used in this paper. Other performance measures could be used as long as the Building Blocks considered can be solved for the selected performance indices.

Our main contribution lies in the definition of new Building Blocks and the automation of the search for the laureate model (more generally, a ranked list of the best possible models). Taken together, these two allow us to represent a wide range of behaviors. Since the search for a laureate model has been automated, performance analysts with a minimal queueing network background can use the resulting tool. It is worthwhile noting that, in addition to the laureate model, our tool can produce the next best candidate (from another Building Block), which may be of interest in some situations.

Our approach has limitations. Since it is based on measurement data, the system considered (or a detailed constructive simulation model of the system) must exist, and there must be a sufficient number of measurement points to adequately capture the behavior of the system. In general, there is no guarantee that our approach will find an adequate model, and a failure of the approach does not necessarily imply that there is no adequate simple model for the given system.

While the laureate model determined by our approach may be a good starting point for constructive modeling or for a search for a good approximation, a potential drawback of our approach is that there is in general no clear readily seen relationship between the parameters of the laureate model and the "natural" parameters of the corresponding constructive model. This limits also the predictive application of the laureate model in that it is not clear how the parameters of the laureate should be modified to reflect a change in the characteristics of the system being modeled.

Despite the above limitations, our method has several advantages. First, the laureate models obtained from our approach are useful to predict performance at workload levels for which measurements may not have been obtained. Hence, our approach may be of help in predicting whether the system considered fulfills or fails to fulfill a quality of service requirement. For instance, based on a projection of the growth in workload, the laureate model provides a quick answer whether a given allowed threshold-value for the average response time will be satisfied or not by the system. Unlike the "classical" constructive approach (such as a proposed framework for e-business applications [3] or disk arrays [34]), our approach reaches this goal without investigating the internal behavior of the system. Second, the nature of the best-fitting Building Block may be of help for constructive modeling of the system. Furthermore, it may provide guidance in the search for simple approximations, by indicating which Building Block might and which ones might not work. Third, the laureate model may reveal hidden information on a system such as a dramatic elongation in its service time as the load increases or an unexpected bottleneck which reduces the intended degree of parallelism in the system. Overall, our approach, when successful, proposes a compact but complete model of congestion and queueing, which we believe to be superior in most cases to the use of an arbitrary mathematical interpolation (e.g. polynomial, Bessel) between the known measurement points.

The particular DFO search method we use tends to be robust and fast as long as the number of Building Block parameters does not exceed 5 or 6. With a larger number of parameters, the search tends to slow down visibly. It is possible that another DFO method might outperform the one we use but at this time, none of those experimented does.

There are several open issues currently under investigation. First among them is the system behavior at or near server saturation, such as shown in Figure 14 (d). It is possible that a generalization of the embedded queue model to include losses would be a good Building Block for this type of behavior. Second, although we have considered measurement sets exhibiting inflexion point(s), we would like to test the robustness of our method on additional real-life system measurements representing other types of "unusual" behavior. Third, in order to demonstrate the extensive applicability of our approach, we are also looking for sets of measurements comprising the loss probability for requests in addition to the average throughput and the average sojourn time.

Overall, we believe that, packaged as a ready-to-use tool, our approach can be of significant value both to the performance analyst in a capacity planning situation, and to the performance modeler in general.

5 Acknowledgments

We would like to thank Safia Kedad and Francis Sourd from UPMC - LIP6 for their valuable help in designing the specific Derivative Free Optimization method we implemented in our automatic model calibration tool.

6 References

[1] Allen, A. O. *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Academic Press, 2nd edition, 1990.

[2] Alouf, S., Nain, P., and Towlsey, D.F. Inferring network characteristics via moment-based estimators, in *INFOCOM*, 2001, pp. 1045-1054.

[3] Bacigalupo, D.A., Turner, J.D., Graham, R. N., and Dillenberger, D. N. A dynamic predictive framework for e-business workload management, in *7th World Multiconference on Systemics,Cybernetics and Informatics (SCI2003) Performance of Web Services Invited Session*, Orlando, Florida, USA, 2003.

[4] Begin, T., Brandwajn, A., Baynat, B., Wolfinger, B., and Fdida, S. Towards an Automatic Modeling Tool for Observed System Behavior, in *European Performance Engineering Workshop (EPEW)*, 2007, pp. 200-212.

[5] Boxma, O.J., Cohen, J.W., and Huffels, N. Approximations of the Mean Waiting Time in an M/G/s Queueing System. *Operations Research*, Vol. 27, 1979, pp. 1115-1127.

[6] Burnham, K.P., and Anderson, D. R. *Model selection and multimodel inference: a practical information-theoretic approach*. 2nd Edition. Springer-Verlag, New York, New York, USA. 2002.

[7] Brandwajn, A. Equivalence and decomposition in queueing systems - a unified approach. *Performance Evaluation*, Vol. 5, 1985, pp. 175-186.

[8] Brandwajn, A., and Jow, L.J. A note on service interruptions. In *Proceedings of the 1985 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, B. D. Gaither, Ed. SIGMETRICS '85, 1985

[9] Brandwajn, A., and Wang, H. A Conditional Probability Approach to M/G/1-like Queues. To appear in *Performance Evaluation*, 2007.

[10] Cao, J., Andersson, M., Nyberg, C., and Kihl, M. Web server performance modeling using an M/G/1/K*PS queue. In *ICT'2003: 10th International Conference on Telecommunications*, Vol. 2, 2005, pp. 1501-1506.

[11] Chandran-Wadia, L., Mahajan, S., and Iyer, S. Throughput performance of the distributed and point coordination functions of an IEEE 802.11 wireless LAN. In *ICCC '02: Proceedings of the 15th International Conference on Computer Communication*, 2002, Washington, DC, USA, pp.36-49.

[12] Cong, J., and Wolfinger, B.E. A unified load generator based on formal load specification and load transformation. In Proceedings of ValueTools 2006: International Conference on Performance Evaluation Methodologies and Tools, 2006, Pisa, Italy.

[13] Conn, A.R., Scheinberg, K, and Toint, P.L. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, Vol. 79, 1997, pp. 397-414.

[14] Crovella, M., and Krishnamurthy, B. *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, Inc, 2006.

[15] Dovrolis, C., Ramanathan, P., and Moore, D. What do packet dispersion techniques measure? In *INFOCOM*, 2001, pp. 905-914.

[16] Gray, G. A., and Kolda, T.G. Algorithm 856: APPSPACK 4.0: Asynchronous Parallel Pattern Search for Derivative-Free Optimization. *ACM Transactions on Mathematical Software*, Vol. 32, 2006, pp. 485-507.

[17] Heidelberger, P., and Lavenberg, S.S. Computer performance evaluation methodology. *IEEE Trans. Computers*, Vol. 33, 1984, pp. 1195-1220.

[18] Jackson, J.R. Networks of Waiting Lines. *Operations Research*, Vol. 5, 1957, pp. 518-521.

[19] Kappel, F., and Kuntsevich, A.V. An Implementation of Shor's r-Algorithm. *Computational Optimization and Applications*, Vol. 15, 2000, pp. 193-205.

[20] Kaufman, J.S. Approximation Methods for Networks of Queues with Priorities. *Performance Evaluation*, Vol. 4, 1984, pp. 183-198.

[21] Kelly, F.P. *Reversibility and Stochastic Networks*. John Wiley & Sons, 1979, New York.

[22] Kimura, T. Approximations for Multi-Server Queues: System Interpolations. *Queueing Systems: Theory and Applications*, Vol. 17, 1994, pp. 347-382 .

[23] Kleinrock, L. *Queueing Systems*, Volume 1: Theory. John Wiley & Sons, 1975.

[24] Kouvatsos, D.D., and Tabet-Aouel, N.M. Product-Form Approximations for an Extended Class of General Closed Queueing Networks. In *14th International Symposium on Computer Performance, Measurement and Evaluation*, Edinburgh, Scottland, 1990, pp. 301-315.

[25] Kühn, P. Analysis of Complex Queueing Networks by Decomposition. *In 8-th International Teletraffic Congress*, Melbourne, 1976, pp. 236-1 - 236-8.

[26] Liu, Z., Wynter, L., Xia, C.H., and Zhang, F. Parameter inference of queueing models for IT systems using end-to-end measurements. *Performance Evaluation*, Vol. 63, 2006, pp. 36-60.

[27] Nash, J. E. and Sutcliffe, J. V. River flow forecasting through conceptual models, Part I - A discussion of principles. *Journal of Hydrology*, Vol. 10, 1970, pp. 282-290.

[28] Paxson, V.E. *Measurements and Analysis of End-To-End Internet Dynamics*. Doctoral Thesis, University of California at Berkeley, 1998.

[29] Powell, M.J.D. Unconstrained minimization algorithms without computation of derivatives. *Bollettino della Unione Matematica Italiana*, Vol. 9, 1974, pp.60–69.

[30] Powell, M.J.D. A view of algorithms for optimization without derivatives. In *Cambridge NA Reports, Optimization Online Digest - June 2007.*

[31] Quintero, A., Elalamy, Y., and Pierre, S. Performance evaluation of a broadband wireless access system subjected to heavy load. *Computer Communications*, Vol. 27, 2004, pp. 781-791.

[32] Salamatian, K., and Fdida, S. A framework for interpreting measurement over Internet. In *Proceedings of the ACM SIGCOMM workshop on models, methods and tools for reproducible network research*, Karlsruhe, Germany, 2003, pp. 87-94.

[33] Scherr, A. *An Analysis of Time-Shared Computer Systems*. MIT Press, Cambridge, MA, 1967.

[34] Varki, E., Merchant, A., Xu, J., and Qiu, X. Issues and challenges in the performance analysis of real disk arrays. *IEEE Trans. Parallel Distrib. Syst*., Vol.15, 2004, pp. 559-574.

[35] Wang, J., and Keshav, S. Efficient and accurate Ethernet simulation. In 24th Annual IEEE International Conference on Local Computer Networks, 1999, Boston, MA, pp. 182–191.

[36] Wolfinger, B.E., Zaddach, M., Heidtmann, K.D., and Bai, G. Analytical modeling of primary and secondary load as induced by video applications using UDP/IP. *Computer Communications*, Vol. 25, 2002, pp. 1094-1102.