Probability in the Engineering and Informational Sciences, **32**, 2018, 396–408.
doi:10.1017/S0269964817000237

THROUGHPUT AND BOTTLENECK ANALYSIS OF TANDEM QUEUES WITH NESTED SESSIONS

A. Hristov, J.W. Bosman, R.D. Van der mei, and S. Bhulai
CWI, Stochastics,
Science Park 123, 1098XG Amsterdam,
The Netherlands
and
Department of Mathematics, Faculty of Sciences,
Vrije Universiteit Amsterdam, 1081HV Amsterdam
The Netherlands
E-mail: A.V.Hristov@cwi.nl; J.W.Bosman@cwi.nl; R.D.van.der.Mei@cwi.nl; S.Bhulai@vu.nl

Various types of systems across a broad range of disciplines contain tandem queues with nested sessions. Strong dependence between the servers has proved to make such networks complicated and difficult to study. Exact analysis is in most of the cases intractable. Moreover, even when performance metrics such as the saturation throughput and the utilization rates of the servers are known, determining the limiting factor of such a network can be far from trivial. In our work, we present a simple, tractable and nevertheless relatively accurate method for approximating the above mentioned performance measurements for any server in a given network. In addition, we propose an extension to the intuitive "slowest server rule" for identification of the bottleneck, and show through extensive numerical experiments that this method works very well.

Keywords: bottleneck identification, layered queueing network, nested sessions, performance analysis, tandem queue, throughput

1. INTRODUCTION

Networks in which the service time at a given server is dependent on the queueing behavior of other queues are called Layered Queueing Networks (LQNs). Such systems find application in various fields such as computer science (Franks et al. [7], Rolia and Sevcik [15]), health care (Yom-Tov and Mandelbaum [23]), telecommunication software systems (Shousha et al. [18]), and assembly lines (Buitenhek, van Houtum, and Zijm [3]). In this paper, we analyze a specific type of LQNs, which are characterized by having exactly one server per layer. We refer to such systems as tandem queues with nested sessions. As an example, consider a queueing network that models the customer dynamics at a gas station. Drivers have to first fuel their cars at the fuel pump and then pay to a cashier inside the station. In this case, there are two nodes, that is, servers: the fuel dispensers and the cashiers. Each one of these servers is characterized by a service rate and a number of available slots, that is, sessions. Note that the customers occupy a place at the first node (the fuel dispensers) throughout the whole procedure – clients leave and free the space at the fuel pump only after filling gas and paying the cashier. Therefore, the service speed of the second server (the cashiers) influences the sojourn time at the first node as well.

1.1. Approximation Approaches

The dependency between the various layers in an LQN may be significant and therefore must be taken into account. This makes the mathematical analysis of such networks challenging. Solutions based on approximate versions (Bard [1], Chandy and Neuse [4], Schweitzer [17]) of the mean value analysis algorithm (Reiser and Lavenberg [14] are researched in Franks et al. [7], Rolia and Sevcik [15], Woodsode et al. [22]). Other algorithms for approximating the performance metrics of LQNs include those developed in Herzog and Rolia [9], Tribastone [20,21], where stochastic process algebras are used for the analysis. Next to the above-mentioned approaches for generalized LQNs, there are a number of researches on special cases of such networks. A specific polling queueing system analyzed in Dorsman, Perel, and Vlasiou [5] can be considered as an instance of an LQN. Another example are systems with exactly two layers. Such networks are studied in Dorsman, van der Mei, and Vlasiou [6] by means of the power-series algorithm (Blanc [2]) and in Perel and Yechiali [11,12,13] with the help of matrix-analytic methods (Neuts [10]). The algorithm presented in our paper is approximating the performance metrics of an LQN without a restriction on the number of layers. However, each layer should contain exactly one server.

1.2. Bottleneck Identification

The goal of any bottleneck identification technique is to find the limiting factor for the performance of a given system. In queueing networks without a layered structure this task is rather trivial – the most loaded server is the one that is slowing down the system (Roser, Nakano, and Tanaka [16], Tregunno [19]). However, in an LQN it might be the case that more than one server is influencing the throughput. In such cases, we need a new definition of a bottleneck and a new technique to identify it. In this paper, by a bottleneck we refer to the server whose service rate modification results in the largest change in throughput for the system as a whole. This definition agrees with the one for traditional queueing models in case the network is not layered: the most loaded server is the one whose service rate modification is most influential for the throughput of the system. However, the dependency of the service times among servers with nested sessions turns the bottleneck identification in such networks into a far more complicated task. This is due to the fact that the most utilized server is not always the one that plays the biggest role in slowing down the system (Franks et al. [8]).

1.3. Contributions

The first contribution of this paper is the presented algorithm for approximating the performance of tandem queues with nested sessions. Given that the system is overloaded we obtain the throughput, that is, the saturation throughput, and the utilization rate of each server. As a second contribution, we formulate an extended definition for an LQN bottleneck. Finally, we introduce a novel layer-specific measure that could be used as a bottleneck identifier.

The reminder of the paper is organized as follows. Section 2 introduces the model that we consider throughout the paper and the algorithm that can be applied to such models in order to approximate the performance metrics. Furthermore, Section 3 extends the analysis of those systems by describing a method for bottleneck identification. The accuracy of the presented algorithms is shown in Sections 4 and 5 correspondingly. Finally, in Section 6 we discuss further research and state our conclusions.

2. THE MODEL

In this section, we present the LQNs that we consider in the paper. The model assumptions and the notation that is used throughout the paper are introduced in this section. Finally, we further elaborate on our definition of a bottleneck.

2.1. Saturated Tandem Queues with Nested Sessions

Figure 1 will serve as an example in this section. It models a system consisting of three tandem queues with nested sessions. In such a network, customers acquire service from all the nodes before leaving the system. They visit the nodes according to a predefined path, which is the same for all of them. Customers start at the first node and only after receiving full service there, they go to the next node (the path is from left to right in Figure 1). Customers free the resource they occupy at a given node only after they receive their service from all the nodes in the network. In comparison, in traditional tandem queueing networks without a nested structure, the customers free the resource at a node as early as they finish with their service at the node itself. Furthermore, the terminology used in systems with a nested structure differs from the one describing traditional queueing networks in a few important aspects. First, the nodes are referred to as servers. Second, the possible number of places that can be occupied by customers at those nodes are called sessions.

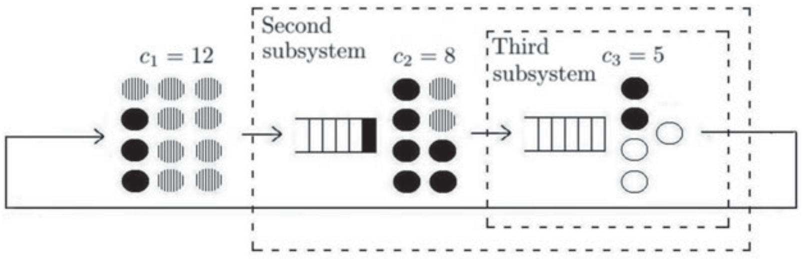
As we are interested in the maximum possible throughput that can be achieved for a given network and pinpointing the bottleneck in case a queue starts to build up, we analyze the system under the assumption that it is saturated. This implies that all sessions at the first server are constantly occupied. Therefore, we can model the system as a closed network with the number of customers equal to the number of sessions at the first node. Next to that, we assume the service time of server i to be exponentially distributed.

As input parameters for such models we use:

$N :=$ the number of layers in the network;

$c_i :=$ the number of sessions at the ith server, where $1 \leq i \leq N$;

$\mu_i :=$ the service rate at the ith server, where $1 \leq i \leq N$ and $\mu_i = 1/\beta_i$.



○ := session free

● := session occupied from customer being served at the specific layer

▥ := session occupied from customer being served at a lower layer

FIGURE 1. A three-layered model.

Furthermore, we denote the layer containing the first server as the top layer or equivalently the first layer. On one layer lower, that is, on the second layer, is situated the second server and the queue behind it, and so on. The network consisting of the ith layer, together with all the layers below it, is referred to as the ith subsystem.

To illustrate, we refer to Figure 1. In the presented network, there are $N = 3$ layers with a number of sessions per server: $c_1 = 12, c_2 = 8$, and $c_3 = 5$. The occupied sessions are marked with black or striped depending whether the user is requiring service from the specific server or waiting for a service from another node. For example, although all 12 sessions are occupied at the first server, only three of them are being served there, whereas the other nine are waiting for/receiving service from either the second or the third node. Therefore, in total there are nine customers at the second subsystem.

2.2. Performance Metrics

The goal of our approximation algorithm is to derive the following performance metrics:

$$\begin{aligned} T^{(i)} &:= \text{the saturation throughput of the } i\text{th subsystem;} \\ T^{(i)}_{\max} &:= \text{the throughput of the } i\text{th subsystem assuming no waiting times;} \\ U^{(i)}_{\text{norm}} &:= \text{the "normal" utilization rate of the } i\text{th server;} \\ U^{(i)}_{\text{eff}} &:= \text{the "effective" utilization rate of the } i\text{th server,} \end{aligned}$$

where:

- “normal” utilization is the average portion of sessions that are occupied at the corresponding server. The sessions can be occupied due to a request that currently receives service from the server or such that holds the session because it still needs to receive service at the sub-layers.
- “effective” utilization is the average portion of sessions that are occupied by requests that receive service from the corresponding server.

To show the difference between the “normal” and the “effective” utilization rate, we again refer to the system in Figure 1. Assume that the specific state depicted at Figure 1 represents the average number of sessions occupied in the long run across the three servers. To obtain the normal utilization of a given server, that is, $U^{(2)}_{\text{norm}}$, one has to calculate the fraction of the occupied sessions. This gives $U^{(2)}_{\text{norm}} = 1$. However, only six out of the eight customers at the second node are acquiring service there, whereas the other two are at the third server. Therefore, $U^{(2)}_{\text{eff}} = 0.75$.

Furthermore, we argue that one may assume $c_i \geq c_j$ for $i < j$ as, due to the layered structure of the network, the maximum number of sessions that can be occupied on a server j is $\min\{c_i \mid i < j\}$.

The models we consider have strictly one server per layer. Therefore, the sojourn time of a client at a given server i is exactly the sum of the waiting time at server i, w_i, the service time acquired at this node β_i, and the sojourn time in the $(i + 1)$th subsystem.

2.3. Bottleneck Analysis

Having obtained the maximum possible throughput of a given system, we determine what would be the most efficient way to increase this performance. We identify the bottleneck as the server that has the highest impact on the throughput of the network.

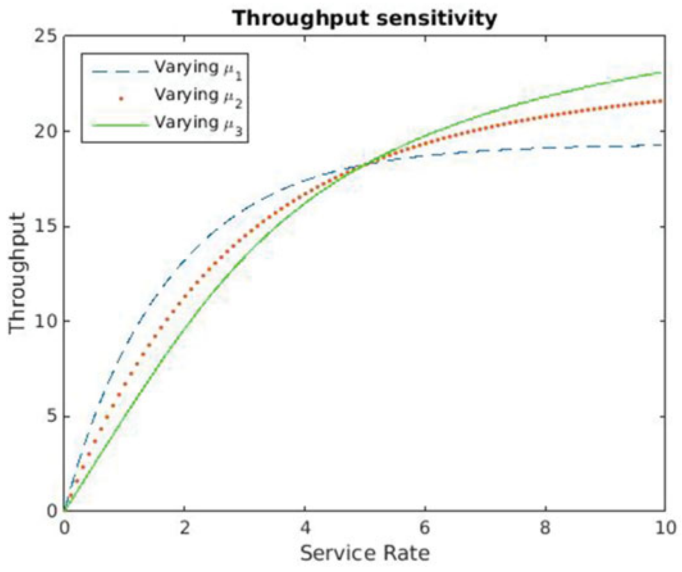


FIGURE 2. Throughput change in accordance to variation of the service rates.

As an example, we take the model in Figure 1 with $\mu_1 = \mu_2 = \mu_3 = 5$ as a baseline. We plot the throughput in the three cases of μ_1, μ_2, or μ_3 varying while having the other two parameters fixed at their baseline values. As it can be seen in Figure 2, changes in the service rate of the third node correspond to largest gains/losses of the overall throughput. Hence, we conclude that in this case the bottleneck is the third server.

3. PERFORMANCE ANALYSIS OF THE NETWORK

In this section, we describe our algorithm for approximating the performance metrics and our bottleneck identification technique for tandem queues with nested sessions. We present the method used to derive the throughput and the utilization rates by first applying it to a two-layered system. Second, we show how one can use a recursive scheme for approximating the performance metrics of a network with any given number of layers $N > 2$. Finally, we describe our bottleneck identification technique, which relies on the saturation throughput of the various subsystems.

3.1. Exact Solution for $N = 2$

In this section, we analyze the special case of a two-layered system. We model the network as a continuous time Markov chain with corresponding state space $\mathbf{S} := \{0, 1, 2, \ldots, c_1\}$, where state i denotes the total number of occupied sessions at the second server. One can easily verify that this Markov chain has a generator matrix $\mathbf{Q}$ with the following non-zero, non-diagonal entries:

$$q_{k,k+1} = \mu_1(c_1 - k), \qquad k = 0, 1, \ldots, c_1 - 1,$$
$$q_{k,k-1} = \mu_2 \min\{c_2, k\}, \qquad k = 1, 2, \ldots, c_1,$$

where $q_{i,k}$ is the transition rate from state i to state k. In addition, we define

$$\rho_k := \frac{q_{k-1,k}}{q_{k,k-1}}, \quad k = 1, 2, \ldots, c_1. \tag{1}$$

As the analyzed system is a Birth–death process, one can obtain the stationary distribution $\pi = \left[\pi_0, \pi_1, \ldots, \pi_{c_1}\right]$ as a product form:

$$\pi_0 = \left(1 + \sum_{i=1}^{c_1} \prod_{j=1}^{i} \rho_j\right)^{-1},$$

$$\pi_k = \pi_0 \prod_{i=1}^{k} \rho_i, \quad k = 1, 2, \ldots, c_1.$$

Furthermore, using the stationary distribution π, one can obtain the desired performance metrics in the following manner:

$$T^{(1)} = \sum_{i=0}^{c_1} \pi_i (c_1 - i)\mu_1, \tag{2}$$

$$U_{\text{eff}}^{(1)} = \sum_{i=0}^{c_1} \pi_i \frac{(c_1 - i)}{c_1}, \tag{3}$$

$$U_{\text{eff}}^{(2)} = U_{\text{norm}}^{(2)} = \sum_{i=0}^{c_1} \pi_i \frac{\min\{i, c_2\}}{c_2}. \tag{4}$$

3.2. Recursive Scheme for $N > 2$

We introduce a few notations that are used in this section:

- $\pi^{(i,j)}$:= the stationary distribution of the ith subsystem conditioned that it contains exactly j customers;
- $T^{(i,j)}$:= the saturation throughput of the ith subsystem conditioned that it contains exactly j customers;
- $U_{eff}^{(\ell,i,j)}$:= the "effective" utilization rate of the ℓth server in case the ith subsystem is studied in isolation and contains exactly j customers;
- $U_{norm}^{(\ell,i,j)}$:= the "normal" utilization rate of the ℓth server in case the ith subsystem is studied in isolation and contains exactly j customers.

In the following, we describe how one can approximate $\pi^{(i,j)}$, $T^{(i,j)}$, $U_{\text{eff}}^{(\ell,i,j)}$, and $U_{\text{norm}}^{(\ell,i,j)}$, for any given $1 \le i \le N-1$; $0 \le j \le c_{i+1}$, and $1 \le \ell \le i$. To analyze the ith subsystem in isolation we assume it to be saturated. Therefore, all the sessions on the ith server should be occupied. Hence, we take $\min\{j, c_i\}$ as the number of sessions at the ith server. Moreover, due to the assumption that there are an infinite number of customers in the ith queue, there is no difference in the networks with $j \ge c_i$. Therefore, in the following analysis we let $j \le c_i$.

The approach is similar to the one described in the case of a two-layered system. Again, we model the i-layered network as a continuous time Markov chain with a state space $\mathbf{S} := \{0, 1, 2, \ldots, j\}$, where the state denotes the total number of customers in the $(i-1)$-layered subsystem. In this case, however, we use the following approximation for the transition rates:

$$q_{k,k+1} = \mu_i(j-k), \quad k = 0,1,\dots,j-1,$$
$$q_{k,k-1} = T^{(i+1,k)}, \quad k = 1,2,\dots,j.$$

Having the transition rates, one can use those values in Eq. (1) to obtain ρ_k for $1 \le k \le j$. Furthermore, by the same method described in Section 3.1, one can derive the stationary distribution $\pi^{(i,j)}$. Once the vector $\pi^{(i,j)}$ is obtained, one can calculate the performance metrics for the corresponding (i,j) tuple as follows:

$$T^{(i,j)} = \sum_{k=0}^{j} \pi_k^{(i,j)}(j-k)\mu_i,$$

$$U_{\text{eff}}^{(\ell,i,j)} = \begin{cases} \sum_{k=0}^{j} \pi_k^{(i,j)} \frac{j-k}{c_i}, & \ell = i, \\ \sum_{k=0}^{j} \pi_k^{(i,j)} U_{\text{eff}}^{(\ell,i+1,j)}, & i < \ell \le N, \end{cases}$$

$$U_{\text{norm}}^{(\ell,i,j)} = \begin{cases} \frac{j}{c_i}, & \ell = i, \\ \sum_{k=0}^{j} \pi_k^{(i,j)} U_{\text{norm}}^{(\ell,i+1,j)}, & i < \ell \le N. \end{cases}$$

As it can be seen, the results for the ith subsystem depend on the values for the $(i+1)$-th one. Therefore, starting from the last, that is, the $(N-1)$th, subsystem one can obtain the desired metrics by the method for the two-layered network. Using the technique described above, one can further recursively derive the values for higher subsystems and eventually the ones for the initial system: $T^{(1,c_1)}$, $U_{\text{eff}}^{(\ell,1,c_1)}$, and $U_{\text{norm}}^{(\ell,1,c_1)}$ for $1 \le \ell \le N$.

3.3. Bottleneck identification technique

In this section, we explain our technique for bottleneck identification. We want to determine the server whose service rate is the most influential one to the overall throughput. Therefore, similar to the "slowest server rule" – SSR, we first calculate each server's speed in isolation $c_i\mu_i$, where $1 \le i \le N$. Next to that, due to the nested structure of the system, one should take into account the dependence between the various servers. Thus, we introduce a server specific metric, which we refer to as the "effectiveness" rate of the server. One can obtain this metric as follows:

$$T^{(i)}/T_{\max}^{(i)} = \frac{c_i}{\sum_{i \le k \le N}(\beta_k + w_k)} \Big/ \frac{c_i}{\sum_{i \le k \le N} \beta_k} = \frac{\sum_{i \le k \le N} \beta_k}{\sum_{i \le k \le N}(\beta_k + w_k)},$$

for $1 \le i < N$. Note that this is exactly the ratio between the total service time and the actual expected sojourn time. Therefore, the higher this ratio is (the closer to 1 it is), the less waiting occurs in the nodes below it, and hence the less influential this nested structure is for the specific server. On the other hand, a small ratio would stand for a large waiting time in lower nodes, which means that the server is not working at its full potential.

In the following, we summarize our bottleneck identification procedure:

- *Step 1:* Obtain $T^{(i)}$ and $T_{\max}^{(i)}$ for all $1 \le i \le N$. The value of $T^{(i)}$ can be approximated by applying our algorithm on the ith subsystem, whereas $T_{\max}^{(i)}$ can be directly derived from the model parameters:

$$T^{(i)}_{\max} = \frac{c_i}{\sum_{i\le k\le N}\beta_k};$$

- *Step 2:* Assign the following "Eff" score to each server:

$$\mathrm{Eff}^{(i)} = \frac{T^{(i)}}{T^{(i)}_{\max}} c_i\mu_i,$$

where $1 \le i \le N$;
- *Step 3:* Determine the server with the smallest "Eff" score as the bottleneck.

4. APPROXIMATION RESULTS

In this section, we present the results for the approximation algorithm. In addition, we also show how our technique relates to methods introduced in similar studies.

The system parameters that we vary in the numerical tests are the number of layers N, the number of sessions per server, $c = (c_1, c_2, \ldots, c_N)$, and the service rate at each node, $\mu = (\mu_1, \mu_2, \ldots, \mu_N)$. To evaluate our method we want to identify possible relationship between the model parameters and the algorithm's accuracy. Since each layer is associated with two variables, extensive test suites for systems with many layers are unfeasible. Therefore, we first analyzed the performance of the algorithm for $N = 3$, in which case we could examine more than 2 million different model instances with a broad range of values for the vectors c and μ. We used the results of these initial tests to find out how the number of sessions and the service rates influence the approximation accuracy. Next to that, we identified the worst case scenarios for three-layered systems. Based on the findings, we could tailor the test settings for networks with more layers in such a way that we get the average and the worst case performance indicators for the algorithm. Moreover, we verified that the observed relationship between the system parameters and the performance of our method remain valid also for networks with more layers, for example, ten layers. In these cases, we compared the approximated metrics to values obtained by a simulation as the exact solutions are computationally intractable.

4.1. Results for Three-layered Networks

First, we designed three test suites with different number of sessions per server $c = (c_1, c_2, c_3)$. Namely, one that represents a small system with $c = (3, 2, 1)$, one for $c = (12, 8, 5)$, and the third having $c = (100, 90, 70)$. In all test suites we vary the values of the service rates μ_1 and μ_2 from 0.1 to 10 with a step size of 0.1. Next to that, without loss of generality, we have fixed $\mu_3 = 1$. The reason is that only the ratio between the various service rates is important with respect to the approximation error. This comes from the fact that scaling the rates is equivalent to scaling the time, which does not influence the performance metrics derived by the algorithm.

At this point, we examine only three-layered systems as they can also be solved exactly, which makes the analysis faster and more accurate in comparison to simulation. This allows us to perform a vast number of test cases and draw conclusions based on the results.

As it can be seen from the plots in Figure 3 for the three-layered systems, the throughput approximation error in each of the test suites is largest in a specific region and goes to 0 outside of it. Therefore, we will not examine the average approximation error as it would

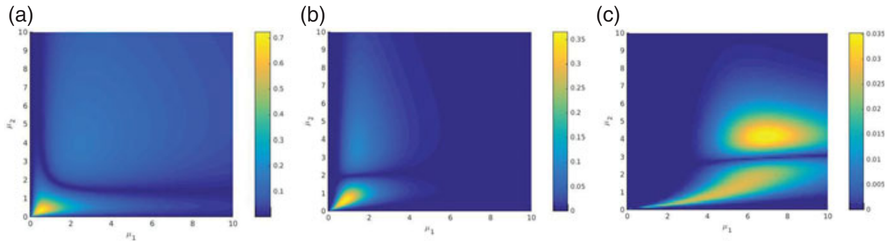


FIGURE 3. Throughput approximation errors for the various test suites for three-layered systems. (a) $c = (3, 2, 1)$. (b) $c = (12, 8, 5)$. (c) $c = (100, 90, 70)$.

be highly dependent on the chosen range of parameters. Instead, we focus on the worst-case scenarios.

Another observation is that the fewer the number of sessions per server there are, the greater the error is. For example, the maximum error in the first test suite is 0.7%, which is more than 20 times higher than the one in the case of a relatively large number of sessions.

In order to test these two conclusions, we design 220 additional test suites of three-layered models. The service rates are again set to vary in the same manner: μ_1 and μ_2 go from 0.1 to 10 with a step size of 0.1 and $\mu_3 = 1$. However, this time we examine all possible tuples $c = (c_1, c_2, c_3)$ up to 10 sessions per server, that is, all combinations of c_1, c_2, and c_3 where $c_3 \leq c_2 \leq c_1 \leq 10$.

As expected, all 220 plots of the approximation error look exactly the same as the one from Figure 3(a) – the relatively high error rates form a specific region. In those test suites, the region is around the point for which $c_1\mu_1 = c_2\mu_2 = c_3\mu_3$. The results of those 2.2 million additional test cases also agree with our second observation – the fewer the number of sessions, the worse the algorithm performs. Based on these, one can correctly identify the worst cases from all the tests conducted so far – the highest error of 2.4% is observed at the systems with the following parameters: $c = (2, 1, 1)$ and $\mu = (0.6, 1, 1)$; $c = (3, 1, 1)$, and $\mu = (0.3, 1, 1)$.

4.2. Results for Ten-layered Networks

With the test suites consisting of three-layered systems we examined the dependency between the accuracy of the algorithm and the number of sessions and the service rate of the various servers. Next to that, we want to analyze the influence of the number of layers on the approximation error of the algorithm. Therefore, we look at systems with more than three layers. However, in those cases, we use simulation to obtain benchmark results as the exact solution is computationally intractable.

In the following, we investigate systems with ten layers. As there are 20 different parameters for those systems, extensive test suites where all those parameters are varied are not possible. First, we performed more than 10 thousand test cases with random parameter sets. Complying with our observations so far, the relative error of those tests was in most of the cases insignificant and rarely larger than 0.1%. Therefore, we tried to tailor the parameters in order to find the worst case scenarios, which we can compare to those found in the three-layered systems. The highest error that we found is 7.7% for the ten-layered network with the following parameters: $c = (4, 3, 3, 2, 2, 2, 1, 1, 1, 1)$, and $\mu = (0.5, 0.75, 0.75, 0.9, 0.9, 0.9, 1, 1, 1, 1)$. Next to systems with the above stated number of sessions per server, we analyzed two more test suites with parameters $c = (10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$ and $c = (15, 14, 13, 12, 11, 10, 9, 8, 7, 6)$. We varied the service rates in the range $20 - 500\%$ of the following values: $\mu = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ and $\mu = (1, 1, 1, 2, 2, 2, 2, 3, 3, 3)$ correspondingly. These two additional test suites were designed to examine the algorithm's accuracy error in unfavorable parameter sets – a few number of sessions per node and similar servers' speed. Nevertheless, the approximation error was rarely significant with a 95th percentile value of 3.76%.

Due to its recursive nature, the accuracy of our algorithm indeed gets worse with an increase in the number of layers. Nevertheless, even the largest errors found for ten-layered systems are comparable with the average ones stated for other algorithms (Franks et al. [7], Tribastone [21]). At the same time, one has to consider that our method is designed specifically for tandem queues with nested sessions, whereas the above cited algorithms can be used to solve more general LQNs.

5. BOTTLENECK IDENTIFICATION RESULTS

In the following, we present the results with regards to our bottleneck identification method. To the best of our knowledge, the prominent techniques researched so far are the ones shown in Franks et al. [8] and the intuitive "SSR". Therefore, we compare those three bottleneck identification metrics.

As previously discussed, the bottleneck identified by our method is the server with the lowest $\text{Eff}^{(i)}$ score. Similarly, the server considered as the bottleneck according to the SSR is the one with the smallest $c_i\mu_i$ product. Note that the SSR is used for traditional tandem queues without nested structure, where the throughput of the whole network is determined by the speed of the slowest node in the chain. Therefore, considering that the throughput of a given server i is the number of sessions times the service rate of each one of them, $c_i\mu_i$, it follows directly that the node with the lowest $c_i\mu_i$ is the limiting factor.

Following the technique in Franks et al. [8], we assign one more score $\text{BStrength}^{(i)}$ to each server except for the last one. In summary, the main idea is to take the ratio of the "normal" utilization rate $U_{\text{norm}}^{(i)}$ of server i over the largest "normal" utilization rate of a server below it, $\max_{\{j>i\}} U_{\text{norm}}^{(j)}$. The server with the largest score is the one identified as the bottleneck, that is,

$$\max_{1\le i<N} \text{BStrength}^{(i)} = \max_{1\le i<N} \frac{U_{\text{norm}}^{(i)}}{\max_{j>i} U_{\text{norm}}^{(j)}}.$$

Finally, we describe the method which we use to determine the real bottleneck. According to our definition, this is the server whose speed modification results in the largest change in the saturation throughput of the system. Therefore, given a system with N servers, we examine N modified networks. In each one of those N systems, the service rate of one of the nodes is increased with 0.01%, whereas all other parameters are kept the same. Using exact analysis we derive the saturation throughput of those modified networks and identify the largest one. In this way, we find the server that is the most influential to the performance of the initial system.

Based on the above described test framework, we evaluate the Eff, SSR, and BStrength techniques on test suites, similar to those described in the previous section. Once again, for the three-layered systems we take: $c = (3, 2, 1)$; $c = (12, 8, 5)$, and $c = (100, 90, 70)$ with μ_1 and μ_2 vary from 0.1 to 10 with a step size of 0.1. However, this time, next to the test cases where μ_3 is fixed at 1, we include such with μ_3 fixed at 10 and at 100. The test suites are extended in such a way in order to achieve a fair number of cases in which a specific server

Table 1. Cases of correct bottleneck identification

Technique	Server 1 (%)	Server 2 (%)	Server 3 (%)	Total (%)
Eff	85.07	95.41	99.98	93.74
SSR	83.42	94.16	100.00	92.88
BStrength	36.49	100.00	–	72.24

is the bottleneck. In addition, we evaluated the bottleneck identification techniques for all the configurations of ten-layer system that we described previously.

The results from the test suites with three-layer systems are listed in Table 1. The table shows the percentage of cases in which the corresponding technique identified the bottleneck correctly. Due to the fact that no BStrength score can be assigned to the last node, we exclude the cases that have the third server as a bottleneck when evaluating this specific technique. From the results, one can conclude that our technique is performing better than the other two. Note that the higher in the network the bottleneck is, the harder it is for the three techniques to identify it correctly. This can be explained by the nested structure being more influential for those servers than for the lower ones.

In addition, the more detailed analysis from Figure 4(a) shows that the cases, in which our technique does not give the correct result, lie on the border line of the parameters ranges for which the bottleneck shifts from one server to another. In those cases, two or more servers are almost equally influential to the overall throughput, and hence even though the server identified by our method is not the real bottleneck, it still does limit the performance of the system. Therefore, we believe that the following relative error gives more insight for the performance of the algorithms:

$$E_{\text{rel}} = \frac{T_{\text{new}} - T_{\text{alg}}}{T_{\text{new}} - T_{\text{old}}},$$

where T_{old} stands for the saturated throughput of the tested system, T_{alg} – the one where the rate of the server identified as the bottleneck by the corresponding algorithm is increased with 0.01%, and T_{new} – the one where the rate of the server that is the real bottleneck is modified with 0.01%. The average relative error E_{rel} from all test cases (both the three-layered and the ten-layered configurations) is 0.92% for our technique, 0.95% for the SSR and 38.49% for the BStrength.

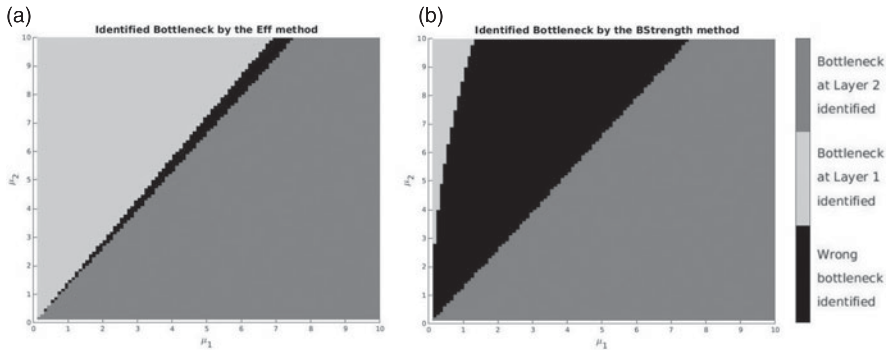


Figure 4. Bottleneck identification results for $c = (3, 2, 1)$ and $\mu_3 = 100$. (a) Eff method. (b) BStrength method.

As a final remark, we would like to point out that our method for bottleneck identification coincides with the SSR in case of a tandem queueing network or in the similar case of an equal number of sessions per server. Therefore, we believe that our method can be considered as a generalization of the SSR.

6. CONCLUSIONS

This paper has studied the performance evaluation of tandem queues with nested sessions. An approximation algorithm for obtaining the servers' throughput and utilization rates was presented. We believe that the high accuracy of the algorithm is an indication of low approximation errors also in case the model is further extended to fit more general LQNs.

In addition, we introduced a server-wise metric that extends the analysis by identifying the bottleneck in a given queueing network. Even though it did not pinpoint the most throughput-limiting server in a small percentage of the cases, it still identified a server that is severely impeding the performance of the system. Moreover, being an extension of the SSR, it is applicable to queueing networks without a layered structure as well. Therefore, we believe that this bottleneck identification method is more general and at the same time more accurate than the existing techniques so far.

References

1. Bard, Y. (1979). Some extensions to multiclass queueing network analysis. In *Proceedings of the 3rd International Symposium on Modelling and Performance Evaluation of Computer Systems*, pp. 51–62.
2. Blanc, J.P.C. (1993). *Performance evaluation of computer and communication systems, Lecture notes in Computer Science*, chapter – Performance analysis and optimization with the power-series algorithm. Berlin, Heidelberg: Springer, pp. 53–80.
3. Buitenhek, R., van Houtum, G.J.J.A.N., & Zijm, H. (2000). AMVA-based solution procedures for open queueing networks with population constraints. *Annals of Operations Research* 93: 15–40.
4. Chandy, K.M. & Neuse, D. (1982). Linearizer: A heuristic algorithm for queueing network models of computing systems. *Communications of the ACM* 25: 126–134.
5. Dorsman, J.L., Perel, N., & Vlasiou, M. (2016). Server waiting times in infinite supply polling systems with preparation times. *Probability in the Engineering and Informational Sciences* 30(2): 153–184.
6. Dorsman, J.L., van der Mei, R.D., & Vlasiou, M. (2013). Analysis of a two-layered network by means of the power-series algorithm. *Performance Evaluation* 70: 1072–1089.
7. Franks, G., Al-Omari, T., Woodside, M., Das, O., & Derisavi, S. (2009). Enhanced modeling and solution of layered queueing networks. *IEEE Transactions on Software Engineering* 35(2): 148–161.
8. Franks, G., Petriu, D., Woodside, M., Xu, J., & Tregunno, P. (2006). Layered bottlenecks and their mitigation. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems (QEST '06)*, pp. 103–114. IEEE Computer Society.
9. Herzog, U. & Rolia, J.A. (2001). Performance validation tools for software/hardware systems. *Performance Evaluation* 45: 125–146.
10. Neuts, M.F. (1981). *Matrix-geometric solutions in stochastic models: an algorithmic approach.* Baltimore: Johns Hopkins University Press.
11. Perel, E. & Yechiali, U. (2008). Queues where customers of one queue act as servers of the other queue. *Queueing Systems* 60: 271–288.
12. Perel, E. & Yechiali, U. (2013). On customers acting as servers. *Asia-Pacific Journal of Operational Research* 30: 1–23.
13. Perel, E. & Yechiali, U. (2016). Finite two layered queueing systems. *Probability in the Engineering and Informational Sciences* 30(3): 492–513.
14. Reiser, M. & Lavenberg, S.S. (1980). Mean-value analysis of closed multichain queuing networks. *Journal of the ACM* 27: 313–322.
15. Rolia, J.A. & Sevcik, K.C. (1995). The method of layers. *IEEE Transactions on Software Engineering* 21(8): 689–700.
16. Roser, C., Nakano, M., & Tanaka, M. (2001). A practical bottleneck detection method. In *Proceedings of the 33rd Conference on Winter Simulation (WSC '01)*, pp. 949–953. IEEE Computer Society.

17. Schweitzer, P.J. (1979). Approximate analysis of multiclass closed networks of queues. In *Proceedings of the International Conference on Stochastic Control and Optimization*, pp. 25–29.
18. Shousha, C., Petriu, D.C., Jalnapurkar, A., & Ngo, K. (1998). Applying performance modelling to a telecommunication system. In *Proceedings of the 1st International Workshop of Software and Performance*, pp. 1–6.
19. Tregunno, P. (2003). Practical analysis of software bottlenecks. Master's thesis, Department of Systems and Computer Engineering, Carleton University.
20. Tribastone, M. (2010). Relating layered queueing networks and process algebra models. In *Proceedings of the 1st Joint WOSP/SIPEW International Conference on Performance Engineering*, pp. 183–194.
21. Tribastone, M. (2013). A fluid model for layered queueing networks. *IEEE Transactions on Software Engineering* 39(6): 744–756.
22. Woodsode, C.M., Neilson, J.E., Petriu, D.C., & Majumdar, S. (1995). The stochastic rendezvous network model for performance of synchronous client-server-like distributed software. *IEEE Transactions on Computers* 44: 20–34.
23. Yom-Tov, G. & Mandelbaum, A. (2008). Queues in hospitals: Semi-open queueing networks in the qed regime. Technical report, Technion, Israeli Institute of Technology.