Stability and Throughput in a Two-Layered Network of Multi-Server Queues

W. van der Weij[a] and R.D. van der Mei[a,b]

[a]CWI, Advanced Communication Networks, Amsterdam, The Netherlands
[b]Vrije Universiteit, Faculty of Sciences, Amsterdam, The Netherlands

We study stability and throughput in a two-layered queueing network consisting of two multi-server nodes, where at any time the busy servers share a common underlying resource in a processor-sharing fashion. We assume general arrival processes, general service-time distributions, arbitrary numbers of servers at both queues, and multiple customer classes, each with a class-specific arrival rate and visit order. For this model we derive a full characterization of the per-queue stability and the per-class throughput, considered as a function of the arrival rates, in a general paramenter setting. The results provide intuition and new fundamental insights into the behavior of queueing networks with this multi-layered structure.

Key words: Multi-server queues; layered queuing models; stability; fair-shared processor; multiple classes of arrival streams.

1 Introduction

In today's information and communication infrastructures we observe a growing diversity and heterogeneity in applications that share parts of the infrastructure. Examples of such infrastructures are Web-based multi-tiered system architectures, with a client tier to provide an interface to the end users, a business logic tier to coordinate information retrieval and processing, and a data tier with legacy systems to store and access customer data. In such environments different applications compete for access to shared infrastructure resources, both on the *software* level (e.g., mutex and database locks, thread pools) and on the *hardware* level (e.g., bandwidth, processing power, disk access). Hence, the performance of these applications is a complex interplay between software and hardware contention. These observations have raised the need to perform a fundamental analysis of *multi-layered* performance models, where servers at one layer are customers at a lower layer. To this end, in this paper we study perhaps one of the simplest non-trivial multi-layered queueing models: a two-layered two-node of multi-server queues, where at any moment the non-idling servers share an underlying resource in a processor-sharing (PS) fashion. The results provide a variety of new and fundamental insights, and as such may be seen as a significant step forward in obtaining an in-depth understanding in the behavior of multi-layered queueing models.

Many of today's application servers need to properly handle huge amounts of transactions within a reasonable time frame. Each transaction typically consists of several sub-transactions that have to be processed some fixed or probabilistic order. To this end, application servers usually implement a number of thread pools, each of which is dedicated to performing a specific sub-transaction. Consider for example the Web server performance model proposed in [3]. Each HTTP transaction consists of one or two sub-transactions: a document-retrieval (DT) step and a script processing (SP) step. To this end, the Web server implements two thread pools, a

dedicated pool of threads performing DT processing, and a pool of threads performing SP processing. The Web server model consists of two multi-server queues, where servers at one queue represent the DT threads, and the servers at the other queue represent SP threads. A particular feature of this model is that at any moment in time the active threads share a common Central Processing Unit (CPU) hardware in a PS fashion. Other examples of performance models with software-hardware interaction are presented in [5, 7].

The literature on single-layered queueing networks is widespread and has been successfully applied in many application areas (see [1] for an overview of the available results). However, only a limited number of papers focus on the performance of multi-layered queuing networks. Rolia and Sevcik [9] propose the so-called Method of Layers (MoL), i.e., a closed queueing-network based model for the responsiveness of client-server applications, explicitly taking into account both software and hardware contention. Another fundamental contribution is presented by Woodside et al. [12], who propose to use the so-called Stochastic Rendez-Vous Network (SRVN) model to analyze the performance of application software with client-server synchronization. The contributions presented in [9] and [12] are often referred to as Layered Queuing Models (LQMs). Another class of layered queuing models are so-called polling models, i.e., multi-queue single-server models where the available service capacity is alternatingly allocated to one of the queues in some round-robin fashion (see [10, 11]). Another related class of models are so-called coupled-processor models, i.e., multi-server models where the service speed of a server at a given queue depends on the number of servers at the other queues (see [2, 4, 6] for results). A common drawback of the available results on multi-layered queuing models is that exact analysis is primarily restricted to special cases, and numerical algorithms are typically required to obtain performance measures of interest (see for example [12]). Consequently, in-depth understanding in the behavior of multi-layered queueing models is limited.

In this paper we consider a two-node network of multi-server queues with a general arrival processes, general service-time distributions, multiple customers classes each with a their own routing scheme and arrival rate, and arbitrary numbers of servers at both queues, where all busy servers share a common underlying resource in a processor-sharing (PS) fashion. For this model, we provide a full characterization of the per-queue stability and the per-class throughput, in a general parameter setting. To this end, we first show that the per-class throughput values satisfy a uniquely solvable set of linear equations, for a given stability subset. Second, we show that there exists a unique stability subset for which this solution is feasible. These two results together provide a full and unique characterization of the throughput and stability figures. Then, for a number of modeling examples we developed closed-form expressions for the stability subsets and the corresponding throughput values. Moreover, we obtain a number of seemingly counter-intuitive results, and provide explanations for them. We believe that the results presented in this paper give new and fundamental insights into the stability and throughput figures in hierarchical queueing networks.

This paper partly extends the results obtained by Van der Mei et al. [8], where expressions are derived for the per-queue stability and throughput in a tandem of $N > 1$ multi-server queues, where queue i has c_i parallel servers and mean service time β_i. A specific feature of the tandem model studied in [8] is that each customer visits all queue in the fixed order $1, 2, \ldots, N$. An interesting feature of [8] is the following *bottleneck property*: if for queue i there exists $j < i$ such that $\beta_j/c_j > \beta_i/c_i$, then queue i will never become unstable, not even when the arrival

rate grows to infinity; in other words, queues that are "behind" a bottleneck queue will never become unstable. In this context, the paper in [8] differs fundamentally from the model considered in the current paper where we consider a network - rather than a tandem - of queues, where customers may follow different routes, so that in general the bottleneck property does not exist. Consequently, the analysis of a network of queues is essentially different from the tandem-queue analysis described in [8].

The remainder of this paper is organized as follows. In Section 2 the model is described and the relevant notation is introduced. In Section 3 we provide a full characterization of the per-queue stability and per-class throughput. In Section 4 we consider a number of specific model instances, and provide expressions for the stability regions and corresponding throughput values, giving new insights into the stability and throughput behavior of the model. Finally, in Section 5 we address a number of challenging topics for further research.

2 Model

Consider a network of two multi-server queues, Q_1 and Q_2, both an infinite buffer space. Let the service times at Q_i are generally distributed with finite mean β_i. Let $\underline{\beta} := (\beta_1, \ldots, \beta_N)$, and define $B := \{\underline{\beta} : \beta_i > 0 \ (i = 1, \ldots, N)\}$. Let c_i the number of servers at Q_i, define $\underline{c} = (c_1, \ldots, c_N)$, and let C be the set of possible combinations of the number of servers, i.e., $C := \{\underline{c} : c_i \in \{1, 2, \ldots\} \ (i = 1, \ldots, N)\}$. Denote by N_i the random variable indicating the number of customers present (i.e., either waiting or being served) at Q_i. Then at any time the number of busy servers at Q_i is $M_i := \min\{N_i, c_i\}$. The total service capacity (which by defintion equals 1) is shared by the active servers in a Processor Sharing (PS) fashion. That is, the service rate of each of the busy servers is $1/\sum_{i=1}^{N} M_i$; if the system is empty, then all servers are idle. We consider four different customers classes, numbered 1, 2, 12 and 21, with the following interpretations. Class-1 customers arrive at Q_1 according to a general arrival process with rate $\lambda_1 \geq 0$, and after receiving service at Q_1 immediately depart from the system; thus, class-1 customers never visit Q_2. Class-2 customers arrive at Q_2 at rate $\lambda_2 \geq 0$, and after receiving service at Q_2 immediately depart from the system. Class-12 customers arrive at Q_1 at rate rate $\lambda_{12} \geq 0$, and after receiving service at Q_1 immediately proceed to Q_2. After receiving service at Q_2 the customers depart from the system. Finally, class-21 customers arrive at Q_2 at rate $\lambda_{21} \geq 0$, and then subsequently visit Q_2 and Q_1 before departing from the system (see Figure 1).

Denote $\underline{\lambda} = (\lambda_1, \lambda_2, \lambda_{12}, \lambda_{21})$ and let $\Lambda := \{\underline{\lambda} : \lambda_1, \lambda_2, \lambda_{12}, \lambda_{21} \geq 0\}$. Denote by λ_i' the departure rate of class-i customers ($i = 1, 2$). Moreover, for class-12 customers we denote by $\lambda_{12}' \geq 0$ the departure rate from Q_1, and by λ_{12}'' the departure rate from Q_2. Similarly, for class-21 customers we denote by $\lambda_{21}' \geq 0$ the departure rate from Q_2, and by λ_{21}'' the departure rate from Q_1. The load at Q_1 (i.e., the amount of arriving work at Q_1) is defined as $\rho_1 := \beta_1(\lambda_1 + \lambda_{12} + \lambda_{21}')$, and similarly, the load offered to Q_2 is defined as $\rho_2 := \beta_2(\lambda_2 + \lambda_{21} + \lambda_{12}')$. The total load offered to the system is denoted by $\rho := \rho_1 + \rho_2$. We will refer to Q_i as stable if and only if the arrival rate of customers at the queue equals the departure rate. Thus, Q_1 is stable if and only if $\lambda_1 = \lambda_1', \lambda_{12} = \lambda_{12}'$ and $\lambda_{21}' = \lambda_{21}''$. Similarly, Q_2 is called stable if and only if $\lambda_2 = \lambda_2', \lambda_{12}' = \lambda_{12}''$ and $\lambda_{21} = \lambda_{21}'$. A queue is called unstable if and only if it is not stable. Finally, we define the notion of a bottleneck: Q_1 is called a *bottleneck* if and only if $\beta_1/c_1 \geq \beta_2/c_2$, and Q_2 is

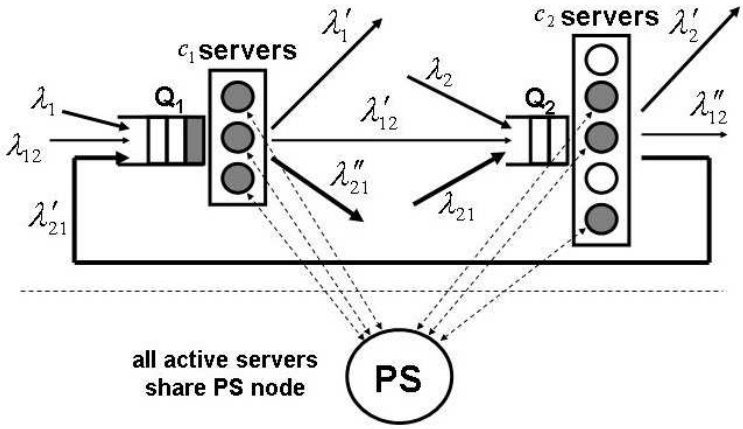

Figure 1: Illustration of the model ($c_1 = 3, c_2 = 5, M_1 = M_2 = 3$).

called a bottleneck if and only if $\beta_1/c_1 \leq \beta_2/c_2$. If $\beta_1/c_1 = \beta_2/c_2$, then both queues are called bottlenecks. Finally, define the set of bottleneck queues by $\mathcal{B} := \{i : Q_i \text{ is a bottleneck}\}$. Notice that by definition $\mathcal{B} = \{1\}, \{2\}$ or $\{1, 2\}$.

3 Analysis

In this section we give an exact characterization of the per-queue stability and the per-class throughput considered as a function of $\underline{\lambda} \in \Lambda$, for given $\underline{\beta} \in B$, $\underline{c} \in C$. For compactness of the presentation the proofs of the various results are omitted.

To analyze the stability of the individual queues, the following notation is useful. For given $\underline{\beta} \in B$, $\underline{c} \in C$, we denote by (S, S) the set of values of $\underline{\lambda} \in \Lambda$ for which both queues are stable. Similarly, (I, S) is the set of values of $\underline{\lambda}$ for which Q_1 is unstable and Q_2 is stable. (S, I) is the set of values of $\underline{\lambda}$ for which Q_1 is stable and Q_2 is unstable, and finally, (I, I) is the set of $\underline{\lambda}$-values for which both queues are unstable. Note that in this way, the set Λ is partitioned into the following four *stability subsets*:

$$\Lambda = (S, S) \ \cup \ (I, S) \ \cup (S, I) \ \cup (I, I). \tag{1}$$

We will now derive an exact characterization of the four stability subsets (S, S), (I, S), (S, I) and (I, I) and give expressions for the corresponding per-class throughput values. To start, we will show that, for given stability subset, the per-queue throughput values can be obtained from a set of linear equations (Lemmas 1 to 4). Subsequently, we show that the stability subsets are uniquely identified (Lemma 5).

Lemma 1 (Per-queue stability and throughput for stable systems)
$\underline{\lambda} \in (S, S)$ *if and only if*

$$(\lambda_1 + \lambda_{12} + \lambda_{21})\beta_1 + (\lambda_2 + \lambda_{12} + \lambda_{21})\beta_2 \leq 1, \tag{2}$$

and in that case the departure rates are:

$$\lambda'_1 = \lambda_1, \quad \lambda'_2 = \lambda_2, \quad \lambda'_{12} = \lambda''_{12} = \lambda_{12}, \quad \lambda'_{21} = \lambda''_{21} = \lambda_{21}. \tag{3}$$

In words, both queues are stable if and only if equation (2) is satisfied, and in that case for each (class, queue)-combination the rate into a queue equals the rate out of that queue.

Lemma 1 characterizes for which parameter settings both queues are stable. Lemmas 2 to 4 below deal with the case in which at least one queue is unstable.

Lemma 2 (Work conserving property)

$$(\lambda_1 + \lambda_{12} + \lambda_{21})\beta_1 + (\lambda_2 + \lambda_{12} + \lambda_{21})\beta_2 \geq 1, \tag{4}$$

if and only if

$$(\lambda'_1 + \lambda'_{12} + \lambda''_{21})\beta_1 + (\lambda'_2 + \lambda'_{21} + \lambda''_{12})\beta_2 = 1. \tag{5}$$

In words, the system as a whole is unstable (i.e., $\rho \geq 1$) if and only if the PS-server runs at full speed, i.e., the total amount of work handled per time unit equals 1.

Lemma 3 (Preservation of throughput ratios)
For any $\underline{\lambda} \in \Lambda$, for both queues the ratios between the per-class rate into a queue and the corresponding output rate is preserved: For Q_1,

$$\frac{\lambda_1}{\lambda'_1} = \frac{\lambda_{12}}{\lambda'_{12}} = \frac{\lambda'_{21}}{\lambda''_{21}} =: \Gamma_1, \tag{6}$$

and for Q_2,

$$\frac{\lambda_2}{\lambda'_2} = \frac{\lambda_{21}}{\lambda'_{21}} = \frac{\lambda'_{12}}{\lambda''_{12}} =: \Gamma_2. \tag{7}$$

In words, for each queue the ratios between the output rates of the different classes are equal to the ratios between the input rates. This property is an immediate consequence of the fact that no priority scheme is used. Note also that in general $\Gamma_i \geq 1$, and that $\Gamma_i = 1$ if and only if Q_i is stable.

Lemma 4 (Fairness for unstable queues)
If $\underline{\lambda} \in (I, I)$, then

$$(\lambda'_1 + \lambda'_{12} + \lambda''_{21})\beta_1 = \frac{c_1}{c_1 + c_2}, \quad (\lambda'_2 + \lambda'_{21} + \lambda''_{12})\beta_2 = \frac{c_2}{c_1 + c_2}. \tag{8}$$

In words, if both queues are unstable, then the rates at which both queues are served are proportional to the numbers of servers at both queues. This property follows directly from the fact that if both queues are unstable, then Q_1 and Q_2 occupy all c_1, c_2 servers.

For given $\underline{\lambda} \in \Lambda$, $\underline{\beta} \in B$ and $\underline{c} \in C$, and stability subset $\mathcal{S} \subset \Lambda$, relations (2) to (8) constitute a set of linear equations for the six output variables $\underline{\lambda}_{out} := (\lambda'_1, \lambda'_2, \lambda'_{12}, \lambda''_{12}, \lambda'_{21}, \lambda''_{21})$ it is readily verified that for any choice of the stability subsets the set of equations has a unique

solution. In other words, for given stability subsets, $\underline{\lambda}_{out}$ is uniquely determined by equations (2) to (8).

Definition
For given $\underline{\lambda} \in \Lambda$, $\underline{\beta} \in B$ and $\underline{c} \in C$ and stability subset $\mathcal{S} \subset \Lambda$, the couple $(\underline{\lambda}, \mathcal{S})$ is called *feasible* if the unique solution $\underline{\lambda}_{out}$ of the set of equations (2)-(8) satisfies the following relations:

$$\lambda_1 \geq \lambda_1' \geq 0, \quad \lambda_2 \geq \lambda_2' \geq 0, \quad \lambda_{12} \geq \lambda_{12}' \geq \lambda_{12}'' \geq 0, \quad \lambda_{21} \geq \lambda_{21}' \geq \lambda_{21}'' \geq 0, \tag{9}$$

and, if $\underline{\lambda} \in (I, S)$ then

$$\frac{(\lambda_1' + \lambda_{12}' + \lambda_{21}'')\beta_1}{c_1} > \frac{(\lambda_2' + \lambda_{21}' + \lambda_{12}'')\beta_2}{c_2}, \tag{10}$$

while the reverse it true for $\underline{\lambda} \in (S, I)$.

In words, the inequalities in (9) state that for each class crossing a certain queue the output rate cannot exceed the input rate, and (10) states that the load-per-server of an unstable queue is stricktly greater than the load-per-server of a stable queue.

Lemma 5 (Uniqueness of the feasible solution)
For each $\underline{\lambda} \in \Lambda$, $\underline{\beta} \in B$ and $\underline{c} \in C$ there exists a unique stability set $\mathcal{S} \subset \Lambda$ such that the couple $(\underline{\lambda}, \mathcal{S})$ is feasible.

In Section 4 we give illustrative examples for determining the stability subsets.

4 Examples

In this section we use the results obtained in Section 3 to obtain explicit expressions for the stability subsets and the corresponding throughput values for a number of model examples.

4.1 Case I: $\lambda_1, \lambda_{12} > 0, \lambda_2 = \lambda_{21} = 0$

Let us consider a tandem of two multi-server queues with only two customer classes, 1 and 12. Lemma 1 implies that $\underline{\lambda} \in (S, S)$ if and only if $(\lambda_1 + \lambda_{12})\beta_1 + \lambda_{12}\beta_2 \leq 1$, and hence,

$$(S, S) = \{\underline{\lambda} \in \Lambda : \lambda_1\beta_1 + \lambda_{12}(\beta_1 + \beta_2) \leq 1\}. \tag{11}$$

Moreover, if $\underline{\lambda} \in (S, S)$ then the output parameters λ_1', λ_{12}' and λ_{12}'' can be expressed in terms of the input parameters λ_1 and λ_{12} as follows:

$$\lambda_1' = \lambda_1 \text{ and } \lambda_{12}' = \lambda_{12}'' = \lambda_{12}. \tag{12}$$

From Lemma 1 it follows that if $\underline{\lambda} \in (I, S)$ then $(\lambda_1 + \lambda_{12})\beta_1 + \lambda_{12}''\beta_2 \geq 1$, or equivalently $(\lambda_1' + \lambda_{12}')\beta_1 + \lambda_{12}''\beta_2 = 1$ (Lemma 2). Moreover, Lemma 3 implies that $\lambda_1/\lambda_1' = \lambda_{12}/\lambda_{12}'$, and because of the stability of Q_2, $\lambda_{12}'' = \lambda_{12}'$. Equation (10) implies that for $\underline{\lambda} \in (I, S)$ we have $(\lambda_1' + \lambda_{12}')\beta_1/c_1 \geq \lambda_{12}''\beta_2/c_2$. Combining these relations, it is readily verified that if $\{1\} \subset \mathcal{B}$, then

$$(I, S) = \{\underline{\lambda} \in \Lambda : \lambda_1\beta_1 + \lambda_{12}(\beta_1 + \beta_2) > 1\}, \tag{13}$$

and if $\{2\} \subset \mathcal{B}$, then

$$(I,S) = \left\{ \underline{\lambda} \in \Lambda : \lambda_1\beta_1 + \lambda_{12}(\beta_1+\beta_2) > 1 \text{ and } \left(\frac{\beta_2}{c_2} - \frac{\beta 1}{c_1}\right)\lambda_1 - \frac{\beta_1}{c_1}\lambda_{12} > 0 \right\}, \tag{14}$$

and that the throughput values are given by the following expressions:

$$\lambda_1' = \frac{\lambda_1}{(\lambda_1+\lambda_{12})\beta_1 + \lambda_{12}\beta_2}, \quad \lambda_{12}' = \lambda_{12}'' = \frac{\lambda_{12}}{(\lambda_1+\lambda_{12})\beta_1 + \lambda_{12}\beta_2}. \tag{15}$$

From Lemma 1 it follows that if $\underline{\lambda} \in (S,I)$ then $(\lambda_1+\lambda_{12})\beta_1 + \lambda_{12}''\beta_2 \geq 1$, or equivalently $(\lambda_1'+\lambda_{12}')\beta_1 + \lambda_{12}''\beta_2 = 1$ (Lemma 2). The stability of Q_2 implies $\lambda_1' = \lambda_1$ and $\lambda_{12}' = \lambda_{12}$. Moreover, equation (10) implies that for $\underline{\lambda} \in (S,I)$ we also have $(\lambda_1'+\lambda_{12}')\beta_1/c_1 < \lambda_{12}''\beta_2/c_2$. Then it is readily verified that if $\{1\} \subset \mathcal{B}$, then $(S,I) = \emptyset$, and if $\{2\} \subset \mathcal{B}$, then

$$(S,I) = \left\{ \underline{\lambda} \in \Lambda : \lambda_1\beta_1 + \lambda_{12}(\beta_1+\beta_2) > 1 \text{ and } \lambda_1 + \lambda_{12} < \frac{c_2}{(c_1+c_2)\beta_1} \right\}, \tag{16}$$

and that the throughput values are given by the following expressions:

$$\lambda_1' = \lambda_1, \quad \lambda_{12}' = \lambda_{12}, \quad \lambda_{12}'' = \frac{1-(\lambda_1+\lambda_{12})\beta_1}{\beta_2}. \tag{17}$$

Using similar arguments, including the use of Lemma 4, it is readily verified that if $\{1\} \subset \mathcal{B}$ then $(I,I) = \emptyset$, and otherwise,

$$(I,I) = \left\{ \underline{\lambda} \in \Lambda : \lambda_1\beta_1 + \lambda_{12}(\beta_1+\beta_2) > 1,\ \lambda_1 + \lambda_{12} \geq \frac{c_2}{(c_1+c_2)\beta_1},\ \left(\frac{\beta_2}{c_2} - \frac{\beta_1}{c_1}\right)\lambda_1 - \frac{\beta_1}{c_1}\lambda_{12} \leq 0 \right\}, \tag{18}$$

and that the throughput values are given by the following expressions:

$$\lambda_1' = \frac{\lambda_1 c_1}{(\lambda_1+\lambda_{12})\beta_1(c_1+c_2)}, \quad \lambda_{12}' = \frac{\lambda_{12}c_1}{(\lambda_1+\lambda_{12})\beta_1(c_1+c_2)}, \text{ and } \lambda_{12}'' = \frac{c_2}{\beta_2(c_1+c_2)}. \tag{19}$$

Figure 2 below illustrates the partitioning into stability subsets. On the left-hand side, we have $\beta_1 = 2, \beta_2 = 4, c_1 = c_2 = 1$, so that $\mathcal{B} = \{2\}$. On the right-hand side $\beta_1 = 4, \beta_2 = 2, c_1 = c_2 = 1$, so that $\mathcal{B} = \{1\}$.

Remark 1:
The results also lead to the following seemingly counter-intuitive observation that when Q_i is unstable for given value of $\underline{\lambda}$ it may well occur an increase of one of the arrival rates can make Q_i stable. To see this, consider for example the model with $\beta_1 = 2$, $\beta_2 = 4$ and $c_1 = c_2 = 1$, see the left-hand side of Figure 2. Now, let us consider the per-queue stability as a function of λ_1 for fixed $\lambda_{12} := 6/40$. Then it is readily verified that if $\lambda_1 \leq 1/40$ then both queues are stable. Moreover, if we increase λ_1 such that $1/40 < \lambda_1 \leq 4/40$, then Q_2 become unstable, while Q_1 is still stable. Subsequently, if λ_1 is further increased to $4/40 < \lambda_1 < 6/40$, then both queues become unstable. Finally, it is interesting to see that if $\lambda_1 \geq 6/40$ then Q_2 becomes stable again, while Q_1 remains unstable. Quite remarkable.
To provide an intuitive explanation for this, note that the load offered to the system by class-12 customers is $\lambda_{12}(\beta_1+\beta_2) = \frac{6}{40}(2+4) = 0.9$, so that the system (as a whole) becomes unstable

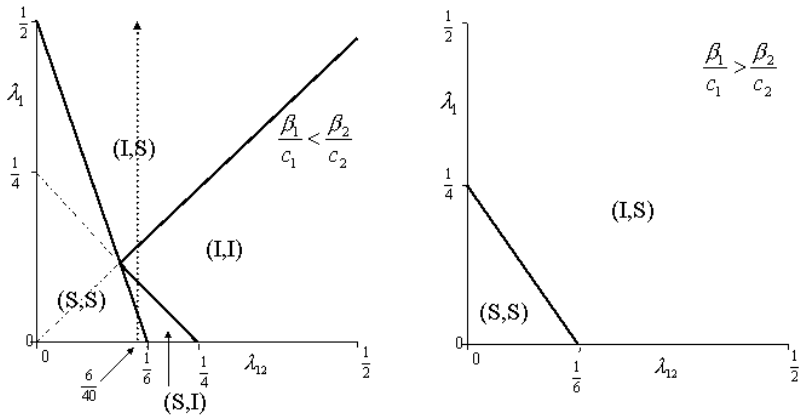

Figure 2: Partitioning into stability subsets for Case I.

when $\lambda_1\beta_1 = 4\lambda_1 > 1 - 0.9 = 0.1$, or equivalently, $\lambda_1 > 1/40$. Then, as λ_1 increases beyond $1/40$ the load on Q_1 is increased, which goes as the expense of the processing power of Q_2, so that Q_2 becomes unstable. Subsequently, if λ_1 is further increased beyond $4/40$, the load on Q_1 is further increased while the unstable Q_2 keeps all c_2 servers busy all time and cannot consume more processing power, and consequently, Q_1 also becomes unstable. Finally, if λ_1 grows large, then the vast majority of customers passing through Q_1 are class-1 customers, so that the rate at which class-12 customers enter Q_2 (i.e., λ'_{12}) tends to become small, while Q_2 still has its fair share $c_1/(c_1 + c_2)$ of the processing capacity, and consequently, Q_2 becomes stable (again) as λ_1 is large enough.

Remark 2:
Note that if Q_2 is the only bottleneck (i.e., $\beta_1/c_1 < \beta_2/c_2$) then the three lines in Figure 2 intersect at a single point $\underline{\lambda}^* := (\lambda_1^*, \lambda_{12}^*)$, with

$$\lambda_1^* := \frac{1}{c_1 + c_2}\left(\frac{c_1}{\beta_1} - \frac{c_2}{\beta_2}\right) \quad \text{and} \quad \lambda_{12}^* := \frac{1}{\beta_2}\frac{c_2}{c_1 + c_2}. \tag{20}$$

According to the definition of stability, we have $(\lambda_1^*, \lambda_{12}^*) \in (S, S)$. At this point, which lies "on the edge" of the stability region, a minor change in $\underline{\lambda}^*$ may lead to instability of one of the queues, or both.

Remark 3:
Comparing the two graphs plotted in Figure 2, it is interesting to see that the partitioning of Λ into stability subsets strongly depends on which queue is a bottleneck. For example, we observe that in the case where Q_1 is the bottleneck (i.e., $\beta_1/c_1 > \beta_2/c_2$), Q_2 never becomes unstable, not even when both arrival rates λ_1 and λ_{12} grow to infinity. To give an intuitive explanation for this, note that all customers that eventually pass through Q_2 (i.e., class-12 customers) first pass through Q_1 before entering Q_2. Then if both Q_1 and Q_2 are running at full speed, the numbers of customers they can handle per time unit is $\alpha_1 := c_1/\beta_1(c_1 + c_2)$ and $\alpha_2 := c_2/\beta_2(c_1 + c_2)$,

respectively. Consequently, if $\beta_1/c_1 > \beta_2/c_2$ then $\alpha_1 < \alpha_2$, so that the output of Q_1 can never saturate Q_2. Alternatively, if Q_2 is a bottleneck, then both Q_1 and Q_2 may become unstable as the arrival rates λ_1 and λ_{12} become large.

4.2 Case II: $\lambda_{12}, \lambda_{21} > 0, \lambda_1 = \lambda_2 = 0$

Let us now consider a network of two multi-server queues with only two customer classes, 12 and 21. Interestingly, this will lead to fundamentally different properties of the stability subsets, which will be discussed below.

From Lemma 1, $\underline{\lambda} \in (S,S)$ if and only if $(\lambda_{12}+\lambda_{21})(\beta_1+\beta_2) \leq 1$, and hence,

$$(S,S) = \left\{ \underline{\lambda} \in \Lambda : \lambda_{12}+\lambda_{21} \leq \frac{1}{\beta_1+\beta_2} \right\}. \tag{21}$$

Moreover, the corresponding throughput values are: $\lambda'_{12} = \lambda''_{12} = \lambda_{12}$ and $\lambda'_{21} = \lambda''_{21} = \lambda_{21}$. Using similar arguments, it is easy to verify that if $\{2\} \subset \mathcal{B}$ then $(I,S) = \emptyset$, and otherwise

$$(I,S) = \left\{ \underline{\lambda} \in \Lambda : \lambda_{12}+\lambda_{21} > \frac{1}{\beta_1+\beta_2} \text{ and } \lambda_{12}+\lambda_{21} < \frac{c_2}{\beta_2(c_1+c_2)} \right\}, \tag{22}$$

and that the corresponding throughput values are

$$\lambda'_{12} = \lambda''_{12} = \frac{\lambda_{12}}{(\lambda_{12}+\lambda_{21})(\beta_1+\beta_2)}, \quad \lambda''_{21} = \frac{\lambda_{21}}{(\lambda_{12}+\lambda_{21})(\beta_1+\beta_2)}. \tag{23}$$

(S,I)**:** Using symmetry, it is easy to seen that if $\{1\} \subset \mathcal{B}$, then $(S,I) = \emptyset$, and otherwise

$$(S,I) = \left\{ \underline{\lambda} \in \Lambda : \lambda_{12}+\lambda_{21} > \frac{1}{\beta_1+\beta_2} \text{ and } \lambda_{12}+\lambda_{12} < \frac{c_1}{\beta_1(c_1+c_2)} \right\}, \tag{24}$$

and that the corresponding throughput values are

$$\lambda''_{12} = \frac{\lambda_{12}}{(\lambda_{12}+\lambda_{21})(\beta_1+\beta_2)}, \quad \lambda'_{21} = \lambda''_{21} = \frac{\lambda_{21}}{(\lambda_{12}+\lambda_{21})(\beta_1+\beta_2)}. \tag{25}$$

Finally, it is easy to verify that if $\{2\} \subset \mathcal{B}$, then

$$(I,I) = \left\{ \underline{\lambda} \in \Lambda : \lambda_{12}+\lambda_{21} > \frac{1}{\beta_1+\beta_2}, \text{ and } \lambda_{12}+\lambda_{21} \geq \frac{c_1}{\beta_1(c_1+c_2)} \right\}, \tag{26}$$

and otherwise,

$$(I,I) = \left\{ \underline{\lambda} \in \Lambda : \lambda_{12}+\lambda_{21} > \frac{1}{\beta_1+\beta_2}, \text{ and } \lambda_{12}+\lambda_{21} \geq \frac{c_2}{\beta_2(c_1+c_2)} \right\}, \tag{27}$$

and the throughput values are given by the following expressions:

$$\lambda'_{12} = \frac{\lambda_{12}c_2}{(\lambda_{12}+\lambda_{21})\beta_2(c_1+c_2)}, \quad \lambda''_{12} = \frac{\lambda_{12}c_1}{(\lambda_{12}+\lambda_{21})\beta_1(c_1+c_2)}, \tag{28}$$

and

$$\lambda'_{21} = \frac{\lambda_{21}c_2}{(\lambda_{12}+\lambda_{21})\beta_2(c_1+c_2)}, \quad \lambda''_{21} = \frac{\lambda_{21}c_1}{(\lambda_{12}+\lambda_{21})\beta_1(c_1+c_2)}. \tag{29}$$

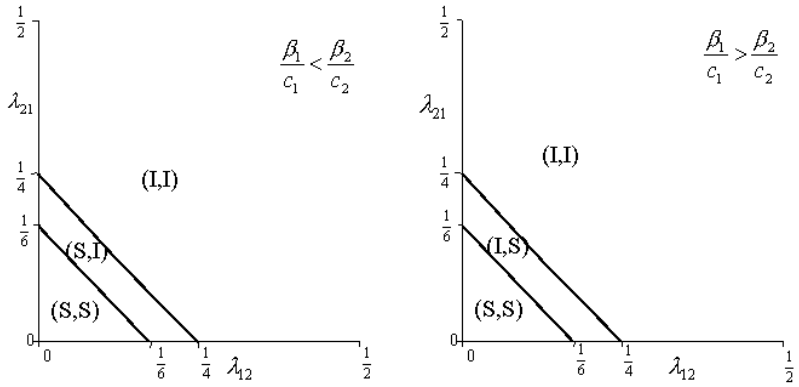

Figure 3: Partitioning into stability subsets for Case II.

Figure 3 below illustrates the partitioning into stability subsets. On the left-hand side, we have $\beta_1 = 2, \beta_2 = 4, c_1 = c_2 = 1$, so that Q_2 is the bottleneck. On the right-hand side $\beta_1 = 4, \beta_2 = 2, c_1 = c_2 = 1$, so that Q_1 is the bottleneck.

Remark 4:
Figure 3 illustrates the fact that in the model considered in Section 4.2 the borderlines are parallel. Consider for example the case $\underline{\lambda} \in (I, S)$, assuming $\beta_1/c_1 > \beta_2/c_2$. Then if λ_{12} is increased with a certain value ϵ while λ_{21} is decreased with the same value ϵ, then the stability of both queues does not change. To see this, we first observe that the load offered to Q_1 remains equal: $\beta_1(\lambda_{12}+\epsilon+\lambda'_{21}-\epsilon) = \beta_1(\lambda_{12}+\lambda'_{21})$. Moreover, the load offered to Q_2 will become $\beta_2(\lambda_{21}-\epsilon+\lambda'_{12}) \leq \beta_2(\lambda_{21}-\epsilon+\lambda_{12}+\epsilon) = \beta_2(\lambda_{21}+\lambda_{12})$. Hence, Q_1 will remain unstable and Q_2 will remain stable.

Remark 5:
We observe that if $\mathcal{B} = \{2\}$ then $(I, S) = \emptyset$. To see this, suppose $\mathcal{B} = \{2\}$ and $\underline{\lambda} \in (I, S)$. Then by definition we have the following inequalities: (1) $c_1/\beta_1 < c_2/\beta_2$, (2) $\lambda_{12} > \lambda'_{12} = \lambda''_{12}$, and (3) $\lambda_{21} = \lambda'_{21} > \lambda''_{21}$. Combining these inequalities it is readily seen that this is in contradiction with (10). Hence, $(I, S) = \emptyset$. On the contrary, it is easy to verify that if $\{1\} \subset \mathcal{B}$ then $(S, I) = \emptyset$.

5 Research challenges

The results presented in this paper raise a number of challenging topics for further research. First, the results presented here may be generalized to a more general network with an arbitrary number of nodes and customers classes. To this end, the insights presented in this paper are highly useful. Second, the results may be used for optimization purposes, where the total throughput is optimized with respect to the number of servers at the different queues. Third, it is interesting to consider how to evaluate and optimize other performance metrics, such as the expected sojourn times.

References

[1] Boxma, O.J. and Daduna, H. (1990). Sojourn times in queueing networks. In: Stochastic Analysis of Computer and Communication Systems. Elsevier Science Publishers.

[2] Cohen, J.W. and Boxma, O.J. (1983). Boundary value problems in queueing system analysis. North Holland, Amsterdam.

[3] Ehrlich, W.K., Hariharan, R., Reeser, P.K., and Van der Mei, R.D. (2001). Performance of Web servers in a distribuetd computing environment. In: Teletraffic Engineering in the Internet Era, proceedings ITC-17 (Salvador-de Balia, Brazil), 137-148.

[4] Fayolle, G. and Iasnogorodski, R. (1979). Two coupled processors: the reduction to a Riemann-Hilbert problem. Zeitschrift für Warscheinlichkeitstheorie und Verwandte Gebiete 47, 325-351.

[5] Harkema, M., Gijsen, B.M.M., Van der Mei, R.D. and Hoekstra, Y. (2004). Middleware performance modelling. Proceedings international Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS (San Jose, CA, July 2004), 733-742.

[6] Konheim, A., Meilijson, I. and Melkman, A. (1981). Processor-sharing of two parallel lines. Journal of Applied Probability 18, 952-956.

[7] Van der Mei, R.D., Hariharan, R. and Reeser, P.K. (2001). Web server performance modeling. Telecommunication Systems 16, 361-378.

[8] Van der Mei, R.D., Gijsen, B.M.M. and Mohy el Dine, S. (2005). Stability and throughput in a layered tandem of multi-server queues. Submitted.

[9] Rolia, J.A. and Sevcik, K.C. (1995). The method of layers. IEEE Transactions on Software Engineering 21, 689-699.

[10] Takagi, H. (1990). Queueing analysis of polling models: an update. In: *Stochastic Analysis of Computer and Communication Systems*, ed. H. Takagi (North-Holland, Amsterdam), 267-318.

[11] Takagi, H. (1997). Queueing analysis of polling models: progress in 1990-1994. In: *Frontiers in Queueing: Models, Methods and Problems*, ed. J.H. Dshalalow (CRC Press, Boca Raton, FL).

[12] Woodside, C.M., Neilson, J.E., Petriu, D.C. and Majumdar, S. (1995). The Stochastic Rendezvous Network model for the performance of synchronous client-server like distributed software. IEEE Transactions on Computers 44, 20-34.