Non-Existence of Stabilizing Policies for the Critical Push-Pull Network and Generalizations

Yoni Nazarathy[a], Leonardo Rojas-Nandayapa[a], Thomas S. Salisbury [1b]

[a]*School of Mathematics and Physics, The University of Queensland, Brisbane, Australia.*
[b]*Department of Mathematics and Statistics, York University, Toronto, Canada.*

Abstract

The push-pull queueing network is a simple two server, two job-stream example in which servers either serve jobs from queues or generate new arrivals. Previous work has shown that there exist non-idling policies that stabilize the system in the positive recurrent sense for all parameter settings in which the network may be rate stable, except for the case where processing rates are equal on each job stream (critical). It was conjectured in Kopzon, Nazarathy and Weiss (2009) that there is no policy that makes the network positive recurrent (stable) in the critical case. Our contribution here is a proof for that conjecture. We also consider generalizations where it is shown that a stabilizing non-idling policy does not exist in the critical case. In this respect we put forward a general sufficient condition for non-stabilizability of queueing networks.

1. Introduction

Controlled queueing networks are primary objects of study in operations research and applied probability as they provide sensible models for a variety of engineering, communications and service situations. Alongside performance analysis and optimal control, stability analysis plays a central role in the theory and has far reaching implications in the design of systems. A comprehensive introduction to stability properties of controlled queueing networks is [1]. See also [2] and [3].

The push-pull queueing network, introduced in [4], is perhaps the simplest example of a queueing network that generates its own input and is able in certain cases (described below) to operate with servers fully utilized while keeping queues stochastically stable. Following [4], the network was analyzed with exponential processing times in [5] and general processing times in [6]. Further network generalizations are in [7]. The novelty of this network is that by allowing servers to split processing power between "arrival generation" and "service of current jobs", one can often find stabilizing control policies

[1]Research supported in part by NSERC.

in which servers never idle. This stands in contrast to the majority of queueing network models (surveyed in [1], [2] and [3]) in which high utilization cannot be achieved without having to endure high congestion levels.

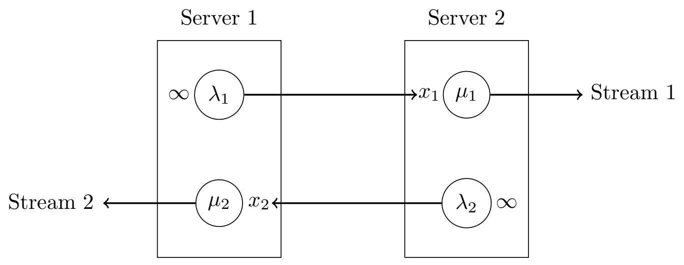

Figure 1: The push-pull network: Two servers are working non-stop on two job streams, $i = 1, 2$. Push operations, labeled λ_i, move jobs to the queues x_i served by pull operations labeled μ_i.

The push-pull network is illustrated in Figure 1. Two servers are working on two job streams. Each job stream begins with a *push* operation (labeled λ_i, $i = 1, 2$) and follows with a *pull* operation (labeled μ_i, $i = 1, 2$). The push operation produces jobs, serving a virtual unlimited supply of raw materials, and moves them down to the pull operation. The pull operation is associated with a queue (labeled x_i, $i = 1, 2$). At any state, a scheduling policy (control) determines for each server whether it should work on *push* or *pull*. Pushing may always be performed as it is assumed that there is an infinite supply of raw materials at the start of each stream. As opposed to that, pulling may only be performed if the associated queue is non-empty. We are interested in non-idling (fully utilizing) policies.

It is standard to associate the network with a probability space. Specifically, four independent *i.i.d.* sequences of non-negative random variables with positive finite mean are taken as primitives of the construction. These random variables signify the durations of consecutive operations. The strictly positive parameters λ_i and μ_i are the processing rates of these operations (i.e. inverses of mean durations). A full construction of the model is described in [6].

In this paper we shall impose a simplifying assumption in which the random variables have a memory-less exponential distribution. In this case, if preemption of operations is allowed and it is assumed that job durations are not known upon commencement, then the state space of the system can be represented by $\mathcal{S} = \mathbb{Z}_+^2$, where a state $(x_1, x_2) \in \mathcal{S}$ implies that there are x_i jobs in the queue i. In our setting, any deterministic and *non-idling* control policy can be represented by a function,

$$\mathcal{P} : \mathcal{S} \to \{\text{push}, \text{pull}\}^2,$$

whose image represents the actions of the servers: $\big(\text{action server 1}, \text{action server 2}\big)$. Since idling is not an option, the control should satisfy for any integer $x \geq 0$ the following conditions

$$\mathcal{P}\big((x,0)\big) \in \{\text{push}\} \times \{\text{push}, \text{pull}\}, \quad \mathcal{P}\big((0,x)\big) \in \{\text{push}, \text{pull}\} \times \{\text{push}\}.$$

Thus it is also implied that $\mathcal{P}\big((0,0)\big) = (\text{push}, \text{push})$.

Given a policy $\mathcal{P}$, the evolution of the network is well represented by a Markov jump process (see for example [8], Chapter II). In this case it is natural to define the following stability properties.

Definition 1.1. *A network controlled by a policy $\mathcal{P}$ is said to be* **stable** *if the associated Markov jump process has a positive recurrent class such that the process enters this class with probability 1. Further, a given network is said to be* **stabilizable** *if there exists a policy $\mathcal{P}$ for which the network is stable. Otherwise the network is* **non-stabilizable***.*

We emphasize that the above definition relates to policies in which the servers are fully utilized. Achieving stability when idling is allowed (no pushing) is trivial.

In [5] (see also [6] for general processing times), the authors have shown that the push-pull network is stabilizable in the case $\lambda_i < \mu_i$, $i = 1, 2$ or in the case $\lambda_i > \mu_i$, $i = 1, 2$. Further, it is obvious by capacity considerations that there is no such policy if $\lambda_1 < \mu_1$, $\lambda_2 > \mu_2$, or in the alternative where the indexes are switched. The remaining open question is the critical case,

$$\lambda_i = \mu_i, \ i = 1, 2.$$

In that case, while there exist simple rate-stable policies in which the associated Markov jump process is null recurrent (see Theorem 1 in [5] and the discussion in Sections 1 and 2 in [6]), it was conjectured in [5] (page 83) that the network is non-stabilizable. Our key contribution in this note is in settling that conjecture. Our proof is based on a simple yet non-trivial martingale stopping argument in which a linear martingale is constructed for *any* possible policy.

A second contribution of the paper is that we are able to extend the method for proving non-stabilizability to more general networks. The novelty of our method is that it puts forward a simple matrix rank criterion as a sufficient condition required for a network to be non-stabilizable. As we show, this condition turns out to be powerful enough to be applied in various types of queueing networks that generate their own input. We prove non-stabilizability for two structured generalizations of the push-pull network in certain critical cases. First we handle a ring with an even number of push-pull servers (Figure 2). Second we handle a two server network with multiple re-entrant lines (Figure 3). Stable policies have been found for both of these types of models for certain non-critical parameter settings in [7].

The remainder of this paper is structured as follows. In Section 2 we prove that the critical push-pull is non-stabilizable. In Section 3 we prove similar results for two generalizations: A ring with an even number of critical push-pull servers and a two server network with re-entrant lines. Closing comments are in Section 4.

2. The Critical Push-Pull is Non-Stabilizable

Note that a policy $\mathcal{P}$ induces a random walk on $\mathbb{Z}_+^2$ with transitions of the form (↳, ↰, $\leftrightarrow$, $\updownarrow$) corresponding to

$$\big((\text{push}, \text{push}), (\text{pull}, \text{pull}), (\text{push}, \text{pull}), (\text{pull}, \text{push})\big).$$

This is a discrete time embedded Markov chain, $\{X_n, n \geq 0\}$ with transition probabilities, $p_{(x_1,x_2),(x_1',x_2')} = \mathbb{P}\big(X_{n+1} = (x_1', x_2') \,|\, X_n = (x_1, x_2)\big)$ defined as follows:

$$p_{(x_1,x_2),(x_1+1,x_2)} = \frac{\lambda_1}{\lambda_1 + \lambda_2}, \quad p_{(x_1,x_2),(x_1,x_2+1)} = \frac{\lambda_2}{\lambda_1 + \lambda_2}, \qquad \text{if} \quad \mathcal{P}\big((x_1, x_2)\big) = (\text{push}, \text{push}),$$

$$p_{(x_1,x_2),(x_1-1,x_2)} = \frac{\mu_1}{\mu_1 + \mu_2}, \quad p_{(x_1,x_2),(x_1,x_2-1)} = \frac{\mu_2}{\mu_1 + \mu_2}, \qquad \text{if} \quad \mathcal{P}\big((x_1, x_2)\big) = (\text{pull}, \text{pull}),$$

$$p_{(x_1,x_2),(x_1-1,x_2)} = \frac{\mu_1}{\mu_1 + \lambda_1}, \quad p_{(x_1,x_2),(x_1+1,x_2)} = \frac{\lambda_1}{\mu_1 + \lambda_1}, \qquad \text{if} \quad \mathcal{P}\big((x_1, x_2)\big) = (\text{push}, \text{pull}),$$

$$p_{(x_1,x_2),(x_1,x_2-1)} = \frac{\mu_2}{\mu_2 + \lambda_2}, \quad p_{(x_1,x_2),(x_1,x_2+1)} = \frac{\lambda_2}{\mu_2 + \lambda_2}, \qquad \text{if} \quad \mathcal{P}\big((x_1, x_2)\big) = (\text{pull}, \text{push}).$$

A given class of $\{X_n\}$ is positive recurrent, if and only if the associated class in the Markov jump process is positive recurrent. This holds since all transitions rates in the chain are bounded from above by $\lambda_1 + \lambda_2 + \mu_1 + \mu_2$ and thus the Markov jump process is non-explosive (see for example [9], Theorem 3.5.3).

In the case where $\lambda_1 = \mu_1 = \mu_2 = \lambda_2$ the process $\{X_n, n \geq 0\}$ is a simple symmetric random walk on a degenerate (*non*-random) environment similar to the random walks studied in [10].

Theorem 2.1. *The critical push-pull network is non-stabilizable.*

Proof Assume that there exists a policy $\mathcal{P}$ such that X_n has a positive recurrent class, $\mathcal{B} \subset \mathcal{S}$. Define,

$$g\big((x_1, x_2)\big) := \lambda_1 x_1 - \lambda_2 x_2.$$

Under the same probability space, define $Z_n = g(X_n)$. It is readily verified by the transition probabilities above that $\mathbb{E}\,[Z_{n+1} | \sigma(X_0, \ldots, X_n)] = Z_n$ and $\mathbb{E}\,[|Z_n|] < \infty$ hence $\{Z_n\}$ is a martingale (for any choice of $\mathcal{P}$).

Pick now two arbitrary states $\mathbf{x}, \mathbf{y} \in \mathcal{B}$ such that $g(\mathbf{x}) \neq g(\mathbf{y})$. It is obvious that two such states exist under the assumption of positive recurrence of $\mathcal{B}$ and the form of $g(\cdot)$. Set $X_0 = \mathbf{x}$ w.p. 1 and define the stopping time $T = \inf\{n \geq 0 : X_n = \mathbf{y}\}$. Since $\mathcal{B}$ is assumed to be positive-recurrent, $\mathbb{E}\,[T] < \infty$. As can be verified with the triangle inequality, $|Z_{n+1} - Z_n| < 1$ a.s. and thus, by the optional stopping theorem (see for example [11], Section 10.10), $\mathbb{E}\,[Z_T] = \mathbb{E}\,[Z_0]$. Hence,

$$g(\mathbf{x}) = \mathbb{E}\,[Z_0] = \mathbb{E}\,[Z_T] = g(\mathbf{y}); \tag{1}$$

a contradiction. Hence there cannot exist a positive recurrent class $\mathcal{B}$. $\square$

3. Generalizations

The key idea of the proof of Theorem 2.1 is to find a function $g(\cdot)$ over $\mathcal{S}$ that is a martingale (harmonic function) for any possible policy $\mathcal{P}$. The fact that a linear harmonic function was successfully employed in the critical push-pull network suggests that the method can be generalized to higher dimensional queueing networks.

In this section we first present general sufficient criteria for non-stabilizability of objects that we refer to as *homogeneous controlled queueing networks*. Then, we employ this result to show that two special cases that are structured generalizations of the critical push-pull network are non-stabilizable.

Our object of study is a controlled discrete time Markov chain $\{X_n,\ \ n \geq\ 0\}$ with state space $\mathcal{S} = \mathbb{Z}_+^M$. Denote by $\mathcal{A}(\mathbf{z})$ the set of actions that can be performed from any state $\mathbf{z} \in \mathcal{S}$,. Let $\mathcal{A} = \bigcup_{\mathbf{z}\in\mathcal{S}} \mathcal{A}(\mathbf{z})$ be the set of all possible actions and assume that $|\mathcal{A}| = L < \infty$. A deterministic policy is then a function, $\mathcal{P} : \mathcal{S} \to \mathcal{A}$, with the restriction $\mathcal{P}(\mathbf{z}) \in \mathcal{A}(\mathbf{z})$. It is assumed that under such policies $X_n \in \mathcal{S}$ for all n. Transition probabilities depend on the selected action $a \in \mathcal{A}(\mathbf{z})$ and are assumed to be specified by $\mathbb{P}_a\big(X_{n+1} = \cdot \,|\, X_n = \mathbf{z}\big)$.

These objects are termed *homogenous* since we assume that,

$$\tilde{P}_a(\mathbf{x}) := \mathbb{P}_a\big(X_{n+1} - \mathbf{z} = \mathbf{x} \,|\, X_n = \mathbf{z}\big),$$

is independent of $\mathbf{z}$. Further they resemble *queueing networks* due to the transition structure that we describe now. Let $\{\mathbf{e}_i\}_{i=1}^M$ denote the canonical row vectors in $\mathbb{R}^M$ and define the set of possible transitions,

$$\mathcal{D} := \{\mathbf{x} \in \mathbb{R}^M \ :\ \tilde{P}_a(\mathbf{x}) > 0 \text{ for some } a \in \mathcal{A}\}.$$

We assume elements of $\mathcal{D}$ are of one of these three forms: $\mathbf{e}_i$, $-\mathbf{e}_i$ or $\mathbf{e}_i - \mathbf{e}_j$ for $i \neq j$. It is further useful to define

$$\mathcal{I}_1 := \{i : \mathbf{e}_i \in \mathcal{D} \text{ or } -\mathbf{e}_i \in \mathcal{D}\}, \qquad \mathcal{I}_2 := \{(i,j) : \mathbf{e}_i - \mathbf{e_j} \in \mathcal{D}\}.$$

Elements of $\mathcal{I}_1$ correspond to arrivals or departures of jobs into the network. Elements of $\mathcal{I}_2$ correspond to movements of jobs between queues in the network.

Our goal is to establish non-stabilizability in certain parameter cases. Similar to Definition 1.1 we say that these more general networks are *non-stabilizable* if there does not exist a policy $\mathcal{P}$ that induces a positive recurrent class that is reached w.p. 1. Mirroring Theorem 2.1, our key step used to prove non-stabilizability is to find a function $g : \mathcal{S} \to \mathbb{R}$ that is harmonic for all policies. The function g should satisfy

$$\mathbb{E}_a[g(X_{n+1}) - g(X_n) \,|\, X_n = \mathbf{z}] = 0, \quad \mathbf{z} \in \mathcal{S},\ a \in \mathcal{A}(\mathbf{z}). \tag{2}$$

The appeal of being homogenous and having a finite action space $\mathcal{A} = \{a_1, \dots, a_L\}$ is that the number of these equations reduces to L:

$$\mathbb{E}_{a_i}[g(X_{n+1}) - g(X_n) \,|\, X_n] = 0, \quad i = 1, \dots, L.$$

It is now sensible to restrict the search of harmonic functions to the class of linear functions, $g(\mathbf{z}) = \boldsymbol{\alpha}'\mathbf{z}$ with $\boldsymbol{\alpha} \in \mathbb{R}^M$ and $\boldsymbol{\alpha} \neq \mathbf{0}$. In this case, proving that g is harmonic reduces to finding $\boldsymbol{\alpha} \neq \mathbf{0}$ such that,

$$\boldsymbol{\alpha}'\, \boldsymbol{\Delta}_i = 0, \quad i = 1, \dots, L, \tag{3}$$

where $\boldsymbol{\Delta}_i := \mathbb{E}_{\,a_i}[X_{n+1} - X_n \,|\, X_n]$ is the *drift vector of action* a_i. We define the $L \times M$ dimensional *action drift matrix* $\mathbf{D}$ as having rows $\boldsymbol{\Delta}_i'$, $i = 1, \dots, L$. Therefore (3) becomes

$$\mathbf{D}\boldsymbol{\alpha} = \mathbf{0}. \tag{4}$$

We are now in a position to state a sufficient condition for non-stabilizability:

Theorem 3.1. *Consider a homogeneous controlled queueing network* $\{X_n,\ n \geq 0\}$ *with action drift matrix* $\mathbf{D}$*. Then it is non-stabilizable if*

$$\operatorname{rank}\big(\mathbf{D}\big) < M, \tag{5}$$

and the following non-degeneracy condition holds: The system (4) *has a solution* $\boldsymbol{\alpha}$ *such that for every* $\mathbf{x} \in \mathcal{S}$ *and every* $a \in \mathcal{A}(\mathbf{x})$, $\mathbb{P}_a\big(\boldsymbol{\alpha}'X_1 \neq \boldsymbol{\alpha}'\mathbf{x} \,|\, X_0 = \mathbf{x}\big) > 0$.

Proof The proof follows the exact same lines as the proof of Theorem 2.1. The rank condition implies that (4) has a non-trivial solution $\boldsymbol{\alpha}$, so there exists a non-trivial linear function $g(\cdot)$ that is harmonic for all policies. The non-degeneracy condition implies that for every $\mathbf{x} \in \mathcal{S}$ there exists a state $\mathbf{y} \in \mathcal{S}$ which is reachable from $\mathbf{x}$ and is such that $g(\mathbf{x}) \neq g(\mathbf{y})$. Hence, if it is assumed that there exists a positive recurrent class $\mathcal{B}$, then a contradiction as in (1) can be reached. The additional condition needed for the optional stopping theorem requiring that $|Z_{n+1} - Z_n|$ be bounded is guaranteed by the form of $\mathcal{D}$. □

Observe that the technical non-degeneracy condition actually implies (5), yet we view the rank condition (5) as the central pillar for establishing non-stabilizability. The lemma below puts forward sufficient conditions for satisfying the non-degeneracy condition:

Lemma 3.2. *Consider a homogeneous controlled queueing network for which there exists an $\boldsymbol{\alpha} \neq \mathbf{0}$ solving (4). If for every $i \in \mathcal{I}_1$, $\alpha_i \neq 0$ and for every $(i,j) \in \mathcal{I}_2$, $\alpha_i \neq \alpha_j$ then the non-degeneracy condition holds.*

Proof For every initial condition X_0 and every action in $\mathcal{A}(X_0)$, almost surely one of these two events takes place: (1) A single coordinate, $i^* \in \mathcal{I}_1$ changes. (2) Two coordinates, denoted $(i^*, j^*) \in \mathcal{I}_2$ change. If (1) occurs then $|\boldsymbol{\alpha}'(X_1 - X_0)| = |\alpha_{i^*}| \neq 0$. If (2) occurs then $\boldsymbol{\alpha}'(X_1 - X_0) = \alpha_{i^*} - \alpha_{j^*} \neq 0$. So in any case, $\boldsymbol{\alpha}' X_1 \neq \boldsymbol{\alpha}' X_0$ a.s. $\square$

3.1. A Push-Pull Ring with an Even Number of Servers

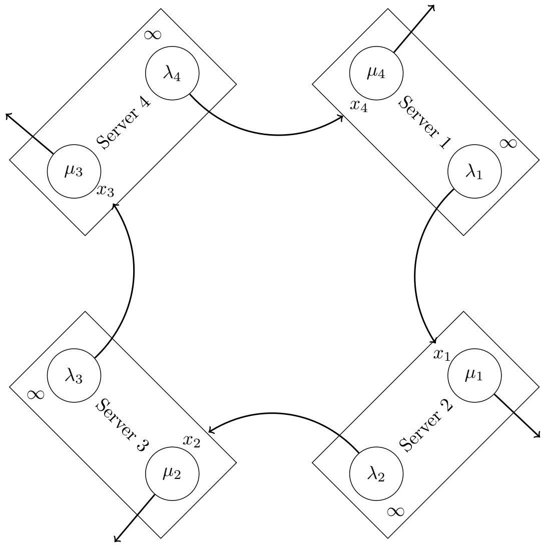

Figure 2: A Ring of $M = 4$ Push-Pull servers.

One way of generalizing the push-pull network is to allow more servers as in Figure 2. In this network there are $M \geq 2$ servers with the same number of streams. Each stream has a push operation at rate λ_i and a pull operation at rate μ_i. Our index notation implies that server i has a choice between *push* to stream i (at rate λ_i) or *pull* from stream $i-1$ (at rate μ_{i-1}). All index arithmetics are modulo M on $\{1, \ldots, M\}$.

Similar to the push-pull network (the special case with $M = 2$), a non-idling policy is a mapping $\mathcal{P} : \mathcal{S} \to \{\text{push}, \text{pull}\}^M$, with the restriction that the pull operation i (on server $i+1$) can only be performed if $x_i > 0$.

Stability properties of this model under a "pull-priority" policy (and general processing times) were investigated in [7], Section 5. In there it is shown that the network is stable if $\lambda_i < \mu_i$, $i = 1, \ldots, M$ as well as if M is odd and $\lambda_i > \mu_i$, $i = 1, \ldots, M$, yet the ratio λ_i / μ_i is "not too large" (see [7] for the full details). We further believe that for odd M, the network is stabilizable in the critical case ($\lambda_i = \mu_i$ for all i) by means of a pull-priority policy. This has not been established in [7], yet stems from the same intuition appearing in [7] based on the concept of a *mode*.

An interesting aspect of push-pull network rings is that *even rings* (network rings with an even number of servers M) are drastically different to *odd rings*. For example, the pull-priority policy cannot be stablilized for even rings in which $\lambda_i > \mu_i$ for all i: To see this, assume the state of the system is,

$$\big(x_1, x_2, \dots, x_n\big) = \big(+, 0, +, 0, \dots, +, 0\big),$$

where '+' indicates a strictly positive quantity. Then, the selected action under pull-priority is

$$\big(\text{push}, \text{pull}, \text{push}, \text{pull}, \dots, \text{push}, \text{pull}\big).$$

In this case there is a strictly positive probability that the '+' queues will grow without bound. The drastic differences between even and odd rings go beyond the pull-priority policy of [7]. We now show:

Theorem 3.3. *The critical push-pull ring with M even is non-stabilizable.*

Proof The push-pull ring, described as a homogeneous controlled random walk, has action drift matrix $\mathbf{D}$ of dimension $2^M \times M$, $\mathcal{I}_1 = \{1, \dots, M\}$ and $\mathcal{I}_2 = \emptyset$. We apply Theorem 3.1 and Lemma 3.2. The rank and non-degeneracy conditions are ensured by the $\boldsymbol{\alpha}$ found in Lemma 3.4 below. □

Lemma 3.4. *In the critical case*

$$\operatorname{rank}\big(\mathbf{D}\big) = \begin{cases} M & \textit{if} \quad M \textit{ is odd,} \\ M-1 & \textit{if} \quad M \textit{ is even.} \end{cases}$$

Further, when M is even a solution of (4) *is*

$$\boldsymbol{\alpha} = \big(\ +\lambda_1^{-1} \quad -\lambda_2^{-1} \quad +\lambda_3^{-1} \quad -\lambda_4^{-1} \quad \dots \quad +\lambda_{M-1}^{-1} \quad -\lambda_M^{-1}\ \big)'.$$

Proof

Define $\beta_j := \lambda_j - \mu_j$, $j = 1, \dots, M$ and for a given action $i \in \{1, \dots, 2^M\}$ let r_i be the sum of rates associated with the action. Then $r_i\,\boldsymbol{\Delta}_i$ has entries that depend on the action i as follows:

Operation on:		
Server $j-1$	Server j	Entry j
push	push	λ_j
push	pull	β_j
pull	push	0
pull	pull	$-\mu_j$

This, in turn, implies the following relation between consecutive entries:

Entry j		Entry $j+1$		
λ_j	$\Rightarrow$	λ_{j+1}	or	β_{j+1}
$-\mu_j$	$\Rightarrow$	$-\mu_{j+1}$	or	0
0	$\Rightarrow$	β_{j+1}	or	λ_{j+1}
β_j	$\Rightarrow$	0	or	$-\mu_{j+1}$

Define the matrix, $\hat{\mathbf{D}} = \text{sign}\big(\mathbf{D}\big)$, where the function sign$(\cdot)$ is taken element-wise. In the critical case ($\beta_i = 0$, $i = 1, \ldots, M$) it is evident that rank$(\hat{\mathbf{D}})$ = rank$(\mathbf{D})$ since $\mathbf{D}_{i,j} = \hat{\mathbf{D}}_{i,j}\lambda_j/r_j$. Further, by considering the structure of consecutive entries in the table above, it is evident that in each row of $\hat{\mathbf{D}}$:

(i) The number of 0's separating two non-zero entries with the **opposite sign** is **odd**.
(ii) The number of 0's separating two non-zero entries with the **same sign** is **even**.

A consequence of (i) and (ii) is that the number of zero entries in each row of $\hat{\mathbf{D}}$ is even since there is an even number of 0-sequences that have an odd number of zeros. Hence in an odd/even ring there is an odd/even number of non-zero entries.

M odd case:
Observe each of the vectors in $\{\mathbf{e}_i : i = 1, \ldots, M\}$ is also a row of $\hat{\mathbf{D}}$ and therefore it has full rank.

M even case:
First, we prove that rank$(\hat{\mathbf{D}}) \geq M - 1$. Second, we show $\hat{\mathbf{D}}\hat{\boldsymbol{\alpha}} = \mathbf{0}$, where,

$$\hat{\boldsymbol{\alpha}} = \begin{pmatrix} +1 & -1 & +1 & -1 & \ldots & +1 & -1 \end{pmatrix}',$$

this immediately implies that $\mathbf{D}\boldsymbol{\alpha} = \mathbf{0}$.

Consider the vectors $\mathbf{f}_i = \mathbf{e}_i + \mathbf{e}_{i+1}$, $i = 1, \ldots, M-1$, observing these are rows of $\hat{\mathbf{D}}$. Next, define $\mathbf{B}$ to be the matrix of size $(M-1) \times M$ with i'th row $\mathbf{f}_i$. Let $\mathbf{B}_{-1}$ be a square matrix of size $(M-1) \times (M-1)$ obtained by deleting the first column of $\mathbf{B}$. Clearly, $\mathbf{B}_{-1}$ has full rank (the determinant of a lower triangular matrix is the product of its diagonal elements). Hence rank$(\hat{\mathbf{D}}) \geq M - 1$.

To show rank$(\hat{\mathbf{D}}) < M$ we show that $\hat{\boldsymbol{\Delta}}'\hat{\boldsymbol{\alpha}} = 0$ for any row $\hat{\boldsymbol{\Delta}} = (\delta_1, \ldots, \delta_M)'$ of the matrix $\hat{\mathbf{D}}$.

Let i_k be the index of the k-th non-zero entry of $\hat{\boldsymbol{\Delta}}$ starting from index 1, and m its total number of non-zero entries. Since m is even, then we can write

$$\hat{\boldsymbol{\Delta}}'\hat{\boldsymbol{\alpha}} = \sum_{k=1}^{m} \delta_{i_k}\hat{\alpha}_{i_k} = \sum_{k=1}^{m/2} \big(\delta_{i_{2k-1}}\hat{\alpha}_{i_{2k-1}} + \delta_{i_{2k}}\hat{\alpha}_{i_{2k}}\big)$$

Define $\ell_k := i_{2k} - i_{2k-1} - 1$ the number of zeros between the non zero consecutive entries $\delta_{i_{2k-1}}$ and $\delta_{i_{2k}}$. By the definition of $\widehat{\mathbf{\Delta}}$ and $\hat{\boldsymbol{\alpha}}$ there are only two possibilities

$$\ell_k = \begin{cases} \text{odd} & \Longrightarrow \quad \delta_{i_{2k-1}} = -\delta_{i_{2k}} \quad \text{and} \quad \hat{\alpha}_{i_{2k-1}} = \hat{\alpha}_{i_{2k}}, \\ \text{even} & \Longrightarrow \quad \delta_{i_{2k-1}} = \delta_{i_{2k}} \quad \text{and} \quad \hat{\alpha}_{i_{2k-1}} = -\hat{\alpha}_{i_{2k}}. \end{cases}$$

This implies that

$$\delta_{i_{2k-1}}\hat{\alpha}_{i_{2k-1}} + \delta_{i_{2k}}\hat{\alpha}_{i_{2k}} = 0, \qquad k = 1, \ldots, m/2,$$

proving that $\hat{\boldsymbol{\alpha}}$ is a solution of $\hat{\mathbf{D}}\hat{\boldsymbol{\alpha}} = \mathbf{0}$. $\square$

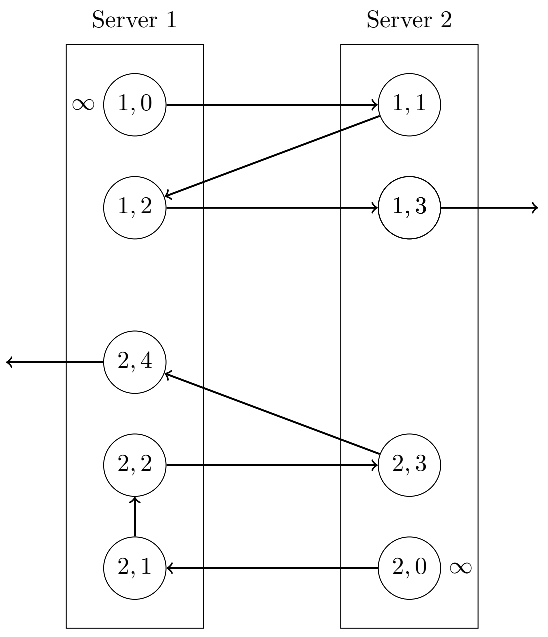

Figure 3: A network of re-entrant lines on two servers. In this example $S = 2$, $n_1 = 3$ and $n_2 = 4$. Hence the number of buffers is $M = 7$. Further the number of actions that a control policy may choose is $L = 20$.

3.2. Re-entrant Lines on Two Servers

We now consider a network as that appearing in Figure 3 essentially generalizing the push-pull by allowing streams of more than two steps that may re-enter servers multiple times. Denote the number of streams by $S \geq 1$. Each stream $i \in \{1, \ldots, S\}$, has $n_i + 1$ operations labeled by $(i, 0), \ldots, (i, n_i)$, where the operation $(i, 0)$ is associated with an infinite supply of materials and the other operations are associated with queues. Assume $n_i \geq 1$. Hence the number of queues in the system is,

$$M = \sum_{i=1}^{S} n_i,$$

and the total number of operations is $M + S$. As in the push-pull network there are two servers, labeled 1 and 2. Each operation (i, j) is performed by a unique server denoted $\sigma(i, j) \in \{1, 2\}$. A control policy is then a rule based on the number of jobs in each of the queues, indicating for each server what operation it needs to work on.

The processing rate of operation (i, j) is denoted by $\mu_{i,j} > 0$. It is useful to denote, for each stream, $i \in \{1, \ldots, S\}$, the set of operations on server $\ell \in \{1, 2\}$ by,

$$C_\ell(i) := \big\{ j \in \{0, \ldots, n_i\} \ : \sigma(i, j) = \ell \big\}.$$

Motivated by [7], Section 4, we say that the network is *critical* if:

$$\sum_{j \in C_1(i)} \mu_{i,j}^{-1} = \sum_{j \in C_2(i)} \mu_{i,j}^{-1}, \quad i = 1, \ldots, S. \tag{6}$$

This means that for any job, the mean processing duration of service required by the job on each of the servers is equal.

The case of $S = 1$ is typically called the re-entrant line (with infinite supply). It was analyzed in [7], Section 3 under a last buffer first serve policy. The case of $S = 2$ is a direct generalization of the push-pull network. It was analyzed in [7], Section 4 under a specific priority policy.

For notational purpuses, it is useful to order the M queues using some arbitrary permutation of the set $\{1, \ldots, M\}$. Denote $\bar{k}(i, j)$ to be the queue served by operation (i, j), $i \in \{1, \ldots, S\}$, $j \in \{1, \ldots, n_i\}$ (note that operations $(i, 0)$ do not have an associated queue). Further denote $\bar{i}(k)$ and $\bar{j}(k)$ as the inverse mappings, i.e., $\bar{k}\big(\bar{i}(k), \bar{j}(k)\big) = k$.

Similar to the push-pull ring of the previous sub-section, it is straightforward to model this network as a controlled random walk. In doing so, it is useful to partition the set $\mathcal{I}_1$ into $\mathcal{I}_{1,+}$, the queues fed by *push* operations at start of each stream and $\mathcal{I}_{1,-}$, the queues drained by *pull* operations at the end of each stream. Further, the set $\mathcal{I}_2$ is associated with operations that are neither at the start or the end of the stream, involving movement of jobs between buffers without changing the total number of jobs in the system. We now have,

$$\begin{aligned}
\mathcal{I}_{1,+} &= \big\{ k \in \{1, \ldots, M\} \ : \ \bar{j}(k) = 1 \big\}, \\
\mathcal{I}_{1,-} &= \big\{ k \in \{1, \ldots, M\} \ : \ \bar{j}(k) = n_{\bar{i}(k)} \big\}, \\
\mathcal{I}_2 &= \big\{ (k, k') \in \{1, \ldots, M\}^2 \ : \ \bar{i}(k) = \bar{i}(k'),\ \bar{j}(k) = \bar{j}(k') - 1 \big\}.
\end{aligned}$$

Denote $L_\ell = \sum_{i=1}^{S} |C_\ell(i)|$ for $\ell = 1,2$. Then the total number of actions is $L = L_1\, L_2$ and the action drift matrix $\mathbf{D}$ is of dimension $L \times M$ as is consistent with previous notation. Denote for each operation (i,j) and each $k \in \{1,\dots,M\}$:

$$\hat{\Delta}_k(i,j) = \begin{cases} \begin{cases} \mu_{i,0} & k = \bar{k}(i,1), \\ 0 & \text{otherwise}, \end{cases} & j = 0, \\ \begin{cases} -\mu_{i,j} & k = \bar{k}(i,j), \\ \mu_{i,j} & k = \bar{k}(i,j+1), \\ 0 & \text{otherwise}, \end{cases} & j \in \{1,\dots,n_i-1\}, \\ \begin{cases} -\mu_{i,n_i} & k = \bar{k}(i,n_i), \\ 0 & \text{otherwise}, \end{cases} & j = n_i. \end{cases}$$

Let $\hat{\boldsymbol{\Delta}}(i,j) \in \mathbb{R}^M$ be the vector of these elements. Now each row of $\mathbf{D}$ corresponds to two actions, (i_1,j_1) such that $\sigma(i_1,j_1) = 1$ and (i_2,j_2) such that $\sigma(i_2,j_2) = 2$. We denote this row by $\boldsymbol{\Delta}(i_1,j_1,i_2,j_2)'$ with,

$$\boldsymbol{\Delta}(i_1,j_1,i_2,j_2) := \frac{1}{||\hat{\boldsymbol{\Delta}}(i_1,j_1) + \hat{\boldsymbol{\Delta}}(i_2,j_2)||_1}(\hat{\boldsymbol{\Delta}}(i_1,j_1) + \hat{\boldsymbol{\Delta}}(i_2,j_2)).$$

Having defined the controlled random walk we are now ready to prove that it is non-stabalizable.

Theorem 3.5. *The critical network is non-stabilizable.*

Proof We find an $\boldsymbol{\alpha} \in \mathbb{R}^M$ such that $\mathbf{D}\boldsymbol{\alpha} = \mathbf{0}$. The elements of $\boldsymbol{\alpha}$ are,

$$\alpha_k = \sum_{j=0}^{\bar{j}(k)-1} \frac{1}{\mu_{\bar{i}(k),j}} (-1)^{\sigma(\bar{i}(k),j)}.$$

For any $i \in \{1,\dots,S\}$, it is straightforward to verify that if $j = 0$,

$$\hat{\boldsymbol{\Delta}}(i,j)'\boldsymbol{\alpha} = \mu_{i,0}\frac{1}{\mu_{i,0}}(-1)^{\sigma(i,j)} = (-1)^{\sigma(i,j)}.$$

Further, if $j \in \{1,\dots,n_i-1\}$ then,

$$\hat{\boldsymbol{\Delta}}(i,j)'\boldsymbol{\alpha} = -\mu_{i,j}\sum_{j'=0}^{j-1}\frac{1}{\mu_{i,j'}}(-1)^{\sigma(i,j')} + \mu_{i,j}\sum_{j'=0}^{j}\frac{1}{\mu_{i,j'}}(-1)^{\sigma(i,j')} = (-1)^{\sigma(i,j)}.$$

Further if $j = n_i$,

$$\begin{aligned}\hat{\mathbf{\Delta}}(i,j)'\boldsymbol{\alpha} &= -\mu_{i,n_i}\sum_{j'=0}^{n_i-1}\frac{1}{\mu_{i,j'}}(-1)^{\sigma(i,j')}\\ &= -\mu_{i,n_i}\Big(\sum_{j'\in C_2(i)}\mu_{i,j'}^{-1} - \sum_{j'\in C_1(i)}\mu_{i,j'}^{-1} - \mu_{i,n_i}^{-1}(-1)^{\sigma(i,n_i)}\Big) = (-1)^{\sigma(i,n_i)}.\end{aligned}$$

In the last equality we use the fact that the network is critical, (6). Hence for any (i,j),

$$\hat{\mathbf{\Delta}}(i,j)'\boldsymbol{\alpha} = (-1)^{\sigma(i,j)}.$$

Now for any row of $\mathbf{D}$, i.e. for any (i_1,j_1), (i_2,j_2) such that $\sigma(i_1,j_1)=1$, $\sigma(i_2,j_2)=2$ we have that,

$$\boldsymbol{\alpha}'\mathbf{\Delta}(i_1,j_1,i_2,j_2) = \frac{1}{||\hat{\mathbf{\Delta}}(i_1,j_1)+\hat{\mathbf{\Delta}}(i_2,j_2)||_1}\big((-1)^{\sigma(i_1,j_1)} + (-1)^{\sigma(i_2,j_2)}\big) = 0,$$

as needed. Since $\boldsymbol{\alpha} \neq \mathbf{0}$, the rank condition of Theorem 3.1 holds.

To see that the non-degeneracy condition is satisfied, we use Lemma 3.2. First, for every $k \in \mathcal{I}_{1,+}$, it is evident that $\alpha_k \neq 0$. Further, for every $k \in \mathcal{I}_{1,-}$, denote $i = \bar{i}(k)$, then

$$\alpha_k = \sum_{j=0}^{n_i-1}\mu_{i,j}^{-1}(-1)^{\sigma(i,j)} = \sum_{j\in C_2(i)}\mu_{i,j}^{-1} - \sum_{j\in C_1(i)}\mu_{i,j}^{-1} - \mu_{i,n_i}^{-1}(-1)^{\sigma(i,n_i)} \neq 0,$$

where the last inequality follows from the fact the network is critical. Finally, for every pair $(k,k') \in \mathcal{I}_2$ we verify

$$\alpha_{k'} - \alpha_k = \frac{1}{\mu_{\bar{i}(k),\bar{j}(k)}}(-1)^{\sigma(\bar{i}(k),\bar{j}(k))} \neq 0,$$

as required. □

4. Closing Comments

The interest in non-stabilizability of the fully-utilizing critical push-pull network stems from the fact that in symmetric non-critical cases it can be stabilized. This is true for both $\lambda < \mu$ and $\lambda > \mu$ in contrast to traditional queueing networks. Hence our contribution that the network is non-stabilizable for $\lambda = \mu$ is of interest.

The consequences to more general networks as appearing in Section 3 are interesting in their own right, since Theorem 3.1 provides a general sufficient condition for non-stabilizability of quite general models.

It should be noted that the class of policies, $\{\mathcal{P}\}$, we consider is that of stationary deterministic policies. One can also consider randomized policies where an action $a \in \mathcal{A}$ is taken with a given probability. Extending the non-stabilizability results to this case is straightforward and carries no surprise. Further, using continuous time martingales, it is merely technical matter to extend to non-stationary policies.

As opposed to non-deterministic and/or non-stationary policies, a non-trivial extension is to consider networks with general processing times (not necessarily exponentially distributed). In this case one way of providing a Markovian description to a network is based on residual service times (see for example [7] and references there-in). In that case, we conjecture that the non-stabilizability results of this paper carry over, yet proving such results may require a different set of tools than the one we have used here.

References

[1] M. Bramson, Stability of Queueing Networks, Springer, 2008.

[2] H. Chen, D. D. Yao, Fundamentals of Queueing Networks: Performance, Asymptotics, and Optimization, Springer, 2001.

[3] S. P. Meyn, Control Techniques for Complex Networks, Cambridge University Press, 2008.

[4] A. Kopzon, G. Weiss, A push pull queueing system, Operations Research Letters 30 (6) (2002) 351–359.

[5] A. Kopzon, Y. Nazarathy, G. Weiss, A push–pull network with infinite supply of work, Queueing Systems 62 (1) (2009) 75–111.

[6] Y. Nazarathy, G. Weiss, Positive Harris recurrence and diffusion scale analysis of a push pull queueing network, Performance Evaluation 67 (4) (2010) 201–217.

[7] Y. Guo, E. Lefeber, Y. Nazarathy, G. Weiss, H. Zhang, Stability and performance for multi-class queueing networks with infinite virtual queues, Preprint.

[8] S. Asmussen, Applied Probability and Queues, Springer-Verlag, 2003.

[9] J. R. Norris, Markov Chains, Cambridge University Press, 1997.

[10] M. Holmes, T. S. Salisbury, Random walks in degenerate random environments, Arxiv preprint arXiv:1105.5105.

[11] D. Williams, Probability with Martingales, Cambridge University Press, 1991.