Queueing Syst (2009) 62: 75–111
DOI 10.1007/s11134-009-9121-1

A Push–Pull network with infinite supply of work

Anat Kopzon · Yoni Nazarathy · Gideon Weiss

Received: 31 January 2007 / Revised: 28 July 2008 / Published online: 20 May 2009
© Springer Science+Business Media, LLC 2009

Abstract We consider a two-node multiclass queueing network with two types of jobs moving through two servers in opposite directions, and there is infinite supply of work of both types. We assume exponential processing times and preemptive resume service. We identify a family of policies which keep both servers busy at all times and keep the queues between the servers positive recurrent. We analyze two specific policies in detail, obtaining steady state distributions. We perform extensive calculations of expected queue lengths under these policies. We compare this network with the Kumar–Seidman–Rybko–Stolyar network, in which there are two random streams of arriving jobs rather than infinite supply of work.

Keywords Queueing · Manufacturing · Markovian multiclass queueing networks · Infinite supply of work · Infinite virtual queues · Threshold policies · Maximum pressure policies · Kumar–Seidman–Rybko–Stolyar network

Mathematics Subject Classification (2000) 60K25 · 68M20 · 90B15

1 Introduction

This paper is concerned with *stochastic processing networks* with *infinite supply of work*, where we focus on the analysis of a specific example of such networks. A *stochastic processing network*, introduced by Harrison [14–17], is a network that uses input items and processing resources to produce various streams of output items. *Infinite supply of work*, as discussed in [1, 2, 26, 32], is provided by *infinite virtual*

Research supported in part by Israel Science Foundation Grants 249/02 and 454/05 and by European Network of Excellence Euro-NGI.

A. Kopzon · Y. Nazarathy · G. Weiss (✉)
Department of Statistics, The University of Haifa, Mount Carmel 31905, Israel
e-mail: gweiss@stat.haifa.ac.il

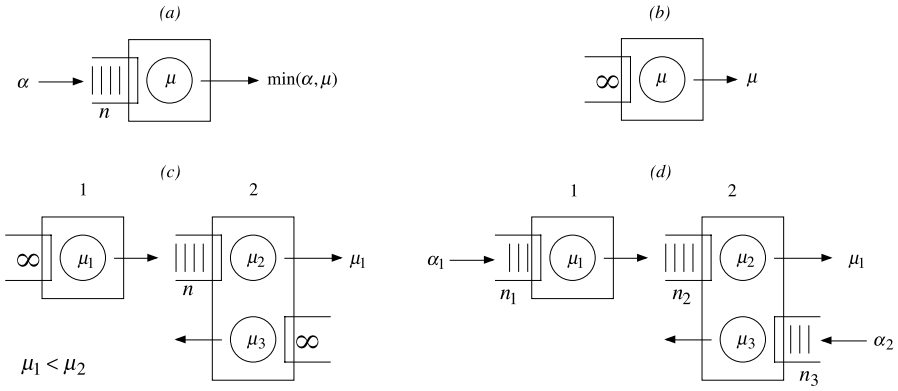

Fig. 1 Systems with infinite supply of work versus exogenous arrivals

queues (IVQ) in which input items for processing are stored and are available at any time in unlimited numbers. To be specific, we envision the IVQs to be part of the network, and processing of items out of these IVQs consumes some of the network's resources. We assume here that each resource of the network has, among the queues which it is processing, at least one IVQ. In that case none of the resources ever needs to idle. In the operation of the network we assume that each of the IVQs has a nominal processing rate at which it introduces items into the network, and each of the output streams has an output rate at which items are produced. These determine the rates at which each input activity, intermediate activity, and output activity is performed, and the offered load of each resource. The network is *rate stable* if each of the standard queues (i.e., not IVQs) in which items are stored in intermediate stages has equal input and output rates, so that there is no linear accumulation of material in these queues. A network operates in *balanced heavy traffic* if it is rate stable and if all the resources are fully utilized. In other words, the input and output rates are such that the offered load to all the resources is equal to $\rho = 1$.

When inputs to the network are exogenous and subject to stochastic variability, a rate stable network in balanced heavy traffic is always congested. As the offered load approaches 1, items accumulate at a rate of $\Theta(\sqrt{t})$. In this case the network may behave as a semi-martingale reflected Brownian motion on the diffusion scale [18, 31].

In stochastic processing networks with infinite supply of work, the situation is radically different: because inputs are not exogenous but are produced by processing of IVQs within the network, we have much more control to cope with stochastic fluctuations and with congestion. Hence we conjecture that such networks can be operated under balanced heavy traffic, with full utilization of all the resources, and yet show no congestion at all: Under appropriate conditions, there exists a wide range of policies that achieve 0 idling in the network and keep all the queues which are not IVQs positive recurrent. Our purpose in this paper is to present a detailed analysis of a simple but nontrivial example of such a network.

The concept of infinite supply of work is quite natural in many practical situations, and in particular it is very relevant to manufacturing systems. Two simple examples

illustrate the effect of infinite supply of work. First, compare a single server queue and a single machine. Consider a single server queue with service rate μ and with a stochastic arrival stream which is independent of the service times and has rate α (see Fig. 1(a)). Then the output from the server will be at rate $\min(\alpha, \mu)$, leading to three modes of behavior: When $\alpha < \mu$ output is at rate α, the queue of customers is positive recurrent (we use the term loosely here and in the following paragraphs, since we have not specified an exact model), and the server idles a fraction $1 - \frac{\alpha}{\mu}$ of the time. When $\alpha > \mu$, output is at rate μ, the server never idles, and the queue grows linearly at a rate $\alpha - \mu$. When $\alpha = \mu$, the system is rate stable, output is at rate μ, the server idles an expected fraction $\Theta(\frac{1}{\sqrt{t}})$ of the time, and the expected queue length grows at a rate $\Theta(\sqrt{t})$. Compare this to a single machine (see Fig. 1(b)) which can process raw parts at the same rate μ. Raw parts will not arrive as a random stream independent of the processing. In fact, the usual practice is to monitor the machine and make sure that it never runs out of raw parts. Infinite virtual queue is an apt name for this, since as far as the machine is concerned, it is serving an infinite queue that is never exhausted, but in reality the queue only contains a few items and is replenished before it runs out. Hence the machine will produce an output stream of finished parts at a rate of μ, and there will be no idling and no congestion.

The second example is also quite familiar. Consider two machines and parts which are processed first by machine 1 at rate μ_1, and next by machine 2 at rate μ_2. An infinite supply of raw parts will keep the first machine busy all the time and produce a stochastic input at rate μ_1 into the second machine. To fully utilize the second machine, we now assume that $\mu_2 > \mu_1$ and add another IVQ of parts which need processing only on machine 2 at rate μ_3 (see Fig. 1(c)). The network will now produce two output streams of parts, stream 1, of parts that are processed by both machines, and stream 2, which is processed only by machine 2. This network will fully utilize both machines with no idling and produce output of stream 1 at rate μ_1 and of stream 2 at rate $\mu_3(1 - \frac{\mu_1}{\mu_2})$ if we use the following type of control: We keep the two IVQs replenished, and whenever the queue of parts of stream 1 that wait for machine 2 is exhausted, machine 2 will switch to stream 2 and will continue processing it for a duration which is a stopping time with finite expectation. The queue of jobs of stream 1 between the two machines will in that case be positive recurrent. In fact machine 2 will behave like a server with vacations (see [23]). The corresponding queueing network (see Fig. 1(d)) with two random arrival streams of rates α_1, α_2 will, under a similar vacation policy, produce outputs at rates $\min(\alpha_1, \mu_1, \mu_2)$ and $\min(\alpha_2, \max(0, \mu_3(1 - \frac{\mu_1}{\mu_2})))$, and be subject to idling and congestion.

A notable property of this second example is that the suggested policy achieves full utilization, no idling, and no congestion, without using the actual processing rates of the two machines. We conjecture that the same can be done for general multiclass queueing networks and for more general processing networks: If each resource has an infinite supply of work, then, under appropriate conditions, there exist policies which fully utilize all the resources and which keep all the standard queues positive recurrent, and these policies can be implemented without knowing the exact processing rates of the various activities.

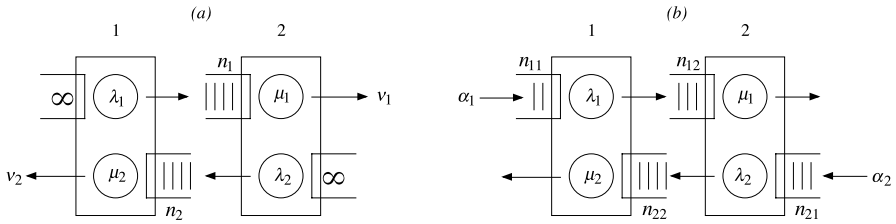

Fig. 2 The Push–Pull network (**a**) and the Kumar–Seidman–Rybko–Stolyar network (**b**)

Our model

Following this preface, we now introduce the *Push–Pull network* which we study in this paper. It was first introduced in our paper [21]. The network is described in Fig. 2(a). It can model a production system with two machines and two products. There are two servers and two types of jobs. Jobs of type 1 are processed first by server 1 and then by server 2, jobs of type 2 are processed in the opposite order, first by server 2 and then by server 1. There is an infinite supply of work for both servers, server 1 has an infinite supply of jobs of type 1 waiting for initial processing, and server 2 has an infinite supply of jobs of type 2 waiting for initial processing. Jobs which complete initial processing move to the other server and queue up for their second processing step. For time $t > 0$, $Q_i(t)$ denotes the number of jobs of type i which have completed their first step of processing at server i and have not yet completed their processing at server $\bar{\imath} = 3 - i$ for $i = 1, 2$. At any time t server i can choose to process a new job of type i out of the IVQ (always available) and send it to the queue of server $\bar{\imath}$; we call that a push operation. Alternatively, if $Q_{\bar{\imath}}(t) > 0$, it can choose to work on a job of type $\bar{\imath}$ and send it out of the network; we call that a pull operation. We let λ_i denote the processing rate of jobs of type i by the push operation at server i, and we let μ_i denote the processing rate of jobs of type i by the pull operation at server $\bar{\imath}$. In Fig. 2(a) we let n_i, $i = 1, 2$, be the values of $Q_i(t)$, the queue length (including job in service) at time t, and we let ∞ denote the infinite supply of jobs available for initial processing. In Fig. 2(b) we show the corresponding queueing network with exogenous arrivals, the so-called Kumar–Seidman–Rybko–Stolyar (KSRS) queueing network, to be discussed in Sect. 7.

Because the Push–Pull network always has work available for both servers, both servers can work all the time, thus achieving full utilization. Such full utilization corresponds to a traffic intensity of 1. Consider some policy in which both servers are working all the time and assume that this policy is rate stable, i.e., input rates equal output rates at all the queues. Denote by θ_i the long-run average fraction of time that server i is working on jobs of type i in a push operation. Since servers are working all the time, $1 - \theta_i$ is the long-run average fraction of time that server i is working on jobs of type $\bar{\imath}$ in a pull operation. If the network is to be rate stable, then we have the following equations for the production rates ν_i of jobs of types $i = 1, 2$:

$$\nu_1 = \theta_1\lambda_1 = (1 - \theta_2)\mu_1,$$
$$\nu_2 = \theta_2\lambda_2 = (1 - \theta_1)\mu_2,$$

which are solved by

$$\theta_i = \frac{\mu_i(\mu_{\bar{\imath}} - \lambda_{\bar{\imath}})}{\mu_1\mu_2 - \lambda_1\lambda_2}, \qquad i = 1, 2,\ \bar{\imath} = 3 - i. \tag{1}$$
$$\nu_i = \frac{\mu_i\lambda_i(\mu_{\bar{\imath}} - \lambda_{\bar{\imath}})}{\mu_1\mu_2 - \lambda_1\lambda_2},$$

There are several cases to distinguish here. If $\mu_i > \lambda_i$, $i = 1, 2$, then the stream of jobs of type i which arrive at server $\bar{\imath}$ has maximum input rate λ_i, and the server can process these jobs at the faster rate μ_i; in this case we say that the network is *inherently stable*. If $\lambda_i > \mu_i$, $i = 1, 2$, then the stream of jobs of type i through server $\bar{\imath}$ has maximum processing rate μ_i, and jobs may arrive at a faster rate of up to λ_i; in this case we say that the network is *inherently unstable*. The case where $\lambda_i = \mu_i$ will be referred to as the *null case*.

In the additional cases of $\lambda_i > \mu_i$, $\mu_{\bar{\imath}} > \lambda_{\bar{\imath}}$ server $\bar{\imath}$ requires longer average processing time than server i for both types. Therefore server $\bar{\imath}$ is a bottleneck, and server i will have slack capacity. Formula (1) in that case yields negative rates and is meaningless. We will not discuss these unbalanced cases any further in this paper.

Our aim in this paper is to find policies for this multiclass queueing network so that the network will complete jobs of types $i = 1, 2$ at the rates ν_i and, at the same time, will keep the queues of jobs between the servers $Q_i(t)$ stable. This is easy to do in the inherently stable case where priority to pull operations is a stable policy. It is less easy to do in the inherently unstable case, and our main results in this paper are policies which achieve this. In the null case we believe that congestion cannot be avoided.

Throughout this paper we will make two simplifying assumptions: We will assume that all processing times are independent and exponentially distributed. In addition, we will assume that preemptions are allowed. Under these two assumptions, the network is memoryless (Markovian), and its state at time t is given by the values $n_i = Q_i(t)$, $i = 1, 2$.

These assumptions can be relaxed, at the cost of getting less explicit results. The behavior of the Push–Pull network in the inherently stable case, under *non-preemptive pull priority*, is analyzed in [21], where explicit stationary distributions are obtained. In our recent paper [27] we consider the Push–Pull network with general processing time distributions, with or without preemptions, and find policies under which our results here continue to hold: the network is working at full utilization with positive Harris recurrent queues.

The rest of the paper is structured as follows. In Sects. 2–5 we study various positive recurrent policies: In the preliminary Sect. 2 we discuss the Markovian structure of the problem and the behavior of the network under preemptive priority to pull operations. In the inherently stable case this policy is stable. In the inherently unstable case this policy is unstable. This motivates threshold policies which we discuss intuitively. In Sects. 3–5 we study the inherently unstable Push–Pull network under various threshold policies. Fixed threshold policies are described in Sect. 3, and a queue balancing diagonal policy is described in Sect. 4. For these policies, we are able to derive the stationary distribution of the Markov jump process explicitly. In Sect. 5 we

define a general class of threshold policies and show that they are stable by proving positive recurrence of the Markov jump process using the Foster–Lyapunov criterion. It also follows from this analysis that the Markov process is exponentially ergodic and that the steady-state tail probabilities decay geometrically.

In Sect. 6 we discuss choosing the parameters of the policies so as to minimize the expected queue lengths. We find that it is easy to choose the parameters of the fixed threshold policy so as to minimize the expected number of jobs in the network, that the queue length are similar to those under the diagonal policy, and that the minimum is quite flat.

In Sect. 7 we contrast our Push–Pull network with the famous KSRS network to which it is similar. We study two types of policies which were suggested for the KSRS network: The affine switching curve polity of Henderson, Meyn, and Tadic [20] and the max pressure policy of Dai and Lin [7]. By comparing the performance of KSRS under these two types of policies to that of the Push–Pull network under fixed threshold and under an adapted max pressure policy, we gain important insights.

A topic which we do not consider in this paper is the behavior of the output processes, $D_1(t)$ and $D_2(t)$, the numbers of jobs of type 1 and type 2 which leave the network in the time interval $(0, t]$. These are studied in our recent paper [27], in which we consider Push–Pull networks with general independent processing times. We use fluid scale analysis and diffusion scale analysis, and we show that for any fully utilizing stable policies, the diffusion scaled output processes converge to a Brownian motion that is independent of the policy, with negative correlation between $D_1(t)$ and $D_2(t)$.

2 Preliminary examination of the network

The state of the network at time t is given by the values n_1, n_2 of $Q_1(t), Q_2(t)$. Figure 3 describes the states and the possible transition rates. These depend on the choice of push or pull operation at each server and are as follows:

$$n_i \to n_i + 1 \quad \text{at rate } \lambda_i \text{ if server } i \text{ chooses push operation,} \tag{2}$$

$$n_i \to n_i - 1 \quad \text{at rate } \mu_i \text{ if server } \bar{\imath} \text{ chooses pull operation, } n_i > 0. \tag{3}$$

Hence: whenever $n_2 > 0$, server 1 can choose to serve the queue Q_2 and move "down" at rate μ_2, or choose to perform push operation on a new job and move "right" at rate λ_1. Similarly, whenever $n_1 > 0$, server 2 can choose to serve the queue Q_1 and move "left" at rate μ_1, or choose to perform push operation on a new job and move "up" at rate λ_2. When $n_2 = 0$, server 1 must choose push operation and move "right" at rate λ_1, and when $n_1 = 0$, server 2 must choose push operation and move "up" at rate λ_2.

We consider now the behavior of this network under the policy of preemptive priority to pull operations over push operations. Once the policy is specified, $(Q_1(t), Q_2(t))$ is a continuous-time discrete-state Markov process (Markov jump process). Furthermore, all the states in $\{(n_1, n_2) : n_1 > 0, n_2 > 0\}$ will be visited at most once, and the network will move to a state in $\{(n_1, 0) : n_1 > 0\} \cup \{(0, n_2) : n_2 > 0\} \cup \{(0, 0)\}$ and stay in that set thereafter.

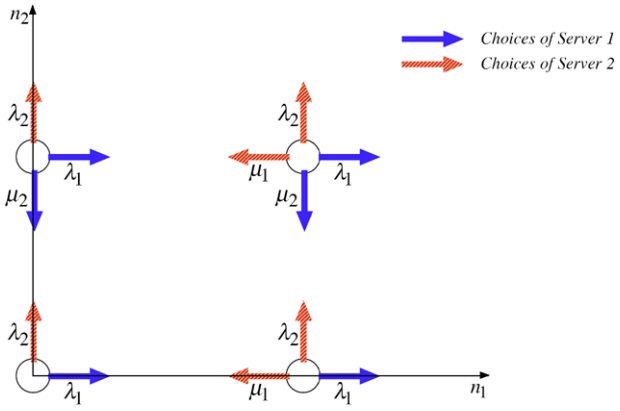

Fig. 3 State of the Push–Pull network and possible transition rates

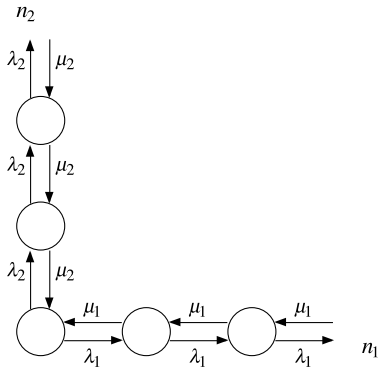

Fig. 4 Transition rates for pull priority policy

Figure 4 describes the transition rates for pull priority. The behavior of $(Q_1(t), Q_2(t))$ under pull priority policy is summarized by:

Theorem 1 *Under pull priority policy, the network will stay in states with* $Q_1(t) > 0$, $Q_2(t) > 0$ *only for a finite time (almost surely).*

Thereafter, if the network is inherently stable, it will alternate randomly between periods of $Q_1(t) > 0$ *and* $Q_2(t) > 0$. *In a period in which* $Q_i(t) > 0$ *it will behave like an M/M/1 queue with arrival rates* λ_i *and service rates* μ_i. *The steady-state probabilities of the network will be*

$$\lim_{t\to\infty} \mathbb{P}\big(Q_i(t) = n_i,\, Q_{\bar{i}}(t) = 0\big) = \frac{(\mu_1 - \lambda_1)(\mu_2 - \lambda_2)}{\mu_1\mu_2 - \lambda_1\lambda_2}\left(\frac{\lambda_i}{\mu_i}\right)^{n_i}, \quad n_i \geq 0. \tag{4}$$

If the network is inherently unstable, then either Q_1 *or* Q_2 *will diverge*:

$$\mathbb{P}\left[\lim_{t\to\infty}\left(\frac{Q_i(t)}{t}, Q_{\bar{i}}(t)\right) = \left(\frac{\lambda_i}{\mu_i} - 1, 0\right)\right] = \frac{\lambda_i - \mu_i}{\lambda_1 + \lambda_2 - \mu_1 - \mu_2}, \quad i = 1, 2. \tag{5}$$

In the null case of $\lambda_i = \mu_i$, $i = 1, 2$, *all the states* $(n_1, 0)$ *and* $(0, n_2)$ *will be null recurrent.*

Proof Clearly all states with $n_1 \cdot n_2 > 0$ are visited no more than once.

The steady-state probabilities for the inherently stable case are obtained immediately from the balance equations (birth and death process).

In the inherently unstable case, the state $(0,0)$ will be visited only a finite, random, geometrically distributed number of times. After each visit to state $(0,0)$, there are three possible events: E_0: The state $(0,0)$ will be visited again, E_1: the process will remain forever in $(Q_1(t) > 0, 0)$, or E_2: the process will remain forever in $(0, Q_2(t) > 0)$. The probability that the process will return to state $(0,0)$ from state $(1,0)$ is $\frac{\mu_1}{\lambda_1}$, see Durrett [10]. Hence,

$$\mathbb{P}(E_1) = \frac{\lambda_1}{\lambda_1 + \lambda_2}\left(1 - \frac{\mu_1}{\lambda_1}\right).$$

Once the process remains forever in states $(Q_1(t) > 0, 0)$, we have $Q_1(t)/t \to \frac{\lambda_1 - \mu_1}{\mu_1}$ almost surely, with a similar result for the event E_2, and (5) follows.

If $\lambda_i = \mu_i$, $i = 1, 2$, then $(0,0)$ is null recurrent as it is for the M/M/1 queue. □

It is straightforward to derive moments of the queue lengths in steady state, in particular, one obtains

$$\lim_{t\to\infty} \mathbb{E}\big(Q_i(t)\big) = \frac{\mu_i \lambda_i}{\mu_1\mu_2 - \lambda_1\lambda_2} \frac{\mu_{\bar{\imath}} - \lambda_{\bar{\imath}}}{\mu_i - \lambda_i}. \tag{6}$$

We now discuss intuitively how we may stabilize the inherently unstable case. To do so we will introduce safety stocks: Server i will stop pushing and give priority to pull only if the queue of type i items at server $\bar{\imath}$ has reached a threshold (safety stock) level s_i. Figure 5 illustrates the transition rates for such a fixed threshold policy.

To see how the safety stocks work and to decide on the required threshold levels, we consider the following Markov jump process, based on a simple random walk in the strip $X(t) = (X_1(t), X_2(t)) \in \{(n_1, n_2) : n_1 = 0, \pm 1, \pm 2, \ldots, n_2 = 0, 1, \ldots, s_2\}$. Jumps of n_1 to the right occur with rate λ_1 out of states $(n_1, 0)$, and to the left at rate μ_1 out of states (n_1, s_2). Jumps of n_2 up occur at rate λ_2 when $n_2 = 0, \ldots, s_1 - 1$ and jumps of n_2 down occur at rate μ_2 when $n_2 = 1, \ldots, s_2$. The vertical coordinate will behave like a finite buffer M/M/1/s_2 queue, independently of the horizontal coordinate, with steady-state probabilities $\pi_{\cdot,0} = \lim_{t\to\infty} \mathbb{P}(X_2(t) = 0) = \frac{1 - \frac{\lambda_2}{\mu_2}}{1 - (\frac{\lambda_2}{\mu_2})^{s_2+1}}$ and $\lim_{t\to\infty} \mathbb{P}(X_2(t) = s_2) = \pi_{\cdot,0}(\frac{\lambda_2}{\mu_2})^{s_2}$. Hence, in steady state the rate of horizontal moves to the right will be $\lambda_1\pi_{\cdot,0}$, and to the left it will be $\mu_1(\frac{\lambda_2}{\mu_2})^{s_2}\pi_{\cdot,0}$. Hence this Markov chain will drift to the right if $\frac{\lambda_1}{\mu_1}(\frac{\lambda_2}{\mu_2})^{s_2} > 1$, and it will drift to the left if $\frac{\lambda_1}{\mu_1}(\frac{\lambda_2}{\mu_2})^{s_2} < 1$.

This indicates that the Push–Pull network will be stable if we choose safety stock threshold levels s_i such that $\frac{\lambda_{\bar{\imath}}}{\mu_{\bar{\imath}}}(\frac{\lambda_i}{\mu_i})^{s_i} < 1$, $i = 1, 2$. We study this fixed threshold policy in the next section.

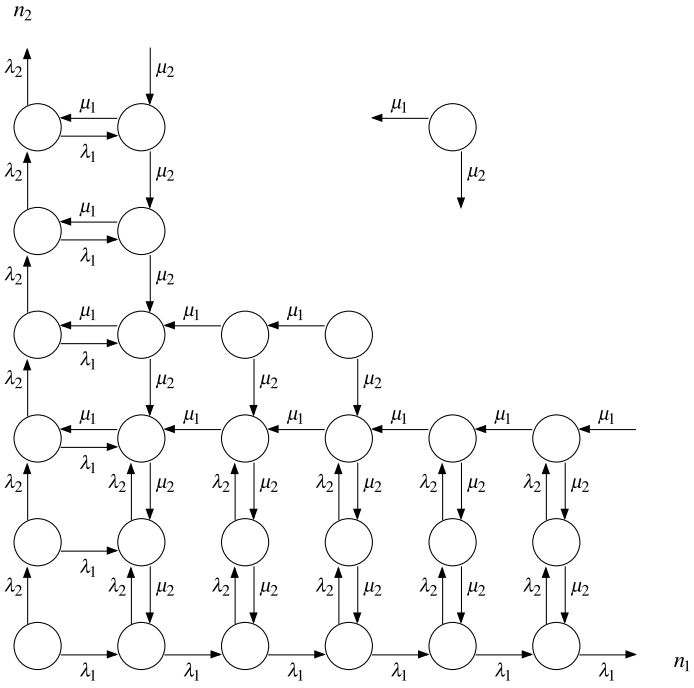

Fig. 5 Transition rates for a fixed threshold policy, $s_1 = 1, s_2 = 2$

The case of $\lambda_i = \mu_i$

When $\lambda_i = \mu_i$, $i = 1, 2$, we saw in Theorem 1 that under pull priority the states $(m, 0)$, and $(0, n)$ form a null recurrent irreducible set and all other states are transient, in other words, the network is null recurrent. We note that (1) does not, in this case, determine ν_i, θ_i, $i = 1, 2$, and in fact we can allocate an arbitrary proportion of the time of each server to type 1 jobs, with the remaining proportion of the time to type 2 jobs, and keep both servers fully occupied. We conjecture that no stationary policy can make the network positive recurrent.

3 Fixed threshold policies

In this section we analyze the fixed threshold policy: This means that server 1 will not start serving the queue Q_2 unless Q_1 has at least s_1 jobs, with a similar rule for server 2 (see Fig. 5). We calculate the steady-state probabilities under this policy. The choice of s_1, s_2 is such that $\frac{\lambda_{\bar{i}}}{\mu_{\bar{i}}}(\frac{\mu_i}{\lambda_i})^{s_i} < 1$, $i = 1, 2$. In this section and in Sects. 4, 5, for easier readability, we shall use m, n rather than n_1, n_2 to denote the number of jobs in Q_1, Q_2.

Theorem 2 *The steady-state distribution of the Markov jump process* $(Q_1(t), Q_2(t))$ *for the Push–Pull network with* $\lambda_i > \mu_i$, *under the fixed threshold policy, is given by*

$$P_{m,n} = \lim_{t\to\infty} \mathbb{P}\big[\big(Q_1(t), Q_2(t)\big) = (m,n)\big]$$
$$= \begin{cases} P_{s_1,s_2} \dfrac{\left(\frac{\lambda_2}{\mu_2}\right)^n + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^n - 1\right)}{\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)}, \\ \quad m = s_1, 0 \le n \le s_2, \\ P_{s_1,s_2} \dfrac{\left[\frac{\lambda_1}{\mu_1} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)\right]^{m-s_1-1}}{\left[\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)\right]^{m-s_1+1}} \\ \quad \times \left\{\frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - \frac{\lambda_1}{\mu_1}\right) + \frac{\lambda_1}{\mu_1}\frac{\lambda_1+\lambda_2-\mu_1-\mu_2}{\lambda_2-\mu_2}\left(\frac{\lambda_2}{\mu_2}\right)^n\right\}, \\ \quad m > s_1, 0 \le n \le s_2, \end{cases} \tag{7}$$

with analogous expressions for $n \ge s_2$ *and* $0 \le m \le s_1$. *The expression for* P_{s_1,s_2} *is*

$$P_{s_1,s_2} = \left[\frac{\lambda_1+\mu_1}{2(\lambda_1-\mu_1)} + \frac{\frac{\lambda_2}{\mu_2}\frac{\lambda_1}{\lambda_1-\mu_1} - \frac{\mu_1}{\lambda_1-\mu_1}}{\left(\frac{\lambda_1}{\mu_1}\right)^{s_1} - \frac{\lambda_2}{\mu_2}} + \frac{\lambda_2+\mu_2}{2(\lambda_2-\mu_2)} + \frac{\frac{\lambda_1}{\mu_1}\frac{\lambda_2}{\lambda_2-\mu_2} - \frac{\mu_2}{\lambda_2-\mu_2}}{\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - \frac{\lambda_1}{\mu_1}}\right]^{-1}. \tag{8}$$

All the remaining states are transient.

Proof Observation of Fig. 5 clearly shows that all the states $\{(m,n) : m < s_1, n < s_2\}$ and $\{(m,n) : m > s_1, n > s_2\}$ are visited at most once and are transient, and that the set of states $\{(m,n) : m \ge s_1, 0 \le n \le s_2\} \cup \{(m,n) : n \ge s_2, 0 \le m \le s_1\}$ is reached after a finite number of steps.

Take $m \ge s_1$ and consider the balance equations around the set of states $(m,0), \ldots, (m,n)$, where $0 \le n < s_2$ (see Fig. 6). We get the balance equations

$$\mu_2 P_{m,n+1} + \lambda_1 P_{m-1,0} = \lambda_2 P_{m,n} + \lambda_1 P_{m,0}, \tag{9}$$

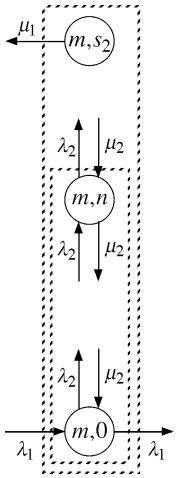

Fig. 6 Balance equations for selected sets of states, under fixed threshold policy

and from this we obtain by induction the recursion relation

$$P_{m,n} = \left(\frac{\lambda_2}{\mu_2}\right)^n P_{m,0} - \frac{\lambda_1}{\lambda_2 - \mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^n - 1\right)(P_{m-1,0} - P_{m,0}),$$
$$0 \le n \le s_2. \tag{10}$$

Since $P_{s_1-1,0} = 0$, we can use (10) to express $P_{s_1,n}$, $0 < n \le s_2$, in terms of $P_{s_1,0}$. We then express $P_{s_1,0}$ in terms of P_{s_1,s_2} and obtain the desired expression for $P_{s_1,n}$, $0 \le n \le s_2$, in (7).

Next we consider $m > s_1$. Observing transitions between $Q_1(t) = m - 1$ and $Q_1(t) = m$ (see Fig. 6), we get the balance equations

$$\mu_1 P_{m,s_2} = \lambda_1 P_{m-1,0}, \quad m > s_1. \tag{11}$$

We substitute this into (10) for $n = s_2$ and obtain a recursion for $P_{m,0}$ in terms of $P_{m-1,0}$,

$$P_{m,0} = P_{m-1,0}\frac{\frac{\lambda_1}{\mu_1} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)}{\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)}, \quad m > s_1. \tag{12}$$

Iterating (12) and expressing $P_{s_1,0}$ in terms of P_{s_1,s_2} we get

$$P_{m,0} = P_{s_1,s_2}\frac{\left[\frac{\lambda_1}{\mu_1} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)\right]^{m-s_1}}{\left[\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)\right]^{m-s_1+1}}. \tag{13}$$

Also, by (12) and (13),

$$P_{m-1,0} - P_{m,0} = P_{s_1,s_2}\frac{\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - \frac{\lambda_1}{\mu_1}\right)\left[\frac{\lambda_1}{\mu_1} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)\right]^{m-s_1-1}}{\left[\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)\right]^{m-s_1+1}}. \tag{14}$$

Finally, substituting (13) and (14) into (10), we obtain (7) for $m > s_1$.

To show the convergence, we consider a fixed $0 \le n \le s_2$ and sum $P_{m,n}$ over $s_1 < m < \infty$. We have this convergence because

$$R_1 = \frac{\frac{\lambda_1}{\mu_1} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)}{\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} + \frac{\lambda_1}{\lambda_2-\mu_2}\left(\left(\frac{\lambda_2}{\mu_2}\right)^{s_2} - 1\right)} < 1, \tag{15}$$

which holds because, by the definition of s_2, $(\frac{\lambda_2}{\mu_2})^{s_2} > \frac{\lambda_1}{\mu_1}$.

The expression for P_{s_1,s_2} follows by normalizing the sum to 1. □

Note

An alternative method to prove Theorem 2 is to consider the Markov jump process as a quasi-birth-and-death process. This was pointed out to us by Moshe Haviv. A similar model is analyzed using these techniques by Haviv and Zlotnikov [19].

Geometric decay of steady-state probabilities

We note that the steady-state probabilities $P_{m,n}$ decay geometrically like R_1^m for all $m > s_1$.

Example

We consider a symmetric Push–Pull network with $\lambda_1 = \lambda_2 = \lambda$, $\mu_1 = \mu_2 = \mu$, and let $r = \mu/\lambda < 1$. For positive recurrence, we take the fixed thresholds $s_1 = s_2 = 2$. We then obtain the steady-state probabilities

$$P_{2,2} = \frac{1-r}{(1+r)(1+2r)},$$

$$P_{2,1} = P_{1,2} = \frac{1-r}{(1+r)(1+2r)}\frac{2r}{2+r},$$

$$P_{2,0} = P_{0,2} = \frac{1-r}{(1+r)(1+2r)}\frac{r^2}{2+r},$$

$$P_{n,2} = P_{2,n} = \frac{1-r}{(1+r)(1+2r)}\frac{r}{2+r}\left(\frac{1+2r}{2+r}\right)^{n-3}, \quad n \geq 3,$$

$$P_{n,1} = P_{1,n} = \frac{1-r}{(1+r)(1+2r)}\frac{3r^2}{(2+r)^2}\left(\frac{1+2r}{2+r}\right)^{n-3}, \quad n \geq 3,$$

$$P_{n,0} = P_{0,n} = \frac{1-r}{(1+r)}\frac{r^2}{(2+r)^2}\left(\frac{1+2r}{2+r}\right)^{n-3}, \quad n \geq 3.$$

The expected return time to state $(2,2)$ is $T_{(2,2)} = \frac{1}{2\mu}\frac{(1+r)(1+2r)}{1-r}$. Each time the network returns to this state, it will randomly move to the horizontal or the vertical strip of states: In the horizontal strip $Q_1(t) \geq Q_2(t)$, while in the vertical strip $Q_2(t) \geq Q_1(t)$. We split the time spent in the state $(2,2)$ equally between the two strips. Hence the network spends an expected time $2T_{(2,2)} = \frac{(1+r)(1+2r)}{\mu(1-r)}$ in each strip before moving to the other strip. Consider the horizontal strip. Let

$$F_2 = P_{2,2} + 2\sum_{n=3}^{\infty} P_{n,2} = \frac{1+r}{(1+r)(1+2r)},$$

$$F_1 = 2\sum_{n=3}^{\infty} P_{n,1} = \frac{2r}{(1+r)(1+2r)},$$

$$F_2 = 2\sum_{n=3}^{\infty} P_{n,0} = \frac{2r^2}{(1+r)(1+2r)}.$$

These probabilities which add up to 1 express the fraction of time that $Q_2(t) = 2$, 1, or 0, respectively, in the strip of states $Q_1 \geq 2$ (the horizontal strip in Fig. 5). Within

this period, when $Q_2(t) = 2$, both queues are served at rate μ. When $Q_2(t) = 1$, only the queue of jobs of type 2 is served at rate μ. When $Q_2 = 0$, both servers are pushing, and there is no service to either queue.

Hence in the horizontal strip the total average service rates of the two queues are:

$$\text{Rate of } Q_1(t) \text{ service: } F_2\,\mu = \mu \frac{1}{1+r}\frac{1+r}{1+2r} = \nu \frac{1+r}{1+2r}$$

$$\text{Rate of } Q_2(t) \text{ service: } (F_2 + F_1)\mu = \mu \frac{1}{1+r}\frac{1+3r}{1+2r} = \nu \frac{1+3r}{1+2r}$$

In summary: For this example, each queue undergoes fluctuations in service rate around the average service rate of ν, where for alternating periods of expected duration $2T_{(2,2)}$, the service rates alternate between $\nu\frac{1+r}{1+2r}$ and $\nu\frac{1+3r}{1+2r}$.

4 Queue balancing diagonal policy

The fixed threshold policy has the following features:

(i) At all times $Q_1(t) \geq s_1$, $Q_2(t) \geq s_2$.
(ii) The geometric decay of the probabilities $\mathbb{P}(Q_1(t) > m)$ is at rate R_1, which is faster as s_2 increases; similarly for $Q_2(t)$.

These features seem conflicting; we would like s_1, s_2 to be small because of (i) and to be large because of (ii). Further features are:

(iii) The fixed threshold policy causes the difference $|Q_1(t) - Q_2(t)|$ to be large.
(iv) As indicated in the example, the fixed threshold policy introduces large fluctuations in the service rates, over cycles of expected durations $\frac{\mu_1+\mu_2}{\mu_1\mu_2}\frac{1}{P_{s_1,s_2}}$.

We now introduce another stabilizing policy which might counter these possibly undesirable features. We consider only the symmetric inherently unstable Push–Pull network with $\lambda = \lambda_1 = \lambda_2 > \mu = \mu_1 = \mu_2$ and suggest a policy which attempts to balance the queues $Q_1(t), Q_2(t)$. Our queue balancing diagonal policy serves the shortest queue, with one server pushing work into it, and the other pulling work out of it, until the difference between the queues decreases to just one customer. When the queue lengths differ by 1 or are equal both servers pull. We calculate the steady-state probabilities under this policy.

Theorem 3 *The steady-state distribution of the Markov jump process $Q_1(t)$, $Q_2(t)$ for the symmetric, inherently unstable Push–Pull network, under the queue balancing diagonal policy, is*

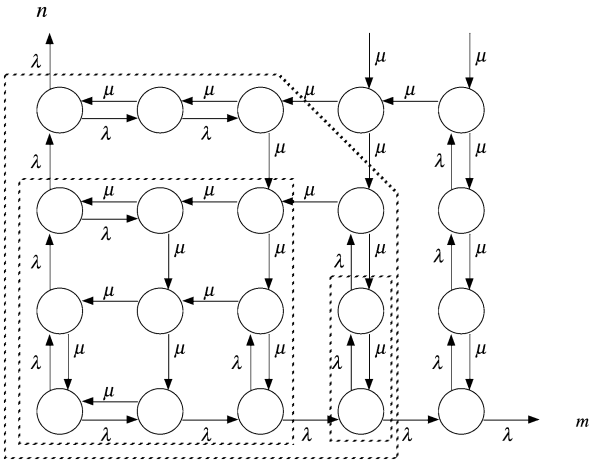

Fig. 7 Transition rates and equation contours for the queue balancing diagonal policy

$$P_{m,n} = \lim_{t\to\infty} \mathbb{P}\big(Q_1(t) = m,\, Q_2(t) = n\big)$$

$$= \begin{cases} P_{0,0} \prod_{i=0}^{m-1} \dfrac{\frac{\lambda}{\mu} + \frac{\lambda}{\lambda-\mu}\left(\left(\frac{\lambda}{\mu}\right)^i - 1\right)}{\left(\frac{\lambda}{\mu}\right)^i + \frac{\lambda}{\lambda-\mu}\left(\left(\frac{\lambda}{\mu}\right)^i - 1\right)}, & m > 0,\ n = 0, \\ P_{m,0} \dfrac{\frac{\lambda}{\lambda-\mu}\left(\frac{\lambda}{\mu}\right)^{m-1} + 2\left(\frac{\lambda}{\mu}\right)^{n+1} - \frac{\lambda}{\mu}\frac{\lambda}{\lambda-\mu}}{\frac{\lambda}{\mu} + \frac{\lambda}{\lambda-\mu}\left(\left(\frac{\lambda}{\mu}\right)^{m-1} - 1\right)}, & m > n > 0, \\ P_{m,0}\dfrac{\lambda}{\mu}, & m = n > 0, \end{cases} \tag{16}$$

where $P_{0,0}$ *normalizes the sum to* 1.

Proof The transition rates for the queue balancing diagonal policy are displayed in Fig. 7, which also shows contours across which we obtain balance equations.

From the transitions in and out of $Q_1(t), Q_2(t) \le m-1$ (see square contour in Fig. 7) and the obvious symmetry $P_{m,n} = P_{n,m}$ we obtain the balance equation

$$P_{m,m-1} = \frac{\lambda}{\mu} P_{m-1,0}, \quad m \ge 1. \tag{17}$$

Equation (10) is valid here as well for $m \ge 2$, $0 \le n \le m-1$, and we use it for $(m, m-1)$. Substituting (17) into (10), we obtain the recursion

$$P_{m,0} = P_{m-1,0} \frac{\frac{\lambda}{\mu} + \frac{\lambda}{\lambda-\mu}\left(\left(\frac{\lambda}{\mu}\right)^{m-1} - 1\right)}{\left(\frac{\lambda}{\mu}\right)^{m-1} + \frac{\lambda}{\lambda-\mu}\left(\left(\frac{\lambda}{\mu}\right)^{m-1} - 1\right)}, \quad m \ge 1. \tag{18}$$

We iterate (18) to obtain the desired expression for $P_{m,0}$.

Substituting the expression for $P_{m,0}$ and for $P_{m-1,0}$ into (10), we then obtain the desired expression for $P_{m,n}$, $m > n > 0$.

Finally, we use the transitions in and out of the set $Q_1(t), Q_2(t) \le m$ excluding $(Q_1(t), Q_2(t)) = (m, m)$ (see large contour in Fig. 7) to obtain that $P_{m,m} = \frac{\lambda}{\mu} P_{m,0}$, which completes the proof of (16).

We now show the convergence. Denote

$$R(m) = \frac{\frac{\lambda}{\mu} + \frac{\lambda}{\lambda-\mu}\left(\left(\frac{\lambda}{\mu}\right)^m - 1\right)}{\left(\frac{\lambda}{\mu}\right)^m + \frac{\lambda}{\lambda-\mu}\left(\left(\frac{\lambda}{\mu}\right)^m - 1\right)}.$$

Then $R(0) = \frac{\lambda}{\mu} > 1$, $R(1) = 1$, and it is easily seen that $R(m)$ is strictly decreasing in m and that $\lim_{m\to\infty} R(m) = \frac{\lambda}{2\lambda-\mu}$.

It is seen from (16, 17) that $P_{m,m} = P_{m,m-1}R(m-1)$ and also that $P(m,n)$ is increasing in n for $n = 0, 1, \ldots, m-1$. We therefore have that

$$P_{m,n} < P_{m,m-1} = \frac{\lambda}{\mu} P_{0,0} \prod_{i=0}^{m-2} R(i), \quad 0 \le n < m-1 \text{ and } 2 \le n = m.$$

Hence,

$$\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} P_{m,n} < P_{0,0}\left[1 + \left(\frac{\lambda}{\mu}\right)^2 + 2\frac{\lambda}{\mu}\sum_{m=1}^{\infty}(m+1)\prod_{i=0}^{m-2} R(i)\right] < \infty. \qquad \square$$

Geometric decay of steady-state probabilities

It is seen from the proof of convergence above that for any fixed n and large $m > n$, $P_{m,n}$ decrease as m increases at a rate which is arbitrarily close to $\frac{\lambda}{2\lambda-\mu}$. The same also holds for $P_{m,m-1}$.

5 Generalized threshold policies

In this section we consider the inherently unstable Push–Pull network and define a family of policies which we call generalized threshold policies. We use the Foster–Lyapunov criterion to prove the stability of the network under these policies. We let the state of the network $Q_1(t) = m$, $Q_2(t) = n$ be denoted by (m,n). An alternative method would have been to prove the stability of the fluid model of the system, as suggested by Dai [6]. We use this simpler approach in a follow up paper to the current paper, in which processing times are general independent rather than exponential [27]. The advantage of the explicit Foster–Lyapunov treatment here is that it also reveals that the process is exponentially ergodic.

Definition 1 Let s_i satisfy $\frac{\lambda_i}{\mu_i}(\frac{\mu_i}{\lambda_i})^{s_i} < 1$, $i = 1, 2$. Let $a \ge s_1$, $b \ge s_2$. Let N_m, $m > a$, and M_n, $n > b$, be nondecreasing integer functions such that $s_2 \le N_m < \frac{b}{a}m$ and $s_1 \le M_n < \frac{a}{b}n$. Define the actions of a policy when $Q_1(t) = m$, $Q_2(t) = n$ as follows.

In states $m \le a$, $n \le b$ use an arbitrary stationary policy.

In states when $m = 0$, $n > b$ and when $m > a$, $0 \le n < N_m$ server 2 will push. In all the remaining states server 2 will pull.

In states when $n = 0$, $m > a$ and when $n > b$, $0 \le m < M_n$ server 1 will push. In all the remaining states server 1 will pull.

Then this policy will be called a generalized threshold policy.

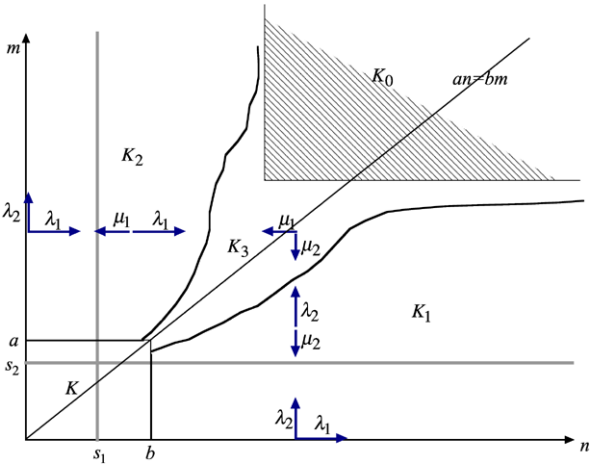

Fig. 8 Directions of transitions for a generalized threshold policy

Generalized threshold policies are described schematically in Fig. 8. We will elaborate on the notation in this figure in the sequel. Our main result in this section is:

Theorem 4 *The Push–Pull network with $\lambda_i > \mu_i$, $i = 1, 2$, is stable under generalized threshold policies, in the sense that it has a single irreducible set of positive recurrent states, and all other states are transient.*

We define the following subsets of states:

$$K = \{(m,n)\colon 0 \le m \le a \text{ and } 0 \le n \le b\},$$

$$\bar{K} = \{(m,n)\colon (m,n) \notin K\},$$

$$K_0 = \{(m,n)\colon m > \sup_{n>b} M_n \text{ and } n > \sup_{m>a} N_m\},$$

$$K_1 = \{(m,n)\colon m > a \text{ and } 0 \le n \le N_m\},$$

$$K_2 = \{(m,n)\colon n > b \text{ and } 0 \le m \le M_n\},$$

$$K_3 = \bar{K} \setminus (K_0 \cup K_1 \cup K_2).$$

The following lemma identifies the single irreducible set of states.

Lemma 1 *There exists a maximal nonempty subset $K^* \subseteq K$ such that all the states in*

$$S = K^* \cup K_1 \cup K_2 \cup K_3$$

communicate.

All the remaining states consisting of $K_0 \cup (K \setminus K^*)$ are transient.

Proof Note that each of the sets K_0, K_3, and $K\backslash K^*$ may be empty. The proof follows from the following steps:

(a) All states in K_0 can be visited at most once, hence they are transient.
(b) All states in K_1 communicate, all states in K_2 communicate.
(c) Every state in K_3 can be reached either from K_1 or from K_2.
(d) From every state in K_3 one can reach either K_1 or K_2.
(e) There are states in K which can be reached from K_1, and there are states in K which can be reached from K_2.
(f) From every state in K it is possible to reach both K_1 and K_2.

From (b, e, f) it is clear that K^* is not empty and that $K^* \cup K_1 \cup K_2$ communicate. From (c, d) they also communicate with K_3. The proof follows. □

We need to show that S is a positive recurrent set. We note first:

Proposition 1 *Let $Z(r)$, $r = 1, 2, \ldots$, be the discrete-time embedded Markov chain of states $(m, n) \in S$ visited by the Markov jump process $Q_1(t), Q_2(t)$. Then the Markov jump process $Q_1(t), Q_2(t)$ is positive recurrent on S if and only if the discrete-time Markov chain $Z(r)$ is positive recurrent on S.*

Proof This follows because all the transition rates out of states in $Q_1(t), Q_2(t)$ are bounded from below by $\min(\lambda_1, \lambda_2, \mu_1, \mu_2)$ and from above by $\lambda_1 + \lambda_2 + \mu_1 + \mu_2$. □

We will show that S is indeed a positive recurrent set for the Markov chain $Z(r)$. We will use the following version of the Foster–Lyapunov criterion, see [3, 11–13].

Theorem 5 (Foster–Lyapunov criterion) *Let $Z(r)$ be a discrete-time irreducible Markov chain on a discrete state space S. Let $\mathcal{K}$ be a finite subset of S. Let $V(x)$, $g(x)$ be two nonnegative functions of $x \in S$, with g bounded. Assume that the following two conditions hold*:

(i) *There exists $h > 0$ such that for all $x \in S\backslash\mathcal{K}$,*

$$\mathbb{E}\big\{V\big(Z\big(g(x)\big)\big) - V(x) | Z(0) = x\big\} < -h;$$

(ii) *there exists B such that for all $x \in \mathcal{K}$,*

$$\mathbb{E}\big(V\big(Z(1)\big) | Z(0) = x\big) < B.$$

Then the Markov chain is positive recurrent.

Proposition 2 *For the Push–Pull network, condition* (ii) *of the Foster–Lyapunov criterion holds for every finite set $\mathcal{K}$.*

Proof This follows because $|Q_i(t+1) - Q_i(t)| \le 1$, $i = 1, 2$ (as it is in every queueing network). □

We let $\mathcal{K} = \{(m, n) : m \le A, n \le B\}$ where $A \ge a$, $B \ge b$, $\frac{B}{A} = \frac{b}{a}$.

Our Lyapunov function $V(m,n)$ is a linear combination of the following three functions:

$$\begin{aligned} U_1(m,n) &= \max(bm, an), \\ U_2(m,n) &= |bm - an|, \\ U_3(m,n) &= \begin{cases} \eta^{K-n} - 1, & m > A,\ 0 \le n \le K, \\ \theta^{L-m} - 1, & n > B,\ 0 \le m \le L, \\ 0, & \text{otherwise}, \end{cases} \end{aligned} \tag{19}$$

where K, L are integers such that $s_2 \le K \le B$, $s_1 \le L \le A$.

We define the Lyapunov function as

$$V(m,n) = \begin{cases} U_1(m,n) + cU_2(m,n) + dU_3(m,n) & (m,n) \notin \mathcal{K}, n \le \frac{b}{a}m, \\ U_1(m,n) + eU_2(m,n) + fU_3(m,n) & (m,n) \notin \mathcal{K}, m \le \frac{a}{b}n, \\ 0, & (m,n) \in \mathcal{K}. \end{cases} \tag{20}$$

Note that $V(x, y)$ is continuous for real x, y outside the rectangle $[0, A] \times [0, B]$ for all values of c, d, e, f.

The number of steps function is

$$g(m,n) = \begin{cases} 2, & b > bm - an > -a, \\ 1, & \text{otherwise}. \end{cases} \tag{21}$$

Proposition 3 *There exist $A, B, K, L, \eta, \theta, c, d, e, f$ such that for all (m,n) outside $\mathcal{K}$,*

$$\mathbb{E}\{V(Z(g(m,n))) - V(m,n) | Z(0) = (m,n)\} < -h < 0.$$

Proof We shall perform the calculations for $n < \frac{b}{a}m$ (i.e., under the diagonal line $mb = na$) and determine A, K, η, c, d only. The calculations above the diagonal are analogous.

We distinguish between N_m bounded and N_m unbounded. When N_m is bounded, since N_m is nondecreasing and integer, there exists A such that for all $m > A$, the value of N_m is equal to the upper bound K. When N_m is unbounded, we can choose the integer K as large as we need, and for any value of K, there will be A such that $N_m > K$ for all $m > A$.

For a function $f(m,n)$, we use the notations $\Delta_m f(m,n) = f(m,n) - f(m-1,n)$, $\Delta_n f(m,n) = f(m,n) - f(m,n-1)$, and $\Delta E f(m,n) = \mathbb{E}(f(Z(1)) - f(m,n) | Z(0) = (m,n))$

Under the diagonal we have

$$\begin{aligned} U_1(m,n) &= bm, \qquad U_2(m,n) = bm - an, \\ U_3(m,n) &= \begin{cases} \eta^{K-n} - 1, & 0 \le n \le K, \\ 0, & \text{otherwise}, \end{cases} \end{aligned} \tag{22}$$

with increments

$$\begin{aligned}
&\Delta_m U_1(m,n) = b, \qquad \Delta_m U_2(m,n) = b, \qquad \Delta_m U_3(m,n) = 0,\\
&\Delta_n U_1(m,n) = 0, \qquad \Delta_n U_2(m,n) = -a,\\
&\Delta_n U_3(m,n) = \begin{cases} -\eta^{K-n}(\eta-1), & 0 < n \le K,\\ 0, & \text{otherwise.}\end{cases}
\end{aligned} \tag{23}$$

The bounded case

We have $N_m = K$, $m > A$. We will not use U_2, so we take $c = 0$.

For $m > A$ and $K < n \le \frac{b}{a}(m-1)$, we get

$$\Delta EV(m,n) = -\frac{\mu_1}{\mu_1+\mu_2} b. \tag{24}$$

This is always < 0.

For $m > A$ and $n = K$, we get

$$\Delta EV(m,n) = -\frac{\mu_1}{\mu_1+\mu_2} b + d\frac{\mu_2}{\mu_1+\mu_2}(\eta-1). \tag{25}$$

This is < 0 if

$$d(\eta-1) < \frac{\mu_1}{\mu_2} b. \tag{26}$$

For $m > A$ and $0 < n < K$, we get

$$\Delta EV(m,n) = -d\frac{\mu_2}{\lambda_2+\mu_2}\eta^{K-n-1}(\eta-1)\left(\frac{\lambda_2}{\mu_2} - \eta\right), \tag{27}$$

$$\le -d\frac{\mu_2}{\lambda_2+\mu_2}(\eta-1)\left(\frac{\lambda_2}{\mu_2} - \eta\right) < 0 \tag{28}$$

if we choose

$$1 < \eta < \frac{\lambda_2}{\mu_2}. \tag{29}$$

For $m > A$ and $n = 0$, we get

$$\Delta EV(m,n) = \frac{\lambda_1}{\lambda_1+\lambda_2} b - d\frac{\lambda_2}{\lambda_1+\lambda_2}\eta^{K-1}(\eta-1). \tag{30}$$

This is < 0 if

$$d(\eta-1) > \frac{\lambda_1}{\lambda_2} b\eta^{-(K-1)}. \tag{31}$$

Recall that $K \ge s_2$ and $1 > \frac{\lambda_1}{\mu_1}(\frac{\mu_2}{\lambda_2})^{s_2}$. Hence we can choose a value of η so that

$$1 < \left(\frac{\lambda_1}{\mu_1}\right)^{1/K} < \eta < \frac{\lambda_2}{\mu_2}, \tag{32}$$

and with this value we will have

$$\frac{\lambda_1}{\lambda_2} b\eta^{-(K-1)} < \frac{\mu_1}{\mu_2} b.$$

We can then choose d so that $d(\eta - 1)$ is strictly between the two values. With this d and η, (26, 29, 31) hold, and $\Delta EV(m,n) < -h < 0$.

The unbounded case

We have $N_m > K$, $m > A$. We will use all three functions U_1, U_2, U_3.

For $m > A$ and $N_m \le n \le \frac{b}{a}(m-1)$, we get

$$\Delta EV(m,n) = -\frac{\mu_1}{\mu_1+\mu_2} b(1+c) + \frac{\mu_2}{\mu_1+\mu_2} ac, \tag{33}$$

which is < 0 if

$$\frac{c}{1+c} < \frac{\mu_1}{\mu_2}\frac{b}{a}. \tag{34}$$

For $m > A$ and $K < n < N_m$, we get

$$\Delta EV(m,n) = -\frac{\lambda_2 - \mu_2}{\lambda_2 + \mu_2} ca, \tag{35}$$

which is always negative.

For $m > A$ and $n = K$, we get

$$\Delta EV(m,n) = -\frac{\lambda_2 - \mu_2}{\lambda_2 + \mu_2} ca + d\frac{\mu_2}{\lambda_2+\mu_2}(\eta - 1). \tag{36}$$

This is < 0 if

$$ca\left(\frac{\lambda_2}{\mu_2} - 1\right) > d(\eta - 1). \tag{37}$$

For $m > A$ and $0 < n < K$, we get

$$\Delta EV(m,n) = -c\frac{\lambda_2 - \mu_2}{\lambda_2+\mu_2} a - d\frac{\mu_2}{\lambda_2+\mu_2}\eta^{K-n-1}(\eta-1)\left(\frac{\lambda_2}{\mu_2} - \eta\right), \tag{38}$$

$$\le -c\frac{\lambda_2 - \mu_2}{\lambda_2+\mu_2} a - d\frac{\mu_2}{\lambda_2+\mu_2}(\eta-1)\left(\frac{\lambda_2}{\mu_2} - \eta\right), \tag{39}$$

where the inequality holds if $1 \le \eta \le \frac{\lambda_2}{\mu_2}$. A sufficient condition for the whole expression to be < 0, for any fixed $c > 0$ and/or $d > 0$, with a value which is independent of m, n, is (29).

For $m > A$ and $n = 0$, we get

$$\Delta EV(m,n) = \frac{\lambda_1}{\lambda_1+\lambda_2}(1+c)b - ca\frac{\lambda_2}{\lambda_1+\lambda_2} - d\frac{\lambda_2}{\lambda_1+\lambda_2}\eta^{K-1}(\eta-1). \tag{40}$$

This is < 0 if

$$d(\eta - 1) > \left(\frac{\lambda_1}{\lambda_2}(1+c)b - ca\right)\eta^{-(K-1)}. \tag{41}$$

To choose $d(\eta - 1)$ so as to satisfy (37, 41), we need to have

$$ca\left(\frac{\lambda_2}{\mu_2} - 1\right) > \left(\frac{\lambda_1}{\lambda_2}(1+c)b - ca\right)\eta^{-(K-1)}. \tag{42}$$

Together with the condition (34), we need to choose

$$\frac{\mu_1}{\mu_2}\frac{b}{a} > \frac{c}{1+c} > \frac{\lambda_1 b}{\lambda_2 a}\frac{\eta^{-(K-1)}}{\frac{\lambda_2}{\mu_2} - 1 + \eta^{-(K-1)}}. \tag{43}$$

Taking any value of η such that $1 < \eta < \frac{\lambda_2}{\mu_2}$ (condition (29)), we can make K large enough so that the right-hand side of (43) is less than the left-hand side and also less than 1. Thus we can choose c so that (42) and (34). We can then choose d so that conditions (37, 41) hold. We have shown how to choose first η, then K, and a corresponding value for A, and finally values of c and d, so that conditions (29, 34, 37, 41) all hold. With these choices, we get $\Delta EV(m,n) < -h < 0$ for the unbounded case.

We can repeat the calculations for the points above the diagonal. We now need to modify the resulting values of A and B first to have $\tilde{A} = \max(A, L)$, $\tilde{B} = \max(B, K)$, so that $K \le \tilde{B}$, $L \le \tilde{A}$, and then to have $\tilde{\tilde{A}} = \max(\tilde{A}, \frac{a}{b}\tilde{B})$, $\tilde{\tilde{B}} = \max(\tilde{B}, \frac{b}{a}\tilde{A})$, so that $\frac{\tilde{\tilde{B}}}{\tilde{\tilde{A}}} = \frac{b}{a}$.

The diagonal region

To complete the proof, we need to analyze the points (m,n) close to the diagonal, $b > bm - an > -a$. Assume that a point $Z(0) = (m,n)$ is exactly on the diagonal, i.e., $bm = an$. Then, in one step,

$$\mathbb{E}\big(V\big(Z(1)\big) - V\big(Z(0)\big) \mid Z(0) = (m,n), bm = an\big) = \frac{\mu_2}{\mu_1 + \mu_2}ca + \frac{\mu_1}{\mu_1 + \mu_2}eb \ge 0.$$

Similarly, if $b > bm - an > -a$, we may get that $\mathbb{E}(V(Z(1)) - V(Z(0))) \ge 0$. This is why we need to look at the difference in two steps, and so we use $g(m,n) = 2$. We need to calculate $\mathbb{E}(V(Z(2)) - V(m,n) \mid Z(0) = (m,n))$.

We can assume that if $b > bm - an > -a$, then $n > K + 2$, $m > L + 2$ (we can increase A, B to make sure). Hence, in the area of interest, V is a combination of U_1, U_2. It is easy to check that

$$U_1(m,n) - U_1(m-1,n-1) \ge \min(a,b), \qquad U_1(m,n) - U_1(m-2,n) \ge 0,$$
$$U_1(m,n) - U_1(m,n-2) \ge 0,$$
$$U_2(m,n) - U_2(m-1,n-1) \ge -(a+b), \qquad U_2(m,n) - U_2(m-2,n) \ge -2b,$$

$$U_2(m,n) - U_2(m,n-2) \geq -2a.$$

We now consider a state (m,n) below or on the diagonal such that server 2 is using pull operation and such that $(m-1,n)$ is above the diagonal (i.e., $b > bm - an \geq 0$). Then clearly $M_n \leq m-1$, and so in state $(m-1,n)$ server 1 will pull. Hence, with probability of at least $\frac{\mu_1\mu_2}{(\mu_1+\mu_2)^2}$, we will move in two steps from (m,n) to $(m-1, n-1)$. A similar argument holds for state m,n above or on the diagonal such that server 1 is using pull operation and such that $(m,n-1)$ is below the diagonal.

Hence, in the case that a pull operation moves across the diagonal, we get, in two steps,

$$\begin{aligned}&\mathbb{E}\big(V\big(Z(2)\big) - V\big(Z(0)\big) \mid Z(0) = (m,n)\big)\\ &\quad\leq -\frac{\mu_1\mu_2}{(\mu_1+\mu_2)^2}\min(a,b) + 2\max(c,e)\max(a,b). \end{aligned} \tag{44}$$

We saw that, in the bounded case, c (or e) is 0 and that, in the unbounded case, c (or e) can be chosen to be arbitrarily small by taking K (or L) large enough in (43). Hence, we can choose K, L, c, e so that $\mathbb{E}(V(Z(2)) - V(Z(0)) \mid Z(0) = (m,n)) < -h < 0$ when $b > bm - an > -a$.

This completes the proof. □

Exponential ergodicity and geometric tails

It is seen from the of proof of Foster's criterion that the Lyapunov function $V(z)$ is uniformly Lipschitz-continuous. Furthermore, the one-step moves of the process are bounded, and so they possess finite exponential moments. Hence, the process $Q_1(t)$, $Q_2(t)$ is actually exponentially ergodic (cf. [25, Proposition A.5.7]).

This implies that the stationary probabilities of $Q_i > m$ fall off geometrically fast in m for any generalized threshold policy (cf. [24, Theorem 16.3.2]).

6 Optimization problems and numerical results

In Sects. 3–5 we have derived policies which keep the queue lengths positive recurrent while achieving full utilization of the servers. These policies are so called throughput efficient, or throughput optimal, in that they complete jobs at the highest possible rate. One can then pose the following question:

Problem 1 For a Markovian Push–Pull network with processing rates λ_i, μ_i and holding cost rates $h_i, i = 1,2$, find a policy which will keep both servers busy at all times and will minimize the long-run average expected holding costs given by $h_1 Q_1(t) + h_2 Q_2(t)$.

Since this is a Markov decision problem with countable state space and a finite number of actions and since stable policies exist, this problem has an optimal solution. Clearly this policy will achieve throughput rates $\nu_i, i = 1,2$. We can also pose a more general problem:

Problem 2 For a Markovian Push–Pull network with processing rates λ_i, μ_i, holding cost rates h_i, and utilization $\rho_i \leq 1, i = 1, 2$, find a policy which will achieve throughput rates $\rho_i \nu_i$ and will minimize the long-run average expected holding costs given by $h_1 Q_1(t) + h_2 Q_2(t)$.

We may speculate, though we have no way of verifying it, that:

Conjecture 1 *Problem 1 is solved by some generalized threshold policy as defined in Sect. 5. Problem 2 is solved by some generalized threshold policy as defined in Sect. 5, with the addition of an idling rule*: *In state* $(\bar{N}_1, 0)$ *idle server* 2. *In state* $(0, \bar{N}_2)$ *idle server* 1.

We cannot say more about the general solution of Problems 1 and 2, but whether Conjecture 1 is correct or not, we should pose the question:

Problem 3 For a Markovian Push–Pull network with processing rates λ_i, μ_i and holding cost rates $h_i, i = 1, 2$, under full utilization, find among all generalized threshold policies the optimal $a, b, N_m, m > a, M_n, n > b$, so as to minimize the long-run average expected holding costs given by $h_1 Q_1(t) + h_2 Q_2(t)$.

We do not see a way of answering this problem in general. What we do in this section is explore some threshold policies numerically. To this purpose, we calculated the expected queue lengths under fixed threshold policies (as defined in Sect. 3) for various values of the threshold and for a range of values of the parameters. We also calculated the expected queue lengths for the queue balancing diagonal policy (as defined in Sect. 4) for various parameter values. Finally we obtained via simulation the expected queue lengths under a hybrid policy. We report here the steady-state expected values of $Q_1(t)$, $Q_2(t)$, $Q_1(t) + Q_2(t)$.

In Table 1 we consider symmetric Push–Pull networks with $\lambda = \lambda_1 = \lambda_2$, $\mu = \mu_1 = \mu_2$. The expected queue lengths are then functions of the ratio μ/λ. In the top part of Table 1 we use fixed threshold policies and investigate the dependence of the expected queue length on the values of the fixed thresholds s_1, s_2. The first line (line 1) of the table reports the expected queue length for the minimum stable thresholds of $s_1 = s_2 = 2$. The optimal fixed thresholds s_1^*, s_2^* and the optimal expected queue lengths for fixed thresholds $(Q_1 + Q_2)^*$ are listed on line 5. In lines 2–4 we use thresholds lower than the optimal s_1^*, s_2^*, and in lines 6–8 thresholds which are higher than the optimal.

Line 9 of the table lists the expected queue lengths under the queue balancing diagonal policy.

The last line (line 10) of the table lists the expected queue lengths under a hybrid policy: We use the queue balancing diagonal policy in the square of $s \times s$ and fixed threshold $s_1 = s_2 = s$ outside that square. The values of s are the optimal ones for this policy. These results for the hybrid policy were estimated through simulation.

Our conclusions from Table 1 are the following:

– As expected, the Push–Pull network becomes more congested as $\mu_i \nearrow \lambda_i$.

Table 1 Expected queue lengths for symmetric Push–Pull network

	μ/λ	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.75	0.8	0.85	0.9	0.95
Threshold policy														
1	Q_1+Q_2	4.11	4.24	4.58	5.05	5.71	6.67	8.13	10.59	12.57	15.56	20.54	30.53	60.51
	$s_1=s_2=2$													
2	Q_1+Q_2	–	–	–	–	–	–	–	–	–	14.84	19.00	21.28	33.99
	s1*	–	–	–	–	–	–	–	–	–	4	4	5	7
	s2* − 2	–	–	–	–	–	–	–	–	–	2	2	3	5
3	Q_1+Q_2	–	–	–	–	–	6.67	8.13	10.59	12.57	12.18	15.46	20.00	33.51
	s1* − 1	–	–	–	–	–	2	2	2	2	3	3	4	6
	s2* − 1	–	–	–	–	–	2	2	2	2	3	3	4	6
4	Q_1+Q_2	–	–	–	–	–	6.83	8.05	10.15	11.85	12.13	15.13	19.83	33.43
	s1*	–	–	–	–	–	3	3	3	3	4	4	5	7
	s2* − 1	–	–	–	–	–	2	2	2	2	3	3	4	6
5	$(Q_1+Q_2)^*$	4.11	4.24	4.58	5.05	5.71	6.58	7.42	8.96	10.23	11.86	14.54	19.53	33.30
	s1*	2	2	2	2	2	3	3	3	3	4	4	5	7
	s2*	2	2	2	2	2	3	3	3	3	4	4	5	7
6	Q_1+Q_2	5.01	5.06	5.24	5.56	6.06	7.18	7.87	9.21	10.36	12.16	14.66	19.67	33.40
	s1*	2	2	2	2	2	3	3	3	3	4	4	5	7
	s2* + 1	3	3	3	3	3	4	4	4	4	5	5	6	8
7	Q_1+Q_2	5.91	5.84	5.79	5.87	6.11	7.70	8.21	9.32	10.31	12.37	14.70	19.74	33.48
	s1* + 1	3	3	3	3	3	4	4	4	4	5	5	6	8
	s2* + 1	3	3	3	3	3	4	4	4	4	5	5	6	8
8	Q_1+Q_2	6.01	6.04	6.18	6.45	6.89	7.99	8.58	9.80	10.88	12.67	15.07	20.01	33.66
	s1*	2	2	2	2	2	3	3	3	3	4	4	5	7
	s2* + 2	4	4	4	4	4	5	5	5	5	6	6	7	9
Queue balancing diagonal policy														
9	Q_1+Q_2	5.44	5.57	5.89	6.34	6.99	7.92	9.36	11.81	13.78	16.76	21.73	31.71	61.61
Hybrid policy														
10	Q_1+Q_2	4.58	4.68	4.97	5.34	5.73	6.32	7.25	8.73	9.84	11.50	14.10	19.07	32.81
	$s\times$	2	2	2	3	3	3	3	4	4	4	5	6	8
	s	2	2	2	3	3	3	3	4	4	4	5	6	8

– It is possible to locate the best fixed thresholds for each set of parameters. For the symmetric network, the optimal thresholds are symmetric. The optimal thresholds are often higher than the minimal thresholds required for stability.

Table 2 Expected queue lengths for different processing speeds

μ/λ	0.1		0.3		0.5		0.7		0.8		0.9		0.95	
$K=1$														
s_1, s_2	2	2	2	2	3	3	3	3	4	4	5	5	7	7
Q_1, Q_2	2.12	2.12	2.53	2.53	3.29	3.29	4.48	4.48	5.93	5.93	9.77	9.77	16.65	16.65
Q_1+Q_2	4.24		5.05		6.58		8.96		11.86		19.53		33.30	
$K=1.5$														
s_1, s_2	2	2	2	2	3	2	4	3	4	3	6	4	9	6
Q_1, Q_2	2.08	2.18	2.37	2.76	4.20	2.42	5.02	4.14	6.31	5.80	10.30	9.78	16.23	18.21
Q_1+Q_2	4.26		5.13		6.62		9.16		12.11		20.08		34.44	
$K=2$														
s_1, s_2	2	2	2	2	3	2	4	3	5	3	7	4	10	5
Q_1, Q_2	2.06	2.24	2.29	3.00	3.98	2.63	4.74	4.65	6.82	5.70	10.26	10.76	16.56	20.03
Q_1+Q_2	4.30		5.29		6.62		9.39		12.51		21.02		36.58	
$K=2.5$														
s_1, s_2	2	2	3	2	3	2	4	3	5	3	8	4	12	5
Q_1, Q_2	2.05	2.30	2.24	3.18	3.85	2.85	4.57	5.15	6.50	6.48	10.57	11.59	17.12	21.99
Q_1+Q_2	4.35		5.42		6.70		9.72		12.97		22.16		39.10	
$K=3$														
s_1, s_2	2	2	3	2	3	2	5	3	6	3	9	4	13	4
Q_1, Q_2	2.04	2.36	2.21	3.24	3.77	3.07	5.44	4.64	7.25	6.24	11.07	12.28	19.00	22.54
Q_1+Q_2	4.40		5.45		6.83		10.08		13.49		23.35		41.54	

- The objective of minimal expected queue length is quite flat around the optimal thresholds.
- The fixed threshold policy with optimal threshold performs better than the queue balancing diagonal policy. The difference increases as the network becomes more congested.
- The hybrid policy performs slightly better than the fixed threshold policy. The optimal thresholds for the hybrid policy are similar to, but slightly higher than, for the fixed threshold policy.

In Table 2 we consider Push–Pull networks with $K = \mu_2/\mu_1 = \lambda_2/\lambda_1$, that is, the ratio of μ_i/λ_i is equal for both types of jobs, but the jobs of type 2 are processed (by both servers) at a rate which is K times faster than for type 1. Note that in this case, $\theta_1 = \theta_2 = 1/2$, while $\nu_2 = K\nu_1$ (see (1)), and the balanced network will spend equal times working on the two different types, and complete K times more of the jobs that require less work. We report the expected steady-state number of jobs in the network for various values of μ_i/λ_i and various values of K, under fixed thresholds, where we choose the optimal levels of the fixed thresholds in each case.

Table 3 Expected queue lengths for asymmetric Push–Pull network

	$\mu_1/\lambda_1 = 0.50$												
μ_2/λ_2	0.05	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.75	0.80	0.85	0.90	0.95
s	7	6	4	3	3	3	3	4	5	7	9	14	28
Q_1	6.05	5.10	3.23	2.43	2.69	3.29	4.84	7.53	9.41	11.86	16.82	26.02	54.23
Q_2	10.00	7.68	6.24	5.38	4.01	3.29	2.75	3.04	3.60	4.92	6.08	9.22	17.81
$Q_1 + Q_2$	16.05	12.78	9.48	7.81	6.70	6.58	7.59	10.57	13.02	16.78	22.90	35.24	72.04
s_1	8	6	5	4	3	3	2	2	2	1	1	1	1
s_2	2	2	2	2	2	3	3	5	6	8	11	17	36
Q_1	7.05	5.12	4.30	3.63	3.21	3.29	3.64	3.88	4.72	3.95	5.51	8.58	16.81
Q_2	3.40	3.67	2.89	2.80	2.90	3.29	3.43	4.57	5.03	7.55	9.07	12.62	24.77
$Q_1 + Q_2$	10.44	8.79	7.20	6.42	6.11	6.58	7.07	8.45	9.76	11.50	14.58	21.20	41.59

We know that for the M/M/1 queue, increasing the processing speed does not change the queue length which is a function of λ/μ only. The main conclusion from Table 2 is that the Push–Pull network behaves similarly: The difference in queue lengths between $K = 1, 1.5, 2.0, 2.5, 3.0$ is quite small. Note however that the optimal thresholds for the two queues are quite different, that is, the slower queue needs a higher threshold.

In Table 3 we consider a different type of asymmetry: We look at Push–Pull networks with $\frac{1}{\mu_1} + \frac{1}{\lambda_1} = \frac{1}{\mu_2} + \frac{1}{\lambda_2}$, i.e., the total amount of work per job (push and pull operation) is the same for both types of jobs, but it is divided differently between the servers. To parameterize this situation we take $\frac{\mu_1}{\lambda_1} = 0.5$ and let $\frac{\mu_2}{\lambda_2}$ vary over a range of values from 0.05 to 0.95. In each of these we have calculated the steady-state expected number of jobs for two different threshold policies: In the top line we use equal thresholds for both queues, and we choose the optimal values of $s_1 = s_2 = s$. In the bottom line we use different thresholds, and we give the values of the thresholds s_1, s_2 and the queue lengths, where s_1, s_2 are chosen optimally.

It can be seen from this table that asymmetry in the division of work between push and pull in the two queues causes congestion: for $\mu_1/\lambda_1 = 0.5$, when μ_2/λ_2 is very small or very large, the queue length become much larger than for the symmetric case of $\mu_2/\lambda_2 = 0.5$. The optimal thresholds are also more different. We note also that choosing equal thresholds causes significantly more congestion than optimal choice of thresholds and accentuates the difference between the queue lengths at the two queues.

7 Comparison with the Kumar–Seidman–Rybko–Stolyar network

The Kumar–Seidman–Rybko–Stolyar (KSRS) network is depicted in Fig. 2(b). It differs from our Push–Pull network in that, instead of infinite supply of work, it has exogenous arrivals of jobs of type i at rate α_i, $i = 1, 2$. It has four queues, jobs of type i arrive at server i and queue up for their first processing step, and then move on

to queue up at server $\bar{\imath}$ for their second processing step. We let $Q_{i,1}(t)$ be the queue length of jobs of type i at server i, and $Q_{i,2}(t)$ be the queue length of jobs of type i at server $\bar{\imath}$. The offered load to server i is $\rho_i = \frac{\alpha_i}{\lambda_i} + \frac{\alpha_{\bar{\imath}}}{\mu_{\bar{\imath}}}$. The actions available for server i at time t are to push if $Q_{i,1}(t) > 0$ and to pull if $Q_{\bar{\imath},2}(t) > 0$. If both queues are empty, the server has to idle.

In this section we compare the behavior of our Push–Pull network to that of the KSRS network under heavy traffic. We observe first that the Push–Pull network can be regarded as a special case of the KSRS network when $\rho_i > 1$, $i = 1, 2$: If this is the case and if $Q_{i,2}$, $i = 1, 2$, are rate stable, then the queues $Q_{i,1}$, $i = 1, 2$, will, after some initial transient period, never be empty. Hence they will act exactly like IVQs. We could, on the other hand, give priority to $Q_{i,1}$, $i = 1, 2$, and if $\alpha_i < \lambda_i$, $i = 1, 2$, then this first-buffer-first-served (FBFS) priority policy will keep these two queues stable, and $Q_{i,2}$, $i = 1, 2$, will then grow linearly in time. A third option, keeping the four queues balanced is achieved by the max pressure policy, as we shall see in Sect. 7.2.

The KSRS network has been studied extensively (cf. [4, 25]) because of the following property: Assume that the processing times are exponential and that the arrival streams are Poisson, so that, under any stationary policy, $(Q_{1,1}(t), Q_{1,2}(t), Q_{2,1}(t), Q_{2,2}(t))$ form a Markov jump process. Assume also that $\rho_i < 1$, $i = 1, 2$. If in addition $\rho_v = \frac{\alpha_i}{\mu_i} + \frac{\alpha_{\bar{\imath}}}{\mu_{\bar{\imath}}} < 1$ (ρ_v is the workload of a so-called *virtual* server), then under any stationary work conserving policy (i.e., servers do not idle if they have any work to do) the resulting Markov jump process is positive recurrent. This result can even be generalized to i.i.d. inter-arrival and service times and positive Harris recurrence [9]. However, if $\rho_v > 1$, then this is not the case. In fact, under pull priority (last-buffer-first-served, LBFS) $\frac{Q_{1,2}(t)}{\mu_1 t} + \frac{Q_{2,2}(t)}{\mu_2 t} \to \rho_v - 1$ almost surely as $t \to \infty$. This was first discovered by Kumar and Seidman in their seminal 1990 paper [22], in which they considered deterministic arrival and service times, and later by Rybko and Stolyar [28], who considered exponential interarrival and service times.

Assume that $\rho_1, \rho_2 < 1$. If $\mu_i > \lambda_i$, $i = 1, 2$ (referred to as the inherently stable case in this paper), then $\rho_1, \rho_2 < 1$ implies also that $\rho_v < 1$. Hence the pull priority policy is stable, and so is any other non-idling policy. If however $\mu_i < \lambda_i$ (referred to as the inherently unstable case in this paper), we can find a constant γ such that if $\gamma < \rho_i < 1$, $i = 1, 2$, then $\rho_v > 1$, and the KSRS network will be unstable under pull priority policy, analogous to our observation for the Push–Pull network (see Sect. 2).

The reason for the instability is that under pull priority, if $Q_{1,2}$ is empty, then for as long as $Q_{2,2} > 0$ server 1 will not send any jobs to $Q_{1,2}$, and so server 2 will only work on $Q_{2,1}$ and will only be busy for a fraction $\frac{\alpha_2}{\lambda_2} < \rho_2$ of the time. In other words, server 2 will be starved, and similarly server 1 will be starved when $Q_{2,2}$ is empty and $Q_{1,2} > 0$. Under pull priority, after some transient period, only one of the queues $Q_{1,2}$, $Q_{2,2}$ will be served at any given time. This motivates the concept of a virtual server.

It is easy to find policies which will avoid starvation of this sort: The minimal correction will be to serve $Q_{i,1}$ whenever $Q_{i,2} = 0$, and this seems to be enough to guarantee stability (we are not aware of a reference in the literature for such a result, but we believe it is true). How to control the KSRS system so as to minimize queue lengths is however a difficult question, which is far from being resolved in the literature.

The question of control becomes more acute as ρ_1 or ρ_2 increase, because in that case some of the queues in the network become congested. In particular, if $\alpha_1 \nearrow \nu_1$ and $\alpha_2 \nearrow \nu_2$, then both $\rho_1, \rho_2 \nearrow 1$, and the system will approach balanced heavy traffic with some congestion. This is indeed in contrast to the Push–Pull network, which we can operate at full utilization (servers busy all the time, $\rho_1 = \rho_2 = 1$) and yet maintain stable queues.

We will now compare the performance of the KSRS network in balanced heavy traffic with that of the Push–Pull network. We will consider two policies for the KSRS network: The affine switching curve policy of Henderson, Meyn, and Tadic [20] (see also [5, 25]) and the max pressure policy of Dai and Lin [7, 8] (see also [29, 30]).

For ease of presentation and avoiding inessential complications, we follow [20] and consider only symmetric networks with $\alpha = \alpha_1 = \alpha_2$, $\lambda = \lambda_1 = \lambda_2$, $\mu = \mu_1 = \mu_2$. This implies $\nu = \nu_1 = \nu_2 = \frac{\lambda\mu}{\lambda+\mu}$, and traffic intensity of both servers is $\rho = \rho_1 = \rho_2 = \frac{\alpha}{\nu}$. To simplify the comparison we will denote the queues of the Push–Pull network by $Q_{1,2} = Q_1$, $Q_{2,2} = Q_2$. The queues $Q_{1,1}$, $Q_{2,1}$ of the KSRS network correspond to the infinite virtual queues of the Push–Pull network.

7.1 The KSRS network with affine switching curve policy

In this section we consider the performance of KSRS under the affine switching curve policy suggested by Henderson, Meyn, and Tadic [20]. We performed extensive simulation studies with versions of this policy and obtained some interesting results, which provide new insights into [20].

As we have already noted, the approach of Henderson, Meyn, and Tadic [20] in analyzing the KSRS network is rather different from our approach in analyzing the push–pull system. Our motivation here is to clarify the points of similarity and difference, as these may not be obvious from a first reading of [20].

Mainly, in the current paper we perform an exact analysis of the jump Markov process, while in [20] the analysis uses fluid and diffusion scaling, and optimality statements refer only to these scaled processes.

Our most significant point in our comparison here is to indicate that for the KSRS network under balanced heavy traffic, in the inherently stable case LBFS with no idling seems optimal, and in the inherently unstable case FBFS is optimal when $\rho \to 1$ as long as $\rho < 1$. This is somewhat in contrast to some of the results reported in [20]. When $\rho > 1$, then in the inherently stable case the behavior of the KSRS becomes identical to that of the Push–Pull network under pull priority. In contrast to that, in the inherently unstable case, for $\rho > 1$, the LBFS policy causes all four queues to diverge, while FBFS policy will cause the queues $Q_{i,2}$ to diverge. It is here that our more delicate policy for the Push–Pull network succeeds in keeping the queues $Q_{i,2}$ stable positive recurrent (exponentially ergodic). The following is our detailed analysis, including simulation results.

The affine switching curve policy is based on the total workload $W_i(t)$ and the immediate workload $w_i(t)$ of each of the servers $i = 1, 2$ defined by

$$W_i(t) = \frac{Q_{i,1}(t)}{\lambda} + \frac{Q_{\bar{\imath},1}(t) + Q_{\bar{\imath},2}(t)}{\mu},$$

$$w_i(t) = \frac{Q_{i,1}(t)}{\lambda} + \frac{Q_{\bar{\imath},2}(t)}{\mu}.$$

The policy is defined by two affine switching curves and two thresholds and is as follows:

- Idle server i if $W_i(t) \le s_i^*[W_{\bar{\imath}}(t) - \bar{W}_{\bar{\imath}}]$.
- When not idling, server i serves $Q_{i,1}$ if $Q_{\bar{\imath},2}(t) = 0$ or if $w_{\bar{\imath}}(t) < \bar{w}_{\bar{\imath}}$,

where the constants s_i^*, $\bar{W}_i$, $\bar{w}_i$, $i = 1, 2$, depend on the parameters of the network. s_i^* is derived from an appropriate linear program (cf. Example 5.3.1.6 in [25]), while $\bar{W}_i$, $\bar{w}_i$ are adjusted by simulation.

Example 1: *inherently stable,* $\lambda = \frac{1}{3}$, $\mu = 1$

For the inherently stable case ($\lambda < \mu$), the policy of [20] is to use $s_i^* = \frac{\lambda}{\mu}$. In [20], this example of $\lambda = \frac{1}{3}$, $\mu = 1$ is only discussed in general terms, and there is no use of simulation to indicate which values of $\bar{W}_i$, $\bar{w}_i$ to choose.

The following choices of $\bar{W} = \bar{W}_1 = \bar{W}_2$, and $\bar{w} = \bar{w}_1 = \bar{w}_2$ are of special interest:

- The choice $\bar{W}_i = 0$ introduces the most idling.
- When $\bar{W}_i = \infty$, the policy is non-idling.
- If we choose $\bar{w}_i = 0$, the policy is to use pull priority at each server (when not idling), which is LBFS.
- If we choose $\bar{w}_i = \infty$, the policy is to use push priority at each server (when not idling), which is FBFS.

We simulated the KSRS network for these values and observed the following mean number of jobs in the network:[1]

	$\bar{W} = 0$	$\bar{W} = \infty$
$\bar{w} = 0$	15.59	15.39
$\bar{w} = \infty$	47.66	36.60

We draw the following conclusions for this example:

- LBFS is preferable to FBFS.
- Under FBFS, non-idling is preferable to idling.
- There is very little effect to idling when priority is according to LBFS.

The seemingly surprising observation that idling has little effect under LBFS has the following explanation: Under LBFS, the queues $Q_{1,2}$, $Q_{2,2}$ will have very few jobs, and most of the jobs in the system will be in the two queues $Q_{1,1}$, $Q_{2,1}$. Assume that $Q_{1,2}(t) = Q_{2,2}(t) = 0$; then $s_1^* = \lambda/\mu \le W_2(t)/W_1(t) \le \mu/\lambda = 1/s_2^*$, so that in heavy traffic, under LBFS, the workload process will automatically stay away from the idling area. This is demonstrated in Fig. 9, in which we plotted the sample path of the workload for one simulation run of the non-idling LBFS policy ($\bar{w} = 0$ and

[1]Each cell was obtained from 60 runs of 5×10^5 time units using exponential processing times with a preemptive policy.

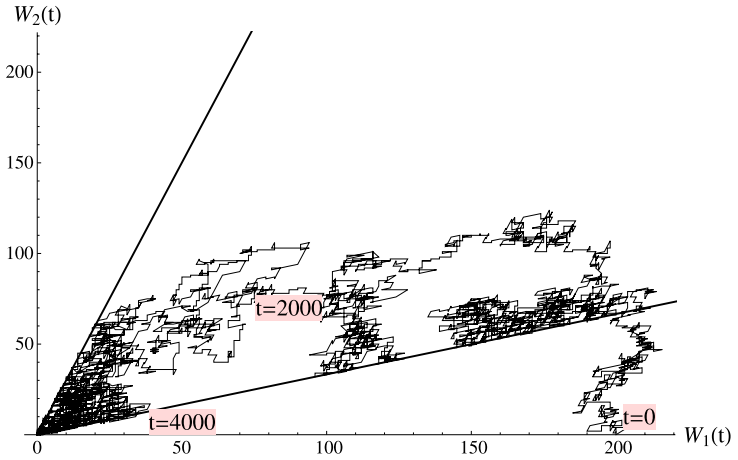

Fig. 9 Sample path of the workload in KSRS, under non-idling LBFS policy, $\lambda = \frac{1}{3}$, $\mu = 1$, and $\rho = 0.9$

$\bar{W} = \infty$). The path starts at time $t = 0$ with $Q_4(t) = 200$ and the other queues empty. It is clearly seen that the sample path drifts upward into the non-idling region, and it stays there for the remaining time without resorting to idling.

We conjecture more generally that, for the KSRS network in heavy traffic, in the inherently stable case, LBFS with no idling minimizes the expected number of jobs in the network.

For the Push–Pull network, our policy is pull priority, and the expected steady-state queue lengths for this numerical example are (from (6))

$$\mathbb{E}(Q_{1,2} + Q_{2,2}| \text{ Push–Pull, Pull Priority}) = \frac{2\mu\lambda}{\mu^2 - \lambda^2} = 0.75.$$

By comparing sample paths it is easy to see that

$$\mathbb{E}(Q_{1,2} + Q_{2,2}| \text{ KSRS, LBFS}) < \mathbb{E}(Q_{1,2} + Q_{2,2}| \text{ Push–Pull Pull Priority}).$$

Of course, as $\rho \to 1$, the expected queue lengths in $Q_{1,1}, Q_{2,1}$ will increase like $\rho/(1-\rho)$.

Example 2: *inherently unstable,* $\lambda = 1, \mu = \frac{1}{3}$ *and* $\lambda = 1, \mu = 0.8$

For the inherently unstable case where $\lambda_i > \mu_i$, $i = 1, 2$, the policy of [20] is to use $s_i^* = 0$, $\bar{W}_i = 0$. The symmetric value of $\bar{w}_1 = \bar{w}_2$ is determined by simulation.

Henderson, Meyn, and Tadic have simulated this example of the KSRS network with $\rho = 0.9$ for a grid of $\bar{w}_1, \bar{w}_2$ values. They used the simulation to determine the optimal values which minimize the expected steady-state number of jobs in the network. Their estimated optimal choice was $\bar{w}_1 = 35$, $\bar{w}_2 = 30$, and $\mathbb{E}(\sum Q) = 17.6$. They point out that clearly the actual optimal choice should be symmetric, $\bar{w} = \bar{w}_1 = \bar{w}_2$.

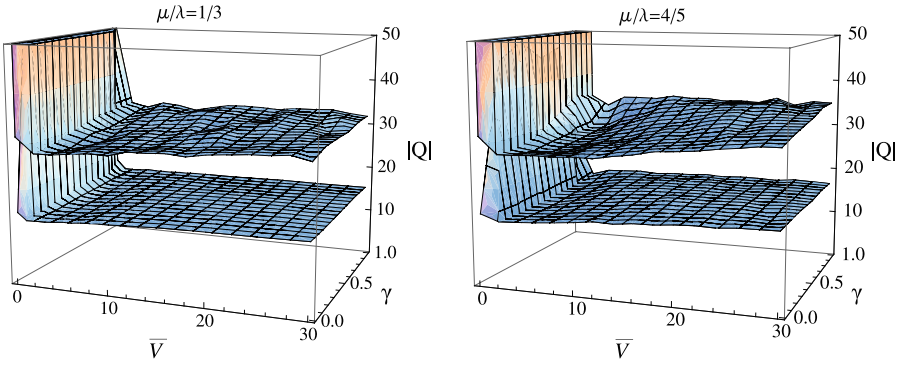

Fig. 10 Simulated expected number of jobs in the KSRS network with $\lambda = 1$, $\mu = 1/3$ and $\lambda = 1$, $\mu = 0.8$ for $\rho = 0.9$ (*bottom* surfaces) and $\rho = 0.95$ (*top* surfaces) over a grid of $\gamma \in [0, 1]$ and $\bar{V} \in [0, 30]$

We have repeated the simulation of KSRS for these parameters and performed a slightly wider search for optimal performance. The safety stocks $\bar{w}_i$ in [20] are in terms of the immediate work load of machine i, $w_i(t)$, which weighs jobs in $Q_{i,1}$ and $Q_{\bar{\imath},2}$ according to their expected processing times. We define generalized immediate workloads $V_i(t) = \gamma Q_{i,1}(t) + (1-\gamma)Q_{\bar{\imath},2}(t)$, $0 \le \gamma \le 1$, $i = 1, 2$. We then use symmetric safety stocks of $\bar{V} = \bar{V}_1 = \bar{V}_2$. Our policy for γ and $\bar{V}$ is defined as follows:

– For servers $i = 1, 2$, service is non-idling, server i will give priority to push operation (serve $Q_{i,1}$) if the generalized immediate work load of server $\bar{\imath}$ is low, that is, if $\gamma Q_{\bar{\imath},1}(t) + (1-\gamma)Q_{i,2}(t) \le \bar{V}$. Otherwise server i will use pull priority (serve $Q_{\bar{\imath},2}$).

Here $\gamma = 0$ means that we only look at the workload in the pull queues, $Q_{i,2}$, $i = 1, 2$. This is equivalent to the fixed threshold policy which we use for the Push–Pull network. In particular, $\gamma = 0$, $\bar{V} = 0$ is the minimal anti-starvation policy: Give priority to pull ($Q_{\bar{\imath},2}$) and work on $Q_{i,1}$ only if either $Q_{i,2} = 0$ or $Q_{\bar{\imath},2} = 0$. Our simulation indicates that this is stable for $\rho < 1$ but creates long queues. On the other hand, as we let $\bar{V}$ increase, for any $0 \le \gamma \le 1$, the policy will more often give priority to push over pull, and hence as we let $\bar{V} \to \infty$, we will actually be using the policy of push priority or FBFS, which is stable.

Our simulation results for $\rho = 0.9, 0.95$ for systems with parameters $\lambda = 1$, $\mu = 1/3$ and $\lambda = 1$, $\mu = 0.8$ over a grid of γ, $\bar{V}$ are given in Fig. 10.[2]

We note some interesting properties:

– For the case $\lambda = 1$, $\mu = 1/3$, the values $\gamma = 0.75$, $\bar{V} = 8$ correspond to the policy in [20] with $\bar{w} = 32$. The queue lengths which we obtained in our simulation were 15.3, which is close to the 17.6 reported in [20]. We note that we have

[2]The grid used for the simulations uses a step size of 0.2 for γ and steps ranging from 1 to 3 for $\bar{V}$. Each point on the grid used between 20 to 100 runs of duration 5×10^5 time units, depending on the observed standard error.

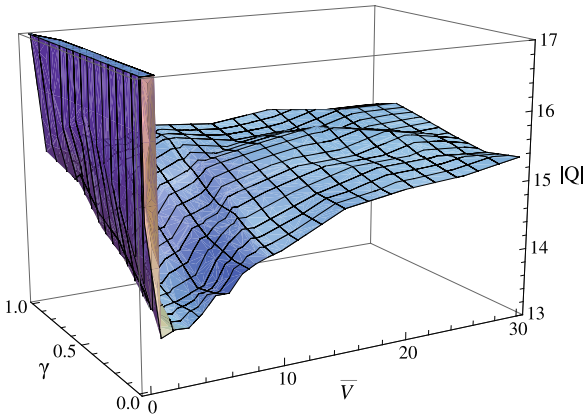

Fig. 11 A closer look at the curve of Fig. 10 for $\lambda = 1$, $\mu = 1/3$, and $\rho = 0.9$

used memoryless exponential inter-arrival and processing times and preemptive priorities in our simulation, while [20] employs a two-parameter discrete-time model for all events. Hence the results are not in complete agreement but are close enough.

- For small values of $\bar{V}$ (including $\bar{V} = 0$), the policy is stable for all γ, but the queue lengths are large.
- As $\bar{V}$ increases, for each fixed γ, a minimum is reached, followed by gradual increase towards an asymptotic value which is the value for FBFS. Hence, contrary to the picture presented in [20], the expected queue lengths are not convex functions of $\bar{V}$, though they are quasi-convex.
- For given γ, we consider the minimum of the queue lengths over all $\bar{V}$. Regarded as a function of γ, it is increasing in γ. The best values are obtained not for $\gamma = \frac{\lambda}{\lambda+\mu}$ as in [20] but for $\gamma = 0$. This can be observed in Fig. 11, which is a rescaled and rotated view of one of the surfaces in Fig. 10. The evaluation of the "best" γ and $\bar{V}$ becomes much harder as ρ increases.
- In all four surface plots, the minimum is very flat. The difference between the minimum value and the value for FBFS is small. Hence, for these two instances of the KSRS network, the policy proposed in [20] as well as our threshold policy do not present a significant improvement over the simple FBFS rule.

KSRS under preemptive FBFS policy

The behavior of KSRS under preemptive FBFS is easy to derive. Assume Poisson arrivals and exponential services. Then $Q_{i,1}$, $i = 1, 2$, behave like two independent M/M/1 queues with arrival rates α_i and service rates λ_i. Furthermore, the departures of $Q_{\bar{\imath},1}$ which are the arrivals of $Q_{\bar{\imath},2}$ form a Poisson process, and they are independent of the arrivals of $Q_{i,1}$. Hence, server i has two independent Poisson arrival streams of rates $\alpha_i, \alpha_{\bar{\imath}}$, which are served according to preemptive priority to $Q_{i,1}$ over $Q_{\bar{\imath},2}$. A further property is that past departures from $Q_{i,1}$ are independent of the present state of $Q_{i,1}$ (by reversibility). Hence the joint stationary distribution of $Q_{i,1}(t), Q_{i,2}(t)$ (the jobs of type i in the system) is of product form. In fact these results are valid for general i.i.d. service distributions in $Q_{i,2}$.

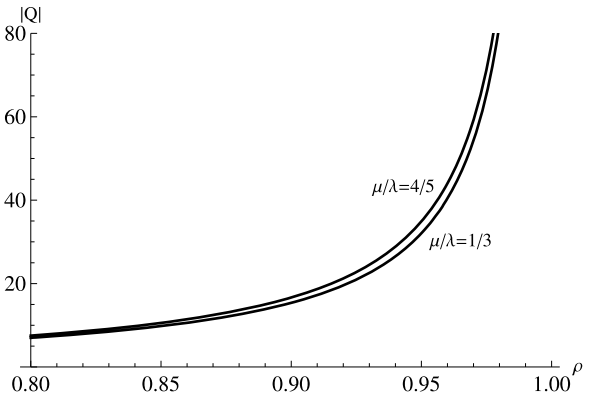

Fig. 12 Mean number of jobs in the KSRS network, under FBFS policy, as a function of ρ, for $\lambda = 1$, $\mu = 1/3$ and $\lambda = 1$, $\mu = 0.8$

By these arguments it is straightforward to see that the expected stationary number of jobs in the symmetric KSRS network under preemptive FBFS is

$$\begin{aligned} \mathbb{E}(Q_{i,1}) &= \frac{\alpha}{\lambda - \alpha}, \\ \mathbb{E}\Big(\sum Q_{i,j}\Big) &= \frac{2\rho}{(1-\rho)} \frac{1 + \left(\frac{\mu}{\lambda}\right)^2 + 2(1-\rho)\frac{\mu}{\lambda}}{1 + (1-\rho)\left(\frac{\mu}{\lambda}\right)^2 + (2-\rho)\frac{\mu}{\lambda}}. \end{aligned} \tag{45}$$

We plot $\mathbb{E}(Q_{1,1} + Q_{1,2} + Q_{2,1} + Q_{2,2})$ as a function of ρ for $\lambda = 1$, $\mu = 1/3$ and for $\lambda = 1$, $\mu = 0.8$ in Fig. 12.

An interesting observation from Fig. 12 is that there is not much difference in the mean number of jobs in the network between $\lambda = 1$, $\mu = 1/3$ and $\lambda = 1$, $\mu = 0.8$. In fact it is seen from (45) that the expected number of jobs for $\rho \sim 1$ is

$$\mathbb{E}\Big(\sum Q_{i,j}\Big) \Big/ \frac{2\rho}{1-\rho} \sim \frac{1 + (\mu/\lambda)^2}{1 + \mu/\lambda}.$$

Returning to the Push–Pull network, we see that in the inherently unstable case its behavior is fundamentally different from the behavior of KSRS under FBFS or under the policy of Henderson, Meyn, and Tadic [20]: Clearly the Push–Pull network cannot be operated under FBFS (i.e., push priority), because it would never serve the pull operations. Our threshold policies strive to give priority to push only as much as is needed to ensure stability. The expected number of jobs in the queues under the optimal fixed threshold policy is (this was calculated in Sect. 6)

For $\lambda = 1$, $\mu = 1/3$: The best fixed threshold is $\gamma = 0$, $\bar{V} = 1$, and $E(Q_{1,2} + Q_{2,2}) \approx 5.25$.

For $\lambda = 1$, $\mu = 0.8$: The best fixed threshold is $\gamma = 0$, $\bar{V} = 3$, and $E(Q_{1,2} + Q_{2,2}) \approx 11.86$.

Note in particular, that for the Push–Pull network, which is running at full utilization, there is a great difference in performance between $\lambda = 1$, $\mu = 1/3$ and $\lambda = 1$, $\mu = 0.8$. As we saw, the difference in KSRS under FBFS is slight.

7.2 The Push–Pull and KSRS networks under max pressure policies

In [7] Dai and Lin introduce a max pressure policy for the control of multi-class processing networks. In particular, for multiclass queueing networks, Dai and Lin show that if the traffic load of the network is $\rho \leq 1$, then under max pressure policy the fluid model of the network is weakly stable. This implies that the stochastic network process is rate stable, i.e., input and output rates of all the queues are the same, and there is no linear accumulation of jobs in the network. Furthermore, if the traffic load of the network is $\rho < 1$, then under max pressure policy the fluid model of the network is stable. This implies that if the arrival processes and service processes of the network are independent renewal processes and if the arrival processes obey a minorization condition, then the stochastic network under max pressure policy can be described by a positive Harris recurrent Markov process (cf. [6]).

In [26] we have proposed an adaptation of the max pressure policy of Dai and Lin to multiclass queueing networks (MCQN) with infinite virtual queues (IVQs). Here the classes are $\mathcal{K} = \mathcal{K}_0 \cup \mathcal{K}_\infty$, where $k \in \mathcal{K}_0$ are standard queues, $Q_k(t) \geq 0$, while $k \in \mathcal{K}_\infty$ are IVQs with infinite supply of work. Each class $k \in \mathcal{K}_\infty$ has a nominal flow rate of α_k, and we represent the state of these IVQs by $R_k(t) = \alpha_k t - D_k(t)$, where $D_k(t)$ counts the departures from k in $(0, t)$. The adaptation of the max pressure policy is to use $R_k(t)$ as a surrogate for the queue length (which is infinite) in all the IVQs. Most of the results of Dai and Lin also hold for this adapted max pressure policy.

In this section we compare the performance of the KSRS network under max pressure policy and of the Push–Pull network under the adapted max pressure policy.

The max pressure policy attempts at any time to achieve the fastest reduction in the sum of squares of the queue lengths in a processing network, and as a result it strives to equalize or balance the number of jobs in the different queues. The policies which we developed for the Push–Pull network in Sects. 3–5 managed to use the infinite supply of work to achieve no idling and yet keep the two standard queues $Q_{1,2}$, $Q_{2,2}$ stable. In the adaptation of the max pressure policy to the Push–Pull network, this is no longer the case, and in fact the two standard queues become congested.

For the KSRS network with independent exponential processing times and independent Poisson arrivals, under max pressure policy, the queue lengths form a Markov jump process. It is positive recurrent if $\rho = \rho_1 = \rho_2 < 1$, null recurrent if $\rho = 1$, and transient diverging to ∞ if $\rho > 1$.

This max pressure policy applied to the KSRS network is as follows:

$$\text{Server 1: Pull if } \mu_2 Q_{2,2}(t) > \lambda_1\big(Q_{1,1}(t) - Q_{1,2}(t)\big), \text{ else Push,}$$

$$\text{Server 2: Pull if } \mu_1 Q_{1,2}(t) > \lambda_2\big(Q_{2,1}(t) - Q_{2,2}(t)\big), \text{ else Push.}$$

To apply the adapted max pressure policy to the Push–Pull network, we need to specify the nominal flow rates for the virtual infinite queues $Q_{1,1}$, $Q_{2,1}$. We will use $\alpha_i = \rho \nu_i$, so that ρ will be the traffic intensity for both servers. Again, the system will be described by a Markov jump process, which is positive recurrent if $\rho = \rho_1 = \rho_2 < 1$, null recurrent if $\rho = 1$, and transient and diverging to ∞ if $\rho > 1$.

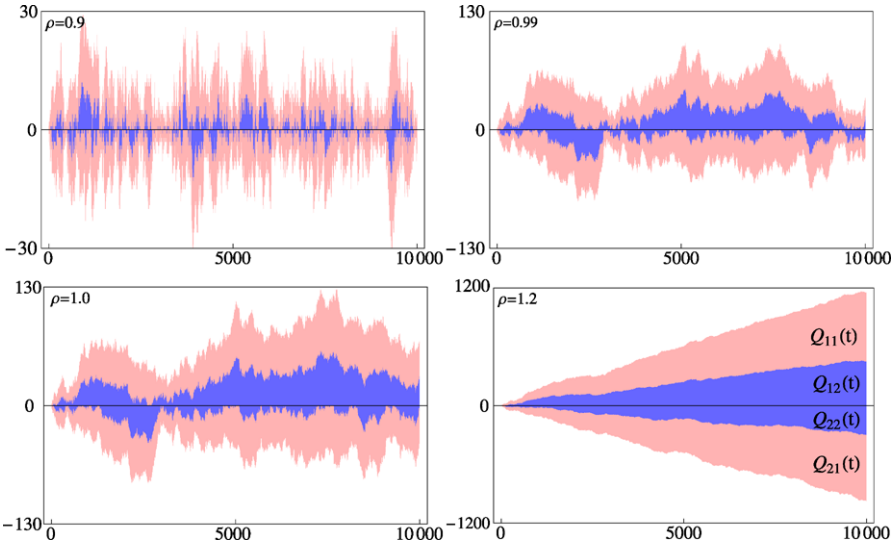

Fig. 13 Queue lengths in the KSRS network under max pressure policy

The max pressure policy adapted for the Push–Pull network is

$$\text{Server 1: Pull if } \mu_2 Q_{2,2}(t) > \lambda_1\big(\alpha_1 t - D_{1,1}(t) - Q_{1,2}(t)\big), \text{ else Push,}$$
$$\text{Server 2: Pull if } \mu_1 Q_{1,2}(t) > \lambda_2\big(\alpha_2 t - D_{2,1}(t) - Q_{2,2}(t)\big), \text{ else Push.}$$

We simulated the KSRS network and the Push–Pull network under these policies. We used the parameters $\lambda = 1$, $\mu = 0.8$ and chose α so that $\rho = \rho_1 = \rho_2$ took values 0.9, 0.99, 1.0, 1.2. The first two correspond to stable networks under heavy and very heavy traffic. The third is the case of full utilization—this is at best null recurrent. The fourth corresponds to an overloaded network. Figures 13 and 14 present traces of single simulation runs processing a large total number of jobs over a time horizon of 10,000 time units.

In Fig. 13 we show the four simulation runs of the KSRS network for the four traffic intensities. The horizontal axis is the time running from 0 to 10,000. The vertical axis measures number of jobs, and we plotted on it four traces of $Q_{1,2}(t)$, $Q_{1,2}(t) + Q_{1,1}(t)$, $-Q_{2,2}(t)$, $-Q_{2,2}(t) - Q_{2,1}(t)$, so that the four layers in the graph correspond to the number of jobs in the four queues and the layers immediately above and below the x-axis correspond to the pull queues.

In Fig. 14 we show the four simulation runs of the Push–Pull network for the four traffic intensities. The axes are the same, and we plotted two traces of $Q_{1,2}(t)$, $-Q_{2,2}(t)$, so the two layers in the graph correspond to the number of jobs in the two queues.

All these runs started with empty queues at time 0. We used a single sequence of exponential inter-arrival and processing times of some 20,000 jobs to generate all these runs (speeding up the arrivals by a constant factor for increasing ρ).

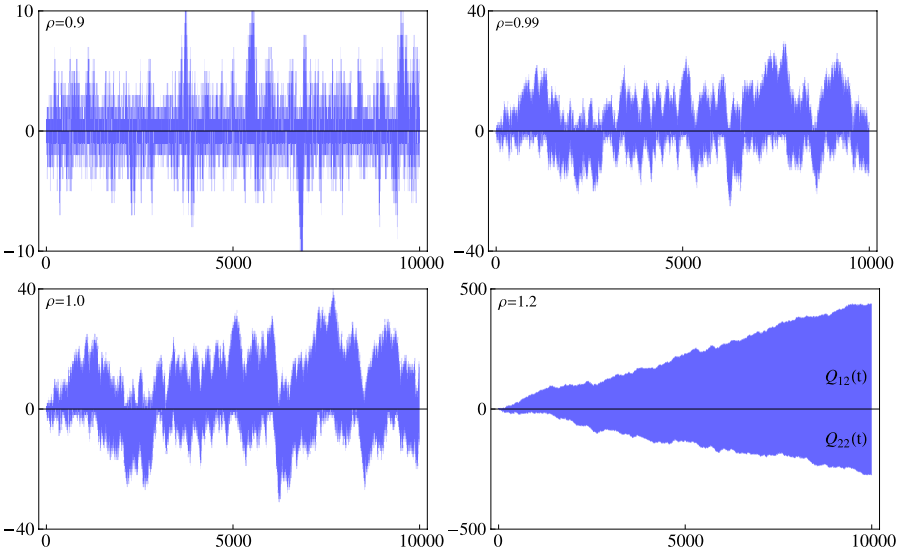

Fig. 14 Queue length in the Push–Pull network under the adapted max pressure policy

From the theory we would expect the queue level for $\rho < 1$ to stabilize at some steady-state level, and this is what we observe for $\rho = 0.9$. The trace for $\rho = 0.99$ does not seem to have enough time to stabilize, and indeed it is very similar to the trace for $\rho = 1$. For $\rho = 1$, the network is at best null recurrent, and so we do not expect the traces to stabilize: they will continue to have longer and longer excursion with very high queue lengths. The overloaded cases clearly show linear growth of the queues.

As a general indication, here are the mean queue lengths from this simulation for $\rho = 0.9$:

	KSRS	Push–Pull
$Q_{1,1} + Q_{2,1}$	18.45	–
$Q_{1,2} + Q_{2,2}$	6.25	4.5

We conclude from these simulations that under the adapted max pressure policy the Push–Pull network behaves very similarly to the KSRS network under max pressure. In particular, since we are not making special use of the infinite supply of work, the two pull queues become congested in heavy traffic.

Acknowledgement We are grateful to an anonymous referee for pointing out that our Lyapunov function in Sect. 5 is uniformly Lipschitz continuous, and hence the Markov process is exponentially ergodic.

References

1. Adan, I.J.B.F., Weiss, G.: A two node Jackson network with infinite supply of work. Probab. Eng. Inf. Sci. **19**, 191–212 (2005)
2. Adan, I.J.B.F., Weiss, G.: Analysis of a simple Markovian re-entrant line with infinite supply of work under the LBFS policy. Queueing Syst. Theory Appl. **54**, 169–183 (2006)

3. Bremaud, P.: Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues. Springer, New York (1999)
4. Chen, H., Yao, D.D.: Fundamentals of Queueing Networks, Performance, Asymptotics and Optimization. Springer, Berlin (2003)
5. Chen, M., Pandit, C., Meyn, S.: In search of sensitivity in network optimization. Queueing Syst. Theory Appl. **44**, 313–363 (2003)
6. Dai, J.G.: On positive Harris recurrence of multi-class queueing networks: a unified approach via fluid limit models. Ann. Appl. Prob. **5**, 49–77 (1995)
7. Dai, J.G., Lin, W.: Maximum pressure policies in stochastic processing networks. Oper. Res. **53**, 197–218 (2005)
8. Dai, J.G., Lin, W.: Asymptotic optimality of maximum pressure policies in stochastic processing networks. Ann. Appl. Probab. **18**, 2239–2299 (2008)
9. Dai, J.G., Vande Vate, J.H.: Global stability of two-station queueing networks. In: Glasserman, P., Sigman, K., Yao, D. (eds.) Proceedings of Workshop on Stochastic Networks: Stability and Rare Events, pp. 1–26. Springer, New York (1996)
10. Durrett, R.: Essentials of Stochastic Processes. Springer, New York (1999)
11. Glynn, P.: Stochastic Systems. Stanford University Graduate Lecture Notes. Stanford University Press, Stanford (2000)
12. Foss, S., Konstantopoulos, T.: An overview of some stochastic stability methods. J. Oper. Res. Soc. Jpn. **47**(4), 275–303 (2004)
13. Foster, F.G.: On the stochastic matrices associated with certain queueing processes. Ann. Math. Stat. **24**, 355–360 (1953)
14. Harrison, J.M.: Brownian models of queueing networks with heterogeneous customer populations. In: Fleming, W., Lions, P.L. (eds.) Proceedings of the IMA Workshop on Stochastic Differential Systems. Springer, Berlin (1988)
15. Harrison, J.M.: Brownian models of open processing networks: Canonical representation of workload. Ann. Appl. Probab. **10**, 75–103 (2000)
16. Harrison, J.M.: A broader view of Brownian networks. Ann. Appl. Probab. **13**, 1119–1150 (2001)
17. Harrison, J.M.: Stochastic networks and activity analysis. In: Suhov, Y. (ed.) In memory of Fridrik Karpelevich. Analytic Methods in Applied Probability. Am. Math. Soc., Providence (2002)
18. Harrison, J.M., Williams, R.J.: Brownian models of multiclass queueing networks. In: Proceedings of the 29th IEEE Conference on Decision and Control, vol. 2, pp. 573–574 (1990)
19. Haviv, M., Zlotnikov, R.: Computational schemes for two exponential servers where the first has a finite buffer (2007, preprint)
20. Henderson, S.G., Meyn, S.P., Tadic, V.B.: Performance evaluation and policy selection in multiclass networks. Discrete Event Dyn. Syst. **13**(1–2), 149–189 (2003)
21. Kopzon, A., Weiss, G.: A push–pull queueing system. Oper. Res. Lett. **30**, 351–359 (2002)
22. Kumar, P.R., Seidman, T.I.: Dynamic instabilities and stabilization methods in distributed real time scheduling of manufacturing systems. IEEE Trans. Automat. Contr. **35**, 289–298 (1990)
23. Levy, Y., Yechiali, U.: Utilization of idle time in an M/G/1 queueing system. Manag. Sci. **22**, 202–211 (1975)
24. Meyn, S.P., Tweedie, R.L.: Markov Chains and Stochastic Stability. Springer, Berlin (1993)
25. Meyn, S.P.: Control Techniques for Complex Networks. Cambridge University Press, Cambridge (2008)
26. Nazarathy, Y., Weiss, G.: Near optimal control of queueing networks over a finite time horizon. Ann. Oper. Res. **170**, 233–249 (2009)
27. Nazarathy, Y., Weiss, G.: Positive Harris recurrence and diffusion scale analysis of a Push–Pull queueing network. Perform. Eval. (2009, to appear). Preliminary version presented at Valuetools 2008
28. Rybko, A.N., Stolyar, A.L.: Ergodicity of stochastic processes describing the operation of open queueing networks. Probl. Inf. Transm. **28**, 199–220 (1992)
29. Tassiulas, L.: Adaptive back-pressure congestion control based on local information. IEEE Trans. Automat. Contr. **40**, 236–250 (1995)
30. Tassiulas, L., Bhattacharya, P.B.: Allocation of interdependent resources for maximal throughput. Stoch. Models **16**, 27–48 (2000)
31. Taylor, L.M., Williams, R.J.: Existence and uniqueness of semimartingale reflecting Brownian motions in an orthant. Probab. Theory Rel. Fields **96**, 283–317 (1993)
32. Weiss, G.: Jackson networks with unlimited supply of work. J. Appl. Probab. **42**, 879–882 (2005)