Routing policies for a partially observable two-server queueing system

Wendy Ellens Péter Kovács* Rudesindo Núñez-Queija Hans van den Berg
University of Amsterdam, Korteweg-de Vries Institute for Mathematics TNO / University of Twente
{w.ellens,p.kovacs,r.nunezqueija}@uva.nl j.l.vandenberg@tno.nl

ABSTRACT

We consider a queueing system controlled by decisions based on partial state information. The motivation for this work stems from road traffic, in which drivers may, or may not, be subscribed to a smartphone application for dynamic route planning.

Our model consists of two queues with independent exponential service times, serving two types of jobs. Arrivals occur according to a Poisson process; a fraction α of the jobs (type X) is observable and controllable. At all times the number of X jobs in each queue and their individual positions are known. Upon its arrival a router decides which queue the next X job should join. Y jobs (fraction $1-\alpha$) are non-observable and non-controllable. They randomly join a queue according to some static routing probability.

We address the following main research questions: 1) what penetration level α is needed for effective control, 2) which policy should be implemented at the router, and 3) what is the added value of having more system information (e.g., average service times)? An extensive simulation study reveals that for heavily loaded systems a low penetration level suffices and that the performance (in terms of the average sojourn time) of a simple policy that relies on little system information is close to w-JSQ (weighted join-the-shortest-queue policy) which is optimal in a fully controllable and observable system. The latter result is confirmed by the analysis of deterministic fluid models that approximate the stochastic evolution under large loads.

CCS Concepts

•**Mathematics of computing** → **Queueing theory;** Mathematical analysis; •**Networks** → **Network performance analysis;** •**Computing methodologies** → *Modeling and simulation;* •**Applied computing** → *Transportation;*

Keywords

Routing, tandem queue, dynamic control, incomplete information, partial control, fluid approximation

1. INTRODUCTION

In this paper we consider a queueing system consisting of two queues and a controller that assigns newly arriving jobs to the queues. A special feature of the system is that only part of the jobs can be observed and controlled, whereas the other jobs cannot be observed explicitly and choose a queue themselves. Thus, the controller has to perform its routing task (aiming at minimising the system delay) based on partial information about the system state. Our primary motivation for studying such a queueing system stems from dynamic road traffic control and on-line navigation systems based on smartphone applications, see e.g. [20]. It is not obvious how to deal in an effective way with the partial observability and controllability of these systems, and how the performance depends on the penetration grade of the application (that is, the fraction of traffic that can be observed).

The aim of the present study is to provide insight into this type of questions by intensively investigating and comparing the performance of various dynamic *routing* strategies for a simplified queueing system. Through extensive simulations we show that heuristic routing policies work quite well even for low penetration levels. Their performance (in terms of the average sojourn time) hardly improves further if the penetration level is increased beyond, say, 25%. The performance of these simple policies that rely on little system information appears to be close to the weighted join-the-shortest-queue (w-JSQ) policy, particularly so for heavily loaded systems, which is optimal in a fully observable and controllable system [7, 21]. This result is confirmed by the outcome of an approximative analytical study. In particular, assuming a highly loaded system, we show that the partial information available for the fluid models of the heuristic policies is enough to provide dynamics that give just as good performance as the fluid approximation of the optimal w-JSQ policy. We conjecture that in the limit the policies give identical trajectories.

To the best of our knowledge our approach to partial observability/controllability in queueing systems is novel. Kuri and Kumar [9] and Mitzenmacher [12], for example, discuss optimal routing for the case that delayed information on the queue lengths is available. In Guo et al. [5] the partial system information available concerns the service time distribution.

*Corresponding author

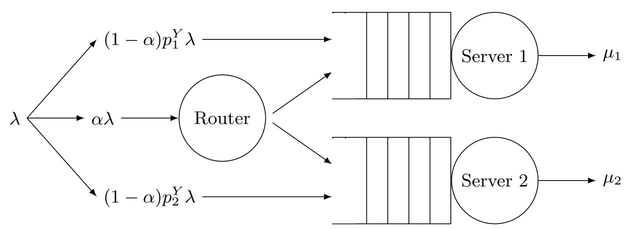


Figure 1: A schematic representation of the queueing system

Other papers, see e.g. [1], consider controllable (foreground) and uncontrollable (background) traffic, but there it is assumed that both traffic types are observable (i.e. the routing of foreground traffic is based on total state information).

When specifically considering road traffic applications, there is an extensive literature on using probe vehicles (which are the observable/controllable part of the traffic) for traffic control. Also field experiments have been conducted, see e.g. [6]. The applications vary from estimations on queue lengths at traffic signals [4], to freeway travel-time prediction [3, 13] and detecting incidents [16]. The number of probe vehicles and the penetration level needed for these estimations to work well are also widely considered [2, 10, 17, 19]. Furthermore, there are systems in place that collect and process data like Vtrack [18] or CroTIS [15], and route guidance applications based on such data, like Waze [20].

The remainder of this paper is organised as follows. First, in Section 2, we give a detailed description of the system model at hand, including the notation, and introduce and discuss the routing policies considered in this paper. Next, in Section 3, these policies are extensively evaluated and compared by simulation. The analytical approach based on fluid approximations, especially useful for highly loaded systems, is provided in Section 4. Finally, in Section 5, we summarise the results of our study, provide conclusions and suggest directions for further research.

2. MODEL

2.1 The queueing system

We consider a two-server queueing system, as depicted in Figure 1. The queues operate independently in a FIFO-manner with exponential service times of rates $\mu = (\mu_1, \mu_2)$. Jobs arrive at the queueing system according to a Poisson process with rate λ and are of one of the following two types.

Jobs of the first type, called X jobs, are controllable and observable. They make up for a fraction α of the total load. X jobs thus arrive according to a Poisson process of rate $\alpha\lambda$. The parameter α is called the *penetration level.* X jobs are *controllable*, because upon their arrival a router decides on which queue they should join. A *policy* is a set of rules according to which the routing decisions for X jobs are taken. Saying that X jobs are *observable*, we mean that at all times their numbers in both queues (possibly including a job that is in service) and their individual positions are known. The numbers of X jobs are denoted by $X(t) = (X_1(t), X_2(t))$. The position of a job in a queue is the number of jobs that must be served before finishing the given job, thus a job currently in service is at position 1. We are especially interested in the positions of the last X jobs in the queue, which will be denoted by $L^X(t) = (L_1^X(t), L_2^X(t))$. Should there be no X jobs present in either queue, we set the corresponding component(s) of L^X to 0.

The second type of jobs, Y jobs, represent background traffic. They are non-controllable and non-observable. They arrive according to a Poisson-process of rate $(1-\alpha)\lambda$ and join one of the two queues according to the static probabilities $p^Y = (p_1^Y, p_2^Y)$, where $p_2^Y = 1 - p_1^Y$. For the router the number of Y jobs in each queue, which will be denoted by $Y(t) = (Y_1(t), Y_2(t))$, is unknown, thus leaving the total queue lengths, denoted by $Q(t) = (Q_1(t), Q_2(t))$, with $Q_i(t) = X_i(t) + Y_i(t)$, for $i = 1, 2$, unknown as well.

The loads on the queues depend on the routing policy and the penetration level. So as to conduct comparisons for α ranging from 0 to 1, we will measure the loads in each of the queues by the actual loads when $\alpha = 0$. For queue i this is given by $\rho_i^Y = p_i^Y\lambda/\mu_i$. When $\rho_1^Y = \rho_2^Y$ we use ρ^Y to denote either of them. For any choice of parameters for which the loads in both queues are below 1, it will thus be possible for the router to maintain the queue stable for all $\alpha \in [0, 1]$.

2.2 Routing policies

In order to route incoming X jobs to a queue, one can think of two types of policies: *dynamic policies* that use state information and *static policies* that do not. We consider the four dynamic routing policies introduced below.

- *Number of X jobs* (denoted by #X): this policy sends an arriving X job to the queue with the fewest X jobs (including a job currently in service).
- *Last X position* (abbreviated to LXP) compares the queues by the position of the last X job and sends an arriving X job to the queue with the lowest last X position.
- *Weighted last X position* (w-LXP) multiplies the last X position by the mean service time $1/\mu_i$ (giving an estimation of the sojourn time for queue i) and sends an arriving X job to the lowest weighted last X position.
- *Estimated weighted last X position* (ew-LXP) does the same as w-LXP except that it does not assume μ to be known. The mean service time is estimated by dividing the sojourn times of the X jobs lastly departed from queue i by their arrival position. The average is over the last w number of departed X costumers, with w a parameter of the policy.

For most of these policies no system information (e.g. the values of α, λ, p^Y and μ)) is assumed to be known by the router. Only in w-LXP μ is used. The first two policies are designed for symmetric queues (equal service rates), the last two for the asymmetric case.

In Section 3 we compare the performance of the above mentioned policies with respect to the expected sojourn time of the jobs (the total time of a job in the system, also called response time). Let S_i denote the expected sojourn time in queue i and S_i^X the expected sojourn time in queue i over X jobs only. We use the same symbols for the average sojourn times in the simulations. The averages over both queues are denoted by S and S^X for all, and for X jobs only, respectively. In the policy comparison we also consider the following two reference policies:

- *Weighted join-the-shortest-queue* (w-JSQ): a dynamic policy that sends an arriving X job to the queue with the fewest jobs (X and Y jobs together). In case of unequal service times, the number of jobs is weighted by the average service time, as for w-LXP. This reference policy assumes full state information (that is, also Y jobs are observable) and knowledge of μ, but no other system information. Since w-JSQ uses more information than our policies, it will likely outperform our policies. Note that it may not be optimal for our partially controllable system, although it is optimal for fully controllable systems [7, 21].

- A *static policy* that applies probabilistic routing. It sends arriving X jobs to queue i with a fixed probability p_i^X. The probabilities p_i^X are set such that the difference between S_1^X and S_2^X is minimised[2]. Note that the policy has the same goal as the policies discussed before: it aims at equalising the sojourn times in both queues. It can be verified that this policy is not the same as the static policy that minimises S. Although the static policy uses no state information, it may not always be outperformed by the dynamic policies, because the static policy uses system parameters $(\alpha, \lambda, \mu, p^Y)$ that are not (all) available to the dynamic policies.

Although all six policies introduced above assume partial controllability (i.e., only X cars can be routed), they differ in terms of the system and state information that is used. In Section 3 we compare their respective performance.

3. SIMULATIONS

In this section we investigate whether the dynamic policies with partial observability and controllability described in Section 2.2 perform better than the static policy and how close their performance (in terms of average sojourn times) is to that of w-JSQ. Furthermore, we want to assess the added value of using extra state and/or system information and how the performance depends on the penetration level α. To this end, we have performed many simulations in MATLAB [11] under various system settings with symmetric and asymmetric service times and background traffic, and different loads and penetration levels α.

3.1 Policy comparison

3.1.1 Symmetric queues

We start by considering the case of equal service rates, $\mu_1 = \mu_2$, see Figure 2 for an example with $\mu_1 = \mu_2 = 1$. We compare the two simple dynamic policies based on the numbers of X jobs (#X) and the positions of the last X jobs (LXP) in the two queues, with the static policy and the full state information policy JSQ. Every data point in this plot, as well as in the following plots, is the result of a simulation experiment with 10 million events (job arrivals and departures, that is).

The performance of LXP is always slightly better than that of #X. This can be explained by the fact that LXP also takes into account the Y jobs that have arrived before the last X job, while #X only measures the number of

[2]Simple calculations yield that this is achieved for $p_1^X = (\mu_1 - \mu_2 + \lambda - 2p_1^Y(1-\alpha)\lambda)/(2\alpha\lambda)$, where S_1^X and S_2^X become equal, or $p_i^X \in \{0,1\}$ if it is impossible to reach the desired equality.

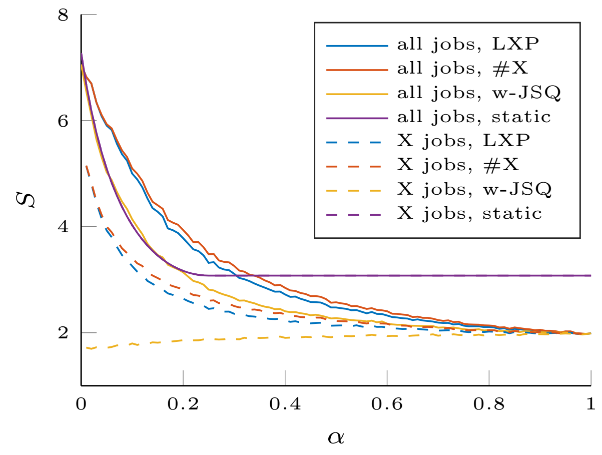


Figure 2: Average sojourn time over both queues as a function of α for several policies, $\rho_1^Y = 0.45$, $\rho_2^Y = 0.9$, $\mu = (1,1)$ and $p^Y = (1/3, 2/3)$. We display both the average over all jobs (solid), and over X jobs only (dashed).

X jobs. The relative difference between the two policies is largest for moderate and somewhat high loads. Under very high loads ($\rho^Y \uparrow 1$) the dynamic policies show comparable performance. In comparison the performance of the static policy is clearly inferior. However, for low α it is better for the overall average sojourn time to apply the static policy. Note that the dynamic policies are always better for the X jobs, and for all jobs when the control level α is relatively large.

3.1.2 Asymmetric queues

We now allow for unequal service rates for the two queues describing the results of a similar set of experiments as in Section 3.1.1. In this case it is natural to use a weighted dynamic policy that takes the service rates into account. We can observe that (non-weighted) LXP also gives better results than #X for asymmetric queues (see Figure 3). Therefore we focus on LXP from now on, and consider its weighted versions w-LXP and ew-LXP. In the former the weights are given, in the latter they are estimated using data of the w lastly departed jobs for each queue separately. We refer to w as the *window size* of ew-LXP and show results for values $w = 1$ and $w = 10$ in our graphs (denoted by ew-LXP(1) and ew-LXP(10) in the legends).

From the simulations in Figure 3 we see that, as expected, the weighted policies (w-LXP and ew-LXP) outperform the unweighted LXP. This turns out to be true across all values of α as can be observed in Figure 4. For practically relevant values of α (that is, relatively small values) this even holds for a small window size of $w = 1$.

In practice the mean service times may vary over time. In this case the policy ew-LXP is particularly relevant. The recommended window size depends on the desired precision and on how fast μ is changing; the memory of the policy should refresh itself on a smaller time scale than the one on which μ changes. For a static μ, a window size $w = 10$ gives relatively accurate results as can be seen in Figure 3. In general, in order to get the same precision, w needs to be larger for lower loads (see Figure 3), because the queue lengths are smaller and therefore the average is taken over

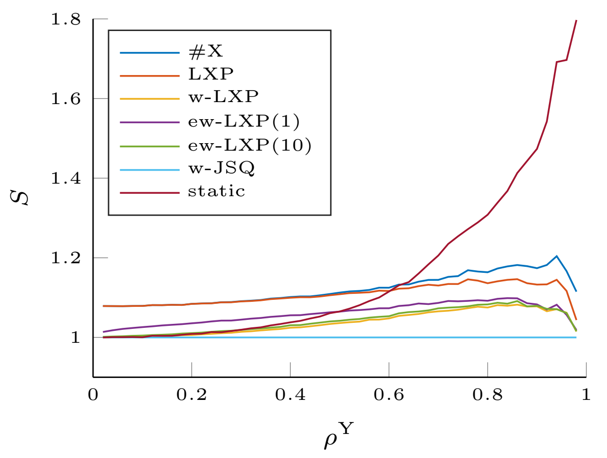


Figure 3: Relative average sojourn time over all jobs and both queues as a function of ρ^Y for several policies, $\alpha = 0.2$, $\mu = (1, 2)$ and $p^Y = (1/3, 2/3)$. The sojourn times are relative to those of w-JSQ.

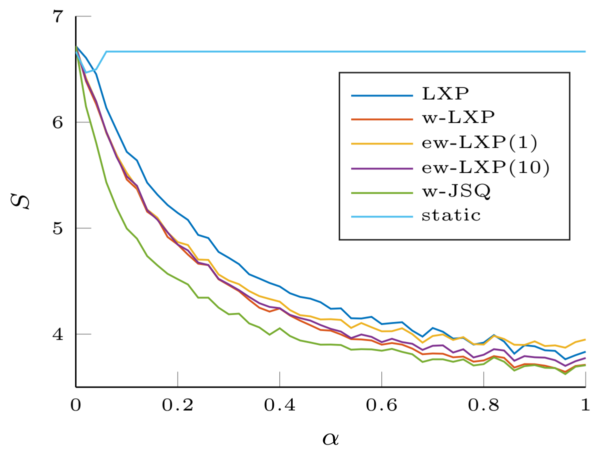


Figure 4: Average sojourn time over all jobs and both queues as a function of α for several policies, $\rho_1^Y = \rho_2^Y = 0.9$, $\mu = (1, 2)$ and $p^Y = (1/3, 2/3)$

a smaller number of jobs. In addition, we observe in Figure 4 that w needs to be larger for higher penetration levels (because the estimates of the average service time are using more overlapping data when there are fewer Y jobs in the system).

The figures and observations discussed in this section refer to the case in which both queues are equally loaded by the Y jobs (that is, $\rho_1^Y = \rho_2^Y$). Note that this does not imply that the queues are symmetric, as the difference between the μ_i can be compensated by the p_i^Y. The observations turned out (in simulations not presented here) to be valid also for unequally loaded queues (e.g. if p^Y is symmetric, while μ is not). We noticed, however, that then the static policy outperforms our dynamic policies for low values of α (as we also observed in Section 3.1.1). So in that case the dynamic policies are only recommended if it is not possible to implement the static policy (because of a lack of system information). For the equally loaded scenario we have seen in Figure 3 that the weighted policies perform much better

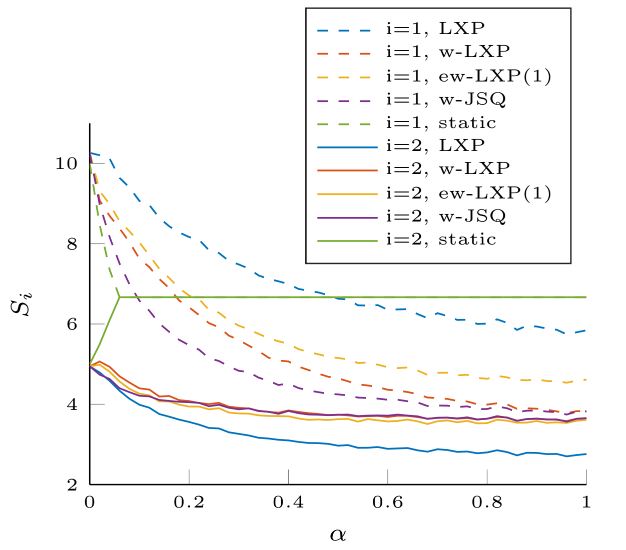


Figure 5: Average (over all jobs) sojourn time as a function of α for both queues separately, $\rho_1^Y = \rho_2^Y = 0.9$, $\mu = (1, 2)$, and $p^Y = (1/3, 2/3)$

than the static policy and that the w-LXP policy is close to the optimal w-JSQ when ρ^Y approaches 1. This fact is explained in detail in Section 4.

3.2 The penetration level

One of the questions we raised in Section 1 was to determine the smallest penetration level α needed for effective control. To shed light on this question, we now study the performance of the various policies as a function of α. From the figures presented in the preceding sections (Figures 2 and 4) we can conclude that the sensitivity of the performance to the penetration level is highest for small values of α (i.e. changes in α cause a larger change in the performance when α is small than when it is large), which can be understood as a “law of diminishing marginal returns”. In most of the scenarios, the average sojourn time for w-LXP and ew-LXP is already rather close to its minimum if α is approximately 25%. The required penetration level depends on the system load; for lower loads a higher α is required and if the load caused by the Y jobs is symmetric, a lower α suffices than if it is asymmetric (because more X jobs are needed to compensate the ‘wrong’ choices of the Y jobs).

First, let us inspect the average sojourn times S_1, S_2 for all jobs (X and Y together). From Figure 5, which is representative for a large collection of graphs for a variety of scenarios, we see that for the dynamic policies the average sojourn times in both queues decrease when α increases. This means that increasing the level of control decreases the average time a job spends in the system, independently of the queue a job is sent to. In contrast, for the static policy the sojourn time in one queue increases when more jobs are being controlled (while the sojourn time in the other decreases as a function of α). As one can notice from Figure 4, this may result in non-monotonous average sojourn times over both queues. However, for the dynamic policies it is always beneficial to the overall average sojourn time to increase the amount of control.

If we consider X jobs only, first we should note that the performance for X jobs is always better than for an average (X or Y) job. This is essential if jobs can choose themselves whether to be in the X class (as in the road traffic example). For the static policy, the behaviour as a function of α is the same as before (when looking at both classes together). However, for w-JSQ the quantities S_1^X and S_2^X are now both *increasing* in α. For this policy, increasing the percentage of controlled jobs improves the overall performance, but deteriorates the performance for the controlled jobs, because the advantage of being directed to the "right" queue becomes smaller if more jobs are being directed to it. This reasoning does not hold for the other dynamic policies, because for those policies increasing α does not only mean more control (which is disadvantageous for the controlled jobs), but also more information (which is advantageous). Consequently, the dynamic policies may show non-monotonous behaviour for one of the queues.

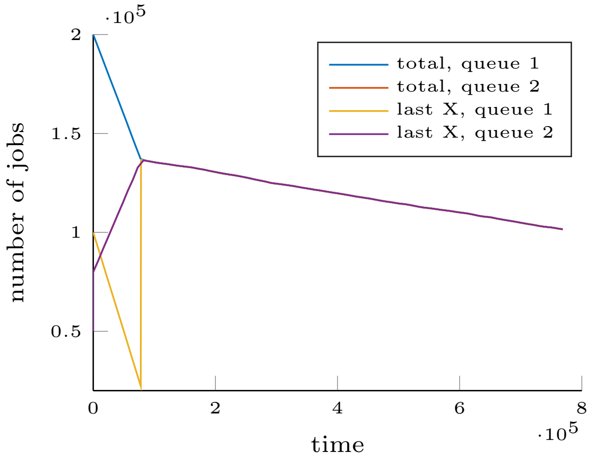
4. FLUID MODEL APPROXIMATION

The policies that we have proposed and studied through simulation do not allow exact (analytical) performance analysis. In this section we approximate the behaviour of some of the policies under high loads using deterministic fluid models. For the deterministic fluid models we show that the partial information exploited by w-LXP is sufficient to give dynamics that yield just as good a performance as the fluid approximation of the "optimal" w-JSQ policy. This result confirms the (in Section 3.1.2 observed) convergence of w-LXP to w-JSQ for loads approaching 1. A fluid approximation can often be shown to be the limiting process (*fluid limit*) of the original stochastic system under an appropriate scaling [14]. Here we do not aim at a technical proof of this limiting procedure, but rather at the analysis of the fluid approximations themselves. Instead of a technical proof, we provide a numerical validation study.

At a high level of abstraction, the fluid approximation may be seen to mimic the dynamics of the stochastic process at *large* system states; i.e., when the queueing processes move far away from the origin. Consequently, when the system load is low the applicability is limited to *transient* performance analysis, because large system states are rarely visited and therefore the fluid approximation does not reflect the typical behaviour of the process. When the system load is high, however, the stochastic process *does* typically move far away from the origin and the analysis of the fluid approximation provides valuable insight into the *stationary* behaviour of the stochastic process as well.

4.1 High-load conditions and fluid limits

Since Little's Law applies to the queue lengths and the sojourn times irrespective of the policy, it is sufficient to concentrate on queue length dynamics to obtain the expectation of the sojourn time (system delay). Clearly, under all of the policies presented in Section 2.2 it is possible to describe the joint queue length processes as a Markov process if the state descriptor contains both queue lengths, *and* the positions of all jobs, of both types, *plus* some information about the past. Note that indeed, for some policies, there is a dependence of the dynamics on the positions of X jobs, and/or the current value of a parameter, which is an estimate based on past system states. Therefore, the state space can be rather involved in general. We focus our analysis on the w-JSQ and w-LXP policies. For these policies



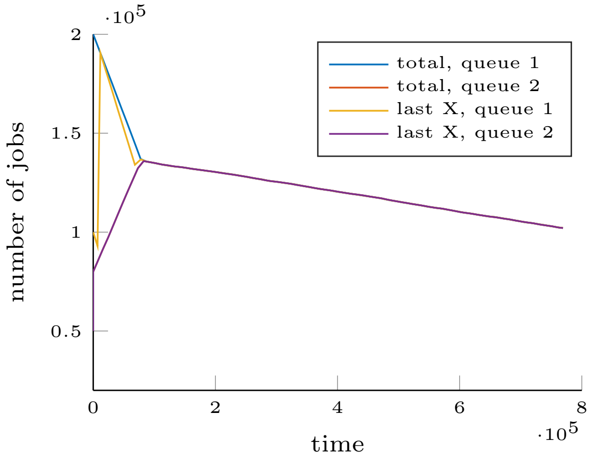


Figure 6: Simulated evolution starting from high initial queue sizes for JSQ (top) and LXP (bottom). In both figures the (red) curve for the number of (X plus Y) jobs in queue 2 and the (purple) curve for the position of the last X job in this queue coincide. After an initial period all four curves coincide.

the random vector $M(t) = \big(Q_1(t), Q_2(t), L_1^X(t), L_2^X(t)\big)$ is Markovian. Let the Markov processes for the two policies be denoted by $M^{\text{w-JSQ}}$ and $M^{\text{w-LXP}}$ respectively.

The condition

$$\lambda/(\mu_1 + \mu_2) < 1, \tag{1}$$

is obviously necessary for these processes to be stable. In addition, we assume α to be sufficiently large, such that with appropriate routing, it is possible to keep both queues from getting overloaded. Besides (1), we therefore require the load of Y jobs on each queue to be below 1, i.e.,

$$p_i^Y (1-\alpha)\lambda < \mu_i, \tag{2}$$

for both $i = 1, 2$. The system is said to be in *heavy traffic* if the boundaries of condition (1) are approached, i.e. if $\lambda \uparrow \mu_1 + \mu_2$. If we impose that the system reaches heavy traffic, while satisfying (2) for both queues, we need

$$\alpha \geq 1 - \frac{1}{p_i^Y} \frac{\mu_i}{\mu_1 + \mu_2}, \tag{3}$$

for both $i = 1, 2$. In particular, if α is 0, it must be that $p_i^Y \lambda \uparrow \mu_i$ for both queues simultaneously, so that $\mu_2 p_1^Y =$

$\mu_1 p_2^Y$.

Although we do not consider heavy-traffic limits, we assume that the system load is high, that is, λ is close to $\mu_1 + \mu_2$. Under such high-load conditions, the number of jobs present in either queue will typically be very large and any *substantial* change in the queue lengths requires such a long time that the arrivals of both types virtually occur in a continuous fashion. This implies that in at least one of the two queues, there must be an X job near the end of the queue. More precisely, for at least one of the two queues, the number of Y jobs standing behind the last X job in line is negligible compared to the total number of jobs in the queue. This intuitive reasoning forms the rationale behind the fluid approximations proposed in Section 4.2. We conjecture that these (deterministic) fluid processes are the fluid limits of the original (stochastic) processes.

To facilitate the discussion, we end this section with the definition of a fluid limit, which formally describes the chosen scaling. If $\{M^{(c)}(t), t \geq 0\}_{c\in\mathbb{N}}$ is a sequence of Markov processes with $\|M^{(c)}(0)\|_1 = c$, then we define

$$\bar{M}^{(c)}(t) = \frac{M^{(c)}(ct)}{c}.$$

The *fluid limit* of this sequence is obtained by letting $c \to \infty$. Under suitable conditions, the limit often turns out to be a deterministic fluid process [14]. We believe (and demonstrate numerically) that this is also the case for the processes $M^{\text{w-JSQ}}$ and $M^{\text{w-LXP}}$.

4.2 Description and analysis of the fluid model

We now propose deterministic fluid processes $m^{\text{w-JSQ}}$ and $m^{\text{w-LXP}}$ to approximate the stochastic processes $M^{\text{w-JSQ}}$ and $M^{\text{w-LXP}}$ respectively, under large load conditions. Each of these processes consists of four components and, similar to our notation before, we generically write $m(t) = (q_1(t), q_2(t), l_1(t), l_2(t))$.

For w-JSQ the evolution satisfies the following system of ordinary differential equations (ODEs). For conciseness we only report the derivatives for positive q_i, and l_i; at level zero the negative term in the derivative is to be removed, since there are no departures in this case.

$$\frac{\mathrm{d}}{\mathrm{d}t}q_1^{\text{w-JSQ}} = \begin{cases} \alpha\lambda + p_1^Y(1-\alpha)\lambda - \mu_1, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 < q_2^{\text{w-JSQ}}/\mu_2, \\ \alpha\lambda\mu_1/(\mu_1+\mu_2) + p_1^Y(1-\alpha)\lambda - \mu_1, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 = q_2^{\text{w-JSQ}}/\mu_2, \\ p_1^Y(1-\alpha)\lambda - \mu_1, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 > q_2^{\text{w-JSQ}}/\mu_2, \end{cases}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}q_2^{\text{w-JSQ}} = \begin{cases} p_2^Y(1-\alpha)\lambda - \mu_2, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 < q_2^{\text{w-JSQ}}/\mu_2, \\ \alpha\lambda\mu_2/(\mu_1+\mu_2) + p_2^Y(1-\alpha)\lambda - \mu_2, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 = q_2^{\text{w-JSQ}}/\mu_2, \\ \alpha\lambda + p_2^Y(1-\alpha)\lambda - \mu_2, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 > q_2^{\text{w-JSQ}}/\mu_2, \end{cases}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}l_1^{\text{w-JSQ}} = \begin{cases} \alpha\lambda + p_1^Y(1-\alpha)\lambda - \mu_1, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 < q_2^{\text{w-JSQ}}/\mu_2, \\ -\mu_1, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 > q_2^{\text{w-JSQ}}/\mu_2, \end{cases}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}l_2^{\text{w-JSQ}} = \begin{cases} -\mu_2, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 < q_2^{\text{w-JSQ}}/\mu_2, \\ \alpha\lambda + p_2^Y(1-\alpha)\lambda - \mu_2, & \text{if } q_1^{\text{w-JSQ}}/\mu_1 > q_2^{\text{w-JSQ}}/\mu_2. \end{cases}$$

These equations reflect that newly arriving X jobs are routed to the weighted shortest queue. Note that the policy's rule in case the weighted queue lengths are equal is irrelevant, because any tie breaking rule forces the process to move along states with $q_1^{\text{w-JSQ}}/\mu_1 = q_2^{\text{w-JSQ}}/\mu_2$. It is worth emphasising that the last-X component of the weighted shortest queue also contains the effect of Y arrivals: both types of arrivals occur in a continuous fashion and are interleaved at the weighted shortest queue. Newly arriving Y jobs thus push the last X position further back.

The above ODE lacks a description for the behaviour of the components $l_i^{\text{w-JSQ}}$ when the process hits the hyperplane $q_1^{\text{w-JSQ}}/\mu_1 = q_2^{\text{w-JSQ}}/\mu_2$. We argued in Section 4.1 that for one of the two queues the last X position must be (almost) equal to the queue length. For w-JSQ this is the case for the weighted shortest queue. Indeed, new X jobs are routed to the weighted shortest queue and the number of Y jobs arriving in between two arriving X jobs is negligible compared to the queue lengths. As a consequence, whenever the hyperplane $q_1^{\text{w-JSQ}}/\mu_1 = q_2^{\text{w-JSQ}}/\mu_2$ is hit, the last X position of the largest weighted queue also moves to the level of the corresponding queue length. Therefore we have to supplement the above ODEs with the following jumps:

$$l_i^{\text{w-JSQ}}(t+) = q_i^{\text{w-JSQ}}(t), \text{ for both } i, \text{ if } q_1^{\text{w-JSQ}}/\mu_1(t) = q_2^{\text{w-JSQ}}/\mu_2(t).$$

Here, by $t+$ we mean immediately after time t.

We now turn our attention to an approximating deterministic fluid system for w-LXP. Much of the discussion above holds in this case as well. The difference is due to the fact that the router now chooses the queue with the lowest weighted last X position, and only switches to the other queue when the weighted last X positions are equal. Thus the ODEs characterising the evolution of $m^{\text{w-LXP}}$ are — similarly to $m^{\text{w-JSQ}}$ — as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}q_1^{\text{w-LXP}} = \begin{cases} \alpha\lambda + p_1^Y(1-\alpha)\lambda - \mu_1, & \text{if } l_1^{\text{w-LXP}}/\mu_1 < l_2^{\text{w-LXP}}/\mu_2, \\ \alpha\lambda\mu_1/(\mu_1+\mu_2) + p_1^Y(1-\alpha)\lambda - \mu_1, & \text{if } l_1^{\text{w-LXP}}/\mu_1 = l_2^{\text{w-LXP}}/\mu_2, \\ p_1^Y(1-\alpha)\lambda - \mu_1, & \text{if } l_1^{\text{w-LXP}}/\mu_1 > l_2^{\text{w-LXP}}/\mu_2, \end{cases}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}q_2^{\text{w-LXP}} = \begin{cases} p_2^Y(1-\alpha)\lambda - \mu_2, & \text{if } l_1^{\text{w-LXP}}/\mu_1 < l_2^{\text{w-LXP}}/\mu_2, \\ \alpha\lambda\mu_2/(\mu_1+\mu_2) + p_2^Y(1-\alpha)\lambda - \mu_2, & \text{if } l_1^{\text{w-LXP}}/\mu_1 = l_2^{\text{w-LXP}}/\mu_2, \\ \alpha\lambda + p_2^Y(1-\alpha)\lambda - \mu_2, & \text{if } l_1^{\text{w-LXP}}/\mu_1 > l_2^{\text{w-LXP}}/\mu_2, \end{cases}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}l_1^{\text{w-LXP}} = \begin{cases} \alpha\lambda + p_1^Y(1-\alpha)\lambda - \mu_1, & \text{if } l_1^{\text{w-LXP}}/\mu_1 < l_2^{\text{w-LXP}}/\mu_2, \\ -\mu_1, & \text{if } l_1^{\text{w-LXP}}/\mu_1 > l_2^{\text{w-LXP}}/\mu_2, \end{cases}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}l_2^{\text{w-LXP}} = \begin{cases} -\mu_2, & \text{if } l_1^{\text{w-LXP}}/\mu_1 < l_2^{\text{w-LXP}}/\mu_2, \\ \alpha\lambda + p_2^Y(1-\alpha)\lambda - \mu_2, & \text{if } l_1^{\text{w-LXP}}/\mu_1 > l_2^{\text{w-LXP}}/\mu_2. \end{cases}$$

As before, if queue j has the lowest weighted last X position at time t, we must have $l_j^{\text{w-LXP}}(t) = q_j^{\text{w-LXP}}(t)$. The jumps on the switching plane are now given by

$$l_i^{\text{w-LXP}}(t+) = q_i^{\text{w-LXP}}(t), \text{ for both } i, \text{ if } l_1^{\text{w-LXP}}/\mu_1(t) = l_2^{\text{w-LXP}}/\mu_2(t).$$

After a transient period the deterministic fluid approximations for both policies w-JSQ and w-LXP live on their switching planes, where respectively $q_1/\mu_1 = q_2/\mu_2$ and $l_1/\mu_1 = l_2/\mu_2$. On the switching planes, jobs are sent to both queues, so that the last X positions are updated continuously and are equal to the queue sizes. It follows that (after a transient period) both fluid processes live on the same line ($q_1/\mu_1 = l_1/\mu_1 = q_2/\mu_2 = l_2/\mu_2$) and take the same decisions, both equalising the load of the two queues.

4.3 Numerical verification of the fluid approximations

We have conducted simulations to verify the suitability of the deterministic fluid approximations for large system loads. When the service rates are different, we observe the same behaviour as with equal μ's (after applying a correction for the different service rates). Thus we limit ourselves here to the symmetric case $\mu_1 = \mu_2$. We set the initial queue lengths and last X positions intentionally away from the switching curve, whilst choosing α such that it satisfies (3).

In our first set of experiments (see Figure 6) we illustrate the appropriateness of the linear deterministic approximations by plotting the trajectories over a very long time horizon and at very large system states. We only plot the simulation curves, as the deterministic approximations would be indistinguishable from them in the graphs. In these experiments we fixed $\alpha = 0.8$, $\mu = (1, 1)$, $p^Y = (1/2, 1/2)$ and $\rho^Y = \rho_1^Y = \rho_2^Y = 0.95$.

First we observe the plot of the trajectories of the queueing processes under JSQ. As long as the two queues have different sizes, all jobs are routed to the shortest queue (queue 2 in this experiment); the last X position in queue 2 is therefore equal to the queue length in queue 2. For queue 1 we see that the last X position decreases faster than the queue length, since new Y jobs are all placed *after* the last X job. The position of the last X job may hit zero and remain at zero until the two queues meet. At that point the last X position in queue 1 increases instantly to become equal to the queue length and from then on both queue lengths *and* the last X positions remain coupled forever.

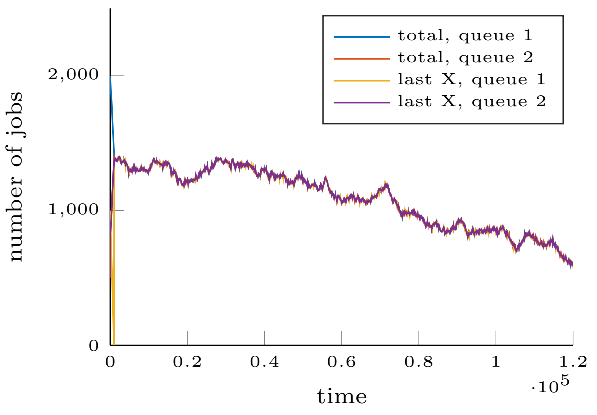
Now we turn to the results under LXP. Again we start off at a very large state for all components, with queue 2 having the lowest queue length *and* the lowest last X-position (these are equal), i.e., the router is sending jobs to queue 2. As soon as the last X position of queue 1 has dropped to the level of that of queue 2, it is increased to the size of queue 1. After a single X job has been sent to queue 1, the next X jobs are being sent to queue 2 again. This pattern repeats until the sizes of the two queues meet, from that point on all four components will have one single value. Note that, since the curve for the last X position in queue 1 can only bounce at a countable number of points, basically all X jobs have been routed to queue 2, as was the case for JSQ. The point where the two queues meet is therefore exactly the same for both policies.

Figure 6 considers extremely large system states, which will only rarely be visited. In Figure 7 we have plotted the trajectories for JSQ and LXP under a load as high as 0.995 and for system states that are more likely to be reached. Other parameters were set as $\alpha = 0.6$, $\mu = (1, 1)$, and $p^Y = (3/5, 2/5)$. We see that the trajectories meet quickly, after which the joint trajectories are well approximated by a linear trend.

Our numerical experiments demonstrate that indeed the proposed fluid limit approximations closely follow the trajectories of the w-JSQ and w-LXP policies at large system states under high loads. The simulations also corroborate our observations in Section 3, where we noticed that in heavy traffic w-LXP performs comparable to w-JSQ.

5. CONCLUSION

We have investigated routing policies for a multiserver queueing system based on partial state information. We took a novel approach in applying the routing control decisions only to a portion of the incoming jobs, as we assumed that the router has control over just part of the traffic. Routing traffic based on partial state information is an important challenge in road traffic control by smartphone applications and on-line navigation systems, because these applications only have access to the position of their own users. Inspired by this application, we considered a two-server queueing system in which part of the jobs (the X jobs) can be observed and controlled (routed to one of the queues), while the other jobs (the Y jobs) act as background traffic which is neither observable nor controllable. We analysed the performance of



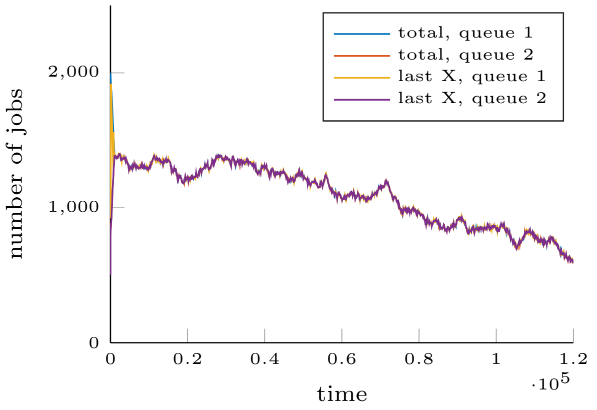


Figure 7: Simulated evolution under a high load for JSQ (top) and LXP (bottom). All curves coincide after a very brief initial period.

our policies (in terms of the average sojourn time) under the assumption of Poisson arrivals and independent exponential service times. We compared the results to join-the-shortest-queue (JSQ) and to a static policy that routes according to fixed routing probabilities using no state information, but full information about the system parameters.

An extensive simulation study revealed that the sojourn time as a function of the penetration level (i.e. the percentage of X jobs) declines fast for small penetration levels, but slowly for large levels. As a consequence, with only a small percentage of controllable jobs it is possible to obtain an average sojourn time which is close to the minimum value. We also observed in the simulations that a simple policy that sends arriving X jobs to the queue with the fewest X jobs performs quite well. If both queues are at least moderately loaded, it is much better than the static policy. A policy that does not base its decisions on the number of X jobs in each of the queues, but on the position of the last X job (that is, its distance to the server in number of jobs) performs even slightly better. Obviously, if the service rates of the two queues are unequal, the last X positions of the queues need to be weighted by their average service times. If this average is unknown, or varying over time, an estimate of it based on the measured sojourn time of only a few recently departed X jobs yields almost as good a performance.

For high loads, the performance of the weighted last X position policy (w-LXP), which uses partial information, is close to the performance of weighted JSQ (w-JSQ), the optimal policy under full information. When the load of both queues converges to 1, the performance of w-LXP (without knowledge of Y jobs in the system) is identical to that of w-JSQ (with full knowledge of Y jobs in the system). We have investigated this remarkable result analytically by means of deterministic dynamical systems approximating the stochastic processes at large system states under high loads. We conjecture the proposed dynamical systems to be the *fluid limits* of the original processes. We have shown that after a transitory phase the dynamical systems corresponding to w-LXP and w-JSQ have identical trajectories for the queue lengths, whatever the initial conditions are. This explains the similarity of the sojourn times under both policies. To justify the analysis of these dynamical systems we performed simulations which confirmed the convergence of the original stochastic processes to the proposed deterministic dynamical systems.

Note that the routing policies proposed and investigated in the present paper for the special case of two partially observable queues can easily be extended to similar systems with more than two queues. This can serve as a model of a road network with multiple possible paths for instance.

Further research on the topic could take various directions. One possibility is to relax the modelling assumptions, by investigating the performance of the proposed routing policies under non-exponential service times at the queues. We would expect similarly good results in that case (particularly in heavy traffic), but the estimation of μ for the ew-LXP policy could be less accurate. Other relaxations or changes in the model could concern non-stationary inputs, state-dependent service rates or service disciplines that are different from FIFO. Other routing policies could also be considered, for instance policies that take into account more system information, such as the arrival rate of Y-jobs, or estimate the corresponding parameters. Note however, that under high loads not much gain in performance can be expected from such policies, as suggested by the fact that policy w-LXP, which is oblivious of Y jobs in the system, performs just as good as w-JSQ with full state information.

6. REFERENCES

[1] S. Bhulai, G. Hoekstra, J. Bosman, and R. van der Mei. Dynamic traffic splitting to parallel wireless networks with partial information: A Bayesian approach. *Performance Evaluation*, 69:41–52, 2012.

[2] M. Cetin, G. List, and Y. Zhou. Factors affecting minimum number of probes required for reliable estimation of travel time. *Transportation Research Record: Journal of the Transportation Research Board*, 1917:37–44, 2005.

[3] M. Chen and S. Chien. Dynamic freeway travel-time prediction with probe vehicle data: Link-based versus path-based. *Transportation Research Record: Journal of the Transportation Research Board*, 1768:157–161, 2001.

[4] G. Comert. Simple analytical models for estimating the queue lengths from probe vehicles at traffic signals. *Transportation Research Part B: Methodological*, 55:59–74, 2013.

[5] P. Guo, W. Sun, and Y. Wang. Equilibrium and optimal strategies to join a queue with partial information on service times. *European Journal of Operational Research*, 214:284–297, 2011.

[6] J. Herrera, D. Work, R. Herring, X. Ban, Q. Jacobson, and A. Bayen. Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment. *Transportation Research Part C: Emerging Technologies*, 18:568–583, 2010.

[7] A. Hordijk and G. Koole. On the optimality of the generalized shortest queue policy. *Probability in the Engineering and Informational Sciences*, 4:477–487, 1990.

[8] A. Hordijk, G. Koole, and J. Loeve. Analysis of a customer assignment model with no state information. *Probability in the Engineering and Informational Sciences*, 8:419–429, 1994.

[9] J. Kuri and A. Kumar. Optimal control of arrivals to queues with delayed queue length information. *IEEE Transactions on Automatic Control*, 40:1444–1450, 1995.

[10] R. Long Cheu, C. Xie, and D. Lee. Probe vehicle population and sample size for arterial speed estimation. *Computer-Aided Civil and Infrastructure Engineering*, 17:53–60, 2002.

[11] Matlab. www.mathworks.com/products/matlab.

[12] M. Mitzenmacher. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems*, 11:6–20, 2000.

[13] C. Nanthawichit, T. Nakatsuji, and H. Suzuki. Application of probe-vehicle data for real-time traffic-state estimation and short-term travel-time prediction on a freeway. *Transportation Research Record: Journal of the Transportation Research Board*, 1855:49–59, 2003.

[14] P. Robert. *Stochastic networks and queues.* Springer, 2003.

[15] T. Roopa, A. Iyer, and S. Rangaswamy. Crotis: Crowdsourcing based traffic information system. In *IEEE International Congress on Big Data (BigData Congress)*, pages 271–277, 2013.

[16] V. Sethi, N. Bhandari, F. Koppelman, and J. Schofer. Arterial incident detection using fixed detector and probe vehicle data. *Transportation Research Part C: Emerging Technologies*, 3:99–112, 1995.

[17] K. Srinivasan and P. Jovanis. Determination of number of probe vehicles required for reliable travel time measurement in urban network. *Transportation Research Record: Journal of the Transportation Research Board*, 1537:15–22, 1996.

[18] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 85–98, 2009.

[19] S. Turner and D. Holdener. Probe vehicle sample sizes for real-time information: the houston experience. In *IEEE Vehicle Navigation and Information Systems Conference (VNIS)*, pages 3–10, 1995.

[20] Waze. www.waze.com.

[21] R. Weber. On the optimal assignment of customers to parallel servers. *Journal of Applied Probability*, 15:406–413, 1978.