# Robust Scheduling and Congestion Control for Flexible Queueing Networks

Ramtin Pedarsani, Jean Walrand and Yuan Zhong

***Abstract*—We consider a general flexible queueing network in which each queue can be processed by several servers, which in turn can serve multiple queues. Important special cases of this model include open multiclass queueing networks [11] and flexible parallel server systems [16].**

**A scheduling policy decides how server capacities are allocated over time. It is robust if it does not depend on network parameters such as arrival and service rates. In this paper, we propose a robust and throughput-optimal scheduling policy for general flexible queueing networks when service rates depend only on the queues. The policy balances all the flows in the network, by minimizing an objective function using stochastic gradient projection over time. We also propose a joint robust scheduling and congestion control policy to maximize the utility of the network. Finally, we provide simulation results to show the performance of the algorithms.**

## I. Introduction

Many modern processing systems, such as data centers, production lines and call centers, exhibit a considerable amount of flexibility. In these systems, processing units can handle multiple demand types, and often have overlapping capabilities. A physical server in a data center cluster can process many kinds of application requests, which in turn may be placed on different servers for service [8], [18]. Likewise, call operators in a call center are often cross-trained to have overlapping skill sets [20]. To best harness the power of this flexibility, it is important to efficiently schedule the processing resources so that all demands are satisfied in a timely manner.

In this paper, we consider a general flexible queueing network, and focus on robust scheduling algorithms that maximize system throughput. An algorithm is robust if it makes scheduling decisions only based on queue sizes, but not on system parameters such as arrival or service rates. It is important that the algorithm be robust, since operating conditions of servers can fluctuate over time, and incoming demand rates can be unpredictable (for example, data center traffic can have unexpected spikes [13]), making estimates of system parameters often unreliable.

At a high level, the system model consists of a finite collection of queues and servers. Both queues and servers are flexible in the sense that each server is capable of serving a (non-empty) subset of the queues, and customers at a queue may be served by more than one server. New customers arrive to the system over time, and let each queue have one dedicated exogenous arrival process. Upon service completion, a customer may join another queue or leave the network, according to a general routing matrix. As one can see, the model considered in this paper is fairly general; important special cases include flexible parallel server systems (e.g., [16]) and the open multi-class queueing networks [11], [6].

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA (email: ramtin@eecs.berkeley.edu; wlr@eecs.berkeley.edu; zhyu4118@berkeley.edu).

As the main contribution of the paper, we propose a scheduling algorithm that is robust to arrival and service rates and is of great interest in practice due to its simplicity. The algorithm uses stochastic gradient descent, and is based on the simple idea of matching incoming customer flow rates to their respective service rates. If system parameters were known, a so-called static planning problem [10] can be solved to obtain the optimal allocation of server capacities, which balances flows in the system. Without the knowledge of system parameters, however, our algorithm updates the allocation of server capacities using queue size information. Our algorithm is provably throughput-optimal in the case that the service rates depend only on the queues. Due to lack of space, we skip the detailed proof which will be available online in the extended version of the paper. Moreover, we provide simulation results that show the convergence of allocation of server capacities to the optimal value, and stability of the queues if the arrival rates are in the capacity region.

Throughput optimality of our scheduling algorithm ensures the efficiency of the system, but in practice, we often need to enforce fairness among different input flows. In this spirit, the second contribution of the paper combines the proposed scheduling algorithm with congestion control with the aim to achieve both fairness among different input flows as well as maximal throughput.

### A. Related Works

In this subsection, we review models and algorithms related to our paper, and put our results in perspective.

Our flexible queueing network model is closely related to the system considered in [1]. [1] studies a class of generalized round robin scheduling policies, extending those described in Section 2.9.2 of [6]. The policies in [1] make use of arrival and service rates, and their throughput properties are analyzed using fluid analysis, hence their approach is quite distinct from ours. We would also like to point out that our network model includes some well-studied queueing systems as special cases. In the case where the queues are not flexible, i.e., each queue has a dedicated server, the system reduces to the open multiclass queueing network (see e.g., [11], [6]). The version of the system with no routing, i.e., when arriving customers leave the network immediately after service completion, is

equivalent to the classical flexible parallel server system, considered in e.g., [16].

[6], [3] contain good reviews of throughput optimal policies in open multiclass queueing networks. For networks of Kelly type, i.e., when the service rate of each server only depend on the server itself, robust policies such as FIFO, Processor Sharing (PS), and Head-of-Line PS have been proved to be throughput optimal [4], [5]. In general multiclass networks, the only known class of throughput optimal scheduling policies are Max-Weight type policies [7], which require knowledge of service rates. [19] considered robust policies such as Longest-Queue-First (LQF), but was only able to show throughput optimality under special cases.

There is considerable interest in the study of robust scheduling algorithms in the context of parallel server systems. The well-know $Gc\mu$ rule (equivalent to a MaxWeight policy with appropriately chosen weights on queues) has been proved to have good performance properties (including throughput optimality) (e.g., [16]), and does not depend on arrival rates. [2] studies performance properties of LQF, which is robust to both arrival and service rates, and establishes its throughput optimality when the activity graph is a tree. [9] establishes the throughput optimality of LQF under a local pooling condition.

Finally, we provide some remarks on the technical approach of algorithms in this paper. They are similar to distributed CSMA algorithms for wireless networks studied in [12]. There the idea of the algorithm is also based on using gradient projection to optimize a certain system objective.

### B. Network Model

Consider a queueing network with $K$ queues and $J$ servers, operating in discrete time[1]. Servers are *flexible* in the sense that each server can serve a (non-empty) set of queues. Similarly, tasks or customers in each queue are *flexible*, so that each queue can be served by a set of servers. For each $j$, let $\mathcal{G}_j$ be the set of queues that server $j$ can serve, let $G_j = |\mathcal{G}_j|$, and let $G = \sum_{j=1}^{J} G_j$. For each $k$, let $\mathcal{S}_k$ be the set of servers that can serve queue $k$, and let $S_k = |\mathcal{S}_k|$. Clearly, $\sum_{k=1}^{K} S_k = G$. Without loss of generality, we also assume that $G_j, S_k \geq 1$ for all $j$ and $k$, so that each server can serve at least one queue, and each queue can be served by at least one server. Thus, $\cup_{j=1}^{J} \mathcal{G}_j = \{1, \ldots, K\}$, and $\cup_{k=1}^{K} \mathcal{S}_k = \{1, \ldots, J\}$. The *activity graph* of the network is the set of all pairs $(k, j)$ such that $k \in \mathcal{G}_j$ (equivalently, $j \in \mathcal{S}_k$).

We suppose that each queue has a dedicated exogenous arrival process (with rates being possibly zero). For each $k$, suppose that arrivals to queue $k$ form an independent Bernoulli process with rate $\lambda_k \in [0, 1]$. Thus, in each time slot, there is exactly one arrival to queue $k$ with probability $\lambda_k$, and no arrival with probability $1 - \lambda_k$. Let $A_k(t)$ to be the cumulative number of external arrivals to queue $k$ up to time $t$. The routing structure of the network is described by the matrix $R = [r_{k'k}]_{1 \leq k', k \leq K}$, where $r_{k'k}$ denotes the probability that a task from queue $k'$ joins queue $k$ after service completion. The random routing is i.i.d. over all time slots. We assume that the network is *open*, i.e., all tasks eventually leave the system. This is characterized by the condition that $(I - R^T)^{-1}$ is invertible, where $I$ is the identity matrix.

Without loss of generality, all servers have service capacity equal to 1. In this paper, we assume that several servers can work simultaneously on the same task, so that their service capacities can be added. This is equivalent to the case of cooperating servers described in [1]. In each time slot, if a task in queue $k$ is served exclusively by server $j$, then the task departs from queue $k$ with probability $\mu_{kj}$ (without loss of generality, suppose $\mu_{kj} \leq 1$ for all $k$, $j$). Thus, $\mu_{kj}$ can also be interpreted as service rates. Write $\mu = [\mu_1^T, \ldots, \mu_K^T]^T \in \mathbb{R}^G$, where $\mu_k = [\mu_{kj}]_{j \in \mathcal{S}_k}$ is the vector of service parameters of type-$k$ tasks when served by different servers in $\mathcal{S}_k$. We call $\mu$ the service rate vector.

To explain the service mechanism of our algorithm, we define the following *allocation vector* $p \in \mathbb{R}_+^G$ (of server capacities):

$$p = [p_1^T, p_2^T, \ldots, p_K^T]^T,$$

where $p_k = [p_{kj}]^T, j \in \mathcal{S}_k$. $p$ is called *feasible* if

$$\sum_{k \in \mathcal{G}_j} p_{kj} \leq 1, \ \forall\ 1 \leq j \leq J. \tag{1}$$

We interpret allocation vectors as randomized scheduling decisions in the following manner. First, without loss of generality, the system parameters can always be re-scaled so that $\sum_{j \in \mathcal{S}_k} \mu_{kj} \leq 1$ for all $k$, by speeding up the clock of the system. Now suppose that at the current time slot, the allocation vector is $p$. For each $j$, $p_{kj}$ can be interpreted as probability that server $j$ decides to work on queue $k$. Then, the head-of-the-line task in queue $k$ is served with probability $\sum_j \mu_{kj} p_{kj}$. Note that $\sum_j \mu_{kj} p_{kj} \leq 1$ by our scaling of the service rates.

**Example 1.** To clarify the network model, we consider a flexible queueing network shown in Figure 1. For concreteness, we can think of this system as a data center with two flexible servers (the two hexagonal boxes), and one type of application with three tiers in succession (the three circles). The activity graph corresponds to the edges associated with the $\mu_{kj}$. $G = 4$ is the number of edges in this graph. The network in this example is similar to a re-entrant line with 3 queues and 2 servers. It is different from the classical re-entrant lines considered in e.g. [15], in that queue 2 can be served by 2 servers. In this network $\mathcal{G}_1 = \{1, 2\}$ and $\mathcal{G}_2 = \{2, 3\}$, $\mathcal{S}_1 = \{1\}$, $\mathcal{S}_2 = \{1, 2\}$, and $\mathcal{S}_3 = \{2\}$. Furthermore, the allocation vector $p$ and service rates vector are of length $G = 4$.

### C. The Static Planning Problem

In this subsection, we introduce a linear program (LP) that characterizes the *capacity region* of the network, defined to be the set of all arrival rate vectors $\lambda$ where there is a scheduling algorithm under which the system is stable[2]. Toward this end, for a given arrival rate vector $\lambda$, we first find the *nominal*

[1]The discrete-time assumption is not essential; the proposed algorithms can also be extended to a continuous-time version of the system, which we do not present due to space constraint.

[2]The stability condition that we are interested in is rate stability.

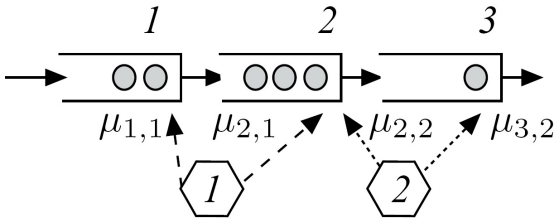


Fig. 1: Flexible re-entrant line with 3 queues and 2 servers

traffic rates $\nu = [\nu_k]_{1\le k\le K} \in \mathbb{R}^K$, where $\nu_k$ is the long-run average total rate at which tasks arrive to queue $k$. For each $k$, $\nu_k = \lambda_k + \sum_{i=1}^K \nu_i r_{ik}$. Thus, we can solve $\nu$ in terms of $R$ and $\lambda$:

$$\nu = (I - R^T)^{-1}\lambda. \tag{2}$$

Note that Eq. (2) is valid, since by definition, $(I - R^T)$ is invertible.

The LP, known as the *static planning problem* [10], is defined as follows.

$$\begin{aligned}
\text{Minimize} \quad & \rho && (3)\\
\text{subject to} \quad & \nu_k \le \sum_{j=1}^J \mu_{kj} p_{kj}, \ \forall\ 1 \le k \le K, && (4)\\
& \rho \ge \sum_{k=1}^K p_{kj}, \qquad \forall\ 1 \le j \le J, && (5)\\
& p_{kj} = 0, \qquad \text{if } k \notin \mathcal{G}_j, && (6)\\
& p_{kj} \ge 0. && (7)
\end{aligned}$$

Let the optimal value of the LP be $\rho^*$. Then $\rho^* \le 1$ is exactly the condition that there exists a feasible allocation $p$ of server capacities. It is also not difficult to see that $\rho^* \le 1$ is a necessary and sufficient condition of system stability. Thus, given $\mu$ and $R$, the *capacity region* $\Lambda$ of the network is the set of all $\lambda \in \mathbb{R}_+^K$, so that the corresponding optimal solution $\rho^*$ to the LP satisfies $\rho^* \le 1$. More formally,

$$\Lambda \triangleq \left\{ \lambda \in \mathbb{R}_+^K : \exists\ p_{kj} \ge 0 \text{ such that } \sum_{k=1}^K p_{kj} \le 1 \ \ \forall\ j, \right.$$
$$\left. \text{and } \nu_k \le \sum_{j=1}^J \mu_{kj} p_{kj} \ \ \forall\ k \right\}.$$

## II. Robust Scheduling Policy

In this section, we propose a robust scheduling policy that is provably throughput-optimal when the service rates depend only on the queues. The policy is robust to arrival and service rates, but not robust to routing probabilities of the network. The key idea is to use a stochastic gradient projection algorithm to update the service allocation vector $p$ such that all the flows in the network are balanced. We first give the precise description of the algorithm, and state the main theorem of the paper. Then, we provide some explanations.

Since service rates only depend on the tasks, only the sum $p_k \triangleq \sum_j p_{kj}$ affects the effective service rate for queue $k$. So, with an abuse of notation and terminology, we call $p = [p_k],\ 1 \le k \le K$ the service allocation vector from now on.

To give a detailed description of the algorithm, first we need some notation. In time slot $n$, let $Q_k^n$ be the size of queue $k$. Let $E^n$ be a diagonal matrix such that $e_{kk}^n = 1\{Q_k^n > 0\}$. Let $\Delta Q^n = [Q_k^{n+1} - Q_k^n]$ be a vector of length $K$ that shows the queue-length changes from time $n$ to $n+1$. Let $\mathcal{C}$ be the polyhedron

$$\mathcal{C} = \{p \in \mathbb{R}^K : \exists\ p_{kj} \ge 0 \text{ such that}$$
$$\forall k\ \sum_j p_{kj} = p_k,\ \forall k, j\ p_{kj} \ge 0,\ \forall j\ \sum_k p_{kj} \le 1\},$$

and for any $K$-dimensional vector $x$, let $[x]_{\mathcal{C}}$ denote the convex projection of $x$ onto $\mathcal{C}$. Finally, let $\{\beta^n\}$ be a decreasing sequence with $\beta^n \to 0$ as $n \to \infty$, $\sum_{n=1}^\infty \beta^n = \infty$, and $\sum_{n=1}^\infty (\beta^n)^2 < \infty$.

Our scheduling algorithm updates the allocation vector $p^n$ in each time slot $n$ in the following manner.

1. We initialize with an arbitrary feasible $p^0$.
2. Update the allocation vector $p^n$ as follows.

$$p^{n+1} = [p^n + \beta^n E^n (I - R^T)^{-1} \Delta Q^n]_{\mathcal{C}}, \tag{8}$$

This completes the description of the algorithm.

Our main result is the following rate stability of the proposed scheduling policy.

**Theorem 1.** *Suppose that the service rates of different tasks do not depend on which server is serving them, i.e. $\mu_{kj} = \mu_k$. Then, the network is rate stable under the proposed scheduling algorithm, i.e.*

$$\lim_{n\to\infty} \frac{Q_k^n}{n} = 0,\ \forall k.$$

We now provide some intuitions for the update (8). The algorithm (8) tries to adaptively find the allocation vector $p^*$ that balances the arrival and departure rate of all the queues. That is, if $\nu_k$ is the average rate through queue $k$, as determined by the flow conservation equations, the vector $p^*$ is such that $\nu_k = \mu_k p_k^*$.

If parameters $\nu$ and $\mu$ were known, we could define the diagonal matrix $M = \text{diag}\{\mu_k\}$. Then, the updates

$$p^{n+1} = [p^n - \beta^n M(\nu - Mp^n)]_{\mathcal{C}}$$

are a gradient projection method that solves the optimization problem

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\|\nu - Mp\|^2 && (9)\\
\text{subject to} \quad & p \in \mathcal{C}, && (10)
\end{aligned}$$

and hence converge to the vector $p^*$. It can be shown that the following "skewed" updates

$$p^{n+1} = [p^n - \beta^n(\nu - Mp^n)]_{\mathcal{C}} \tag{11}$$

also converge to the correct allocation vector $p^*$. We then use $\Delta Q^n$, the changes in queue sizes, to estimate the gradient

term $\nu - Mp^n$. It is easy to show that the $k^{\text{th}}$ entry of $(I - R^T)^{-1}\Delta Q^n$ is an unbiased estimator $\nu_k - \mu_k p_k^n$, if $Q_k^n > 0$:

$$\begin{aligned}
&\mathbb{E}(E^n(I - R^T)^{-1}\Delta Q^n | Q^n) \\
&\quad = E^n(I - R^T)^{-1}\mathbb{E}(\Delta Q^n | Q^n) \\
&\quad = E^n(I - R^T)^{-1}(\lambda + R^T M E^n p^n - M E^n p^n) \\
&\quad = E^n \nu - M E^n p^n \\
&\quad = E^n(\nu - M p^n).
\end{aligned}$$

The derivation also explains the reason to consider the skewed updates (11): the matrix factor $M$ requires knowledge of the service rates, but we want our updates to be robust with respect to system parameters. Replacing $\nu - Mp^n$ by $E^n(I - R^T)^{-1}\Delta Q^n$ in (11) gives us the stochastic updates (8).

Note that matrix $E^n$ in update (8) ensures that the algorithm updates $p_k^n$ only for queues $k$ that are non-empty, since $\left[(I - R^T)^{-1}\Delta Q^n\right]_k$ is no longer an unbiased estimator of $\nu_k - \mu_k p_k^n$ when $Q_k^n = 0$.

## III. Joint Robust Scheduling and Congestion Control

### A. Problem Formulation

Up to now we have proposed a robust scheduling policy for general flexible queueing networks. We expect this policy to stabilize the system for any $\lambda$ in the capacity region, hence ensuring efficiency, but in practice, the system also needs a method to regulate admissions, and enforce fairness among different input flows to the network. A systematic way of selecting the arrival rates is to consider the objective of maximizing the total utility of the tasks arriving to the network via different queues. The standard formulation first proposed by Kelly and co-authors in [14] is to solve the following optimization problem.

$$\text{Maximize} \sum_{k=1}^{K} U_k(\lambda_k) \tag{12}$$

$$\text{subject to capacity constraints.} \tag{13}$$

Our objective is to solve the optimization problem using gradient projection and update the allocation vector $p$ and $\lambda$ simultaneously. To this end, the optimization problem can be written as

$$\begin{aligned}
&\max_{\lambda, p} \sum_{k=1}^{K} U_k(\lambda_k) \\
&\text{subject to } \nu_k \le \mu_k p_k, \quad p \text{ is feasible.}
\end{aligned} \tag{14}$$

### B. Utility Maximizing Algorithm

To solve the optimization problem (14), we solve the dual problem by forming the Lagrangian

$$L(\lambda, \alpha, p) = \sum_{k=1}^{K} U_k(\lambda_k) - \sum_{k=1}^{K} \alpha_k(\nu_k - \mu_k p_k),$$

where $\alpha_k,\ 1 \le k \le K$ are the Lagrange multipliers or shadow prices. We want to maximize $L$ over $\lambda$ and $p$, and minimize

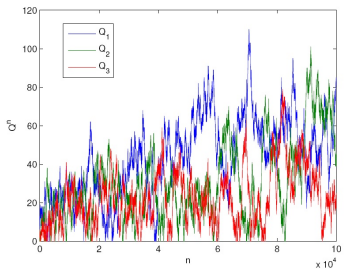


Fig. 2: Queue-length $Q^n$ vs. time

it over $\alpha_k,\ 1 \le k \le K$. To maximize $L$ over $\lambda$, after some manipulations we get

$$\lambda_k^n = \arg\max_x \ U_k(x) - [(I - R)^{-1}\alpha^n]_k x,$$

where $[v]_k$ denotes the $k$th component of the vector $v$. We use stochastic gradient projection to minimize $L$ over $\alpha_k$ and maximize it over $p$. Our scheduling algorithm updates the shadow prices $\alpha^n$ and the allocation vector $p^n$ in each time slot $n$ in the following manner.

1. We initialize with an arbitrary feasible $p^0$ and shadow prices $\alpha_k^0 \ge 0$.
2. At time $n$, we update the allocation vector $p^n$ and Lagrange multiplies $\alpha^n$ for the next time slot as follows.

$$p^{n+1} = [p^n + \beta^n E^n \alpha^n]_{\mathcal{C}} \tag{15}$$

$$\alpha^{n+1} = \left[\alpha^n + \beta^n E^n (I - R^T)^{-1}\Delta Q^n\right]^+, \tag{16}$$

   where $[x]^+$ denotes the projection of vector $x \in \mathbb{R}^K$ on $\mathbb{R}_+^K$.
3. For each queue $k$, set the arrival rate at time $n+1$ to be

$$\lambda_k^{n+1} = \arg\max_x \ U_k(x) - [(I - R)^{-1}\alpha^n]_k x. \tag{17}$$

The reasons behind (15) and (16) are similar to the arguments in Section II.

## IV. Simulations

In this section, we show the simulation results and discuss the performance of the robust scheduling algorithm, and also the joint scheduling and congestion control algorithm.

### A. Robust Scheduling Algorithm

Consider the network shown in Figure 1. The parameters of the network are the following: $\lambda = 1/6$ and

$$\mu_{11} = 1/3, \mu_{21} = 1/4, \mu_{22} = 1/4, \text{ and } \mu_{32} = 1/2.$$

The step size of the algorithm is chosen to be $\beta^n = \frac{1}{n^{0.6}}$ and the initial queue lengths are $[5, 5, 5]$. From (3), it is easy to see that the stability region $\Lambda$ is $\lambda \le \frac{2}{9}$.

Figure 2 shows the queue-length as a function of time. The simulations demonstrate that the queues become empty infinitely often; thus, they are rate stable. However, the average queue-length is large, so the algorithm suffers from bad delay. The reason is that the allocation vector $p^n$ is converging to

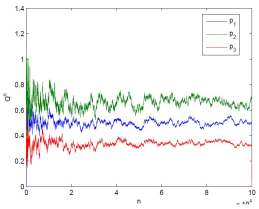

Fig. 3: Allocation vector $p^n$ vs. time

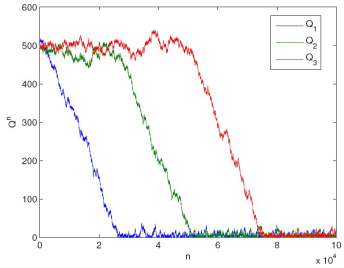

Fig. 4: Queue lengths for modified algorithm with $\epsilon = 0.02$

the value that equalizes the arrival and service rates of all the queues. As an example, if we consider an M/M/1 queue with arrival rate $\lambda$ and service rate $\mu$, then the value of $p^n$ converges to $\frac{\lambda}{\mu}$ which leads to a null recurrent queue. Figure 3 shows how vector $p^n$ converges as a function of time. In Section IV-B, we propose a modified version of the algorithm that reduces the delays.

### B. Reducing Delays

As discussed in Section IV-A, the simulations show that as expected, the allocation vector $p^n$ converges to the allocation vector that makes the queues null recurrent. However, if $\lambda$ is strictly in the interior of $\Lambda$ there exists $\epsilon > 0$ and an allocation vector $p^*$ such that $\nu_k \leq \epsilon + \sum_k \mu_k p_k^*$ for all $k$.

To improve delay, we force $p_k^n$ to converge to $\frac{\nu_k}{\mu_k} + \varepsilon$ for some small $\varepsilon > 0$. To this end, it suffices to modify the update equation in (8) to the following.

$$p^{n+1} = [p^n + \varepsilon \mathbf{1} + \beta^n E^n (I - R^T)^{-1} \Delta Q^n]_{\mathcal{C}},$$

where $\mathbf{1}$ is a $K$-dimensional vector with all entries equal to 1. Figure 4 shows the simulation result for the modified version of the algorithm that improves system delay significantly, even starting from very high initial queue-lengths $[500, 500, 500]$. However, this is at the expense of losing $\varepsilon$-throughput.

### C. Joint Scheduling and Congestion Control

Consider the same network as Section IV-A. However, the arrival rate is not fixed, and it is determined by the admission control policy as mentioned in Section III. The utility function in this simulation is

$$U(\lambda) = \log(\lambda)$$

Clearly, the solution of the utility maximization problem is $\lambda^* = \frac{2}{9}$. Figure 5 validates the convergence of the robust utility maximization algorithm's solution to the optimal solution.

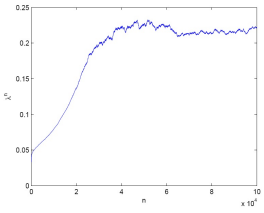

Fig. 5: Arrival rate $\lambda^n$ vs. time


## ACKNOWLEDGMENTS

This work is supported by MURI grant BAA 07-036.18.


## REFERENCES


[1] S. Andradóttir, H. Ayhan and D. G. Down. "Dynamic server allocation for queueing networks with flexible servers". *Operations Research*, vol. 51, pp. 952–968, 2003.

[2] G. Baharian and T. Tezcan. "Stability analysis of parallel server systems under longest queue first", *Mathematical Methods of Operations Research*, vol. 74, pp. 257–279, 2011.

[3] M. Bramson. "*Stability of queueing networks*". Springer, 2008.

[4] M. Bramson. "Convergence to equilibria for fluid models of FIFO queueing networks". *Queueing Systems*, vol. 22, pp. 5–45, 1996.

[5] M. Bramson. "Convergence to equilibria for fluid models of head-of-the-line proportional processor sharing queueing networks". *Queueing Systems*, vol. 23, pp. 1–26, 1996.

[6] J. Dai. "Stability of fluid and stochastic processing networks". *MaPhySto Miscellanea Publication*, No. 9, 1999.

[7] J. Dai and W. Lin. "Maximum pressure policies in stochastic processing networks". *Operations Research*, vol. 53, pp. 197–218, 2005.

[8] J. Dean and S. Ghemawat. "MapReduce: Simplified data processing on large clusters". *Communications of the ACM*, vol. 51, pp. 107–113, 2008.

[9] A. Dimakis and J. Walrand. "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits". *Advances in Applied Probability*, vol. 38, pp. 505–521, 2006.

[10] J. M. Harrison. "Brownian models of open processing networks: Canonical representation of workload". *Annals of Applied Probability*, vol. 10, pp. 75–103, 2000.

[11] J. M. Harrison and V. Nguyen. Brownian models of multiclass queueing networks: current status and open problems. *Queueing Systems Theory Appl.*, vol. 13, pp. 5–40, 1993.

[12] L. Jiang and J. Walrand. A distributed CSMA algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 960–972, Jun. 2010.

[13] R. Kandula, S. Sengupta, A. Greenberg, P. Patel and R. Chaiken. "The nature of data center traffic: Measurements & analysis". *Proceedings of SIGCOMM*, pp. 202–208, 2009.

[14] F. P. Kelly, A. Maulloo, and D. Tan. "Rate control for communication networks: Shadow prices, proportional fairness and stability". *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Mar. 1998 pp. 75–103.

[15] P. R. Kumar. Re-entrant lines. *Queueing Systems*, vol. 13, pp. 87–110, May 1993.

[16] A. Mandelbaum and A. L. Stolyar. "Scheduling flexible servers with convex delay costs: Heavy-trafc optimality of the Generalized c$\mu$-Rule". *Operations Research*, vol. 52, pp. 836–855, 2004.

[17] M. J. Neely, E. Modiano, and, C. P. Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 396–409, Apr. 2008.

[18] P. Padala, K. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal and A. Merchant. "Automated control of multiple virtualized resources". *Proceedings of Eurosys*, pp. 13–26, 2009.

[19] R. Pedarsani and J. Walrand. "Stability of Lu-Kumar networks under Longest-Queue and Longest-Dominating-Queue scheduling". Preprint, 2012.

[20] R. Wallace and W. Whitt. "A staffing algorithm for call centers with skill-based routing". *Manufacturing and Service Operations Management*, vol. 7, pp. 276–294, 2005.