Queues with Skill Based Parallel Servers and a FCFS Infinite Matching Model

Ivo Adan, Marko Boon, Ana Bušić, Jean Mairesse, Gideon Weiss

To cite this version:

Ivo Adan, Marko Boon, Ana Bušić, Jean Mairesse, Gideon Weiss. Queues with Skill Based Parallel Servers and a FCFS Infinite Matching Model. MAMA 2013 workshop of ACM Sigmetrics, 2013, Pittsburgh, United States. pp.22-24, 10.1145/2567529.2567536. hal-01273894

HAL Id: hal-01273894
https://inria.hal.science/hal-01273894
Submitted on 14 Feb 2016
1. A QUEUEING SYSTEM

We consider the following skill based parallel service queueing system: Customers are of types $C = \{c_1, \ldots, c_i\}$, servers are of types $S = \{s_1, \ldots, s_j\}$, and there is a bipartite graph $G$ of compatibilities between $C, S$. The graph has arcs $(i, j) \in G$ if server type $s_j$ has the skill to serve customer type $c_i$. Customers arrive in independent Poisson streams of rates $\lambda_i$, and have absolutely continuous patience distributions $F_i$. There are $n_j$ servers of type $s_j$, and the service times are customer-server-type dependent, the service of a customer of type $c_i$ by a server of type $s_j$ has a random duration distributed as $G_{ij}$, with average $m_{ij}$. We use the terminology of queueing theory throughout, but this type of system, with minor modifications, is useful in modeling call centers, manufacturing systems, organ transplants, multimedia servers, and cloud computing [8].

Performance of such systems is highly dependent on the operating policy. We focus here on first come first served (FCFS), where a server is assigned to the longest waiting compatible customer, coupled with assign longest idle server (ALIS), where a customer is assigned to the compatible server that has been idle for the longest time. FCFS-ALIS is widely used, because it is fair to both customers and servers, it is simple to implement, it requires little information about the parameters and the current state of the system, and it is robust under time varying conditions.

Our goal here is to develop a structured method to support the design and efficient operation of skill based parallel service systems under FCFS. At this level of generality such systems are highly intractable, no analytic results are expected, and asymptotics, e.g. using many server scaling are called for [12, 10, 7, 13]. We suggest an approximation based on a simplified look at the process — if we discard all arrival times, identities of servers, and processing times, and consider just the sequence of customers in order of arrivals, and the sequence of services in the order in which they become available, we have two infinite sequences of customer types and of server types. Making the simplifying assumption that these are i.i.d., and using FCFS, we obtain the FCFS infinite matching model. This model, which is of interest in its own right, is tractable, using symmetry and reversibility, as we describe in Section 2. From it we calculate matching rates at which customers of type $c_i$ are served by servers of type $s_j$ under FCFS. In Section 3 we use these matching rates in the design of our queueing system, with specified performance measures. We demonstrate the effectiveness of this approach by an illustrative example.

This paper answers some questions raised in [10, 6]. It summarizes recent results from in [1, 2] and current research in progress [3, 4].

2. FCFS INFINITE MATCHING

The model of first come first served (FCFS) infinite matching was introduced in [6], to answer some questions raised in [9, 10]. At that point the model looked highly intractable. It was followed by [1], where surprisingly it was found that the model is highly tractable, with a product form solution. A more general FCFS infinite matching model was described in [5]. The infinite matching model was found to be closely related to several queueing models with skill based parallel service [11, 2]. Here we present some novel results.

We consider two independent infinite random sequences: $c_m, s_n$, $m, n \in T$, $c_m$ chosen i.i.d. from $C, S$ with respective probabilities $\alpha_i$ and $\beta_j$, and the bipartite compatibility graph $G$. $T$ is either $N = 1, 2, 3, \ldots$ or $Z = \ldots, -2, -1, 0, 1, 2, 3, \ldots$

Notation: We shall refer to $s^n$ as servers, to $c^m$ as customers. Let $S(c_i)$ be the server types compatible with $c_i$. For subsets of types $C, S$ let $S(C) = \bigcup_{c_i \in C} S(c_i)$, and let $\alpha_C = \sum_{c_i \in C} \alpha_i$, $\beta_S = \sum_{s_j \in S} \beta_j$.

FCFS matching between the two sequences, for $T = N$ is constructed as follows: $s^1$ is matched to the first compatible $c^m$. Thereafter $s^n$ is matched to the first compatible $c^m$ which was not matched to any of $s^1, \ldots, s^{n-1}$. We refer to the sequences with the matchings as the FCFS matching of $s^n, c^m$, $m, n \in N$.

**Definition 1.** Let $c^m$ be matched with $s^n$ in the FCFS matching of the two sequences. The exchange transformation replaces $c^m$ by $s^m = s^n$, and $s^n$ by $c^n = c^m$ (see Figure 1).

Assume all servers up to $N$ were matched and exchanged.
Let $M$ be the location of the first unmatched customer, and $M$ be the location of the last matched customer, which was exchanged by a server. Consider the ordered sequence of servers and customers, $3 = (c^3, s^3, \ldots, s^M)$ with $3 = 0$ if $N = M = M - 1$ (see Figure 1 where $N = 4$, $M = 3$, $M = 6$ and $3 = (c^1, s^1, c^2, s^2)$). We define a Markov chain $Z_N$ with state $3$, where $Z_{N+1}$ will be obtained when $s^N+1$ is matched and exchanged.

### Figure 1: FCFS matching with exchange transformation, $Z_4 = 3 = (c^3, s^4, c^5, s^6)$

**Theorem 1.** The stationary distribution of $Z_N$ is given by

$$
\pi(3) = B \prod_{i=1}^{j} \alpha_i^{#c_i} \beta_j^{#s_j}
$$

where $#c_i$ is the number of customers of type $c_i$, and $#s_j$ is the number of servers of type $s_j$, and $B$ is the normalizing constant.

We note that the clean and beautiful simplicity of this multi-Bernoulli product form of $\pi$ masks the true complexity of the process, which is revealed if we examine what sets are allowed: we note that each $c^m$ in the sequence $3$ must be incompatible with all later $s^n$. The proof of Theorem 1 uses time reversal, and the symmetry in the process. It leads us to make the following conjecture:

**Conjecture 1 (Reversibility).** Let $s^n, c^m$, $m,n \in Z$, be two independent i.i.d. sequences, with FCFS matching between them. Let $s^n, c^m$ be obtained by the exchange transformation, with the corresponding matching. Then $s^n, c^m$ are independent i.i.d. sequences, and the matching between them is FCFS in the reversed time direction.

We can in fact show that $s^n, c^m$ are FCFS in the reversed time direction, and that each of these sequences is i.i.d. but we are still missing the proof that the sequence $s^n$ is independent of $c^m$. From $\pi$ one can obtain various marginal distributions of interest, and various performance measures. Most relevant for our queueing system are the matching rates $r_{c_i,s_j}$ at which customers of type $c_i$ are served by servers of type $s_j$. The formula for that is:

$$
r_{c_i,s_j} = \beta_j \sum_{P_2} B \prod_{k=1}^{k-1} (\beta(k) - \alpha(k))^{-1} \left( \sum_{k=1}^{k-1} \alpha(k) \beta(k) - \alpha(k) \chi_k \right)
$$

where the summation is over $P_j$, the set all the permutations of $S$. For more details on the notation used see [1].

If complete resource pooling does not hold the system decomposes uniquely to subsystems, $(C(1), S(1)), \ldots, (C(k), S(k))$ defined recursively, the first of them being:

$$
C(1) = \arg \min_{C \subseteq C} \beta \alpha \pi(S(1)) \cdot S(1) = S(C(1)).
$$

### 3. A DESIGN ALGORITHM

We now present an algorithm for the design of the queueing system of Section 1. We assume that we want to operate it in ED (efficiency driven) mode, with overloaded servers, and controlled abandonments. We let:

**Input:** The compatibility graph $G$, the arrival rates $\lambda$, the patience distributions $F_i$, and the service-time distributions $G_{ij}$, of which we only need the means $m_{ij}$.

**Quality of service decision:** Partition customer types into $C(l)$, $l = 1, \ldots, L$, for our queueing system are the matching rates $G_{ij}$, for which we still missing the proof that the sequence $s^n$ is FCFS in the reversed time direction.

**Conjecture 1 (Reversibility).** Let $s^n, c^m$, $m,n \in Z$, be two independent i.i.d. sequences, with FCFS matching between them. Let $s^n, c^m$ be obtained by the exchange transformation, with the corresponding matching. Then $s^n, c^m$ are independent i.i.d. sequences, and the matching between them is FCFS in the reversed time direction.

We can in fact show that $s^n, c^m$ are FCFS in the reversed time direction, and that each of these sequences is i.i.d. but we are still missing the proof that the sequence $s^n$ is independent of $c^m$. From $\pi$ one can obtain various marginal distributions of interest, and various performance measures. Most relevant for our queueing system are the matching rates $r_{c_i,s_j}$ at which customers of type $c_i$ are served by servers of type $s_j$. The formula for that is:

$$
r_{c_i,s_j} = \beta_j \sum_{P_2} B \prod_{k=1}^{k-1} (\beta(k) - \alpha(k))^{-1} \left( \sum_{k=1}^{k-1} \alpha(k) \beta(k) - \alpha(k) \chi_k \right)
$$

where the summation is over $P_j$, the set all the permutations of $S$. For more details on the notation used see [1].

If complete resource pooling does not hold the system decomposes uniquely to subsystems, $(C(1), S(1)), \ldots, (C(k), S(k))$ defined recursively, the first of them being:

$$
C(1) = \arg \min_{C \subseteq C} \beta \alpha \pi(S(1)) \cdot S(1) = S(C(1)).
$$

**Output:** Number of servers of each type, $n_j$, the matching rates $r_{ij}$, the cut-off waiting times $W_i$, and the abandonment rates $F_i(W_i)$.

The idea behind the algorithm is that if $\lambda$ is large relative to the typical waiting time $W$, there will be many customers in the system at all times, with a fraction $F(W)$ of abandonments, and the types of successive customers which will receive service will still be i.i.d. $\alpha$. At the same time, with large $\lambda$ the number of servers $n_j$ will be large, and the instants of service completions will be a superposition of many almost stationary, almost independent, point processes which should be approximately Poisson. As a result types of successive available servers will be i.i.d. $\beta$.

We now test our algorithm on simple graph $G$, to show that it performs quite well with moderate number of servers. $C$, $S$ and $G$ are shown in Figure 2. We use the following input data. The proportion of arrivals for the $c_i$ are $\alpha^i = (0.2, 0.5, 0.3)$. The patience distribution for all customers is exponential with mean $\lambda$, so $F(\lambda) = 1 - e^{-\lambda \lambda_{10}}$. The service time distributions $G_{ij}$ are as follows:

<table>
<thead>
<tr>
<th>Service time distributions</th>
<th>Mean $s_1$</th>
<th>Mean $s_2$</th>
<th>Mean $s_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$\exp(0.2) : 5$</td>
<td>$\exp(0.25) : 5$</td>
<td>$\exp(0.125) : 8$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$\text{uniform}(2, 3) : 4$</td>
<td>$\text{uniform}(1, 5) : 3$</td>
<td>$\text{unifor}(3, 3) : 4.5$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$\text{uniform}(2, 3) : 3$</td>
<td>$\text{uniform}(1, 5) : 3$</td>
<td>$\text{unifor}(3, 3) : 4.5$</td>
</tr>
</tbody>
</table>
This leaves preferential service to types over type . We ran extensive simulations, of runs with customers each, for every experiment. Some of our results are:

First design: equal service quality

We choose common waiting time so that abandonment rate is targeted as . This leaves . We choose . This satisfies complete resource pooling. The staffing calculations lead to

\[ n(\lambda) \approx \lambda(1.305, 1.300, 1.981). \]

We use several values of . As expected the results improve as increases, but are already quite good for a moderate number of servers.

Second design: preferential service

Here we use a waiting time of for , and waiting time for for . This partitions the servers into , and . For we choose . The staffing calculations lead to

\[ n(\lambda) \approx \lambda(1.023, 1.712, 1.637). \]

Results of the simulation are given in the following tables.

4. REFERENCES


Table 1: Simulation results for equal service level.

Table 2: Simulation results for preferential treatment.