Queues with skill based parallel servers and a FCFS infinite matching model

Citation for published version (APA):

Document status and date:
Published: 01/01/2013

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
1. A QUEUEING SYSTEM

We consider the following skill based parallel service queueing system: Customers are of types \( C = \{c_1, \ldots, c_J \} \), 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 arc \((i, j) \in G \) 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 \( n_i \) servers of type \( s_i \), 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 expected, and asymptotics, e.g. using many server scaling are expected, and asymptotics, e.g. using many server scaling are expected, and asymptotics, e.g. using many server scaling are expected, and asymptotics, e.g. using many server scaling are expected, and asymptotics, e.g. using many server scaling are expected. 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 respect to 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: \( s^m \) is matched to the first compatible \( c^m \). Thereafter \( s^m \) is matched to the first compatible \( c^m \) which was not matched to any of \( s^1, \ldots, s^{m-1} \). We refer to the sequences with the matchings as the FCFS matching of \( s^m, c^m \), \( m, n \in N \).

**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 \( \tilde{s}^m = s^n \), and \( s^n \) by \( \tilde{c}^m = c^m \) (see Figure 1).

Assume all servers up to \( N \) were matched and exchanged.

---

*Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands, {i.j.b.f.adan, m.a.a.boon}@tue.nl Research supported in part by the Netherlands Organization for Scientific Research (NWO).

†INRIA/CNS, 23 avenue d’italie, CS 81321, 75214 Paris Cedex 13, France, ana.busic@inria.fr.

‡LIAFA, CNRS et Universite Paris 7, case 7014, 75205 Paris Cedex 13, France, mairesse@liafa.univ-paris-diderot.fr.

§Department of Statistics, The University of Haifa, Mount Carmel 31905, Israel, gweiss@stat.haifa.ac.il. Research supported in part by Israel Science Foundation Grant 711/09.

Copyright is held by author/owner(s).
Let $M$ be the location of the first unmatched customer, and $\overline{M}$ be the location of the last matched customer, which was exchanged by a server. Consider the ordered sequence of servers and customers, $3 = c_2^\dagger \ldots, c_{2^m}$ with $3 = 0$ if $\overline{N} = \overline{M} = \overline{M} - 1$ (see Figure 1 where $\overline{N} = 4, \overline{M} = 3, \overline{M} = 6$ and $3 = (c_1^\dagger, s_1^\dagger, c_2^\dagger, s_2^\dagger)$). 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.

$$C \begin{array}{cccc}
\circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ
\end{array}$$

$$S \begin{array}{cccc}
\circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ
\end{array}$$

Figure 1: FCFS matching with exchange transformation, $Z_4 = 3 = (c_3^\dagger, s_4^\dagger, c_5^\dagger, s_6^\dagger)$

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

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

(1)

where $\# c_i$ is the number of customers of type $c_i$ and $\# s_j$ is the number of servers of type $s_j$ in $3$, and $B$ is the normalizing constant.

A necessary and sufficient condition for finite $B$ and ergodicity is complete resource pooling defined by:

$$\alpha_C < \beta_{S(C)}, \text{ for all non trivial subsets } C \subset C.$$ 

(2)

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 states 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 $s^n, c^m, m, n \in \mathbb{Z}$, 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:

$$r_{c_i,s_j} = \beta_j \sum_{\alpha_i \psi} B \prod_{k=1}^{j-1} (\beta_k(1 - \alpha_k))^{-1} \left( \sum_{h=1}^{j-1} \phi_{h} \frac{\alpha_k}{\beta_k} - \frac{\alpha_k}{\beta_k} \right) \prod_{l=1}^{k-1} \frac{\beta_l - \alpha_l}{\beta_l - \alpha_l}$$

$$= \frac{\prod_{l=1}^{k-1} \beta_l - \alpha_l}{\beta_l - \alpha_l} + \frac{\phi_{j}}{\psi_{j}} \prod_{l=1}^{j-1} \beta_l - \alpha_l,$$

(3)

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, $(\mathcal{C}(1), S(1)), \ldots, (\mathcal{C}(k), S(k))$ defined recursively, the first of them being:

$$\mathcal{C}(1) = \min_{\alpha \in C} \frac{\beta_{S(C)}}{\alpha}$$

$$S(1) = \mathcal{S(C(1)}).$$

3. A DESIGN ALGORITHM

We now present an algorithm for the design of the queuing 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 $\lambda_i$, 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 $\mathcal{C}(l), l = 1, \ldots, L$, where higher $l$ implies more preferential service. Let $\mathcal{S}(l) = \mathcal{S(C(l))} \cup \bigcup_{\alpha \in C(l)} S(\alpha)$. Specify target waiting times $W_1 > W_2 > \ldots > W_L$.

Continue to design each subsystem separately. We drop the superscripts and use notation $c_i \in C, s_j \in S, W$ within each subsystem:

Division of labor decision: Specify the fraction of services performed by each type of server, $\beta_j$.

Calculations: Define total service rate $\lambda = \sum \lambda_i (1 - F_i(W))$ and let $\alpha_i = \lambda_i (1 - F_i(W))/\lambda_i$. Verify complete resource pooling within the subsystem, and compute the matching rates $r_{c_i,s_j}$ using (3). Calculate required staffing:

$$n_j = \frac{\lambda}{m_{ij} \sum c_i \in C(s_j)}$$

(4)

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 $a_i = (0.2, 0.5, 0.3)$. The patience distribution for all customers is exponential with mean 10, so $F_i(t) = 1 - e^{-0.1t}$. The service time distributions $G_{i,j}$ are as follows:

<table>
<thead>
<tr>
<th>Service time distributions : Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{i,j}$</td>
</tr>
<tr>
<td>$c_1$</td>
</tr>
<tr>
<td>$c_2$</td>
</tr>
<tr>
<td>$c_3$</td>
</tr>
</tbody>
</table>
\[\alpha_1^0 = \frac{1}{3} \quad \alpha_2^0 = \frac{1}{5} \quad \alpha_3^0 = \frac{1}{10}\]

Figure 2: The “almost complete” network. Nodes on the top row represent customer types, and the ones on the bottom row server types. The dashed arcs are present in the case of complete resource pooling, but vanish in the case of incomplete resource pooling.

We explore two designs. The first provides equal service quality to all types of customers, the second design gives preferential service to types \(c_1, c_3\) over type \(c_2\). We ran extensive simulations, of 100 runs with 100,000 customers each, for every experiment. Some of our results are:

### First design: equal service quality

We choose common waiting time \(W = 1\), so that abandonment rate is targeted as \(\theta = F_c(W_i) = 1 - e^{-\alpha_i} \approx 0.095\). This leaves \(\alpha = \alpha^0 = (0.2, 0.5, 0.3)\). We choose \(\beta = (0.3, 0.4, 0.3)\). This satisfies complete resource pooling. The calculated matching rates are

\[
[r_{ij}] = \begin{bmatrix}
0 & 117 & 47 \\
21 & 175 & 64 \\
21 & 115 & 0
\end{bmatrix},
\]

and the required staffing, as a function of the total arrival rate \(\lambda\) works out at:

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

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

### Second design: preferential service

We use a waiting time of \(W_1 = 1\), for \(c^{(1)} = \{c_2\}\), and waiting time \(W_2 = 0.5\) for \(c^{(2)} = \{c_1, c_3\}\). This partitions the servers into \(S^{(1)} = \{s_1, s_3\}\), and \(S^{(2)} = \{s_2\}\). For \(S^{(1)}\) we choose \(\beta_1 = \beta_3 = 0.5\), while of course for \(S^{(2)}\) \(\beta_2 = 1\). 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
