A State Dependent Chat System Model

Per Enqvist[1] and Göran Svensson[1,2]
[1]*Division of Optimization and Systems Theory, Kungliga Tekniska Högskolan, Lindstedtsv 25, Stockholm, Sweden*
[2]*Teleopti WFM, Teleopti AB, Karlavägen 106, Stockholm, Sweden*

Keywords: Chat System, Queueing Network, Modeling.

Abstract: The main purpose of this paper is to introduce a model of a chat based communication system, as well as developing the necessary tools to enable resource optimization with regards to a measure of the service quality. The system is modeled by a Markov process in continuous time and with a countable state space. The construction of the intensity matrix corresponding to this system is outlined and proofs of a stationary state distribution and an efficient way of calculating it are introduced. A numerical example for system optimization when the service measure is the average sojourn time is included as well as a heuristic algorithm for quicker solution generation.

1 INTRODUCTION

In this paper a chat based communication system model is developed and used. It is based on the patent (Svensson, 2018). The type of service system investigated is a Markov process $\{X(t), t \in \mathbb{R}_+\}$ on a countable state-space $\mathcal{X}$. The main difference from a more traditional queueing system, such as a $M/M/s$-queue, is that the service intensities are state dependent. This state dependency is introduced to model a server (agent) being able to work on several tasks in parallel. In particular, such a system can model a chat based communication system at a modern contact center, which will be the focus of this paper. However, the main ideas may just as easily be applied to any similar phenomena. Here, this feature is modeled by attributing variable service rates to servers under different workloads. The number of jobs a server is currently serving in parallel is called the concurrency level. This setting shares many similarities with the field of processor sharing, such as (Kleinrock, 1967), (Cohen, 1979), and limited processor sharing, see (Yamazaki and Sakasegawa, 1987), (Avi-Itzhak and Halfin, 1989) and in particular it is similar to limited and variable processor sharing (Rege and Sengupta, 1985), (Gupta and Zhang, 2014). The key difference lies in the fact that the amount of service needed to complete a job is not known until it is finished, whereas in the processor sharing framework the size of the job is known once it enters the system.

In Section 2 the state space based queueing model is introduced, with focus on the intensity matrix, denoted by Q. In Section 3 the closed form solutions for the problem are discussed, first for a single agent handling several tasks in parallel and then for general instances of modeled queueing system. In Section 4 several measures of *Quality of Service* (QoS) are discussed, relevant optimization formulations are introduced and some numerical examples are given. For the QoS measures the differences as compared with similar measures for telephone systems are deliberated on. These measures are intended to represent a relevant indication of customer and provider satisfaction. Many such measures may be devised, in this text the focus will be on the the customer sojourn time. Given a QoS measure and a minimum requirement to be fullfilled the optimal number of concurrent customers per agent and group can be determined, as well as the optimal routing in terms of where to route an arrival as well as setting the maximum concurrency level for the agents.

2 SYSTEM MODEL

A queueing system, where the servers can work on several tasks in parallel is modeled. Since this work pertains to chat-based communication in a contact center environment, the servers of classical queueing systems may be referred to as agents and the arrivals will be considered to be arriving customers or clients in a chat queue, they will sometimes also be referred

Enqvist, P. and Svensson, G.
A State Dependent Chat System Model.
DOI: 10.5220/0007391001210132
In *Proceedings of the 8th International Conference on Operations Research and Enterprise Systems (ICORES 2019)*, pages 121-132
ISBN: 978-989-758-352-0

to as tasks or jobs. Once an agent answers an open chat, it is assumed to be the start of the service period of that specific chat dialogue and the end of the waiting time in queue for that customer. Any change of state happens instantaneously. All state changes (jump process) will be right-continuous with left limits everywhere with probability one (càdlàg).

2.1 Model Components

Let $\{X(t); t \in \mathbb{R}_+\}$ be a Markov process on a countable state space $\mathcal{X}$, in continuous time t. The state space is given by all possible combinations of job distributions over server groups and including a buffer for waiting customers. Consider such a process when the parameters allow a steady state solution, i.e., the system is stable as $\lim t \to \infty$. Then the transition probabilities depend only on the current state, not on time, and the state distribution is independent of initial conditions. Using the idea of a chat based system where servers may work on several jobs in parallel the underlying state space can be constructed, which correspond to the number of jobs in the system and their distributions over the available servers and the buffer. The parameters determining the state space and the transition rates include the number of servers (or groups of servers if they are not exchangable), the rate at which arrivals occur, the service rates under different customer distributions and the routing of customers to servers or server groups.

2.1.1 Arrivals

In the following all customer arrivals are considered to be independent and identically distributed, further it is assumed that all tasks are equal in the sense that they are indistinguishable from one another in terms of service required, i.e., there exist a single class of customers. The arrival process is also assumed to be independent of the service process.

The rate at which new arrivals enter the queueing system is taken to follow a homogeneous Poisson process with rate parameter λ. It is assumed that there is one common buffer for all arriving customers, if there are no service slots available.

2.1.2 Servers and Server Groups

All servers belong to some group, indexed by $\mathcal{G} = \{1, \ldots, G\}$. Agents from the same group are exchangable. Let there be $s^i \in \mathbb{N}$ agents in group $i \in \mathcal{G}$ and let $\mathbf{s} = [s^1\ s^2\ \ldots\ s^G]^T$ be the corresponding staffing vector.

An agent can be idle or actively serving a number of customers up to the maximum concurrency limit $n^i, i \in \mathcal{G}$. Let $\mathbf{n} = [n^1\ n^2\ \ldots\ n^G]^T$ represent the vector of maximum limits of the number of jobs a server may simultaneously work on. Then the maximum number of customers that may be receiving service is given by $J_{max} = \mathbf{n}^T\mathbf{s}$, i.e., the number of service slots.

The individual agent's state will be defined by the number of concurrently served customers.

Let $\mathcal{Y}_i = \{1, \ldots, n^i\}$ denote the state space for a typical agent of group $i \in \mathcal{G}$, corresponding to the number of customers being served. An agent's state may then be denoted by s^i_j, corresponding to agent $j \in \{1, 2, \ldots, s^i\}$ in group $i \in \mathcal{G}$.

The state of group i at some time point t can be captured by a state vector $X_i(t) \in \mathbb{N}^{n^i+1}$, which counts the number of agents in each state for the group

$$X_i(t) = \sum_{j=1}^{s^i} \sum_{k=0}^{n^i} \mathbb{I}(s^i_j = k) e_k, \quad i \in \mathcal{G}, \tag{1}$$

where $\mathbb{I}$ denotes the indicator function and where e_k is the unit vector in the direction of $k \in \mathcal{Y}_i$. Some additional organizational structure may be imposed for book keeping purposes when needed.

The service rate of an agent will depend on the state of that agent, i.e., the number of customers being served. The total service rate, pertaining to an agent, is assumed to be split equally between the served customers. The service times are assumed to be exponentially distributed with intensity parameter μ^i_k, which depends on the group $i \in \mathcal{G}$ and the current state $k \in \mathcal{Y}_i$ of the agent.

The total service rate when all agents are serving their maximum number of customers will be denoted by $\mu_{tail}(\mathbf{n}) = \sum_{i=1}^{G} s^i \mu^i_{n^i}$. Depending on the choice of $\mathbf{n}$, μ_{tail} will take different values. It is often fruitful to set this value equal the maximum possible service capacity.

Figure 1 depicts three agents, from the same group, serving a varying number of clients. Note the varying service rates. In general there is no need for further restrictions on the agent's service intensities, however, it is reasonable to expect that the service per customer is decreasing for increasing concurrency levels.

Assumption 2.1 *In this article it will be assumed that the service intensity per customer, μ^i_k/k, is a decreasing function of the number of concurrent customers. The total service rate, μ^i_k, of an agent may increase as more concurrency is allowed, see Figure 2. This assumption implies that the total service rate of an agent is an integer concave function in terms of the number of clients being served.*

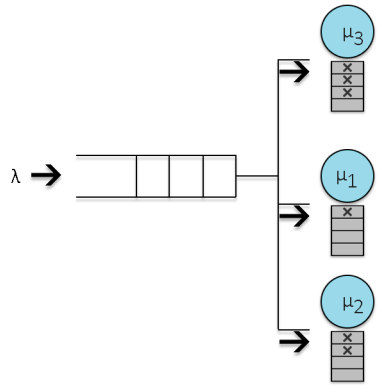

Figure 1: System of three agents, where agents 1 and 3 serve several customer simultaneously. Figure by Carl Rockman.

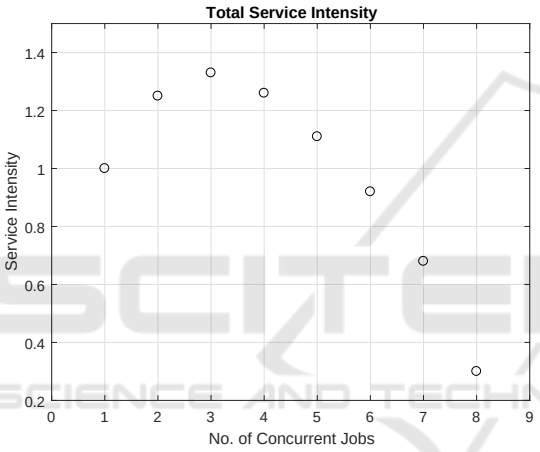

Figure 2: Total service rate for a single server under different number of concurrent customer loads.

2.1.3 Buffer

If an arriving customer can not be served immediately then the customer is placed on hold, waiting in a buffer. The buffer may be infinite in size and is following the queueing discipline of *first come first served* (FCFS). A client waits in this queue until an agent has a free slot. Once the service slots have been filled there will be a waiting queue forming. The state of the buffer is denoted by $X_b \in \{0,1,\dots\}$.

2.1.4 Routing

The process of matching arrivals to servers is handled by a state dependent routing rule $\mathcal{A} = \{a^i(x)\}_{i=1}^{G}$ for $x \in \mathcal{X}$. Since it is state dependent the stationary solution do not satisfy the insensitivity property, i.e., the product form solution. The term $a^i(x)$ corresponds to the probability that the next arrival will be routed to group i given that the system is in state x, see Assumption 2.2.

$\mathcal{A}$ provides the controls for the system when there are available service slots while the rule needs to be extended to include the routing when there are no available service slots. Let $\mathcal{R} = \mathcal{A} \cup \{r_b\}$, where r_b represents routing a new job to the buffer. Since the servers in the same group are exchangeable it does not matter which specific one is in which specific state, all that is needed is the distribution of the servers of the group over the states. If a customer gets routed to the buffer then this routing will be followed by a second routing to the first available agent.

The routing rule $\mathcal{R}$ includes only inter-group routing while the intra-group routing will be done by sending a new arrival to an appropriate agent, which under most service measures will be to the agent with the lowest current workload by Assumption 2.1. If there are several servers, within a group, with an equal number of jobs the new arrival is distributed uniformly. The specific routing is dependent on which QoS measures are under consideration.

An assignment rule could also be made to incorporate other factors, such as fair work distribution between agents, priorites of groups and more. The one condition on the routing rule is that it preserves the Markovian property of the system.

Let $L : \mathcal{X} \to \mathbb{N}$ be a counting measure that returns the total number of clients in the system given the current state.

Assumption 2.2 *For each $k \in \{1,\dots,n^i\}$, $i \in \mathcal{G}$ and $\mathcal{R}$ as given above, satisfies*

(i) $r_b, a^i \geq 0$ *and* $r_b + \sum_{i=1}^{G} a^i(x) = 1$, *for all* $x \in \mathcal{X}$,

(ii) $\mathcal{A} = 0$ *(identically zero) and* $r_b = 1$ *if and only if* $L(x) \geq J_{max}$,

(iii) $r_b \in \{0,1\}$.

2.2 Matrix Formulation

Traditional queueing systems are derived from the pure birth-death process. The assumptions in Section 2.1 gives rise to a different type of queueing system. Defining a Markov process in terms of generator matrices provides a compact means of formulating the system model.

When all agents are fully occupied then new arrivals end up waiting in the queueing buffer. This part of the system behaves as a birth-death process with an arrival intensity λ and the total service rate of μ_{tail}. The corresponding intensity matrix structure is tridiagonal.

Pooling agents with identical performance into groups limits the size of the system. See Figure 3

and 4 for examples. The figures highlights the advantages of grouping agents, in terms of system size. It is realistic to expect that agents are grouped in a contact center, in terms of the service they can provide. From the examples depicted in the figures it can be seen that the given routing rule aims at distributing new arrivals as evenly as possible. Clients leaving the system may result in an uneven distribution by which is meant that the total system service intensity is lower than the maximum possible intensity for that number of jobs.

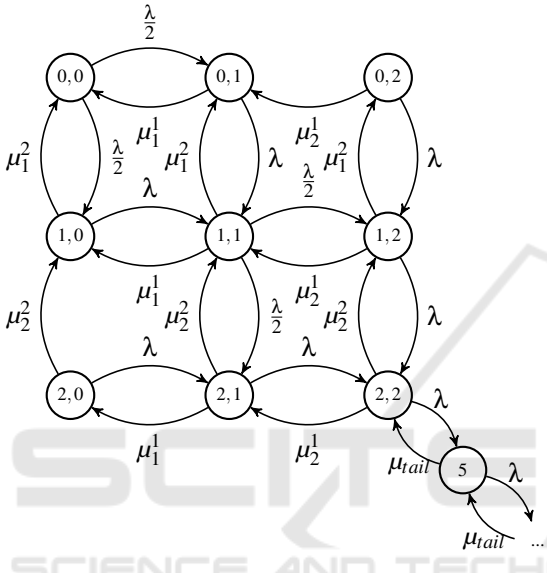

Figure 3: Example of a system with $G = 2, n^1 = n^2 = 2$ and $s^1 = s^2 = 1$ with assignments to the least used agent, and uniformly distributed if there is a tie.

2.2.1 System Intensity Matrix

As mentioned, a compact way to describe the queueing system is in the form of an intensity matrix Q and the corresponding state probability vector $\bar{p}$, which determines the stationary state probabilities. Any continuous time Markov process with some regularity condition on the initial distribution can be uniquely related to an intensity matrix Q.

Definition 2.3 (Intensity Matrix). *A matrix $Q = (q_{ij})_{1 \le i,j \le M}$ for some system size M, possibly infinite, is defined as an intensity matrix (infinitesimal generator) if it satisfies the following conditions:*

(i) $0 \le -q_{ii}$ *for all* $i \in \{1,\dots,M\}$,

(ii) $0 \le q_{ij}$ *for all* $i, j \in \{1,\dots,M\}$ *with* $i \neq j$,

(iii) $\sum_{j=1}^{M} q_{ij} = 0$ *for all* $i \in \{1,\dots,M\}$.

Note that the inequality in (i) is strict here.

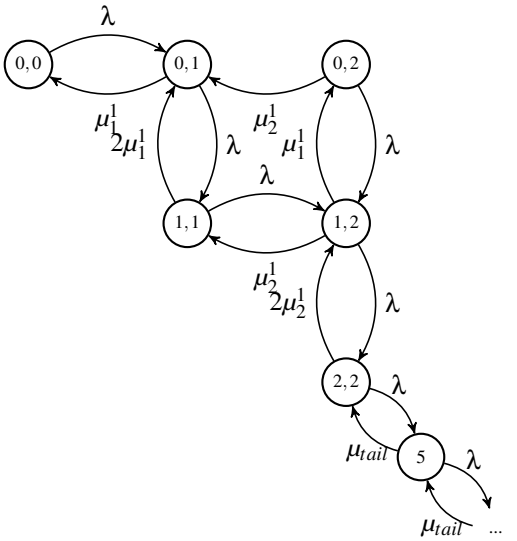

Figure 4: Example of a system with $G = 1, n^1 = 2$ and $s^1 = 2$ with assignments to the least used agent, and uniformly distributed if there is a tie.

The intensity matrix governs the rate of state changes of the Markov process $X(t)$. The state of the Markov system at any given time t is determined by

$$X(t) = \begin{bmatrix} X_1 \\ \vdots \\ X_G \\ X_b \end{bmatrix}(t). \tag{2}$$

Only one state change may occur at any given time. There are four such possible types of changes that may occur:

(i) An arrival occurs and is routed to an available agent,

(ii) An arrival occurs and is routed to the buffer,

(iii) A departure occurs and a service slot becomes available,

(iv) A departure occurs and the buffer is decreased.

All state changes occur on group level to keep the numerical size to a minimum. For case (i) and (iii) a change of state can then be handled on group level by letting e_k be a unit vector in the direction of $k \in \mathbb{R}^{n^i+1}$ and using this to update the system state on group level. By assuming that at most one server in the whole network transitions at any given time, the group specific transition from state $x \in \mathcal{Y}_i$ to $y \in \mathcal{Y}_i$ may be denoted by $x_i \mapsto x_i + (e_y - e_x)$. This corresponds to a unique state change of the Markov process $X(t)$ on $\mathcal{X}$.

In case (ii), i.e., that all servers are occupied when a new arrival occurs then the routing sends the arrival

to the buffer which changes states accordingly $X_b \mapsto X_b + 1$.

If there is a departure when the buffer is not empty, case (iv), then the buffer changes as $X_b \mapsto X_b - 1$ and the state of the corresponding group remains unchanged as $x_i \mapsto x_i + (e_x - e_x)$.

The current state of the system is given by $X(t)$, which is fully determined by the distribution of jobs between the different groups and the buffer. Each state is represented by a row in the intensity matrix Q.

When an arrival occurs it does so with an intensity of λ and if there is at least one service slot free then it gets routed to an available agent, which corresponds to a thinned state dependent Poisson process. When there are no free service slots the new arrival gets routed to the buffer. The intensities with which the arrivals are routed to the different groups, or the buffer, are given by

$$\begin{cases} \lambda_i(x) = a^i(x)\lambda, & x \in \mathcal{X}_{1:J_{max}}, \\ \lambda_i(x) = r_b\lambda, & x \in \mathcal{X}_{(J_{max}+1):M}, \end{cases} \quad \text{for all } i, \tag{3}$$

where $\mathcal{X}_{1:J_{max}}$ correspond to there being at least one service slot free to recieve an arriving client and $\mathcal{X}_{(J_{max}+1):M}$ when there are no available servers.

On the other hand departures leave the system from group i in state $X_i(t)$ with intensity

$$\mu^i(x) = [\mu_0^i \, \mu_1^i \, \dots \, \mu_{n^i}^i] X_i(t), \quad i \in \mathcal{G}. \tag{4}$$

Definition 2.4 (System Intensity Matrix). *Let Q be the intensity matrix corresponding to the Markov process $X(t)$, satisfying Definition 2.3 and with jump intensities given by Equations (3) and (4). The states for which $X_b = 0$ are first in order followed by the states for which $X_b > 0$, in ascending order of the total number of clients in the system. Furthermore, let the first row correspond to the empty system state and the $(J_{max}+1)$:th row be the state for which all service slots are filled but the buffer is empty. Let $N = J_{max} + 1$.*

This intensity matrix can be partitioned into four submatrices, as shown in Equation (5). This partition will make further analysis of the system more tractable.

$$Q = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) \tag{5}$$

The first submatrix, A, correspond to states for which the buffer is empty, see Definition 2.4. It is a sparse band matrix of size $N \times N$. The size becomes considerably smaller if the servers are grouped into a few groups.

The two submatrices B and C may both be infinite but contain only one non-zero element each.

$$B = \begin{pmatrix} 0 & 0 & \cdots \\ \vdots & \ddots & \cdots \\ \lambda & 0 & \cdots \end{pmatrix}, \tag{6}$$

$$C = \begin{pmatrix} 0 & \cdots & 0 & \mu_{tail} \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \end{pmatrix}. \tag{7}$$

The fourth submatrix, D, corresponds to the states for which there is a queue. The intensity submatrix D has the expected tridiagonal form of a traditional $M/M/\cdot$ system, i.e., it corresponds to a standard birth-death process and may be infinite in size.

$$D = \begin{pmatrix} d & \lambda & 0 & 0 & \cdots \\ \mu_{tail} & d & \lambda & 0 & \cdots \\ 0 & \mu_{tail} & d & \lambda & \cdots \\ \vdots & & \ddots & \ddots & \ddots \end{pmatrix}, \tag{8}$$

where $d = -(\lambda + \mu_{tail})$ except for the last row where it is $-\mu_{tail}$.

2.3 System Structure

The use of the intensity matrix, given by Equation (5), requires that the system matrix can be created for different numbers of groupings of agents and different maxima on concurrency levels. The structure of the submatrices B,C and D are given in (6) - (8). It is only the system size that varies the structure of those submatrices. The construction of A is given by Equations 3 and 4. The matrix A depends on a given assignment rule, $\mathcal{A}$, which determines the routing of new arrivals within the system. The submatrix A is of finite size for all realistic systems, it might however be of interest to study its behaviour under different limiting schemes, compare with the Halfin-Whitt heavy traffic regime (Whitt and Halfin, 1981).

In the empty system, corresponding to the first row of Q, only arrivals may occur. The exact order of the rows is important when implementing a numerical model of the system, but here it suffices that the first and last row of A are clearly defined.

Example: For some ordering of the states the intensity matrix for Figure 3 has the following structure

$$\left(\begin{array}{cccccccc|ccc}
d_1 & \frac{\lambda}{2} & 0 & \frac{\lambda}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
\mu_1^1 & d_2 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & \mu_2^1 & d_3 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & \cdots \\
\mu_1^2 & 0 & 0 & d_4 & \lambda & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & \mu_1^2 & 0 & \mu_1^1 & d_5 & \frac{\lambda}{2} & 0 & \frac{\lambda}{2} & 0 & 0 & \cdots \\
0 & 0 & \mu_1^2 & 0 & \mu_2^1 & d_6 & 0 & 0 & \lambda & 0 & 0 & \cdots \\
0 & 0 & 0 & \mu_2^2 & 0 & 0 & d_7 & \lambda & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \mu_2^2 & 0 & \mu_1^1 & d_8 & \lambda & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & \mu_2^2 & 0 & \mu_2^1 & d_9 & \lambda & 0 & \cdots \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu_{tail} & d_{10} & \lambda & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots
\end{array}\right),$$

where the d_i:s are the negative sum of the off-diagonal row elements.

The routing rule will be determined in relation to one or more QoS measures, see Section 4. Different measures may imply different routing rules, e.g., under some circumstances minimizing the average waiting time in the system and the average waiting time in the buffer produces different routing and even different concurrency levels for the servers.

3 STATE PROBABILITY DISTRIBUTION

An important goal of modeling a system in steady state is to determine if there exists a stationary state distribution, $\bar{p}$, and the necessary conditions associated with it. The solutions to this question will be answered first for a single agent and then for a system of multiple agents and groups.

3.1 Single Agent, Multiple Tasks Queueing System

Looking at a queueing system with a single agent that can serve up to n customers concurrently. Assuming that the system is in steady state and of the type $M/M/n$. It is also assumed that the service rate per customer is a nonincreasing function of the number being served and that the service amount is evenly distributed amongst the clients. Since only one agent is considered the group index may be dropped.

Introduce the state probabilities p_i for $i = 0, 1, \ldots$. The arrivals are assumed to follow a time homogeneous Poisson process with intensity parameter λ. Only one class of customers is considered. The following expressions will be used in determining the state probability distribution for the given queueing system in equilibrium

$$\rho_i = \begin{cases} i\frac{\lambda}{\mu_i}, & i \leq n \\ n\frac{\lambda}{\mu_n}, & i > n, \end{cases} \tag{9}$$

where μ_i is the total service intensity for state $i \leq n$ and μ_n for $i > n$ when there are i tasks in the queue.

Let

$$\rho(i) = \begin{cases} \prod_{j=1}^{i} \rho_j & \text{if } i \leq n \\ \rho_n^{i-n} \prod_{j=1}^{n} \rho_j & \text{if } i > n. \end{cases} \tag{10}$$

The load, $\rho(i)$, depends on the number of total tasks in the system, both tasks that are being served and those in the buffer . To achieve steady state the system must be either finite or $\frac{\lambda}{\mu_n} = \frac{1}{n}\rho_n < 1$ must hold if the buffer size is infinite. The probability of each state can be expressed in terms of the empty system state probability, p_0, via the local or global flow balance equations

$$p_i = \begin{cases} \frac{1}{i!}\rho(i)p_0 & \text{if } i \leq n \\ \left(\frac{1}{n}\right)^{i-n} \frac{1}{n!}\rho(i)p_0 & \text{if } i > n. \end{cases} \tag{11}$$

From Markov theory it is well known that for a birth-death process where $\lambda/\mu_n < 1$ there exist a stationary distribution. Since $\sum_{i=0}^{\infty} p_i$ is a sum of probabilities, when the steady state condition holds, there is a solution for p_0 determined by the following expression for p_0

$$p_0 = \left(1 + \sum_{i=1}^{n-1} \frac{1}{i!}\rho(i) + \frac{\rho(n)}{n!}\frac{n}{n-\rho_n}\right)^{-1}. \tag{12}$$

Since λ and the μ_i:s are known we can calculate p_0 and hence also p_i, for any $i \in \mathbb{N}$.

3.2 Multiple Agents and Multiple Tasks Queueing System

Consider the situation where there are several agents handling incoming tasks. These agents may be grouped into pools with other agents with whom they are exchangeable. We will look at the case where there is only one class of jobs. The state probabilities will be calculated for the potentially infinite queueing system.

To solve the complete system described in Section 2 a few preliminary results will have to be shown.

A stationary distribution $\bar{p}$ can be found when $\mu_{tail} > \lambda$ holds. It is shown that there exist a limiting distribution, which is equivalent to the stationary state distribution, by means of finding the unique solution.

Use the partitioning of the intensity matrix Q from Equation (5). Suppose that $Q \in \mathbb{R}^{M\times M}$, $A \in \mathbb{R}^{N\times N}$

where $N = J_{max} + 1$ and that $M \geq N$. The steady state solution for the Markov system then satisfies

$$xQ = \mathbf{0}. \tag{13}$$

It is helpful to partition the state vector into two parts, matching the partition in (5) so that $x = (x_1\, x_2)$, where $x_1 \in \mathbb{R}^N$ and $x_2 \in \mathbb{R}^{M-N}$. Then Equation (13) can be stated as

$$(x_1\, x_2)\begin{pmatrix} A & B \\ C & D \end{pmatrix} = (0\,0). \tag{14}$$

Since D is invertible due to tridiagonal birth-death structure, the following holds

$$\begin{cases} x_1A + x_2C = 0 \\ x_1B + x_2D = 0 \end{cases} \implies \begin{cases} x_2 = -x_1BD^{-1} \\ x_1(A - BD^{-1}C) = 0. \end{cases} \tag{15}$$

Remark 3.1. *The first Equation in (15) indicates that the probabilites of the states of the x_2-vector only depend on the last element of the x_1-vector since submatrix B only contains one nonzero element, in the last row of the first column.*

That the constructed intensity matrix Q is irreducible will be used repeatedly, thus, it is prudent to show that this is indeed the case.

Lemma 3.2. *The system matrix Q is irreducible if N is finite.*

Proof. When all service slots are filled, $L(X(t)) \geq N$, the process has a pure birth-death structure and thus this part is irreducible and communicating with the system state $L(X(t)) = N$.

For the situation when $L(X(t)) \leq N$, the irreducibility may be shown state by state. By construction no state is absorbing, the arrival and service intensities are finite and $\lambda, \mu_k^i > 0$. The state $L(X(t)) = 0$ and $L(X(t)) = N$ communicate since all accepted routing rules route new arrivals to some system state with one more job in the system with positive probability and any distribution of occupied servers can always reach $L(X(t)) = 0$ since all service rates are positive. Thus all intermediate states between $L(X(t)) = 0$ and $L(X(t)) = N$ are reachable, with positive probability, from $L(X(t)) = N$. Thus since any configuration can reach the empty system state and this state communicates with $L(X(t)) = N$ which in turn communicates with all states for which $L(X(t)) > N$ the whole chain of Q is irreducible. □

To solve the system it is useful to first show that the A matrix has full rank for all systems where the size of the intensity matrix Q is at least of size $N + 1 \times N + 1$. Let the $N \times N$ matrix $\tilde{A}$ be

$$\tilde{A} = \begin{cases} A, & M = N, \\ A + \lambda e_N e_N^T, & M \geq N + 1, \end{cases} \tag{16}$$

where e_N is a unit vector in the N:th direction and where the intensity matrix Q has been ordered such that the N:th row corresponds to the state for which all agents are fully occupied but the buffer is empty and where the states following the N:th state are ordered in terms of increasing number of customers in the system.

Remark 3.3. *For an irreducible intensity matrix $Q \in \mathbb{R}^{M \times M}$, where $M \geq N$, it holds that rank$(\tilde{A}) = N - 1$, since it represents a finite irreducible Markov chain.*

Lemma 3.4. *For an irreducible Markov system defined by the intensity matrix Q of Section 2.2.1, with $Q \in \mathbb{R}^{M \times M}$ and where $M \geq N + 1$ it holds that the submatrix A of Q has rank N.*

Proof. Let Q be the following intensity matrix

$$Q = \begin{pmatrix} A & b \\ c^T & d \end{pmatrix}, \tag{17}$$

where $b \in \mathbb{R}^N$, $c^T \in \mathbb{R}^N$ and $d \in \mathbb{R}$. Then $Q \in \mathbb{R}^{(N+1)\times(N+1)}$ represents an irreducible Markov system. By construction, $Q\mathbf{1} = 0$, the final column of Q can be expressed as a linear combination of the other columns

$$\begin{pmatrix} b \\ d \end{pmatrix} = -\sum_i \begin{pmatrix} A \\ c^T \end{pmatrix}_i. \tag{18}$$

Then the submatrix $(A\, c^T)^T$ in (17) has at most N independent columns. It remains to show that c^T is a linear combination of the rows of A. The sub matrix $(A\, c^T)^T$ also has N independent rows and it suffices to show that c^T is a linear combination of the rows in A. To do that turn to the matrix $\tilde{A}$, which by construction has rank$(\tilde{A}) = N - 1$, since it is an intensity matrix for a finite irreducible Markov system. Then there exists a vector $x^* \in \mathbb{R}^N$ such that $x^* > 0$ and $(x^*)^T\tilde{A} = 0$. The question becomes if there is a $\gamma \neq 0$ that satisfies

$$((x^*)^T\ \ \gamma)\begin{pmatrix} A \\ c^T \end{pmatrix} = (x^*)^T A + \gamma c^T = 0, \tag{19}$$

where $c^T = e_N^T \mu_{tail}$. Using $\tilde{A}$ from Equation (16) to obtain

$$\begin{aligned} 0 = (x^*)^T\tilde{A} &= (x^*)^T(A + \lambda e_N e_N^T) \\ &= (x^*)^T A + \lambda (x^*)^T e_N e_N^T. \end{aligned} \tag{20}$$

By choosing $\gamma = \frac{\lambda (x^*)^T e_N}{\mu_{tail}} = \frac{\lambda (x^*)_N}{\mu_{tail}} \neq 0$ a nontrivial solution to Equation (19) is found and hence c^T can be written as a linear combination of the rows of A which in turn implies that rank$(A) = N$, i.e., A has full rank. □

From the structure of B, C and D the following holds

$$BD^{-1}C = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & y^* \end{pmatrix} = y^* e_N e_N^T \qquad (21)$$

for some scalar y^* and $\hat{A} = A - BD^{-1}C = A - y^* e_N e_N^T$, where $\hat{A} = A - BD^{-1}C$ is the Schur complement of Q. Since $x_1(A - BD^{-1}C) = 0$ according to Equation (15), the characteristic equation becomes

$$\mathcal{X}_{\hat{A}}(y) = \det(A - y^* e_n e_n^T) = 0. \qquad (22)$$

Furthermore since the characteristic equation can be expressed as $\mathcal{X}_{\hat{A}}(y) = ay + b = 0$ and $\mathcal{X}_{\hat{A}}(y = 0) \neq 0$ from Lemma 3.4 there exist a unique generalized eigenvalue y. The unique value $y^* = -\lambda$ gives that $\hat{A} = \tilde{A}$.

The goal is to find the limiting distribution for the system in steady state and to confirm that this is indeed equal to the stationary distribution $\bar{p}$. As before, let the first N states correspond to the states for which the buffer is empty, $X_b = 0$. Partition the distribution vector into two parts as done previously, $x = (x_1\ x_2)$, where x_1 corresponds to the states $L(X(t)) \leq N$ and x_2 the states for which $L(X(t)) > N$. From general theory of Markov systems it is known that the solution for an irreducible system in steady state can be obtained by using the $M-1$ independent equations from the intensity matrix and augmenting the system by use of the fact that the state vector is a probability distribution, i.e. x adds to one, $\sum_i (x)_i = 1$, to obtain a solution for the unique stationary state probability distribution $\bar{p}$.

It will be shown that it is sufficient to solve for a smaller problem in terms of the first N equations, which are independent since $rank(A) = N$ from Lemma 3.4. The solution is obtained by using $x_1 \tilde{A} = 0$, from Equation (16), and the fact that the rest of the state probabilities only depend on $(x_1)_N$ i.e., the last element of the x_1-vector. Let

$$Q_{\mathbb{I}} = \begin{pmatrix} 1 & | & & | \\ \vdots & q_2 & \dots & q_M \\ 1 & | & & | \end{pmatrix} \qquad (23)$$

$$\tilde{A}_{\mathbb{I}} = \begin{pmatrix} 1 & | & & | \\ \vdots & \tilde{a}_2 & \dots & \tilde{a}_N \\ \psi & | & & | \end{pmatrix} \qquad (24)$$

where $\psi = \sum_{i=0}^{M-N} \left(\frac{\lambda}{\mu_{tail}}\right)^i$ and where the $\tilde{a}_i$ represent column i of the $\tilde{A}$ matrix.

Proposition 3.5. *Suppose that Q is the intensity matrix for an irreducible Markovian system, as given in Section 2.2.1, in steady state and Equations (23)-(24) holds, then solving*

$$(x_1\, x_2)\, Q_{\mathbb{I}} = (1\, 0 \dots 0) \qquad (25)$$

is equivalent to solving

$$x_1 \tilde{A}_{\mathbb{I}} = (1\, 0 \dots 0) \qquad (26)$$

and calculating the state probabilities in x_2 as $(x_2)_i = (x_1)_N \left(\frac{\lambda}{\mu_{tail}}\right)^i$ for $i = 1, 2, ..., M-N$, where $(x_1)_N$ represents the last element of the x_1-vector.

Proof. Since $x := (x_1\, x_2)$ is a probability distribution it holds that

$$x_1 \mathbf{1}_N + x_2 \mathbf{1}_{M-N} = 1. \qquad (27)$$

where $\mathbf{1}_i$ is a vector of i ones. Due to construction of the intensity matrix and the local flow balance equations it also holds that

$$(x)_{j+1} = \begin{cases} \frac{\lambda}{\mu_{tail}}(x)_N = \frac{\lambda}{\mu_{tail}}(x_1)_N, \text{ for } j = N, \\ \frac{\lambda}{\mu_{tail}}(x)_j = \frac{\lambda}{\mu_{tail}}(x_2)_{j-N}, \text{ for } j \geq N+1. \end{cases} \qquad (28)$$

Using ψ and Equations (27) and (28) together gives

$$1 = x_1 \mathbf{1}_N + x_2 \mathbf{1}_{M-N} = (x_1)_{1:N-1} \mathbf{1}_{N-1} + (x_1)_N + \sum_{i=1}^{M-N} \left(\frac{\lambda}{\mu_{tail}}\right)^i (x_1)_N = (x_1)_{1:N-1} \mathbf{1}_N + (x_1)_N \psi. \qquad (29)$$

This result is used in calculating the matrix multiplication of the first column of the intensity matrix.

$$xQ_{\mathbb{I}} = (x_1\, x_2) \left(\begin{array}{c|ccc|c} 1 & | & & | & \\ \vdots & a_2 & \dots & a_N & B \\ 1 & | & & | & \\ \hline 1 & | & & | & \\ \vdots & c_2 & \dots & c_N & D \\ 1 & | & & | & \end{array}\right) = (1\, 0 \dots 0) \qquad (30)$$

$$(x_1\, x_2) \left(\begin{array}{c|ccc|c} 1 & | & & | & \\ \vdots & a_2 & \dots & a_N & B \\ \psi & | & & | & \\ \hline 0 & | & & | & \\ \vdots & c_2 & \dots & c_N & D \\ 0 & | & & | & \end{array}\right) = [1\, 0 \dots 0]. \qquad (31)$$

Solving Equation (31) in terms of x_1, using the result from Equation (15) results in

$$x_1 \left(\begin{pmatrix} 1 & | & & | \\ \vdots & a_2 & \dots & a_N \\ \psi & | & & | \end{pmatrix} - BD^{-1}C \right) = x_1 \tilde{A}_{\mathbb{I}} = (1\, 0 \dots 0), \qquad (32)$$

where $BD^{-1}C$ is given by Equation (22), with $y^* = -\lambda$. □

Corollary 3.6. *If* $M \to \infty$ *then* $\psi \to \frac{\mu_{tail}}{\mu_{tail}-\lambda}$ *and* $\lim_{M\to\infty} x = \bar{p}$.

Proof. By the definition of ψ and for a fixed value of N, i.e., the size of A, which is constant for a given system, ψ is given by

$$\psi = \lim_{M\to\infty} \sum_{i=N}^{M} \left(\frac{\lambda}{\mu_{tail}}\right)^{(i-N)} = \frac{\mu_{tail}}{\mu_{tail}-\lambda}. \qquad (33)$$

Using results for a geometric series in Equation (33). This being the unique solution to Equation (30), thus, it is also the stationary state solution. □

This method partitions the system in two, where the second part may be handled as a pure birth-death process with corresponding probability structure. For a deeper study of such phenomena see (Boucherie, 1993).

Remark 3.7. *The size of the generator matrix becomes very large even for moderate sized systems. The advantage of using a small number of groups play a significant role in keeping the size of the system managable as does the employment of sparse numerical methods. The intensity matrix Q is a sparse matrix. The level of sparsity increases if the routing is deterministic.*

4 OPTIMAL NUMBER OF CONCURRENT SERVICES PER AGENT AND SERVICE LEVEL

In the previous sections a model describing a system of grouped indistinguishable agents, able to handle several tasks simultaneously at varying service rates, has been introduced. One important question is how to choose the planned level of maximum concurrency, such that the system is optimal with respect to some QoS measure/s. Another is how to choose the number of servers to employ in each group. To answer these questions one or more measures of QoS has to be decided upon. There are many choices for such measures, see for example (Gans et al., 2003). Once the measure, or measures, have been choosen, the performance of the system w.r.t. that measure can be determined and optimized. The optimization process may be performed iteratively, but to keep the amount of calculations down a heuristic method for estimating this number will be given below, with respect to the average sojourn time for a customer.

4.1 Quality of Service Measures

Finding a model that fits the underlying system is often but a stepping stone in the process of managing a queueing network. To evaluate different system configurations some metric is needed. In service systems the metric is commonly referred to as a *Quality of Service* measure.

When looking at some of the measures used in traditional call centers it becomes apparent that some modifications are in order. Two common measures are *Average Speed of Answer* (ASA) and *Traffic Service Factor* (TSF), where the first is given as the expected value of the amount of time the client has to wait on service to begin, i.e., the call is answered and the second one gives the fraction of clients that start their service within a given time. For a chat system, as described above, both these time measures can be made zero by opening an infinite number of service slots. However, the service rate per customer would most likely be appaling. This suggests that the QoS measure used should include aspects of the time spent actually recieving service as well as the waiting time. Some simple such measures might be the average head count process, i.e., the number of clients in the system at a given time t or equivalently the expected sojourn time. If used sensibly the mean waiting time in queue can successfully be used as the measure. It is worth noting that some routing rules and system configurations may lead to shorter queueing times while increasing the mean sojourn time, e.g., letting the agent concurrency level surpass the one providing maximum total service rate may shorten the waiting queue at the expense of longer sojourn times.

Definition 4.1. *The sojourn time of a client is defined as the time a customer spends in the system, from arrival to the system until the departure from the system. Let* $W \in \mathbb{R}^+$ *be a random varible denoting the sojourn time then the expected sojourn time is*

$$EW = \mathbb{E}(W). \qquad (34)$$

Using the minimum average sojourn time for a customer has the advantages of being fairly easy to handle and Little's law may be applied. For an individual customer it might however be less favourable since waiting times may be long for some customers, the tail events. A drawback of using the mean sojourn time measure is that it can be difficult to calibrate. In the event that another measure is preferred the minimum average sojourn time measure can still be used as a initial value of an iterative search process.

In the general case both the maximum concurrency level and the number of agents per group constitutes the targets but for the mean sojourn time the

maximum concurrency level for minimizing the average number of clients in the system is just the concurrency level where the total service rate is maximized. This follows from the fact that at each number of clients in the system, $L(X(t))$, the rate at which service is performed for the system is equal or higher than for any other choice of n while the arrival rate is the same. Thus, for the sojourn time measure it suffices to find the optimal number and distribution of servers over the groups. This translates to an assignment rule that routes new clients as evenly as possible if the total service rate is an integer concave function of the concurrency level.

A special case is when there is only one group with a linear increase in total service rate as a function of the concurrency level. Such a system may be represented by a $M/M/sn$ queueing system.

Another type of service measure that might be worth considering is a fairness one, such as that the time spent idling should be fairly evenly distributed between groups and agents. In the example of Section 4.3 below it will be shown why this would be an interesting metric.

4.2 Optimization Formulation

Given the QoS measure, estimates of the system parameters and the costs of agents the problem may be formulated as an optimimization problem. It is also assumed that forecasts concerning the arrival process are available. There are two main perspectives on the optimization problem, the first being how many agents are enough to fullfill demands of the service quality as captured by the QoS measure,

$$(P)\begin{pmatrix} \min\limits_{s} & \sum\limits_{i} C_i(s) \\ \text{subj. to} & QoS(s,n) \leq b \\ & s,n \in \mathbb{Z}^+ \end{pmatrix}, \qquad (35)$$

and the second being given a budget how to best staff the chat system to provide the best possible service. The second formulation is mostly of interest when the budget does not allow for sufficient staff to actually fullfill the QoS level requirements.

$$(B)\begin{pmatrix} \min\limits_{s} & QoS(s,n) \\ \text{subj. to} & \sum\limits_{i} C_i(s) \leq B \\ & s,n \in \mathbb{Z}^+ \end{pmatrix}, \qquad (36)$$

C is the agent cost function, b the required service level and B is a budget constraint. It is assumed that agents from the same group cost the same.

The optimization formulations may easily be reformulated to include multiple QoS measures and restrictions on agent availability.

4.3 Numerical Example

For a system that is measured on minimizing the average number of clients both the expected sojourn time and the expected queueing time may be used as measures. In many cases it is easier to work with the time a customer has to wait in line rather than the sojourn time, since the average queueing time can be pushed close to zero, while the sojourn time will depend on the arrival rate and the single customer service rates.

When solving these types of problems it is possible to find the optimal solution via an iterative combinatorial approach, however, such a method is very costly in terms of the number of calculations needed. This really becomes an issue when dealing with large systems, with a high degree of concurrency and many groups. Even efficient and sparse solvers will struggle to deliver solutions quickly. Thus, a simple heuristic may be employed to achieve near optimal solutions for realistic parameters. Given some starting distribution of agents over the groups such that, $s^i_j > 0$, for all $i \in \mathcal{G}$ and $j \in \{1,\ldots,s^i\}$, then one agent at a time is added according to some prediction function $\kappa(\mathbf{s}) : \mathbb{N}^G \to \mathcal{G}$. This function calculates marginal gains for each group in terms of service rate per cost of agent, and then returns the corresponding group index with the largest marginal gain. The service rate used is based on a prediction of what state an additional agent would be in and uses the corresponding service rate.

To decide which group receives the additional agent the heuristic looks at the current system solution and determines the most likely (weighted frequency) concurrency level of each group. These levels are used to calculate the corresponding marginal gains, $\frac{\mu^i_k}{C_i}$, to compare the benefits of adding agents to the different groups and picks the one with the largest marginal gain. In the next step the system is solved with the predicted agent distribution and if the new solution predicts another agent to be added to the same group then the choice is accepted. However, if the new solution indicates that a new group is to receive an additional agent then all possible updates are compared and the QoS measures determine the best update. The drawback of using this heuristic is that there are situations where the algorithm reacts too slowly, as can be seen in the example below.

In general the heuristic will provide means to control a base distribution of agents between groups, which may be dictated by contracts, etc. It will also generate near optimal solutions at a much lower computational cost than the combinatorial approach. It works best when the service level of the groups dominate each other consistently. Near efficient points are

Algorithm 1: Starting with a staffing vector such that QoS(**s**) is defined, heuristically determine a new staffing vector such that QoS(**s**) $\leq b$.

Data: $G, \mathbf{s}, \mathbf{n}, \lambda, \mu_k^i$:s, b
initialize ;
$s^i > 0 \ \forall i \in \mathcal{G}$;
solve system and calculate QoS(**s**) ;
while *QoS*(**s**) $> b$ **do**
  compute $k = \kappa(\mathbf{s})$;
  let $\mathbf{s}_{tmp} \rightarrow \mathbf{s} + e_k$;
  solve system for $\mathbf{s}_{tmp}$;
  compute $k_{tmp} = \kappa(\mathbf{s}_{tmp})$;
  if *k equals* k_{tmp} **then**
    $\mathbf{s} \rightarrow \mathbf{s}_{tmp}$;
  else
    for *i=1:G* **do**
      solve system for $\mathbf{s} \rightarrow \mathbf{s} + e_i$
    end
    choose **s** s.t.
    $\text{QoS}(\mathbf{s}) = \min_{i=1:G} \text{QoS}(\mathbf{s} + e_i)$;
  end
end

calculated as a side effect of the heuristic, which may be used to solve either (35) or (36).

Example

In this example the expected sojourn times, the expected waiting times and the idleness of average agents of group 1 and 2, respectively, have been investigated. The solutions are in terms of the expected sojourn times. The parameters used are as follows $G = 2, n_1 = n_2 = 2$ and both groups start with $s_1 = s_2 = 8$ agents each. Servers from the first group are more efficient than members of the second group when working with a single customer while the agents of group two have higher service rates for two concurrent clients. The arrival rate is given by $\lambda = 13.5$ and the service rates are $\mu_0^1 = 0, \mu_1^1 = 0.6, \mu_2^1 = 0.8, \mu_0^2 = 0, \mu_1^2 = 0.5, \mu_2^2 = 0.9$ and both types of agents have $C = 1$. The servers in Figure 5 will have a decreasing workload as more servers are added, with a jump when the system changes from adding servers to group two to group one, which can be seen in Figure 6. The complete set of values may be viewed in Table 1 and 2.

The example was chosen such that the heuristic would perform suboptimally. It seems the solution, both the optimal and the heuristic, behaves in accordance with the "law of diminishing returns" (see (Koole and Pot, 2011) for a discussion), i.e., the QoS measure is integer convex in the number of servers. Furthermore, it can be noted that although the optimal

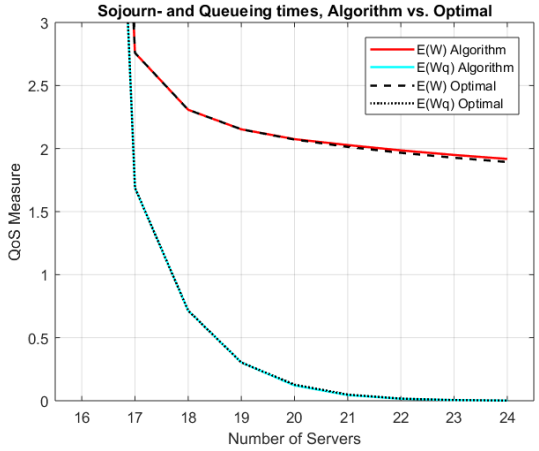

Figure 5: The expected sojourn times and the expected queueing times for the optimal solution as compared to the algorithm. Where E(W) is the expected sojourn time and E(Wq) the time waiting in the queue.

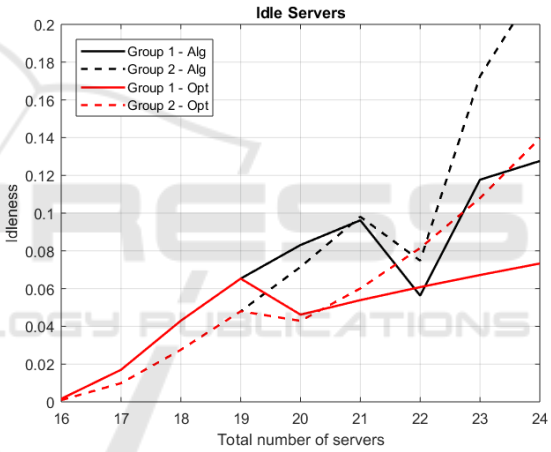

Figure 6: The percentage of the time agents from group 1 and 2 are idle for the algorithm solution and the optimal solution.

Table 1: The algorithm solution generated.

Agents	Qos $E(W)$	$E(W_q)$	Idleness grp1	grp2	Numbers grp1	grp2
16	11.7223	11.5862	0.0016	0.0009	8	8
17	2.7588	1.6885	0.0170	0.0099	8	9
18	2.3078	0.7150	0.0430	0.0276	8	10
19	2.1526	0.3037	0.0653	0.0480	8	11
20	2.0746	0.1214	0.0831	0.0715	8	12
21	2.0278	0.0450	0.0962	0.0981	8	13
22	1.9856	0.0158	0.0563	0.0748	9	13
23	1.9494	0.0051	0.1177	0.1725	10	13
24	1.9177	0.0015	0.1276	0.2228	11	13

solution generates solutions with lower sojourn times the waiting times in queue are lower for the heuristic solution which illustrates the point made in Section 4.1. The intuition is that the waiting times are more dependent on the total service provided when the concurrency level is maximized.

Table 2: The optimal solution generated.

Agents	Qos $E(W)$	$E(W_q)$	Idleness grp1	grp2	Numbers grp1	grp2
16	11.7223	11.5862	0.0016	0.0009	8	8
17	2.7588	1.6885	0.0170	0.0099	8	9
18	2.3078	0.7150	0.0430	0.0276	8	10
19	2.1526	0.3037	0.0653	0.0480	8	11
20	2.0718	0.1276	0.0462	0.0429	9	11
21	2.0132	0.0489	0.0539	0.0601	10	11
22	1.9664	0.0170	0.0608	0.0815	11	11
23	1.9271	0.0054	0.0672	0.1079	12	11
24	1.8934	0.0016	0.0733	0.1396	13	11

Looking at Figure 6 it can be seen that the work distribution between the groups, in terms of time spent in the idle state, is quite uneven. Thus a fairness measure might be relevant to mitigate some of that effect.

5 SUMMARY AND CONCLUSIONS

We have shown how a queueing system, where the servers handle several tasks simultaneously and where the total service rate for a server varies with the number of concurrent jobs handled, can be constructed. The construction is general under the imposed conditions of the system being in steady state, irreducible and Markovian. It can be constructed in such a way that each agent is considered to be its own group, which means that the impact of each agent on the system can be measured and estimated. However the number of system states will increase very quickly which in practice will limit the number of groups that can be considered. Also it would, in most cases, be hard to find sufficient data to estimate each agent separately to any degree of precision. This system can be controlled in two ways, by assigning agents and by means of the routing rules.

It is shown that it is sufficient to solve a smaller system of linear equations than the whole system, to calculate the steady state probabilities. The size of this smaller system is $\mathbb{R}^{N\times N}$. Once this smaller system has been solved, the rest of the state probabilities can be calculated recursively by a given formula.

By introducing a measure of the quality of service we can say something about how the system performs under different conditions. By using the average system time for a customer as the measure of QoS, it is shown how the optimal choice of maximum number of simultaneous tasks should be chosen to minimize average customer sojourn time. The solution is compared to a heuristic method which is found to provide results close to the true optimum.

A brief comparison between QoS measures of traditional call centers and that of chat systems is included where the conclusion is that traditional measures must be handled sensibly or poor system performance may result.

REFERENCES

Avi-Itzhak, B. and Halfin, S. (1989). Response times in gated m/g/1 queues: The processor-sharing case. *Queueing Systems*, 4(3):263–279.

Boucherie, R. J. (1993). Aggregation of markov chains. *Stochastic Processes and their Applications*, 45(1):95–114.

Cohen, J. (1979). The multiple phase service network with generalized processor sharing. *Acta Informatica*, 12(3):245–284.

Gans, N., Koole, G., and Mandelbaum, A. (2003). Telephone call centers: Tutorial, review, and research prospects. *Manufacturing Service Oper. Management*.

Gupta, V. and Zhang, J. (2014). Approximations and optimal control for state-dependent limited processor sharing queues.

Kleinrock, L. (1967). Time-shared systems: a theoretical treatment. *Journal of the ACM (JACM)*, 14(2):242–261.

Koole, G. and Pot, A. (2011). A note on profi
t maximization and monotonicity for inbound call centers. *Operations Research*, (59(5)):1304–1308.

Rege, K. M. and Sengupta, B. (1985). Sojourn time distribution in a multiprogrammed computer system. *AT&T Technical Journal*, 64(5):1077–1090.

Svensson, G. (2018). Product and computer system for a chat based communication system. US 10009468 B1, USA.

Whitt, W. and Halfin, S. (1981). Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29(3):567–588.

Yamazaki, G. and Sakasegawa, H. (1987). An optimal design problem for limited processor sharing systems. *Management Science*, 33(8):1010–1019.