EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems
Research Article EAI.EU

Optimisation of Server Selection for Maximising Utility in Erlang-Loss Systems

Maciej Pietowski[1], Quoc-Tuan Vien[2], Truong Khoa Phan[3]

[1]Research Computing Services, University of Cambridge, UK
[2]Faculty of Science and Technology, Middlesex University, UK
[3]Department of Electronic and Electrical Engineering, University College London, UK

Abstract

This paper undertakes the challenge of server selection problem in Erlang-loss system (ELS). We propose a novel approach to the server selection problem in the ELS taking into account probabilistic modelling to reflect a practical scenario when user arrivals vary over time. The proposed framework is divided into three stages, including i) developing a new method for server selection based on the $M/M/n/n$ queuing model with probabilistic arrivals; ii) combining server allocation results with further research on utility-maximising server selection to optimise system performance; and iii) designing a heuristic approach to efficiently solve the developed optimisation problem. Simulation results show that by using this framework, Internet Service Providers (ISPs) can significantly improve QoS for better revenue with optimal server allocation in their data centre networks.

Received on 21 October 2019; accepted on 31 October 2019; published on 05 November 2019

Keywords: Utility function, Erlang-Loss System (ELS), Server Selection

doi:10.4108/eai.24-10-2019.161367

1. Introduction

Although a variety of technologies [1] have emerged to provide enhanced services, a tremendously increasing number of users causes heavy congestion when many users simultaneously access the limited network resources, e.g. shared databases or file servers. The users suffering from either limited access or poor performance might decide to stop using the offered services, which leads to decreasing revenue for service providers. A common-sense solution is to simply provide more resources; however, it is not preferable as it would trigger high cost as well as a waste of resources. Therefore optimising available resources for a maximal network utility is a task of great importance to network designers.

With the continuing growth of a number of applications consuming high bandwidth such as video streaming, online video gaming, etc., the limited spectrum resources would be drained quickly causing serious traffic problems even on very well-design/maintained networks [2, 3]. Prioritising diverse types of traffic based on Quality-of-Service (QoS) is critical to managing the access to resources [4]. Specifically, the levels of QoS priorities can help to ensure applications to receive adequate bandwidth for acceptable performance in accordance with the required QoS from the users. For instance, the highest QoS priority would be reserved for emergency services like remote surgery and natural disaster, while the lowest QoS is generally assigned to the voice or data calls.

As user population having access to Internet services expands rapidly, it is necessary to continuously improve the accessibility of Internet resources by deploying multiple, distributed server sites or implementing cloud-based systems. The distribution of Internet services across the network allows us to place the servers geographically closer to the end-users, which accordingly improves the QoS and also enhances the service scalability by sharing the load among several server farms. However, this raises a principal issue of how to direct clients to the most appropriate service locations. Selecting a random server may result in compromising QoS, e.g. extensive delays due to distance and load on the servers. A correctly designed server selection mechanism should consider

*Corresponding author. Email: Mp929@cam.ac.uk

certain important factors, such as service independence, support of multiple policies, high availability, wide-area load balancing or low overhead. On the other hand, service providers should be able to provide a minimum number of servers to guarantee good QoS for all the user requests.

Dealing with the client-server model, a well-known queuing system, namely the Erlang-loss system (ELS), has been largely used to model telecommunication applications. In the ELS, users are not allowed to enter the system if all the servers are busy. In other words, the maximum number of users that the ELS can serve is not more than the number of available servers and the arriving users will be blocked if there is no available server. The ELS has been considered in various communication systems where the system capacity is generally restricted by the number of servers.

In brief, the main contributions of our work are as follows:

- We develop a new method for server selection based on the $M/M/n/n$ queuing model with probabilistic arrivals. Specifically, two cases of the interarrival of users are considered, which include i) independent and identically distributed (i.i.d.) and ii) independent and non-identically distributed (i.n.d.) interarrivals. We then analyse and evaluate blocking probability and server utilisation for all the cases.

- We combine server selection results with further research on the utility-based solution to optimise system performance. An optimisation problem is formulated to find the optimal number of servers so as to maximise utility per server (UpS)[1] subject to blocking probability, server utilisation, and network latency constraints.

- As this is an NP-hard problem, we design a heuristic approach to efficiently solve the problem. Based on simulation results, we show that the algorithm works well with different network conditions while maximising system performance.

This paper is organised as follows: we start by surveying related work in Section 2. We then present the ELS and performance analysis in Section 3. In Section 4, we introduce our QoS utility-based model and extend the model to the ELS in Section 5. We evaluate the algorithm in Section 6 and conclude the work in Section 7.

[1]The UpS is defined as the total network utility over the number of utilised servers.

2. Related Works

2.1. Erlang-Loss System

The $M/M/n/n$ queue is the most recognisable system in queuing theory, and it has been the selected area of study by many researchers. One of the early examinations of the ELS with state-dependent arrival and service rates was investigated by Brumelle in the late 70s [5]. CPU shared system and "birth-death" process were considered, where the state of the systems also consider the number of busy servers, the time until the next arrival and the remaining workload for the customers in the system. Krishnan [6] presented a simple and straightforward alternative to obtain results on convexity for arrival and service rates in the ELS and stopover in the Erlang delay system. In [7], Iversen & Mirtchev proposed a simple Erlang B model that can be used in a tele-traffic system for full accessibility with a generalised Poisson input stream. The idea in [7] was situated on the logic continuation of the Poisson distribution and the Erlang B formula. Procedures based on birth and death processes and state-dependent arrival rates were adopted verifying the simplicity and uniformity of the Erlang B model in representing both heavy and low traffic, and this makes the model attractive for modelling traffic, network evaluation and analysis.

A different approach to the $M/M/n/n$ can be found in [8], where Yao & Knessl outlined two parallel $M/M/n/n$ queues with n servers in each queue and no waiting lines. If both have the same occupancy, then the arrival is routed to any of them. Asymptotic approximations to the joint steady-state distribution of finding m and n servers occupied in the first and in the second queue were obtained along with finding the minimum allocation of the number of busy servers in the second queue. The Erlang loss model can be used for tackling the issues of clogging in a shared resource, that has been researched extensively, for instance in [9–13]. This problem has been modelled as a single-class ELS or multi-class ELS model for both single link and network. It is well-known that the product-form solution exists for the multi-class ELS model of a single link [10]. Kaufman [9] and Roberts [14] independently discovered one-dimensional recursive formula for computing the blocking probability, which simplified the computation. Based on [10], Nilsson et al. [11] proposed a more stable algorithm to compute blocking probability. As for provisioning purpose, Hampshire et al. [12] introduced a valid approach to the issues mentioned above.

Examining utilisation and Erlang traffic theory in asynchronous networks based on IP technology, Kavacky, et al. [15] proposed a test network model with a video traffic source. The difference between

two-link shaping methods was identified, and a start-time fair queuing was shown to be more suitable for the video traffic, establishing that the Erlang model is an appropriate choice for Variable Bit Rate traffic calculations on losses. Another research on the Erlang model has been applied for packet loss estimation in VoIP networks [16], where Misuth & Baronak presented a way of calculating the values of input variables, i.e. traffic load in Erlangs and number of lines, based on the characteristics of the codec in use and link interconnecting communication nodes. These values help to obtain packet loss probability, and their expectations were verified by simulations. It was also shown in [16] that the buffer utilisation in network nodes could positively influence the measured packet loss probability, and the Erlang B model could be used to determine the upper bound of packet loss probability in the general worst-case scenario where no buffer is available.

Considering the $M/M/n/n$ queue with two types of arrival rates and various levels of priorities, Smith et al., [17] evaluated all priority frameworks and derived the average number of primary and secondary users and the blocking probabilities related to them. The steady-state probabilities were derived for the case of not given priority to the primary user. The derived expressions were simplified in a short form, which can also lead to straightforward results for the performance metrics. In the case where priority is allocated to the dominant users, the second users might have to drop a resource with a dropping probability. The ELS is also being used in cloud computing and traditional data centres. Recently, Li, 2016 [18] has studied the common resource capacity management problem in theoretical loss network systems with the applications in cloud computing. A stochastic optimisation framework was developed to find the optimal capability for diverse types of resources. Considering the Erlang fixed-point approximation, a new quality-efficiency-driven fixed-point approximation for blocking probability was obtained, and the optimisation formed on the quality-efficiency-driven approximation can be solved with an iterative algorithm. *In this paper, we design a novel $M/M/n/n$ queuing model for server selection problem with probabilistic arrivals including i.i.d. and i.n.d. interarrivals.* While the former is generally assumed to simplify the analysis, the latter reflects well the practical scenarios when the users may arrive with different rates at different time.

2.2. Utility Function

The utility function measures user satisfaction and it is a function of received QoS. Most of the work in literature considers the utility function as a non-decreasing function of the effective transmission rate (bandwidth) [19–21] or signal to interference ratio (SIR) of occurring connections [22]. This paper considers the utility function as a non-increasing function of latency. The function includes a special first region $[0, T_{min}]$ where the utility score stays the same, so this presents a better way of how services are operating. Utility-based optimisation improves comprehensive system performance. However, there is no, or just a few real functional solutions were deployed so far due to their complexity [21, 22]. For example, the proposed algorithms in [21] require alteration of the TCP stack at the end-hosts. The utility function presented in the paper does not involve any modification in the TCP stack at the host-end so can be conveniently deployed in actual networks.

Many cloud services are running on geographically distributed data centres for better reliability and performance, which depends on server selection. Xu & Li [23] considered the emerging problem of joint request mapping and allocating with distributed data centres. A general convex optimisation was formulated, where the location is linked with the performance and costs. An efficient distribution algorithm was proposed to decompose a large-scale global problem into many sub-problems. Carrera et al., [24] presented a system that automatically allocated resource to clustered web applications. It is established on a utility-driven application distribution algorithm to achieve equalised satisfaction across applications.

Utility-maximisation server selection is a fundamental problem to tackle because users want to have access to resources in the most efficient way. Phan et al. [25, 26] presented a method for selection of replicated servers distributed over a wide area, allowing applications and network providers to trade-off costs with QoS for their users. Compared to the closest server selection approach, the proposed utility framework in [25] helps to reduce blocking probability while maintaining excellent utility for users. A polynomial optimisation algorithm was developed to allocate user service requests to servers located on the utility while satisfying transit cost constraint and an efficient low-overhead distributed model was proposed to deal with a small-scale subset of data requirements. Yuan et al., [27] proposed a user-oriented QoE-driven multimedia service delivering a solution in the context of the content delivery network (CDN) architecture. The major improvement of the QoE-based server selection strategy is taking both the underlying network conditions and video quality into account. The experiment results in [27] confirmed that the proposed strategy based on neural network achieves significant improvements regarding user perception compared to traditional QoS-based methods. *In this paper, we extend the work in [25, 26] by considering $M/M/n/n$ queuing model together with the utility-maximising problem. This helps to identify*

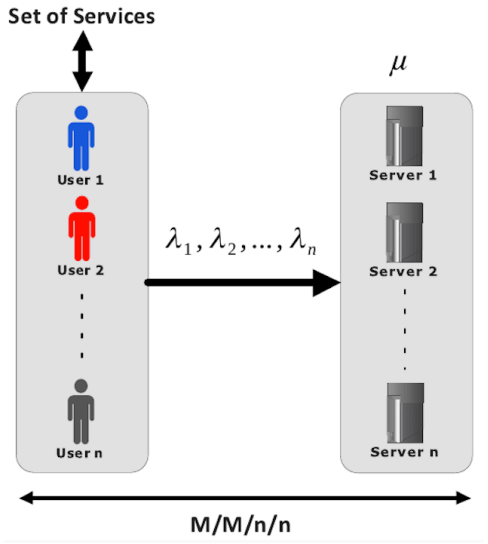


Figure 1. Erlang-Loss System (ELS).

the optimal number of servers, and thus it is expected to save a considerable resource instead of deploying all available servers in the system.

3. Erlang-Loss System and Its Performance Analysis

Figure 1 illustrates an ELS model where users only enter the system when the servers are available and the number of users that can be served is not more than the number of available servers. If there is no available server, then these users will be blocked or lost to the system. The ELS is generally described via Kendall's notation as $M/M/n/n$ which represents an n-server queue with Poisson arrivals, exponentially distributed service time, finite system capacity with a maximum of n users in the system, infinite population and first-in first-out discipline.

In this paper, we consider two cases of the interarrivals of the users, including i) independent and identically distributed (i.i.d.) interarrivals and ii) independent and non-identically distributed (i.n.d.) interarrivals. For brevity, let us denote the ELS with i.i.d. interarrivals and the ELS with i.n.d. interarrivals by (S1) and (S2), respectively. Suppose there are T time frames in a day and probabilistic arrivals are considered where the probability that there are user(s) coming in the i-th time frame, $i = 1, 2, \ldots, T$, is α_i satisfying $\sum_{i=1}^{T} \alpha_i = 1$.

Let us denote λ_i as the arrival rate of users within the i-th time frame. Two systems (S1) and (S2) can be defined as follows:

- *System (S1)*: the interarrivals between users within T time frames are i.i.d. following exponential distribution with an identical arrival rate of $\lambda_1 = \lambda_2 = \cdots = \lambda_T = \lambda$.
- *System (S2)*: the interarrivals between users within T time frames are i.n.d. following exponential distribution with different arrival rates of $\lambda_i \neq \lambda_j, \forall i \neq j, \{i, j\} \in \{1, 2, \ldots, T\}$.

In both systems, the service time at a server follows an exponential distribution with a service rate of μ. Let ρ_i, $i = 1, 2, \ldots, T$, denote the traffic intensity in the i-th time frame which is defined as a measure of the ability of a server to serve requests within the i-th time frame. The traffic intensity of system (S1) with i.i.d. arrivals is thus given by

$$\rho = \frac{\lambda}{\mu}, \tag{1}$$

while the traffic intensity of system (S2) in the i-th time frame with i.n.d. arrivals is

$$\rho_i = \frac{\lambda_i}{\mu}. \tag{2}$$

In the following, we analyse blocking probability and server utilisation in order to evaluate the performance of these two systems. In the $M/M/n/n$ queue, the blocking probability is defined as a steady-state probability that there are n users in the system and server utilisation is defined as a percentage of time that a server is busy serving requests from the users.

3.1. Performance Analysis of System (S1)

Following birth-death process in system (S1), the steady-state probability that there are k users, $k = 1, 2, \ldots, n$, in the i-th time frame, $i = 1, 2, \ldots, T$, denoted by $P_{k,i}^{(1)}$, can be determined by

$$P_{k,i}^{(1)} = P_{0,i}^{(1)} \frac{\lambda_i^k}{k!\mu^k} \overset{(a)}{=} P_{0,i}^{(1)} \frac{\lambda^k}{k!\mu^k} = P_{0,i}^{(1)} \frac{\rho^k}{k!}, \tag{3}$$

where $P_{0,i}^{(1)}$ denotes the steady-state probability that system (S1) is idle in the i-th time frame and (a) is due to the i.i.d. arrivals, i.e. $\lambda_i = \lambda$. With a note that $\sum_{j=0}^{n} P_{j,i}^{(1)} = 1$, it can be arrived at

$$P_{0,i}^{(1)} = \left(\sum_{j=0}^{n} \frac{\rho^j}{j!} \right)^{-1}. \tag{4}$$

Substituting (4) into (3), we obtain

$$P_{k,i}^{(1)} = \frac{\frac{\rho^k}{k!}}{\sum_{j=0}^{n} \frac{\rho^j}{j!}}. \tag{5}$$

Considering T time frames, the probability that there are k users in the system is given by

$$P_k^{(1)} = \sum_{i=1}^{T} \alpha_i P_{k,i}^{(1)} \overset{(a)}{=} \frac{\frac{\rho^k}{k!}}{\sum_{j=0}^{n} \frac{\rho^j}{j!}}, \tag{6}$$

where (a) is due to the fact that $\sum_{i=1}^{T} \alpha_i = 1$ and $P_{k,i}^{(1)}$ in (5) is independent of the time frame i.

By definition, the blocking probability of system (S1) is thus determined by

$$P_n^{(1)} = \frac{\frac{\rho^n}{n!}}{\sum_{j=0}^{n} \frac{\rho^j}{j!}}. \tag{7}$$

Let $U_A^{(1)}$ denote the utilisation of all servers in system (S1). $U_A^{(1)}$ is also known as the average number of busy servers over T time frames. From (6), $U_A^{(1)}$ can be computed by

$$U_A^{(1)} = \sum_{k=1}^{n} k P_k^{(1)} = \sum_{k=1}^{n} \frac{\frac{\rho^k}{(k-1)!}}{\sum_{j=0}^{n} \frac{\rho^j}{j!}}. \tag{8}$$

The server utilisation in system (S1), denoted by $U_S^{(1)}$, is therefore given by

$$U_S^{(1)} = \frac{U_A^{(1)}}{n} = \frac{1}{n} \sum_{k=1}^{n} \frac{\frac{\rho^k}{(k-1)!}}{\sum_{j=0}^{n} \frac{\rho^j}{j!}}. \tag{9}$$

3.2. Performance Analysis of System (S2)

Considering system (S2), the steady-state probability that there are k users, $k = 1, 2, \ldots, n$, in the i-th time frame, $i = 1, 2, \ldots, T$, i.e. $P_{k,i}^{(2)}$, can be similarly given by

$$P_{k,i}^{(2)} = P_{0,i}^{(2)} \frac{\lambda_i^k}{k! \mu^k} = P_{0,2}^{(1)} \frac{\rho_i^k}{k!}. \tag{10}$$

Here, $P_{0,i}^{(2)}$ denotes the steady-state probability that system (S2) is idle in the i-th time frame, which can be deduced as

$$P_{0,i}^{(2)} = \left(\sum_{j=0}^{n} \frac{\rho_i^j}{j!} \right)^{-1}. \tag{11}$$

Substituting (11) into (10), we obtain

$$P_{k,i}^{(2)} = \frac{\frac{\rho_i^k}{k!}}{\sum_{j=0}^{n} \frac{\rho_i^j}{j!}}. \tag{12}$$

Over T time frames, the probability that there are k users in the system (S2) is also given by

$$P_k^{(2)} = \sum_{i=1}^{T} \alpha_i P_{k,i}^{(2)} = \sum_{i=1}^{T} \frac{\alpha_i \frac{\rho_i^k}{k!}}{\sum_{j=0}^{n} \frac{\rho_i^j}{j!}}. \tag{13}$$

The blocking probability of system (S2) is therefore obtained by

$$P_n^{(2)} = \sum_{i=1}^{T} \frac{\alpha_i \frac{\rho_i^n}{n!}}{\sum_{j=0}^{n} \frac{\rho_i^j}{j!}}. \tag{14}$$

Similarly, the utilisation of all servers in system (S2), i.e. $U_A^{(2)}$, can be computed by

$$U_A^{(2)} = \sum_{k=1}^{n} k P_k^{(2)} = \sum_{k=1}^{n} \sum_{i=1}^{T} \frac{\alpha_i \frac{\rho_i^k}{k!}}{\sum_{j=0}^{n} \frac{\rho_i^j}{j!}} \tag{15}$$

and the server utilisation in system (S2), i.e. $U_S^{(2)}$, is given by

$$U_S^{(2)} = \frac{U_A^{(2)}}{n} = \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{T} \frac{\alpha_i \frac{\rho_i^k}{k!}}{\sum_{j=0}^{n} \frac{\rho_i^j}{j!}}. \tag{16}$$

4. QoS Utility-based Model

Inspired by the work in [25] and [26], our QoS utility-based model (QUM) is considered to implement server selection for an ELS subject to QoS requirements. In the QUM, there are basically two types of service providers, including i) *Application Service Providers (ASPs)* who are organisations providing cloud-based applications to their users from either their facilities or another source, and ii) *Internet Service Providers (ISPs)* who resolve user requests by implementing various resolution algorithms. The ASPs employ a utility function of the service that can be understood as the latency requirements for the services, while the ISPs compromise the QoS with the traffic cost.

Prior to introducing the utility function in the QUM, let us consider the following example:

Example 1. *As shown in Fig. 2, two servers are deployed to provide services for two users following a M/M/2/2 queuing model with a finite system capacity of two users. It is assumed that both User 1 and User 2 require a voice service over the Internet which can be provided by either Server 1 or Server 2 and the latency* $t_i^{(j)}$, $\{i, j\} \in \{1, 2\}$, *between User i and Server j are set as* $t_1^{(1)} = 7$ *ms,* $t_1^{(2)} = 20$ *ms,* $t_2^{(1)} = 20$ *ms and* $t_2^{(2)} = 22$ *ms. Each server, however, can serve only one user at a time. This accordingly raises an issue which server should be selected to serve a user.*

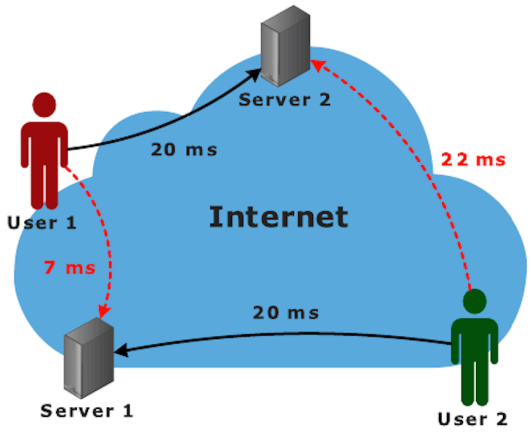


Figure 2. An Example of QoS Utility-based Model (QUM).

Employing the traditional minimum-distance based selection algorithm, the optimal solution should be User 1 - Server 1 and User 2 - Server 2 as the minimum total latency of 29 ms. Notice that there is almost no difference between audio and real speech if latency is less than or equal to 20 ms [26]. In this scenario, User 1 receives the best QoS whereas User 2 might receive some disruption in the voice quality as it is above the safe latency threshold (20 ms) for voice services. Therefore, a better solution should be: User 1 to acquire the service from Server 2, suffering a latency of 20 ms while still maintaining the best QoS of voice service. In return, User 2 is served by Server 1 with a latency of 20 ms. This means that both users can obtain the best QoS using this server selection solution.

The observation in Example 1 leads to the definition of the utility function for each pair of user and service, which can be represented by a linear function of the latency. Assume there exists a k-th server among n servers to serve a j-th service, $j = 1, 2, \ldots, N_S$, of a coming i-th user, $i = 1, 2, \ldots, N_U$, where N_S and N_U denote the number of services and the number of users, respectively. We have the following definition of the utility function for each pair of user and service [26]:

Definition 1. *The utility function in a QUM for a pair of the i-th user, $i = 1, 2, \ldots, N_U$, and the j-th service, $j = 1, 2, \ldots, N_S$, is given by*

$$u_{ij} \triangleq \begin{cases} \frac{1}{\delta_{ij}} & \text{if } t_{ij} < T_{ij}^{(l)}, \\ \frac{T_{ij}^{(u)} - t_{ij}}{\delta_{ij}(T_{ij}^{(u)} - T_{ij}^{(l)})} & \text{if } T_{ij}^{(l)} \le t_{ij} \le T_{ij}^{(u)}, \\ 0 & \text{if } t_{ij} > T_{ij}^{(u)}, \end{cases} \tag{17}$$

where $\delta_{ij} \ge 1$ is the priority level of the j-th service requested by the i-th user, t_{ij} is the network latency for the i-th user to have the j-th service, and $T_{ij}^{(l)}$ and $T_{ij}^{(u)}$ are the lower and upper thresholds, respectively, of the latency for the required QoS.

Remark 1. *It can be noticed in (17) that the reciprocal of the priority level means a higher utility is expected when a user requests a service with a higher priority level. This reflects the practical network perspective when different users require the same service, then the user having a higher priority but achieving the same latency should achieve a higher utility.*

Remark 2. *Given a fixed service priority level, i.e. fixed δ_{ij}, the utility function in (17) consists of three cases which can be interpreted with respect to two thresholds $T_{ij}^{(l)}$ and $T_{ij}^{(u)}$ of the latency as follows:*

- *Case 1 ($t_{ij} < T_{ij}^{(l)}$): An "excellent" QoS is achieved with a high utility. In this case, a network latency lower than $T_{ij}^{(l)}$ provides a higher utility approaching the best performance. However, such performance enhancement is unnoticeable, and thus it can be regarded as of the same QoS.*
- *Case 2 ($T_{ij}^{(l)} \le t_{ij} \le T_{ij}^{(u)}$): The network latency is in an acceptable range providing a QoS ranked from "good" to "poor". The "fair" point can be included for further ranking depending on services, though it does not cause any change on the gradient of the graph.*
- *Case 3 ($t_{ij} > T_{ij}^{(u)}$): The latency is unacceptable with "no service". In other words, service request is blocked if the network latency exceeds $T_{ij}^{(u)}$.*

5. QUM-based Server Selection in the ELS

In this section, we first present the network design aspects required for server selection taking into account various performance measures in the ELS along with those in the QUM. An optimisation problem is then developed, which aims to maximise UpS by implementing a QUM-based server selection in the ELS. For convenience, notations used throughout the paper are summarised in Table 1 in chronological order of their appearances.

5.1. Network Design Aspects

Adopting QUM to the ELS, QoS constraints in both the ELS and QUM need to be taken into account. It is crucial to consider the following network design aspects:

Blocking Probability. In order to guarantee that all users can be served in the ELS, the number of user arrivals to the system should not be more than the number of servers. Otherwise, they will be dropped or blocked. This means that the blocking probability should not exceed a QoS threshold. Let P_{thre} denote the blocking probability threshold. We then have the following constraint on the blocking probability, i.e.

Table 1. Summary of main notations

Notation	Meaning
T	number of time frames in a day
λ_i	arrival rate of users in the i-th time frame
α_i	probability of users arriving in the i-th time frame
μ	service rate at server
ρ_i	traffic intensity in the i-th time frame
n	number of servers
$P_n^{(m)}$	blocking probability of the m-th system
$U_S^{(m)}$	server utilisation in the m-th system
N_S, N_U	number of users and services, respectively
$\mathfrak{U}$, $\mathfrak{S}$, $\mathfrak{N}$	user, service and server sets, respectively
u_{ij}	utility function of a pair {i-th user, j-th service}
δ_{ij}	priority level of the j-th service requested by the i-th user
t_{ij}	network latency for the i-th user to have the j-th service
$t_{ij}^{(k)}$	network latency for the i-th user to have the j-th service from the k-th server
$x_{ij}^{(k)}$	fraction of the i-th user with the j-th service processed by the k-th server
a_{ij}	auxiliary parameter of a pair {i-th user, j-th service}
$T_{ij}^{(l)}$, $T_{ij}^{(u)}$	lower and upper thresholds, respectively, of the network latency
P_{thre}	blocking probability threshold
$U_{S,thre}$	server utilisation thresholds
$N_{\max}$	maximum number of available servers

$P_n^{(m)}$, $m \in \{1, 2\}$, for each of two systems (S1) and (S2):

$$P_n^{(m)} \le P_{thre}, \tag{18}$$

where $P_n^{(1)}$ and $P_n^{(2)}$ are given by (7) and (14), respectively.

Server Utilisation. Due to the limitation of the server, it may not spend all resources to serve requests from the users. Its utilisation is subject to a specified QoS constraint. Let us denote the server utilisation threshold by $U_{S,thre}$. The server utilisation in an ELS ($U_S^{(m)}$), $m \in \{1, 2\}$, should satisfy:

$$U_S^{(m)} \le U_{S,thre}, \tag{19}$$

where $U_S^{(1)}$ and $U_S^{(2)}$ are given by (9) and (16), respectively.

User-Service Allocation. It is expected that all requests from the i-th user, $i \in \mathfrak{U} = \{1, 2, \ldots, N_U\}$, should be served by at least one server among n available servers. Let $x_{ij}^{(k)}$, $0 \le x_{ij}^{(k)} \le 1$, denote the fraction of the i-th user with the j-th service, $j \in \mathfrak{S} = \{1, 2, \ldots, N_S\}$, which is processed by the k-th server, $k \in \mathfrak{N} = \{1, 2, \ldots, n\}$. We accordingly have the following constraint:

$$\sum_{k=1}^{n} x_{ij}^{(k)} = 1. \tag{20}$$

Network Latency. The latency for the i-th user, $i \in \mathfrak{U}$, to get the j-th service, $j \in \mathfrak{S}$, from the k-th server, $k \in \mathfrak{N}$, should be not exceeding $T_{ij}^{(u)}$, i.e.

$$t_{ij}^{(k)} \le T_{ij}^{(u)}. \tag{21}$$

Let $a_{ij} \triangleq t_{ij} - T_{ij}^{(l)}$, where t_{ij} is the network latency given by:

$$t_{ij} = \sum_{k=1}^{n} t_{ij}^{(k)} x_{ij}^{(k)}. \tag{22}$$

The utility function in (17) for the case when $T_{ij}^{(l)} \le t_{ij} \le T_{ij}^{(u)}$ can be rewritten as:

$$u_{ij} = \frac{T_{ij}^{(u)} - T_{ij}^{(l)} - a_{ij}}{\delta_{ij}(T_{ij}^{(u)} - T_{ij}^{(l)})}. \tag{23}$$

Here, a_{ij} is regarded as an auxiliary parameter which can be used to compute the utility function for a pair of the i-th user and the j-th service as:

$$a_{ij} = \left(1 - u_{ij}\delta_{ij}\right)\left(T_{ij}^{(u)} - T_{ij}^{(l)}\right). \tag{24}$$

In order to maintain an acceptable network latency, a_{ij} should satisfy the following constraint:

$$\begin{aligned} a_{ij} &\ge \left(t_{ij} - T_{ij}^{(l)}\right)^{+} \\ &= \left(\sum_{k=1}^{n} t_{ij}^{(k)} x_{ij}^{(k)} - T_{ij}^{(l)}\right)^{+}, \end{aligned} \tag{25}$$

where $x^{+} \triangleq \max(x, 0)$.

5.2. Optimisation Problem for QUM-based Server Selection in the ELS

In the above network design aspects, the user-service allocation is shown to be critical in every network, especially when a number of constraints, either explicit or implicit, need to be taken into account. With the aim of efficiently allocating the users and their required services with an optimal number of servers, we can formulate an optimisation problem that maximises UpS

as follows:

$$\max_{n} \frac{\sum_{i\in\mathfrak{U},j\in\mathfrak{S}} u_{ij}}{n} \tag{26}$$

subject to the following constraints:

$$(C1): \quad P_n^{(m)} \le P_{thre}, m = 1,2 \tag{27}$$

$$(C2): \quad U_S^{(m)} \le U_{S,thre}, m = 1,2 \tag{28}$$

$$(C3): \quad a_{ij} \ge \left(\sum_{k=1}^{n} t_{ij}^{(k)} x_{ij}^{(k)} - T_{ij}^{(l)}\right)^{+}, \forall i \in \mathfrak{U}, j \in \mathfrak{S} \tag{29}$$

$$(C4): \quad t_{ij}^{(k)} \le T_{ij}^{(u)}, \forall i \in \mathfrak{U}, j \in \mathfrak{S}, k \in \mathfrak{N} \tag{30}$$

$$(C5): \quad 0 \le x_{ij}^{(k)} \le 1, \forall i \in \mathfrak{U}, j \in \mathfrak{S}, k \in \mathfrak{N} \tag{31}$$

$$(C6): \quad \sum_{k=1}^{n} x_{ij}^{(k)} = 1, \forall i \in \mathfrak{U}, j \in \mathfrak{S} \tag{32}$$

$$(C7): \quad 1 \le n \le N_{\max}, n \in \mathbb{Z}, \tag{33}$$

where $N_{\max}$ denotes the maximum number of available servers that can be used in the system. In the above optimisation problem, the constraints (C1)-(C6) are defined as in the network design aspects.

It can be noticed that the optimisation problem in (26) is an integer programming problem since its design variable is the number of servers which is restricted to be integer (see (C7)). This problem is thus an NP-hard problem. In the following, let us introduce an heuristic approach to solve the problem in (26).

We first put the constraints in groups and then sequentially consider each group of constraints as in the following steps:

- *Step 1*: From the QoS constraints in the ELS, this step finds the minimum number of servers, denoted by $N_{\min}$, required for the user-application services by solving

$$N_{\min} = \min n \tag{34}$$

$$\text{st. (C1), (C2), (C7).} \tag{35}$$

  With a limited number of available servers, i.e. $N_{\max}$ in constraint (C7), the minimum number of servers can be easily found by an iterative search. After this step, the constraints (C1) and (C2) in the optimisation problem in (26) can be removed, whereas the constraint (C7) is replaced by

$$(C8): \quad N_{\min} \le n \le N_{\max}. \tag{36}$$

- *Step 2*: Given the limited number of available servers, this step finds the optimal number of servers to maximise UpS in (26) subject to the constraints (C3)-(C6) and (C8). Notice that although some constraints of the ELS are relaxed, the optimisation problem is still in the NP-hard form with a stricter integer set of the number of servers. Therefore, an iterative searching approach can be employed, where we validate the total utility of all users and services with respect to different number of servers in the range $[N_{\min}, N_{\max}]$ until achieving the maximum UpS.

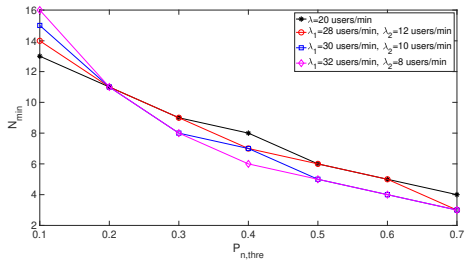
For clarity, the finding of the optimal number of servers with the proposed two-step iterative search for QUM-based server selection in the ELS is summarised in Algorithm 1.

Algorithm 1 Two-Step Iterative Search

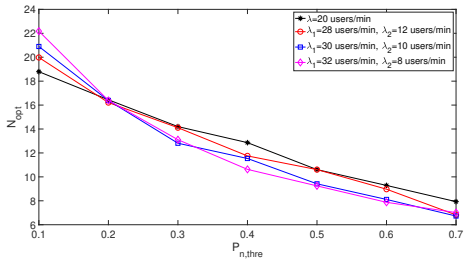
Input: P_{thre}, $U_{S,thre}$, $\{[T_{ij}^{(l)}, T_{ij}^{(u)}]\}$, $\{\alpha_i\}$, $\{\delta_{ij}\}$, $N_{\max}$
Step 1: Find the minimum number of servers
Set $n = 0$
repeat
  $n = n + 1$
  Find $P_n^{(m)}$ and $U_S^{(m)}$, $m = 1,2$, using (7), (14), (9) and (16)
until $P_n^{(m)} \le P_{thre}$ **and** $U_S^{(m)} \le U_{S,thre}$ (see (C1) and (C2))
$N_{\min} = n$
Step 2: Find the optimal number of servers
Set $UpS_{\max} = 0$, $n_{opt} = N_{\min}$
for $n = N_{\min}$ **to** $N_{\max}$ (see (C8)) **do**
  for all $i \in \mathfrak{U}$ **and** $j \in \mathfrak{S}$ **do**
    if $t_{ij}^{(k)} \le T_{ij}^{(u)}$ **and** $0 \le x_{ij}^{(k)} \le 1$ **and** $\sum_{k=1}^{n} x_{ij}^{(k)} = 1$ (see (C4), (C5), (C6)) **then**
      Find $t_{ij} = \sum_{k=1}^{n} t_{ij}^{(k)} x_{ij}^{(k)}$ (see (22))
      if $t_{ij} \ge T_{ij}^{(l)}$ **then**
        $a_{ij} = t_{ij} - T_{ij}^{(l)}$
      else
        $a_{ij} = 0$
      end if
      Find u_{ij} using (23)
    end if
  end for
  Compute $UpS = \sum_{i,j} u_{ij}/n$
  if $UpS \ge UpS_{\max}$ **then**
    $UpS_{\max} = UpS$
    $n_{opt} = n$
  end if
end for
Output: n_{opt}

6. Simulation Results

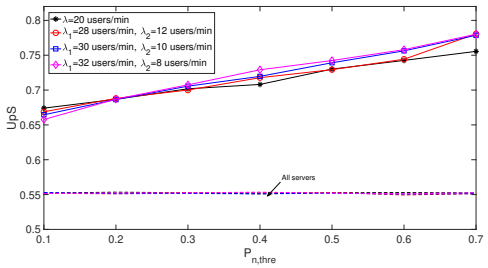
In this section, simulation results of QUM-based server selection in the ELS are presented. The



(a) Minimum number of servers vs. $P_{n,thre}$



(b) Optimal number of servers vs. $P_{n,thre}$



(c) UpS vs. $P_{n,thre}$

Figure 3. Impacts of user arrival rates

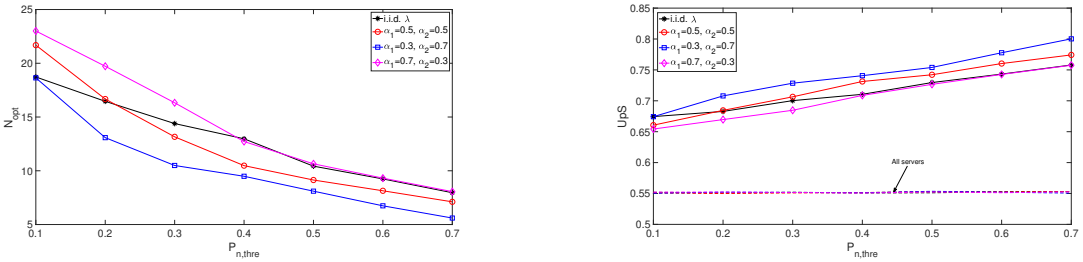
proposed algorithm is implemented and validated using MATLAB. Specifically, we first demonstrate the impacts of queuing parameters in the ELS, i.e. user arrival rates, user arrival probabilities, and service rate, on the optimal number of servers along with the maximum UpS that can be achieved with the QUM-based server selection. We then evaluate the performance of the proposed approach for different services with different latency requirements. Different server utilisation requirements are also taken into account to verify the effectiveness of the proposed algorithm. Furthermore, in the simulation, both i.i.d. and i.n.d. interarrivals with probabilistic arrivals are considered to show the practicability of employing QUM-based server selection.

6.1. Impacts of User Arrival Rates

Considering the impacts of user arrival rates on the system performance with the proposed two-step iterative search, Figs. 3a, 3b and 3c plot the minimum number of servers, i.e. $N_{\min}$, the optimal number of servers, i.e. N_{opt}, and UpS, respectively, versus blocking probability threshold, i.e. $P_{n,thre}$. In the simulation, the minimum and the optimal number of the servers are sequentially determined by employing Steps 1 and 2 in Algorithm 1, whereas the maximum UpS is obtained with the optimal number of servers. The parameters used in the simulations are as follows: $T^{(u)} = 150$ ms, $T^{(l)} = 20$ ms, $\delta = 1$, $\mu = 2$ users/min, $\alpha = [0.5, 0.5]$, $U_{S,thre} = 0.9$ and $N_{\max} = 40$. The arrival rate is set as $\lambda = 20$ users/min for i.i.d. arrival, while three different sets of arrival rates over two-time frames are considered for i.n.d. case, i.e. $\{(\lambda_1, \lambda_2)\} \in \{(28, 12), (30, 10), (32, 8)\}$ users/min. As shown in Figs. 3a and 3b, the minimum and the optimal number of required servers monotonically decreases as the blocking probability threshold increases. This is due to the fact that we allow increasing utility value in (26) by sacrificing blocking probability in (27) which is shown in Fig. 3c. It is also shown that a lower number of servers are required with i.n.d. arrivals on average when compared to the i.i.d. case and higher UpS is achieved with the proposed algorithm over the case when all servers are deployed. This accordingly means a considerable resource can be saved with probabilistic arrival modelling for enhanced UpS.

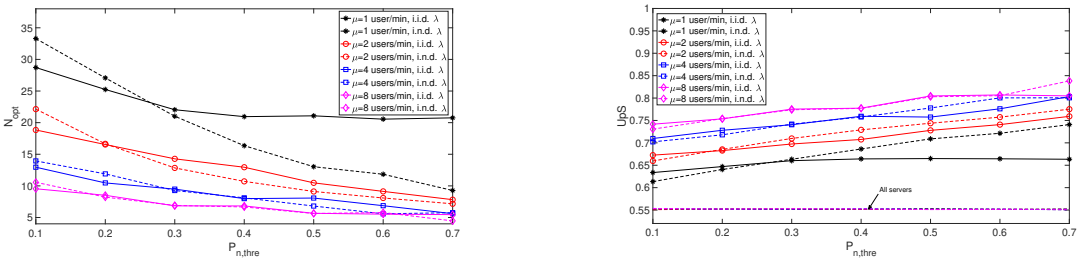
6.2. Impacts of User Arrival Probabilities

We evaluate the algorithm with different user arrival rates. Figs. 4a and 4b plot the optimal number of servers, i.e. N_{opt}, and the corresponding UpS, respectively, versus blocking probability threshold, i.e. $P_{n,thre}$, with respect to two scenarios of the interarrival of the users including i.i.d. and i.n.d. arrivals. The configurations we use are as follows: $T^{(u)} = 150$ ms, $T^{(l)} = 20$ ms, $\delta = 1$, $\mu = 2$ users/min, i.i.d. $\lambda = 20$ users/min, i.n.d. $\lambda = [32, 8]$ users/min, and $U_{S,thre} = 0.9$. $N_{\max} = 40$. For the i.n.d. arrivals, three sets of arrival probability are considered, including $\{\alpha_1, \alpha_2\} = \{(0.5, 0.5), (0.3, 0.7), (0.7, 0.3)\}$. Similarly, as shown in Fig. 4, the number of optimal servers reduces while UpS increases when we increase blocking probability



(a) Optimal number of servers vs. $P_{n,thre}$ **(b)** UpS vs. $P_{n,thre}$

Figure 4. Impacts of user arrival probabilities



(a) Optimal number of servers vs. $P_{n,thre}$ **(b)** UpS vs. $P_{n,thre}$

Figure 5. Impacts of service rates

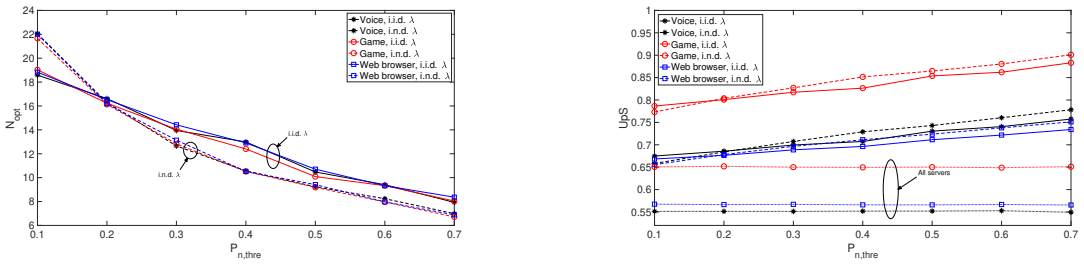
threshold $P_{n,thre}$. Additionally, it can be observed that, when the users arrive with a higher rate, it does not always mean that more servers are required. In fact, in probabilistic modelling, a lower probability of the user arrivals would result in less servers even the users arrive at a higher rate. Furthermore, the proposed algorithm is shown to save a considerable number of servers for a higher UpS in all cases of the arrival rates and their probabilities.

6.3. Impacts of Service Rate

We next simulate the impact of service rate on system performance. Figs. 5a and 5b plot the optimal number of servers, i.e. N_{opt}, and the corresponding UpS, respectively, against the blocking probability threshold, i.e. $P_{n,thre}$, with respect to various service rates. Both i.i.d. and i.n.d. arrivals are considered. Similar to Fig. 3, the simulation parameters are set as follows: $T^{(u)} = 150$ ms, $T^{(l)} = 20$ ms, $\delta = 1$, i.i.d. $\lambda = 20$ users/min, i.n.d. $\lambda = [32, 8]$ users/min, $\alpha = [0.5, 0.5]$, $U_{S,thre} = 0.9$, and $N_{\max} = 40$. It can be seen in Fig. 5 that the number of optimal servers monotonically decreases as a function of service rate, while still achieving a higher UpS for both cases of i.i.d. and i.n.d. arrivals. This indeed can be intuitively verified due to the increased service rate. Also, considering probabilistic modelling, the i.n.d. arrivals, which reflects well the practical scenario, are again shown to require a much lower number of servers when the blocking probability threshold increases.

6.4. Impacts of Latency Requirements in Different Services

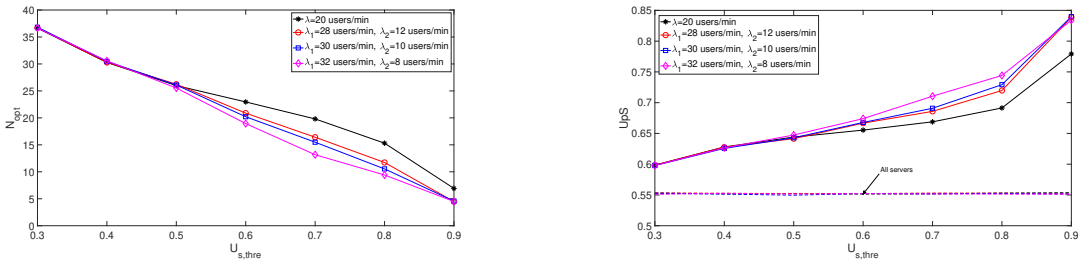
Evaluating the impacts of latency requirements in different services, Figs. 6a and 6b sequentially plot the optimal number of servers, i.e. N_{opt}, and the corresponding UpS versus the blocking probability threshold, i.e. $P_{n,thre}$, with the following simulation parameters: $N_{\max} = 40$, $\delta = 1$, i.i.d. $\lambda = 20$ users/min, i.n.d. $\lambda = [32, 8]$ users/min, $\alpha = [0.5, 0.5]$, and $U_{S,thre} = 0.9$. Three services, i.e. voice, game and web browser, are considered having the upper and lower thresholds of latency requirement defined by $T^{(u)} = [150, 100, 10000]$ ms and $T^{(l)} = [20, 50, 100]$ ms, respectively. As shown in Fig. 6, a smaller number of servers are required for i.n.d. arrivals of all kinds of service when compared to those for i.i.d. arrivals with no probabilistic modelling. Although the number of servers is approximately the same for all services, the game application achieves much higher UpS than both the voice and web browser applications. This is due to the fact that a tight latency requirement is required in the game application compared to that of other services, while the same number of servers can be deployed. Again, the proposed algorithm is shown to achieve higher UpS for all kinds of service with probabilistic modelling over the i.i.d. arrivals as well as the case of employing all servers. This accordingly verifies the effectiveness of the proposed algorithm for various delay-limited services.



(a) Optimal number of servers vs. $P_{n,thre}$

(b) UpS vs. $P_{n,thre}$

Figure 6. Impacts of latency requirements in different services



(a) Optimal number of servers vs. server utilisation threshold w.r.t. different user arrival rates

(b) UpS vs. server utilisation threshold w.r.t. different user arrival rates

Figure 7. Impacts of server utilisation requirements

6.5. Impacts of Server Utilisation Requirements

As regards server utilisation requirements, Fig. 7a and 7b plot the optimal number of servers and the corresponding UpS, respectively, versus server utilisation threshold, i.e. $U_{S,thre}$. The following parameters are used: $T^{(u)} = 150$ ms, $T^{(l)} = 20$ ms, $\delta = 1$, $\mu = 2$ users/min, $\alpha = [0.5, 0.5]$, $P_{n,thre} = 0.9$, and $N_{\max} = 40$. Similar to Fig. 3, the arrival rate is set as $\lambda = 20$ users/min for i.i.d. arrivals and three different sets for i.n.d. arrivals, i.e. $\{(\lambda_1, \lambda_2)\} \in \{(28, 12), (30, 10), (32, 8)\}$ users/min. It can be observed in Fig. 7 that the optimal number of required servers monotonically decreases as the server utilisation threshold increases. Also, a lower number of servers are required with probabilistic modelling for i.n.d. arrivals when compared to the i.i.d. case and a higher UpS is shown to achieve for all cases over the employment of all servers. This accordingly brings advantage to the proposed algorithm in resource saving with probabilistic arrival modelling.

7. Conclusions

In this paper, we have presented a novel utility-maximising server selection for an ELS, i.e. $M/M/n/n$ queuing model. Through simulation results, we have shown that the proposed algorithm works well with different network conditions to save a considerable number of servers for a high utility. In general, the minimum and the optimal number of the required servers monotonically decrease as the blocking probability threshold increases. More importantly, the probabilistic modelling with i.n.d. arrivals, which reflects well the practical scenario, has been shown to achieve a higher utility with a lower number of servers when compared to the i.i.d. arrival model. As future work, we will extend the utility function to support more QoS metrics along with the design of an online algorithm that can adapt the solutions to the changes of practical networks in real time.

References

[1] H. Q. Tran, C. V. Phan, Q.-T. Vien, An Overview of 5G Technologies, Emerging Wireless Communication and Network Technologies, Springer (2018).

[2] T. K. Phan, J. Moulierac, C. N. Tran, N. Thoai, Xcast6 Treemap Islands: Revisiting Multicast Model, in: ACM CoNEXT Student Workshop, 2012.

[3] J. Li, T. K. Phan, W. K. Chai, D. Tuncer, G. Pavlou, D. Griffin, M. Rio, DR-Cache: Distributed Resilient Caching with Latency Guarantees, in: IEEE INFOCOM, 2018 (2018).

[4] L. P. Thiruvasakan, Q.-T. Vien, J. Loo, G. Mapp, A QoS-Based Flow Assignment for Traffic Engineering in Software-Defined Networks, Advances in Intelligent Systems and Computing, Springer (2020).

[5] S. L. Brumelle, A Generalization of Erlang's Loss System to State Dependent Arrival and Service Rates,

Mathematics of Operations Research (1978).
[6] K. R. Krishnan, The Convexity of Loss Rate in an Erlang Loss System and Sojourn in an Erlang Delay System with Respect to Arrival and Service Rates, IEEE Transactions on Communications (1990).
[7] V. B. Iversen, S. T. Mirtchev, Generalised Erlang Loss Formula, Electronics Letters (1996).
[8] H. Yao, C. H. Knessl, On the Shortest Queue Version of the Erlang's Loss Model, Applied Mathematics (2008).
[9] J. Kaufman, Blocking in a Shared Resource Environment, IEEE Transactions On Communications (1981).
[10] F. Kelly, Blocking Probabilities in Large Circuit-switched Networks, Advance Applied Prob. (1986).
[11] A. A. Nilsson, M. J. Perry, A. Gersht, V. B. Iversen, On Multi-rate Erlang-B Computations, in: Proceedings of ITC, 1999.
[12] R. C. Hampshire, W. A. Massey, D. Mitra, Q. Wang, Provisioning for Bandwidth Sharing and Exchange, Telecommunications Network Design and Management (2003).
[13] K. Alnowibet, Nonstationary Erlang Loss Queues and Networks, Ph.D Dissertaion, North Carolina State University (2004).
[14] J. W. Roberts, A Service System with Heterogeneous User Requirements, in: Performance of Data Communications Systems and Their Applications, 1981.
[15] M. Kavacky, E. Chromy, J. Suran, Evaluation of Erlang Models in IP Network, in: ICETA, 2011.
[16] T. Misuth, I. Baronak, Application of Erlang B Model in Modern VoIP Networks, in: International Conference on Telecommunications and Signal Processing (TSP), 2011.
[17] P. J. Smith, A. Firag, P. A. Dmochowski, M. Shafi, Analysis of the M/M/N/N Queue with Two Types of Arrival Process: Applications to Future Mobile Radio Systems, Journal of Applied Mathematics (2012).
[18] J. Li, Optimal Resource Capacity Management for Stochastic Loss Network Systems with Applications in Clouds and Data Centers, in: Conference on Decision and Control (CDC), 2016.
[19] H. Chan, P. Fan, Z. Cao, A Utility-based Network Selection Scheme for Multiple Services in Heterogeneous Networks, in: International Conference on Wireless Networks, Communications and Mobile Computing, 2005.
[20] X. Duan, Z. Niu, J. Zheng, Utility Optimization and Fairness Guarantees for Multimedia Traffic in the Downlink of DS-CDMA Systems, in: IEEE GlobeCom, 2003.
[21] R. J. La, V. Anantharam, Utility-based Rate Control in The Internet for Elastic Traffic, IEEE/ACM Transactions on Networking (2002).
[22] M. Xiao, N. B. Shroff, E. K. P. Chong, A Utility-based Power Control Scheme in Wireless Cellular Systems, IEEE/ACM Transactions on Networking (2003).
[23] H. Xu, B. Li, Joint Request Mapping and Response Routing for Geo-distributed Cloud Services, in: INFOCOM, 2013.
[24] D. Carrera, M. Steinder, J. Torres, Utility-based Placement of Dynamic Web Applications with Fairness Goals, in: Network Operations and Management Symposium, 2008.
[25] T. K. Phan, D. Griffin, E. Maini, M. Rio, Utility-maximizing Server Selection, in: IFIP Networking, 2016.
[26] T. K. Phan, D. Griffin, E. Maini, M. Rio, Utility-centric Networking: Balancing Transit Costs with Quality of Experience, IEEE/ACM Transactions on Networking (2018).
[27] S. Yuan, T. Lin, S. Bai, S. Ci, User-Oriented QoE-Driven Server Selection for Multimedia Service Provisioning in Content Distribution Networks., in: Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015.