*electronics*

MDPI

*Article*

# Modeling and Performance Evaluation of Multi-Class Queuing System with QoS and Priority Constraints

**Fahad Mazaed Alotaibi [1,*], Israr Ullah [2] and Shakeel Ahmad [3]**

[1] Faculty of Computing and Information Technology (FCIT), King Abdulaziz University, Jeddah 21589, Saudi Arabia
[2] Department of Computer Science, Virtual University of Pakistan, Lahore 54000, Pakistan; israr.ullah@vu.edu.pk
[3] Faculty of Computing and Information Technology in Rabigh (FCITR), King Abdulaziz University, Jeddah 21589, Saudi Arabia; sarahmad@kau.edu.sa
* Correspondence: fmmalotaibi@kau.edu.sa; Tel.: +966-530-273-935

**Abstract:** Many service providers often categorize their users into multi-classes, depending on their service requirements. Each class has strict quality of service (QoS) demands (e.g., minimum required service rate or transfer time) that must be ensured throughout its service. In some cases, priorities are also assigned in a multi-class user's environment to ensure that the important class user shall be serviced first. In this paper, we have developed a novel Markov chain based analytical model to investigate and evaluate a multi-class queuing system with a strict QoS requirement and priority constraints. Experimental analysis is conducted for two users classes, i.e., class-1 (may be free/student users) and class-2 (may be paid/research users). Each class requests have strict QoS requirements in terms of the minimum required rate (MRR) that must be ensured throughout its lifetime once the request is admitted into the system. Secondly, class-2 requests have preemption priority over class-1, i.e., if there is no room for newly arriving class-2 requests, then one or more active flows of class-1 can be ejected in order to accommodate high-class requests. Model results are validated through simulation results and performance measures of our interest include blocking probability (BP) of individual classes and the overall system, effect of higher-class jobs on lower-class jobs, and link capacity utilization. The proposed model can be instrumental in developing advanced connection admission control (CAC), efficient resource dimensioning, and capacity planning of the queuing system.

**Keywords:** markov chains; performance evaluation; multi-service queuing system; QoS; deadline and priority constraints



**Citation:** Alotaibi, F.M.; Ullah, I.; Ahmad, S. Modeling and Performance Evaluation of Multi-Class Queuing System with QoS and Priority Constraints. *Electronics* **2021**, *10*, 500. https://doi.org/10.3390/electronics10040500

Academic Editor: Christos J. Bouras
Received: 29 December 2020
Accepted: 11 February 2021
Published: 20 February 2021



## 1. Introduction

Communication and information technologies have made tremendous growth in the recent past. At the same time, many scientific and non-scientific applications are putting complex demands on these networks. Grid and Cloud computation technologies offer many useful applications that are based on high-speed computation and communication [1]. Some of these applications require data transmission to be completed within certain time bounds while others demand certain QoS to be maintained throughout its service [2–4]. The network resources are often shared among various users and they may be assigned priorities over one another (preemptive and non-preemptive) [5,6]. Moreover, network resource dimensioning and capacity planning needs to be done efficiently, depending on the arriving traffic rate and pattern. In short, complex user demands and growing technologies offer too many challenges for the researchers to find a match between the two and they have attracted a lot of research attention. Mathematical and analytical modeling techniques have proven to be an effective and ideal tool for capturing these system behaviors and undertaking performance evaluations under varying conditions.

Grid/Cloud computing environment provides an abstract view of the underlying resources and seamless representation as a single entity to end-user [7]. The users just need to focus on their tasks without worrying about the underlying architecture. The resources may include supercomputing devices, large storage capacities, and high-speed communication links, etc. In cases where the resource's placement is geographically distributed, we can only estimate the capacity of bottleneck link along the path of data transfer, but no control over the traffic and its allocation. For capacity planning and network dimensioning, we consider such a Grid/Cloud computing environment, where all of the resources are under the control of a single, centralized entity, e.g., Grid'5000 [8].

Any new system or proposed technique can be evaluated for correctness and effectiveness in three different ways.

- **Real Implementation:** this is done by performing a real experiment on designated tools and devices. Although this approach will give us the exact results but it involves too much labor and cost (in terms of time and money). Moreover, the design may often require slight modification and tuning, but this approach is not flexible enough to accommodate these minor adjustments, and it will result in an increase in cost and delay. Therefore, this is not the best way to start with.
- **Simulation:** this is done by performing simulations using simulators that closely reflect real-world intended scenarios. Although, the simulator may not capture exactly all real-world parameters and, hence, its results may be slightly different than the true results, but still gives nice insight into the system behavior. The simulation models are easy to be developed and used to have a quick initial glance of system behavior with proposed modifications. The parameters can be easily fine-tuned to achieve optimal results at zero cost. Simulators can be used as an effective start-up tool, but their results cannot be fully trusted, as they may not capture the exact real world.
- **Analytical Modeling:** this is done by developing a mathematical model of the intended system and then the model can be used to evaluate the performance of a system under varying conditions to analyze its behavior. These models are often based on certain assumptions regarding some of the system parameters that are often criticized and considered as a flaw. In reality, anything other than real implementation is based on some assumptions in one way or the other. The assumptions are not just made blindly, rather they are supported by strong and valid arguments. Assumptions are based on a closed approximation of the real-world conditions.

Simulation modeling and analytical modeling are both considered to be the most efficient way of doing initial performance analysis and are often used in conjunction to validate each other. They are useful where the real system is not existing and yet to be developed. Once a technique is proven working through modeling and simulation, then its success is also more likely in real implementations.

Modeling and performance evaluation of multi-class queuing networks has gained a lot of research attention [9–12]. Typically, network system models are mostly based on Markov Chains. The bottleneck link can be viewed as a Single Server Queue and solved using Continuous Time Markov Chain (CTMC). Laplace and Fourier's transformations are also frequently used in the solution of these queues. Some researchers have used the concept of Linear Programming by mapping this bottleneck link utilization problem to an optimization problem. Petri-nets are used in modeling scientific workflows that enable scientists to describe their work as a series of tasks without worrying about resource allocation and coordination. Several solution techniques can be found in [13–16].

This research is mainly aimed at developing a novel analytical model that is flexible enough to capture network behavior under multi-class flows with strict QoS requirements, such as deadline and priority constraints in Grid/Cloud networks. This work is an extension of our previous study [17] in which, we have presented an analytical model for multi-class deadline constrained data transfer without considering preemption priorities. To the best of our knowledge, no such model has been developed until the write-up of this document, which can capture multi-class queuing system behavior with strict QoS

demands and priority constraints. The proposed model is a representation of a system with multi-class users; each having certain priority and QoS constraints. An example of such a system can easily be found in several daily life queuing systems. For the sake of demonstration, we restrict our study to Grid/Cloud computing environment and the same can be extended/applicable to any queuing system with stated characteristics. Furthermore, we consider the Grid/Cloud computing environment with dedicated communication lines, because the model is only applicable where the notion of QoS is valid, while the traditional Internet is known for its best-effort services. Our goal is to design an integrated and unified model that can be used for performance evaluation of the system with multi-class users having a deadline and priority constraints. The proposed model will be useful in high-speed network dimensioning, QoS provisioning, and capacity planning.

The rest of the text is organized, as follows: Section 2 presents a brief review of related work. Section 3 presents a brief description of the network systems and their corresponding characteristics. Section 5 depicts our proposed model. Section 6 presents the performance analysis. The paper is concluded in Section 7, with an outlook to our future work.

## 2. Related Work

With the advancement in communication and information technology, users and organization QoS demands are also growing and becoming more challenging. This section presents a brief review of various related models proposed in the literature. Each model captures network behavior under different conditions and user requirements. Bonald et al. conducted performance modeling and analysis of elastic flows in [18]; however, deadline constraints are not included in their proposed model. They have modeled the bottleneck link as an M/G/1-PS queue for fairness analysis and onward mean throughput approximation of TCP protocol. Bandwidth dimensioning model was developed by Berger et al. in [19] to estimate bandwidth share of individual connection in high-speed networks. They have considered a single bottleneck link in the network. Operations in semiconductor manufacturing are modeled as M/M(a,b)/c/PR priority queue by the authors of [20]. They have considered two priority classes without modeling deadline constraints. AlQahtani et al. developed an analytical model for 3G wireless networks for performance analysis of various control schemes in [21]. They have analyzed four different traffic classes, i.e., two non-real-time and two real-time.

Fodor G. et al. developed a model to estimate throughput guarantees and compute blocking probabilities for three kinds of flows in [22] i.e., (a) Rigid/Non-adaptive streaming flows with strict throughput requirement, e.g., voice calls, (b) adaptive streaming flows has a peak bandwidth requirement $b_2$, but they can be squeezed down to $b_2^{min}$ to accommodate other flows. Their holding time is independent of allocated $BW$ e.g., an adaptive video flow with codec enabled, and (c) elastic flows have lower and upper throughput bounds. The model is based on the extension of the classical loss model that was originally designed for ATM and circuit-switched networks. The concept of Partial Overlap (POL) is used in this model to divide the available capacity into two (1) $BW_{com}$ reserved for rigid flows (2) $BW_{ELS}$ reserved for Elastic and Adaptive flows. According to [22], the acceptable blocking probability threshold for each class is assumed as $BP_1^{max}$, $BP_2^{max}$ and $BP_3^{max}$ and $N_1$, $N_2$ and $N_3$ are the max no. of jobs of each class that can be accommodated, respectively. Because $BW_{com}$ is dependent upon $BP_1^{max}$ and it can be calculated easily using the Erlang-B formula. After fixing $BW_{com}$, we can calculate the max. no. of jobs of rigid flows $N_1$ as

$$N_1 = \frac{BW_{com}}{b_1} \tag{1}$$

where $b_1$ is the peak bandwidth requirement of individual rigid flow.

The values of $N_2$ and $N_3$ are iteratively calculated using an algorithm, called the Iterative Link Allocation procedure. The algorithm starts with some large values of $N_2$ and $N_3$, and it calculates their respective blocking probabilities $BP_2^{max}$ and $BP_3^{max}$ using CTMC. The values of $N_2$ and $N_3$ are decremented after every iteration until it results in such values

for which $BP_2 \leq BP_2^{max}$ and $BP3 \leq BP_3^{max}$. It aims at establishing a trade-off between $BP$ and throughput as larger values of $N_2$ and $N_3$ will certainly reduce their respective $BP$, but it will result in their throughput degradation. In [23], the authors proposed an Autonomic Distributed Streaming Service (ADSS) model for the application that involves data streaming between remote systems with/without in-transit data processing. The proposed ADSS model enables the intermediate node to change their behavior in response to the environmental conditions, i.e., network congestion or destination receiving rate. In such cases, ADSS can opportunistically exploit intermediate processing nodes in order to perform partial/complete in-transit processing on data, or it can temporarily store the data into the hard disk to avoid buffer overflow and data loss. Provided that data arrival rate at an intermediate node is $\lambda$, now, depending upon the reception rate of next-hop node and network congestion level, ADSS will automatically exploit perform in-transit processing on data at rate $\mu$ or temporarily store the data onto a hard disk with the rate $\omega$. The model takes current values $\lambda$, $\mu$, and $\omega$ as input and calculates future values for $\mu$, $\omega$, and the number of processing units to use for the next interval of time. ADSS is implemented using Reference Net (a kind of Petri-Nets) that helps in achieving required synchronization between associating processing nodes. The model applies to applications with end-to-end QoS requirements and can combine in-transit processing with data transmission.

Network slicing and software-defined networking (SDN) are the two most commonly used solutions for provisioning QoS in 5G networks. However, the efficient utilization of the network resources requires precise modeling of the traffic. Santhosha et al. developed a multi-class network model using SDN and network slicing to quantify network performance [24]. Heterogeneous flows are assumed from customers with different varying intensities without considering the deadline or priority constraints. A simulation-based model is presented in [25] in order to study the stability region in multi-class queuing networks. The requests are processed based on the first-come-first-serve policy without having priorities. Baris et al. studied the abandonment behavior of multi-class customers due to network congestion in [26]. Each class customer request receives different reward and cost rates, and their proposed model attempts to maximize their expected utilities. Likewise, many other studies can be found in the literature with emphasis on multi-class traffic modeling [27–29]. However, none of these studies consider deadline constrained bulk data transfers with preemptive priorities. Rami et al. studied the multi-class queuing system with dynamic priorities that are dependent upon the workload without considering the deadline constraints [30]. An improved scheduling policy is presented in [31] for a real-time queuing system with rewards and deadlines while ignoring the priorities.

In [32], the authors considered a multi-server queuing system with three priority classes and two servers. Each class of customers has its arrivals and service rates. They have used numerical analysis methods to solve the system of linear equations and calculate each class blocking probability and average queue length using system steady-state probabilities. Kannan et al. worked on scheduling bulk file transfers with deadline constraints by dividing the time scale into uniform time slices [33]. Bandwidth adjustments are made at the start of every time slice. They have also explored file transfer over multi-paths and found significant improvement in throughput as compared to a single path. In [34], Bin et al. studied the problem of scheduling bulk data transfer with a deadline constrained to find the optimal bandwidth allocation scheme, resulting in minimizing the overall network congestion. They have solved this problem for optimality using the maximum concurrent flow problem. In [35], the authors presented a novel model for multi-class deadline constrained network flows with equal sharing of residual link capacity. They have modeled the underlying shared bottleneck link as an M/M/1/K-PS Queue and solved it while using multi-dimensional Continuous Time Markov Chain (CTMC). The model can be easily extended to any number of classes with varying arrival and service rates. The model is being validated using NS-2 and offline simulation, and used for the calculation of Blocking Probability (BP) of individual classes as well as the overall system. The authors

also presented an algorithm for network dimensioning and capacity planning based on their model.

In the Grid/Cloud computing environment, resources are often reserved in advance to perform certain tasks. Therefore, designated data must be made available at those resources within certain time bounds, and this is usually known as the deadline constraint of the data transfers. Moreover, data transfer requests may be categorized into various classes, depending upon their minimum bandwidth requirement. A system of multi-class deadline constrained bulk data transfers is modeled in [35], where the classes are differentiated based on their minimum bandwidth requirement. Here, we are interested in extending this work by assigning each class a relative priority with preemption. This may reflect a system with multi-users, each having its priority, e.g., In Grids/Clouds, we may have two simple classes of users, as follows: (a) paid users/scientists whose request will be given the highest priority. (b) free users/students, whose request will be given the least priority. The same may be extended to any number of classes assigned with relative priorities with preemption.

Multi-class flow models with preemptive priorities have previously been explored in the literature, but none of them consider the deadline constraint. Our work is mainly focused on developing an analytical model for multi-class deadline-constrained data transfer requests with preemptive priorities. To the best of our knowledge, no such model exists in the literature by the write up of this document.

## 3. Regarding Analytical Modeling

The following subsections present a brief description of the network systems and their corresponding characteristics.

### *3.1. Network Representation*

Any network can be represented by a connected graph $G(V, E)$, where $V$ is the set of all nodes in the network and $E$ is a set of edges between nodes. Often, flows in a high-speed network require multi-hop data transmission between the source and destination located at remote stations (the terms requests and flows are used interchangeably). Network performance and throughput of the flows sharing the same path depends on the efficient utilization of bottleneck link on the path with capacity $C$. As stated earlier, we are considering a network environment where communication links are under the control of a single entity, so that QoS demands of various flows can be fulfilled. Most of the models that were proposed in the literature aimed at the optimal utilization of the bottleneck link. Various bottleneck link bandwidth sharing schemes have been proposed and analyzed. It also helps in model simplification.

Grid/Cloud-based applications often require data transmission to be completed within certain time bounds, such that certain QoS to be maintained throughout its service. The network resources are shared among various users and they may be assigned priorities over one another (preemptive and non-preemptive). Here, we limit our model to capture system behavior under two-classes data transfer mechanisms with deadline and priority constraints.

### *3.2. System Parameters*

The model takes various system parameters as input and all evaluation is based on these parameters. Typical input parameters include:

- Bottleneck link capacity $C$.
- Arrival rate $\lambda_i$ of individual $i^{th}$ class flows into the system.
- Service rate $\mu_i$ of individual $i^{th}$ class flows.
- Probability distribution of arrivals and services (Poisson and exponential distribution are considered for arrivals and services, respectively).

Note: in some cases, the arrivals/services rates may be considered as system state dependent, which is out of the scope of this study.

### *3.3. Performance Measures*

Performance measures of our interest include blocking probability of overall system and individual classes, the effect of higher-class jobs on lower-class jobs, and link capacity utilization. This study will help in the efficient resource dimensioning and capacity planning of the queuing system. Important measures include:

- Blocking Probability (BP) of the system and individual classes.
- Comparative analysis of preemptive and non-preemptive models.
- Percentage of lower-class flows being ejected by higher class flows.
- Percentage Link utilization, etc.

## 4. Problem Formulation

We are interested in the investigation and performance evaluation of a multi-class queuing system with strict QoS (deadline) and priority constraints. For the sake of demonstration, we apply our model to Grid/Cloud computing environment with two simple classes of users, as follows: (a) paid users/scientists, whose request will be given the highest priority, (b) free users/students, whose request will be given the least priority. This model can be extended/applicable to any queuing system with stated characteristics and any number of classes.

The Grid/Cloud computing network can be represented by a connected graph $G(V, E)$, where $V$ is the set of all nodes (storage/computing resources) in the network and $E$ is a set of edges (communication links) between nodes. Often, data transfer requests require multi-hop data transmission between source and destination located at remote stations. Let us say that $p_{i,j}$ is the path between source $v_i$ and destination $v_j$. Network performance and throughput of the flows sharing the same path depend upon the efficient utilization of bottleneck link on the path with capacity $C$.

*Definitions:*

1. Data Transfer Request: a data transfer request $r = (\nu_r, \omega_r, \phi_r)$ is a tuple, where $\nu_r$ is the volume of $r$, $\omega_r = [\eta_r, \psi_r]$ is the active window (from arrival time $\eta_r$ to deadline $\psi_r$) and $\phi_r$ is the path connecting source $S_r$ and destination $D_r$ of the request $r$.
2. $MRR_r$: Minimum Required Rate $MRR_r$ of the request $r$ is calculated on the basis of its volume and active window, as follows:

$$MRR_r = \frac{\nu_r}{\psi_r - \eta_r}$$

3. $BP$: blocking Probability ($BP$) is the ratio of total rejected requests and the total number of submitted requests.
4. Residual capacity $C_r$ is the remaining capacity of the link and it can be calculated, as follows:

$$C_r = C - \sum_{i=1}^{R} MRR_i \times N_i \tag{2}$$

where $R$ is the total number of classes and $N_i$ is the number of requests of $i^{th}$ class.

5. Active request is the term used for all the accepted requests that are currently in the flow.

Consider a shared bottleneck link having capacity $C$. Data transfer requests are categorized into $R$ classes that are based on their minimum required rates. Each class is assigned a priority $\tau$ i.e., $\tau_i$ is the priority of $i^{th}$ class request. A request is accepted if

- It is $MRR_r$ can be fulfilled. At any time instant $t$, a request of an $i^{th}$ class is accepted if

$$C_r \geq MRR_r$$

- In cases where $C_r < MRR_r$ and there are enough active request of lower classes, such that

$$\left( C_r + \sum_{i=1}^{Q} MRR_i \times N_i \right) \geq MRR_r$$

where $Q$ is the list of accepted lower class requests. In this case, sufficient requests of lower classes will be ejected in order to accommodate the incoming request of the higher class.

The state of the system $S$ at any time instant $t$ can be represented as:

$$S_t = (N_1, N_2, N_3, \ldots, N_R)$$

There are three possibilities to share the available residual capacity when $C_r > 0$.

- No-Sharing ($NS$) Scheme: residual capacity $C_r$ is unused and it results in poor utilization of link capacity.
- Equal-Sharing ($ES$) Scheme: $C_r$ is shared equally among the active flows [35] and this scheme results are better than the no-sharing scheme.
- Weighted-Sharing ($WS$) Scheme: $C_r$ is distributed among active flow proportional to their class $MRR$ [17], and this scheme results in improved capacity utilization.

The sharing of residual capacity $C_r$ as per the above schemes is explained with an example in Figure 1 with $C = 7$ Gbps, where the current state of the system is $(2, 1)$ i.e., two active flows of class 1 and one active flow of class 2. We can easily compute that $C_r = 3$ Gbps and the Figure 1 explains how it is shared among the active flows, as per the three schemes. In this study, experiments are conducted with an equal sharing scheme only.

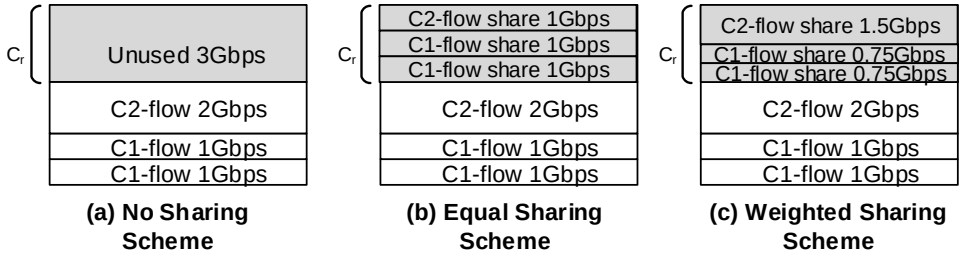



**Figure 1.** Various schemes for sharing residual capacity $C_r$.

## 5. Proposed Model

Markov chains are successfully used for performance evaluation of many different types of queuing systems. For given system parameters, we can easily find performance measures, like BP, link utilization, mean flow time, etc. These performance measures are helpful in system dimensioning and capacity planning for provisioning better QoS.

In a queuing system, the users are often classified into multiple classes, depending upon their service requirement and paying capacity. In such a multi-class environment, priority is often also assigned to each class signifying their level of importance. Various models are proposed for the analysis of multi-class priority queuing systems. These models are based on varying system parameters, as per the nature of the application, different arrival and services distribution, queuing mechanism, and priority handling (preemptive or non-preemptive, resume or restart). In the queuing system, lower class requests are blocked for two reasons: (a) blocked due to non-availability of capacity in the system and (b) ejected by the higher class. Aggregating these two types of probabilities, we will obtain the overall BP of the corresponding lower class. Most of the models proposed in the literature can help in finding the overall BP of the lower class. To the best of our knowledge, there is no such model that can provide us with insight into the two components of the BP of lower classes stated above.

The proposed model presents a novel and more intuitive approach for treating Markov chains to find the BP of individual classes. By using this novel approach, we can obtain

the detailed BP of a particular class from which we can easily obtain blocking due to higher classes ejection and blocking due to system capacity. Typically, by solving Markov chains, we get the steady-state probability (SSP) vector $\pi$ from initial one-step transition probabilities, but, here, we are interested in finding steady transition probabilities (STP), i.e., long-term probabilities of the system taking each transition. Next, we explain this concept with a simple example.

Consider a simple CTMC $(M/M/1/2)$ having three states, as shown in Figure 2, and the similarity rate matrix Q for this simple chain is given below

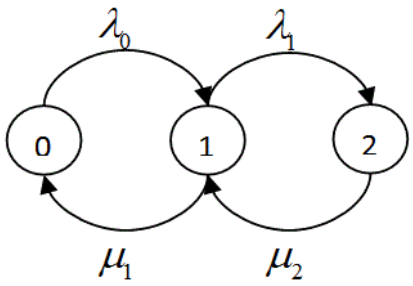



**Figure 2.** M/M/1/2 Markov Chain.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} -\lambda_0 & \lambda_0 & 0 \\ \mu_1 & -(\lambda_1+\mu_1) & \lambda_1 \\ 0 & \mu_2 & -\mu_2 \end{pmatrix} \end{array}$$

We can find one-step transition probability matrix $P$ from the above matrix Q using the following formula

$$P = \frac{Q}{Max(q_{ii})} + I$$

The Markov chain that is given in Figure 2 will look like that shown in Figure 3 in terms of one-step transition probabilities.

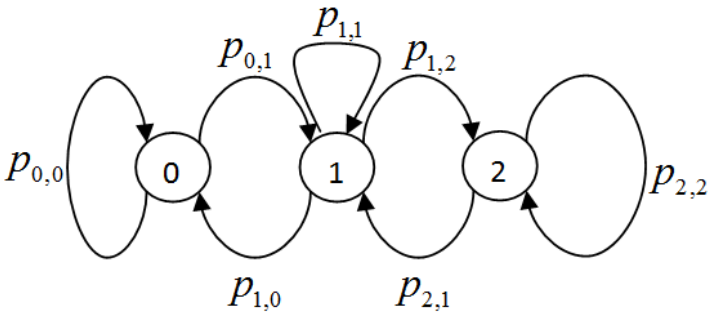



**Figure 3.** M/M/1/2 with one-step transition probabilities.

We are interested in finding steady transition probabilities (STP) $P_{i,j} \forall i, j$, i.e., the long term probability of the system taking each transition. In the next section, first, we will explain STP and how it can help provision the deep insight of blocking of the lower class in multi-service priority queuing system. Afterward, the concept of normalized arrival probabilities (NAP) is presented, i.e., another way of computing the blocking probability with proof of its correction while using a simple M/M/1/N queue as shown in Figure 4.

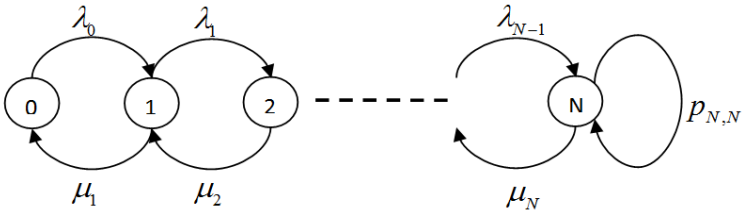



**Figure 4.** Typical M/M/1/N queue.

*5.1. Steady Transition Probabilities (STP)*

The concept of steady transition probabilities (STP) is just a detailed view of the Markov chain, and we can obtain steady-state probabilities from steady transition probabilities and vice versa. As stated earlier, STP is the long-term probabilities of the system taking each transition, and these can be calculated in two ways.

- Inverted Markov Chains
- Using SSP and one-step transition probability matrix $P$

5.1.1. Inverted Markov Chains

By solving the Markov chain, we obtain steady-state probabilities, i.e., the long-term probability of the system being in every state. Using Inverted Markov Chains, we simply consider transitions as the states of the Markov chain and we need one step transition-to-transition probabilities in order to calculate STP. Consider the simple Markov chain with three states and inter-state transition probabilities, as given in Figure 5.

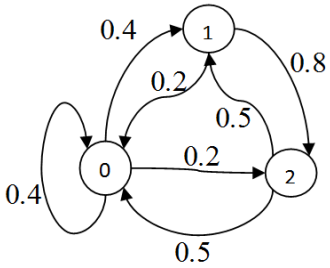



**Figure 5.** Sample M/M/1/2 queue.

It is easy to get its one step probability matrix $P$, as below.

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0.4 & 0.4 & 0.2 \\ 0.2 & 0 & 0.8 \\ 0.5 & 0.5 & 0 \end{pmatrix} \end{array}$$

Once, we obtain the one step transition probability matrix $P$, the Iterative (Power) method [36] can be used to calculate the steady state probability vector $\pi$, as follows:

$$\pi^0 P = \pi^1$$

$$\pi^1 P = \pi^2$$

$$\cdots$$

$$lim_{n\to\infty} \pi^n P = \pi^n$$

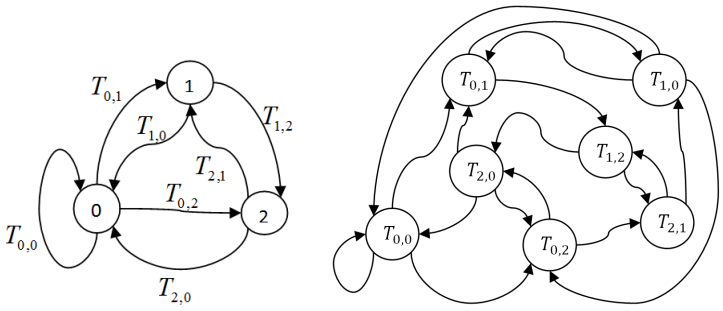
where $\pi^0$ is initial (random) probability distribution vector with condition $\sum_{i=1}^{n} p_i = 1$. After solving for above chain (Figure 5), we get

$$\pi = (0.37036, 0.30863, 0.32103)$$

i.e.,

$$P(S_0) = 0.37036$$
$$P(S_1) = 0.30863$$
$$P(S_2) = 0.32103$$

We now redraw Figure 5 by relabeling each transition as $T_{i,j} \forall i, j \in S$, as shown in Figure 6a, which shows the original Markov chain for sample M/M/1/2 queue along with the corresponding inverted Markov chain given in Figure 6b.




(a) Original Markov chain (b) Transformed/Inverted Markov chain

**Figure 6.** Transformation of sample M/M/1/2 queue.

The one step transition to transition probability matrix is given below

$$P = \begin{array}{c} \\ T_{0,1} \\ T_{0,2} \\ T_{0,0} \\ T_{1,2} \\ T_{1,0} \\ T_{2,0} \\ T_{2,1} \end{array} \begin{array}{c} \begin{array}{ccccccc} T_{0,1} & T_{0,2} & T_{0,0} & T_{1,2} & T_{1,0} & T_{2,0} & T_{2,1} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0.8 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0.2 & 0 & 0 \end{pmatrix} \end{array}$$

As it follows the Markov property, we can now find the steady transition probability vector $\pi$ in the same way and, after solving, we get

$$\pi = \{0.148148148, 0.074074074, 0.148148148, 0.24691358, 0.061728395, 0.160493827, 0.160493827\}$$

i.e.,

$$P(T_{0,1}) = 0.148148148$$
$$P(T_{0,2}) = 0.074074074$$
$$P(T_{0,0}) = 0.148148148$$
$$P(T_{1,2}) = 0.24691358$$
$$P(T_{1,0}) = 0.061728395$$
$$P(T_{2,0}) = 0.160493827$$
$$P(T_{2,1}) = 0.160493827$$

We can easily see that for any state $s$

$$\sum_{\forall i} P(T_{i,s}) = \sum_{\forall j} P(T_{s,j}) = P(s)$$

i.e., the sum of all transition probabilities into state $s$ is equal to the sum of transition probabilities out of state $s$ and that is equal to the probability of being in state $s$ e.g., for $S_1$, we can easily see that,

$$P(T_{0,1}) + P(T_{2,1}) = P(T_{1,2}) + P(T_{1,0}) = P(S_1)$$

$$0.148148148 + 0.160493827 = 0.24691358 + 0.061728395 = 0.30863$$

$$0.308641975 = 0.308641975 = 0.30863$$

The same can be observed for all other states. This shows that STP gives us a more detailed view of the system long term probabilities.

5.1.2. Alternative Approach Based on $SSP$ and $P$

From the previous results, we can easily deduce that

$$P(T_{i,j}) = P(S_i).p_{i,j} \tag{3}$$

For example,

$$P(T_{0,1}) = P(S_0).p_{0,1}$$
$$P(T_{0,2}) = P(S_0).p_{0,2}$$
$$P(T_{0,0}) = P(S_0).p_{0,0}$$

$\Rightarrow$

$$P(T_{0,1}) + P(T_{0,2}) + P(T_{0,0}) = P(S_0).(p_{0,0} + p_{0,1} + p_{0,2})$$

As $p_{1,1} + p_{1,2} + p_{1,3} = 1$ therefore we get

$$P(T_{0,1}) + P(T_{0,2}) + P(T_{0,0}) = P(S_0)$$

This method gives us a simple way to calculate STP, and this is more convenient in terms of computation as for large size CTMC, the size of one-step transition-to-transition probability matrix will grow enormously and it will require greater computations. In other words, we can simply calculate the steady transition probability of any transition $T_{i,j}$ using Equation (3).

*5.2. Normalized Arrival Probabilities (NAP)*

STP gives us the long term probabilities of the system taking any transition. Mainly, we have two types of transitions in CTMC, i.e., arrivals and departures. For capacity planning, we are often interested in system BP, which is only related to arrivals only. Let $T$ be the set of all transition probabilities, then we can express it as

$$T = T_A + T_D$$

where $T_A$ and $T_D$ are the set of arrival and departure probabilities, respectively.

Here, we are only interested in arrival probabilities and let the summation of all arrival probabilities be $D$, i.e.,

$$\sum P(T_{i,j}) = D \; \forall T_{i,j} \in T_A$$

It can be observed that, for constant arrival and service rates,

$$\sum P(T_{i,j}) = \frac{\lambda}{\lambda + \mu}$$

We divide each arrival transition by $D$ to obtain the Normalized Arrival Probability (NAP), i.e.,

$$\bar{P}(T_{i,j}) = \frac{P(T_{i,j})}{D} \Rightarrow \sum \bar{P}(T_{i,j}) = 1$$

This $NAP$ gives us the distribution of arrivals, e.g., if the total no. of arrival into the system is $A$, then the arrival count along with each arrival transition $AC(T_{i,j})$ is given by

$$AC(T_{i,j}) = \bar{P}(T_{i,j}) \times A$$

Thus, we obtain the approximate no. of arrivals on each arrival transition.

### *5.3. Using NAP to Compute BP of $M/M/1/N$ Queuing System*

In this section, we will show how to use NAP to find the BP of the system. We will also prove that its result is the same as the BP calculated using traditional SSP. For instance, see Figure 4, in which the blocking probability of the system is the probability of the system being in state N i.e., $P(N)$, and the same result can be obtained using $NAP$.

This is very intuitive to choose $\bar{P}(T_{N,N})$ only because all other arrivals are accommodated by the system and they cause a transition from one state to another. $T_{N,N}$ is the only looping transition in $M/M/1/N$, i.e., arrivals along with this transition cause no change in the system's state (loopback). In other words, all of the arrivals along $T_{N,N}$ are blocked by the system. That is why we say that the BP of the system in NAP is the looping transitions in the case of $M/M/1/N$ i.e., $\bar{P}(T_{N,N})$ and this is more intuitive. Next, we will try to prove the following

$$P(N) = \bar{P}(T_{N,N})$$

For sake of illustration, we limit our queue size to $N = 2$ i.e., $M/M/1/2$ with arrival $\lambda$ and service rate $\mu$, as shown in Figure 2.

Similarity, the Rate Matrix $Q$ of above $M/M/1/2$ queue is given below.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} -\lambda & \lambda & 0 \\ \mu & -(\lambda+\mu) & \lambda \\ 0 & \mu & -\mu \end{pmatrix} \end{array}$$

We can find steady-state probabilities of this simple $M/M/1/2$ by solving the following birth–death equation.

$$\lambda P_0 = \mu P_1 \Rightarrow P_1 = \frac{\lambda}{\mu} P_0$$

$$P_2 = \frac{\lambda^2}{\mu^2} P_0$$

we know that

$$P_0 + P_1 + P_2 = 1$$

$$P_0 + \frac{\lambda}{\mu} P_0 + \frac{\lambda^2}{\mu^2} P_0 = 1$$

$$P_0 = \frac{1}{1 + \frac{\lambda}{\mu} + \frac{\lambda^2}{\mu^2}}$$

$$P_0 = \frac{\mu^2}{\lambda^2 + \lambda\mu + \mu^2}$$

$$P_1 = \frac{\lambda}{\mu} P_0$$

$$P_1 = \frac{\lambda}{\mu} \frac{\mu^2}{\lambda^2 + \lambda\mu + \mu^2} = \frac{\lambda\mu}{\lambda^2 + \lambda\mu + \mu^2}$$

and the BP of the system is $P_2$

$$P_2 = \frac{\lambda^2}{\mu^2} P_0$$

$$P_2 = \frac{\lambda^2}{\mu^2} \frac{\mu^2}{\lambda^2 + \lambda\mu + \mu^2} = \frac{\lambda^2}{\lambda^2 + \lambda\mu + \mu^2}$$

We now try to find the same result using NAP, which is calculated by using SSP and one-step transition probabilities. To obtain one-step transition probabilities, we use the following formula

$$P = \frac{Q}{Max(q_{ii})} + I$$

Hence,

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} & 0 \\ \frac{\mu}{\lambda+\mu} & 0 & \frac{\lambda}{\lambda+\mu} \\ 0 & \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \end{pmatrix} \end{array}$$

Redraw Figure 2 using one-step transition probabilities, we get the picture that is shown in Figure 7 (transition probability with zero value are ignored)

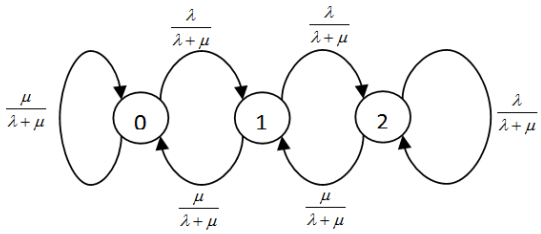



**Figure 7.** M/M/1/2 queue with one-step transition probabilities.

We can see that, among these six transitions, three are arrivals, i.e., $T_{0,1}, T_{1,2}, T_{2,2}$. We can find $NAP$, as below

$$P(T_{0,1}) = P(0).p_{0,1} = \frac{\mu^2}{\lambda^2 + \lambda\mu + \mu^2} \frac{\lambda}{\lambda + \mu} = \frac{\lambda\mu^2}{z}$$

where

$$z = (\lambda + \mu)(\lambda^2 + \lambda\mu + \mu^2)$$

Similarly,

$$P(T_{1,2}) = P(1).p_{1,2} = \frac{\lambda\mu}{\lambda^2 + \lambda\mu + \mu^2} \frac{\lambda}{\lambda + \mu}$$

$$= \frac{\lambda^2\mu}{z}$$

$$P(T_{2,2}) = P(2).p_{2,2} = \frac{\lambda^2}{\lambda^2 + \lambda\mu + \mu^2} \frac{\lambda}{\lambda + \mu}$$

$$= \frac{\lambda^3}{z}$$

We can now compute that normalized arrival probability $\bar{P}(T_{2,2})$, as below

$$\bar{P}(T_{2,2}) = \frac{\frac{\lambda^3}{z}}{\frac{\lambda\mu^2}{z} + \frac{\lambda^2\mu}{z} + \frac{\lambda^3}{z}}$$

$$= \frac{\lambda^2}{\lambda^2 + \lambda\mu + \mu^2}$$

Thus, we have proved that

$$P(2) = \bar{P}(T_{2,2})$$

This can be easily be extended to $M/M/N/N$. Moreover, for constant arrival rate $\lambda$ and service rate $\mu$, we can easily find out that sum of all services transitions

$$\sum P(T_{i,j}) = \frac{\mu}{\lambda + \mu} \quad \forall \; T_{i,j} \in T_D$$

and the probability of the system being in an idle state is as below

$$\bar{P}(T_{0,0}) = P(0) = \frac{\mu^2}{\lambda^2 + \lambda\mu + \mu^2}$$

### *5.4. Model Implementation*

We have modeled the bottleneck link of the network as a constant capacity $C$ server. Arrivals of multi-class requests are assumed to follow Poisson and the services are exponentially distributed with mean volume $V$. Thus, the system is modeled as a multi-dimensional Continuous Time Markov Chain (CTMC), as shown in Figure 8. Given the system is in state $S_i$, then the arrival of $c^{th}$ request will result in a transition to state $S_j$ and completion of a $c^{th}$ class job will result in a transition to state $S_k$.

$$S_i = (N_1, N_2, \ldots, N_c, \ldots, N_R)$$

$$S_j = (N_1, N_2, \ldots, N_c + 1, \ldots, N_R)$$

$$S_k = (N_1, N_2, \ldots, N_c - 1, \ldots, N_R)$$

As arrival of all classes is equally likely and they are generated using Poisson distribution, therefore the transition rate from state $S_i$ to $S_j$ uponthe arrival of a $c^{th}$ class request will become:

$$\lambda_{i,j} = \lambda_c \tag{4}$$

Upon the completion of a request of class $c$, the system will make a transition from state $i$ to state $k$. As in this study, the experiments are only conducted with an equal sharing scheme, and the service rate for this scheme is calculated, as follows:

$$\mu_{i,k} = \left\{ \begin{array}{cc} \frac{N_c}{V}\left(MRR_c + \frac{C_r}{\sum_{i=1}^{R} N_i}\right) & for N_c > 0 \\ 0 & otherwise \end{array} \right\} \tag{5}$$

where $V$ is the mean size of the requests and $N_c$ are the total number of active flows of class $c$ in-state $i$.

Figure 8 presents a sample CTMC for two classes with $C = 5$ Gbps. Class 2 jobs have preemption priority over class 1. It can be noted that, un states 4 and 5, there is no room for newly arriving requests of class 2, therefore transition is made to state 9, which results in the ejection of 1 and 2 requests of class 1, respectively. Likewise, the transition from states 8 and 9 to state 11 also results in the ejection of class 1 jobs.

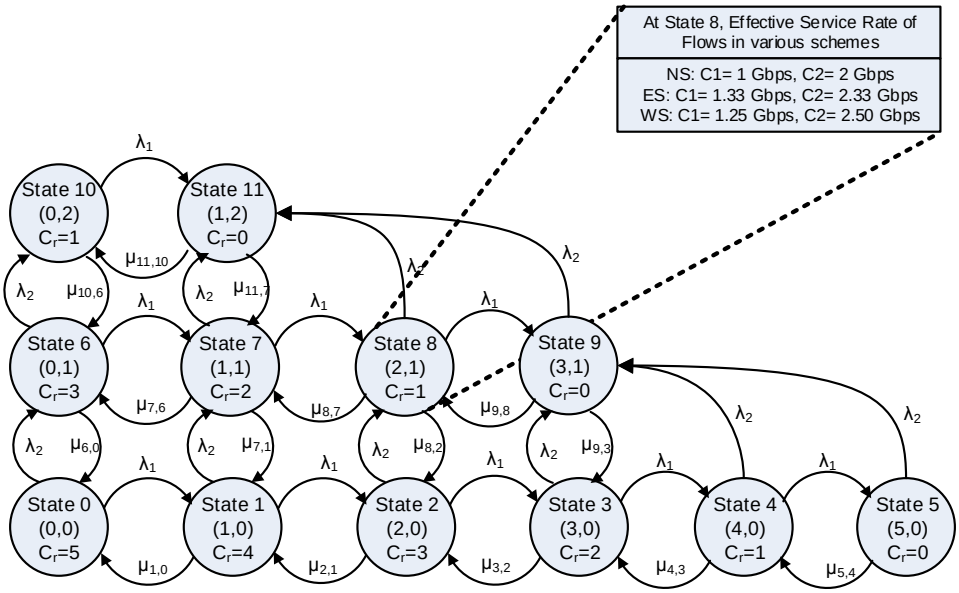



**Figure 8.** Sample two-dimensional Continuous Time Markov Chain (CTMC) for two classes with $MRR_c = c\ Gbps\ \forall\ c \in \{1, \ldots, R\}$ and link capacity $C = 5$ Gbps.

The total number of states in the CTMC grows exponentially with the increase in the link capacity $C$ and total number of classes, as shown in Figure 9. A state $S$ in CTMC is valid if:

$$\sum_{i=1}^{R} (N_i \times MRR_i) \leq C$$

where $N_i$ is the total number of active flows of $i^{th}$ class having a minimum flow rate $MRR_i$.

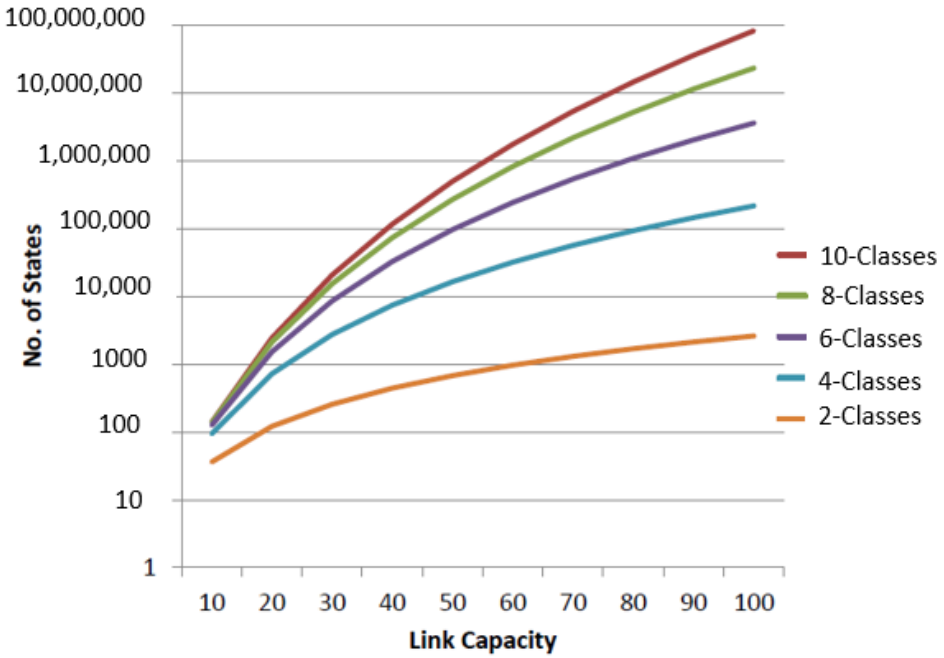



**Figure 9.** Impact of link capacity on number of states in CTMC with varying number of classes in the system.

After generating all of the possible states and corresponding transition probabilities, CTMC is solved using the iterative method, and we get steady-state probability vector $\pi$ of

the system, which is then used to compute blocking probabilities of the overall system and individual classes and subsequent performance analysis.

5.4.1. Computation of BP

The blocking probabilities of the overall system and individual classes are computed while using the steady-state probability vector $\pi$. To compute $BP$ of class $x$, set $S_{B_c}$ of all those states in CTMC is required where a new request of class $x$ cannot be accommodated. Thus, the blocking probability of high priority class $x$ can be computed, as below:

$$BP_x = \sum_{\forall s \in S_{B_x}} p_s$$

where $p_s$ is the long term probability of the system being in state $s$.

The blocking probability of lower priority class $y$ can be computed, as below:

$$BP_y = \sum_{\forall s \in S_{B_{y1}}} p_s + \sum_{\forall t \in S_{T_A}} \bar{P}(T_{i,j}) \times MRR_x$$

where $S_{T_A}$ is a subset of normalized arrival probabilities, which results in the ejection of lower-class requests.

Blocking probability of the overall system can be computed as below:

$$BP = \left[ \sum_{c=1}^{R} (BP_c) \right] \times \frac{1}{\lambda} \tag{6}$$

5.4.2. Computation of Link Utilization

Percentage link utilization $C_{util}$ of the system having link capacity $C$ is computed from state probability vector $\pi$, as follows:

$$C_{util} = \frac{\sum_{\forall s \in \Theta} p_s \times (C - C_r(s))}{C} \times 100$$

where $C_r(s)$ is the link residual capacity in state $s$.

## 6. Performance Evaluation

The objectives of performance evaluation are:

- To validate the proposed model results.
- To highlight effect on overall system blocking probability, due to preemptive priority as compared to the non-preemptive model.
- To conduct a class-wise comparative analysis of blocking probabilities for preemptive and non-preemptive models.
- To present a detailed analysis of lower-class blocking probabilities.
- To perform analysis of link capacity utilization with varying traffic intensities.

The proposed model validation is conducted through simulation using an ad-hoc simulator that was developed in Microsoft Visual Studio 2017 using Visual Basic .NET (VB.NET). The simulation model considers an ideal network environment and it does not capture the network/packet-level details such as losses and overheads. In every simulation experiment, 100,000 requests/flows are generated using Poisson distribution. The flow volumes are exponentially distributed with mean a value of $V$. Table 1 presents the summary of configuration for different parameters that are related to models and simulations. The reported results are the average values for 10 different simulation runs for each experimental setup.

For the sake of simplicity and without losing any generality, the arrival rate of all classes is considered to be the same, i.e.,

$$\lambda_c = \frac{\lambda}{R}$$

where $R$ is the total number of classes and $\lambda$ is the arrival rate of all requests.

We know that traffic intensity $\rho$ can be computed, as below:

$$\rho = \frac{\lambda \times V}{C}$$

For a given/desired traffic intensity, the mean flow size $V$ can be obtained as

$$V = \frac{\rho \times C}{\lambda}$$

**Table 1.** Configuration setup for simulation and model.

| S. No. | Parameter/Variable | Value/Range | |
|---|---|---|---|
| | | **Simulation** | **Model** |
| 1 | Number of classes | 2 | |
| 2 | Arrival rate | 0.25 | |
| 3 | Link capacity | 20, 30, 40 Gbps | |
| 4 | Traffic intensity | 0.5–2.0 (step 0.1) | |
| 5 | Total number of requests | 100,000 | NA |
| 6 | Size of individual flow | Exp. distributed in range 40–160 (step 8) for $C = 20$ 60–240 (step 12) for $C = 30$ 80–320 (step 16) for $C = 40$ | NA |

Figure 10 shows the blocking probabilities that were calculated for various traffic intensities using analytical model and simulation while considering the link capacity of $C = 30$ Gbps. Model and simulation results are both nicely aligned for all traffic intensities varying from 0.5 to 2.0. These results clearly show that the simulations validate the model. For traffic intensities that are below 1.0, the overall system blocking probability is very low (acceptable). However, a significant increase in the blocking probabilities can be observed as traffic intensity approaches 2.0, where more than 50% of requests are blocked by the system. Furthermore, these results also confirmed that the system blocking probability is not linearly increasing with the increase in traffic intensity.

Next, we study the effect on the overall system blocking probability due to preemptive priority as compared to the non-preemptive model [17]. Figure 11 presents the comparative analysis of blocking probabilities for preemptive and non-preemptive models results with $C = 30$ Gbps. For traffic intensities that are below 1.0, there is no significant difference in blocking probabilities of the two models, and this is due to the underutilization of link capacity. However, a gradual increase in difference among the blocking probabilities of the two models can be observed as traffic intensity approaches 2.0 where approx. 50% and 55% of requests are blocked by the system in case of the non-preemptive and preemptive model, respectively. These results show that the preemptive model results in less than a 5% (absolute) increase in the system overall blocking probability when compared to its counterpart non-preemptive model.

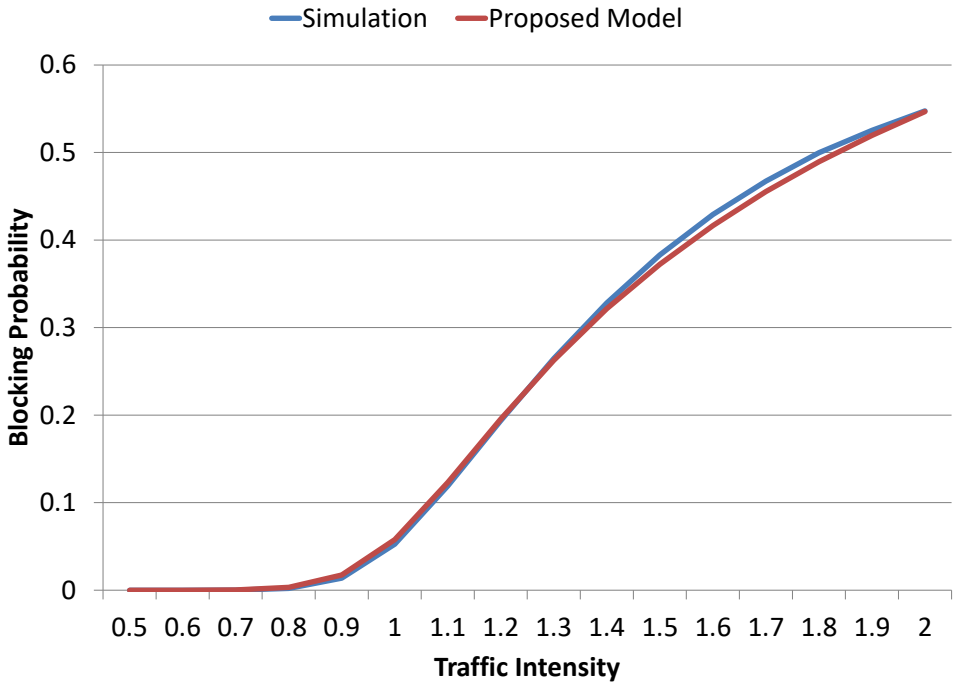



**Figure 10.** Validation of Model results through simulation with $C = 30$ Gbps

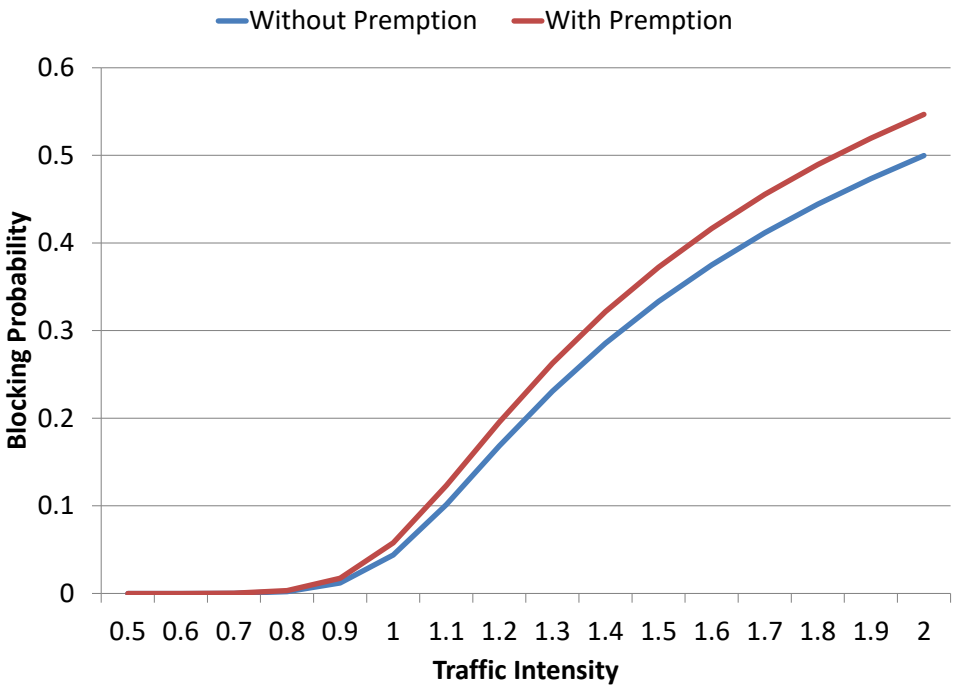



**Figure 11.** Comparative analysis of blocking probabilities for preemptive and non-preemptive model results with $C = 30$ Gbps

An increase in the system overall blocking probability by the preemptive model is not particularly significant, i.e., less than 5% (absolute) when compared to its counterpart non-preemptive model. However, a detailed investigation of individual class probabilities revealed a significant increase in the lower-class (class 1) probabilities, as shown in Figure 12. Once again, for traffic intensities that are below 1.0, the difference in individual class blocking probabilities of the two models is very low, which is due to the underutilization of link capacity. However, a significant increase in difference among the individual class blocking probabilities of the two models can be observed with an increase in traffic intensity. In the case of the non-preemptive model, for a traffic intensity of 2.0, the blocking

probabilities of class 1 and class 2 are 38% and 61%, respectively. When both of the classes are treated equally by the system, then class requests are experiencing high blocking probability due to their high QoS requirement i.e., $2MRR$. Whereas, in the case of the preemptive model, the same blocking probabilities changed to 94% and 15%, for class 1 and class 2, respectively. The significantly high blocking probabilities of class 1 (94%) is due to two reasons: (a) being blocked by the system due to unavailability of required QoS ($1MRR$) as a result of high utilization of system capacity and (b) ejected by the system to make room for high priority jobs. The whole link capacity is available for class 2 requests, as if class 1 requests do not exist (virtually) and, therefore, the blocking probability of class 2 is reduced from 61% to 15%, for traffic intensity of 2.0.

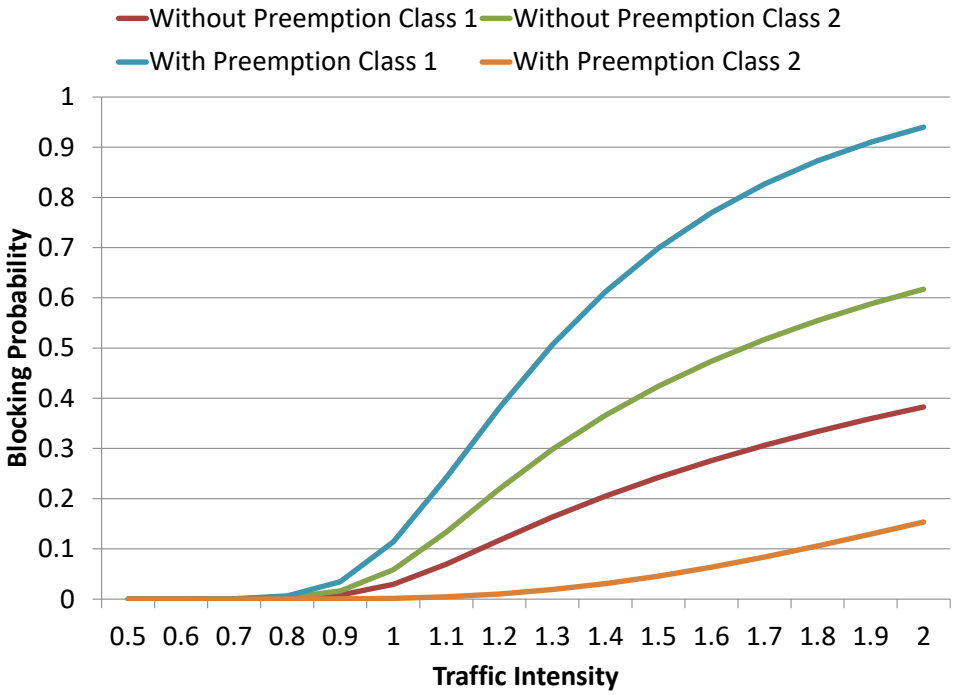
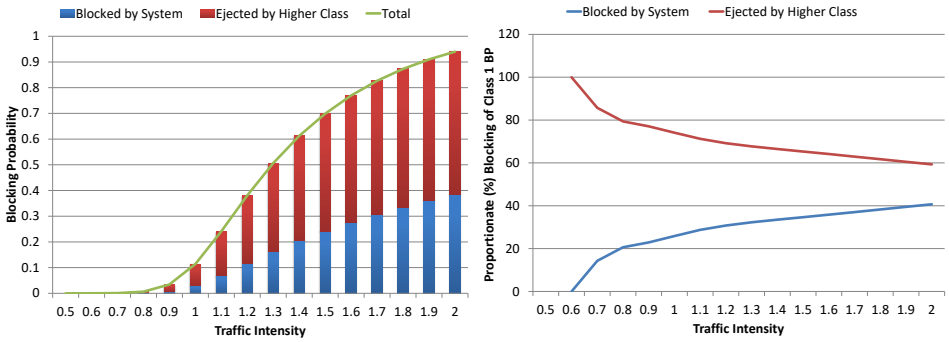



**Figure 12.** Class-wise comparative analysis of blocking probabilities for preemptive and non-preemptive Model results with $C = 30$ Gbps.

Class-wise comparative analysis of blocking probabilities, as given in Figure 12, indicate a significant increase (147.43%) in the blocking probabilities of class 1 for the preemptive model when compared to the non-preemptive model. This is due to two reasons: (a) being blocked by the system due to unavailability of required QoS (b) ejected by the system to make room for high-priority jobs. Figure 13 shows the detailed analysis of blocking probabilities components for class 1 while using preemptive model results with $C = 30$ Gbps, No. of classes $R = 2$ and $MRR_c = c\ Gbps\ \forall\ c \in \{1, \ldots, R\}$. Figure 13a provided detailed insight regarding class 1 blocking probabilities, along with the contribution of each component, in total, the blocking probabilities. We can observe that a major portion of the class 1 requests are blocked due to ejection by the arrival of higher class jobs as compared to blocking by the system due to the unavailability of the required QoS. This is due to the relatively higher QoS requirement of class 2 jobs i.e., having $MRR = 2$ Gbps. In other words, when the residual capacity is zero, the arrival of the class 2 job will cause an ejection of two requests (in progress) of class 1 if available. This is also evident from Figure 13b, which provides proportionate (%) blocking of class 1 blocking probability. For lower traffic intensities, a relatively low percentage of class 1 requests are blocked by the system as compared to the ones ejected by higher classes. For instance, with a traffic intensity of 1.0, around 10% requests of class 1 are blocked and, out of these 10% blocked requests, around 25% are blocked by the system, and 75% are ejected due to the arrival of higher class requests. Whereas, with a traffic intensity of 2.0, a total of around 94% requests of

class 1 are blocked and, out of these 94% blocked requests, around 40% are blocked by the system whereas 60% are ejected due to the arrival of higher class requests. In other words, more requests of class 1 are blocked by the system with an increase in traffic intensity due to high link utilization.




(a) Detailed view of class 1 blocking probabilities (b) Proportionate (%) blocking of class 1 BP

**Figure 13.** Analysis of blocking probabilities components for class 1 using preemptive model results with $C = 30$ Gbps.

Figure 14 shows the percentage link capacity utilization by proposed model with varying traffic intensities for $C = 20, 30, 40$ Gbps. For traffic intensities that are below 1.0, the link capacity utilization is below 35%, i.e., significant link capacity is available most of the time, which is the main reason for having significantly low blocking probabilities for traffic intensities that are below 1.0. A gradual increase in the link capacity utilization can be observed as the traffic intensity increases beyond 1.0 up to 1.5, but, afterward, there is no significant improvement in link capacity utilization. This shows that, as we approach towards the maximum achievable link utilization, an increase in traffic intensity contributes less in maximizing link utilization and, in contrast, it results in a drastic increase in the system blocking probability, which is evident from earlier results. Figure 14 also shows that, with an increase in traffic intensity, link utilization exhibits a converse behavior with an increase in the link capacities. For instance, with lower link capacity ($C = 20$ Gbps), link utilization grows faster in the early stages and gets slower towards the end to reach the maximum. Conversely, with higher link capacity ($C = 40$ Gbps), the growth in link utilization is slower in the beginning and it gets faster towards the end to reach the maximum.

In order to further illustrate the bottleneck link capacity utilization, we have conducted another set of experiments with varying requests arrival rate $\lambda = \{0.20, 0.25, 0.30, 0.35, 0.40\}$ having a mean volume size of 120 Gbps and the results are shown in Figure 15. It is evident from the results that, for low bottleneck link capacities, the link utilization is very high, i.e., around 90% for all arrival rates. As we increase the bottleneck link capacity, a gradual decrease in link utilization can be observed. For lower arrival rate $\lambda = 0.20$, the decrease in link utilization is faster when compared to the results of a higher arrival rate $\lambda = 0.40$. For instance, for $\lambda = 0.20$, when the bottleneck link capacity $C$ is increased from 20 Gbps to 40 Gbps, the link utilization is reduced from 64% to 5.57%. Whereas, for $\lambda = 0.40$, when the bottleneck link capacity $C$ is increased from 20 Gbps to 40 Gbps, the link utilization is reduced from 90.55% to 75.46%.

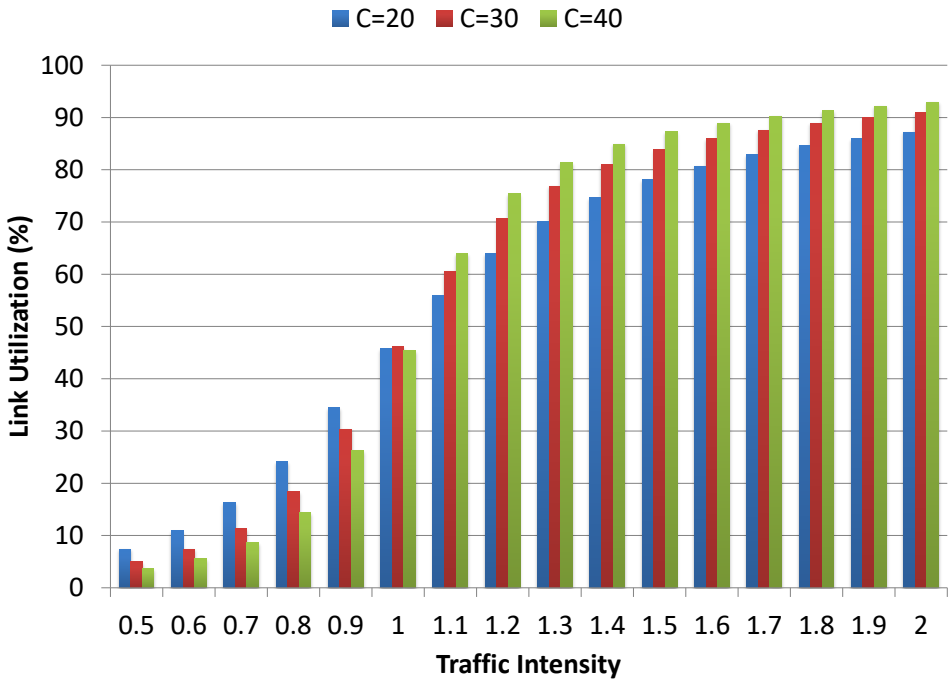



**Figure 14.** Percentage link capacity utilization by proposed model with varying traffic intensities for $C = 20, 30, 40$ Gbps.

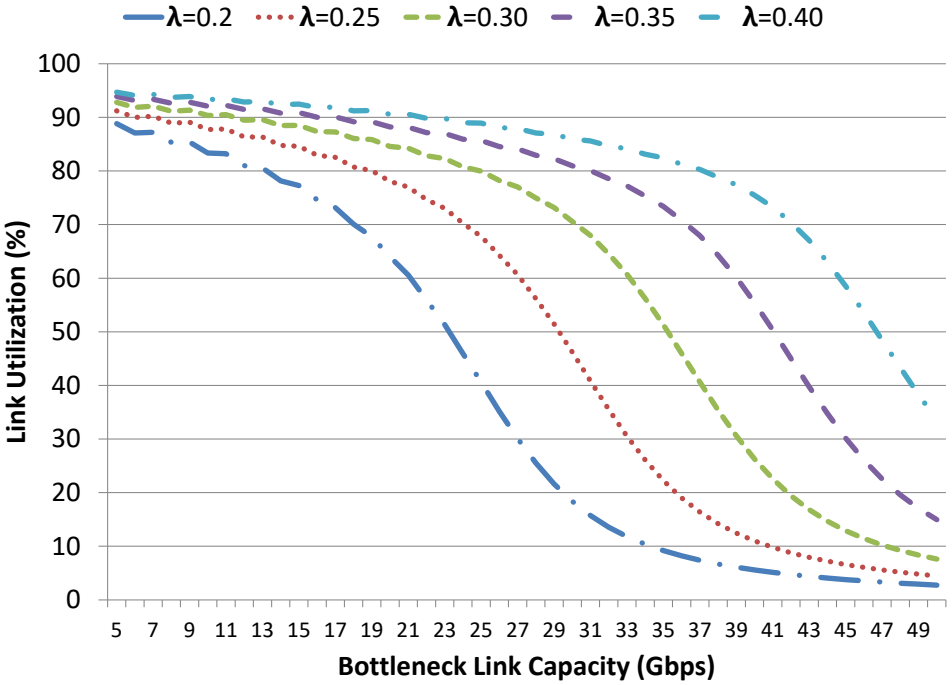



**Figure 15.** Percentage link capacity utilization by proposed model with varying arrival rates $\lambda = \{0.20, 0.25, 0.30, 0.35, 0.40\}$ and mean volume size $V = 120$ Gbps.

Algorithm 1 can be used for network capacity planning in order to compute optimal bottleneck link capacity for a given traffic intensity and requests an arrival rate, such that the overall network blocking probability remains within a certain acceptable range, i.e., $[BP_{lim} - \alpha, BP_{lim} + \alpha]$.

The experiments are conducted for a certain case study with varying requests for arrival rate $\lambda = \{0.20, 0.25, 0.30, 0.35, 0.40\}$ having mean volume size of 120 Gbps. Here, we are interested in finding the optimal bottleneck link capacity, such that the overall

network blocking probability remains with a certain acceptable range i.e., $BP_{lim} = 0.05$ with $\alpha = 0.002$. There are two classes of user requests and $MRR$ for class-1 and class-1 requests are 1 Gbps and 2 Gbps, respectively. Furthermore, class-2 requests have preemptive priority over class-1. Figure 16 provides the proposed model results for the aforementioned case study. The results show that the overall blocking probability of the system gets decreased with the gradual increase in bottleneck link capacity and, finally, we obtain different optimal bottleneck link capacity for each arrival rate, as indicated in Figure 16. With the increase in the arrival rate of incoming requests, we need to increase the bottleneck link capacity in order to have the overall blocking probability of the system below the desired range. For instance, the optimal bottleneck link capacity is 31 Gbps for the request arrival rate $\lambda = 0.25$ in order to have the overall blocking probability around 0.05. Whereas, for request arrival rate $\lambda = 0.40$, the optimal bottleneck link capacity results in 47 Gbps. This is just an example to illustrate the utility of the proposed model in the capacity planning of a network with given traffic conditions.

**Algorithm 1** Network Capacity Planning

**Require:** $V, \lambda, C_{max}, BP_{lim}, \alpha$

**Ensure:** $C_{opt}$

$C_{min} \leftarrow 0$

$BP \leftarrow 1$

$flag \leftarrow false$

**while** $flag \neq true$ **do**

$C_{cur} \leftarrow (C_{min} + C_{max})/2$

Generate system states $S$ for $C_{cur}$

Compute states transition probabilities for $S$ using $\lambda$ and Equation (5)

Compute states-state probability vector $\pi$

Update $BP$ using Equation (6)

**if** $BP \in [BP_{lim} - \alpha, BP_{lim} + \alpha]$ **then**

$C_{opt} \leftarrow C_{cur}$

$flag \leftarrow true$

**end if**

**if** $BP < BP_{lim}$ **then**

$C_{max} \leftarrow C_{cur}$

**else**

$C_{min} \leftarrow C_{cur}$

**end if**

**end while**

**return** $C_{opt}$

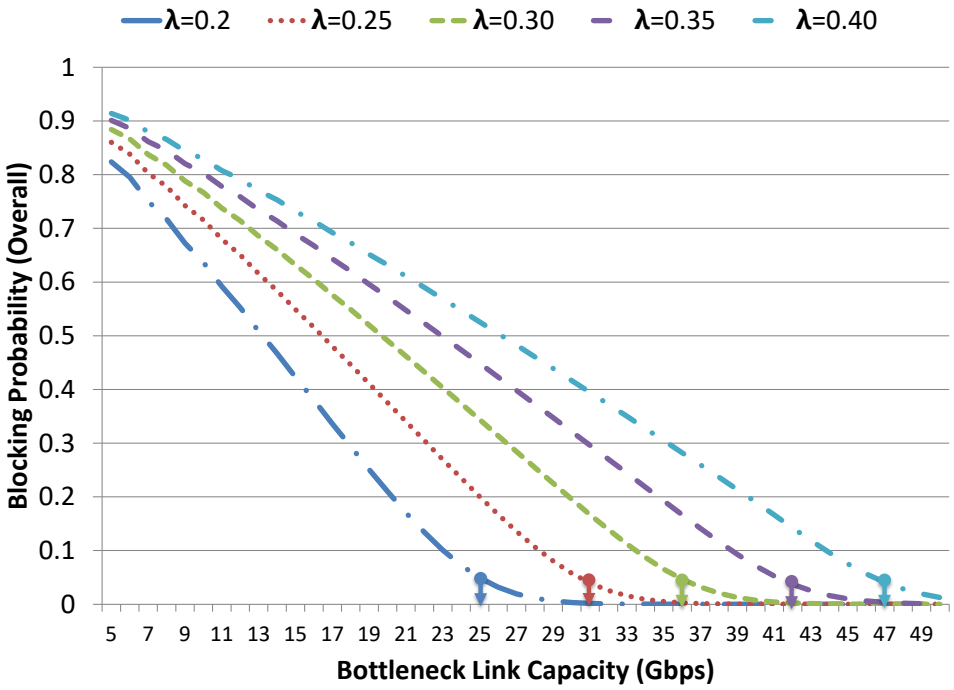



**Figure 16.** Network capacity planning for varying arrival rates $\lambda = \{0.20, 0.25, 0.30, 0.35, 0.40\}$ and mean volume size $V$ = 120 Gbps.

## 7. Conclusions and Future Work

In this paper, we have presented a novel analytical model for a multi-service queue with deadline and priority constraints. The model is validated through simulations of bulk data transfers using the equal sharing scheme of residual capacity. The proposed model results in less than a 5% increase of the system overall blocking probability when compared to its counterpart non-preemptive model. Detailed class-wise comparative analysis of blocking probabilities revealed that a significant increase (147.43%) in the lower class (class 1) blocking probabilities was observed when compared to its blocking probability results by the non-preemptive model. After further investigations regarding class 1 blocking probabilities, it was observed that a major portion of the class 1 requests are blocked due to ejection by the arrival of higher class jobs as compared to blocking by the system due to the unavailability of required QoS. The main reason for having significantly low blocking probabilities for traffic intensities that were below 1.0 was found to be the poor link capacity utilization, i.e., below 35%. These results also showed that, as we approach towards the maximum achievable link utilization, an increase in the traffic intensity contributes less in maximizing link utilization and, in contrast, it results in a drastic increase in the system blocking probability.

In the future, we are looking forward to extending this study by conducting experimental analysis with some real-world data of similar networks and parameters of various distribution schemes, like Poisson, Bounded Pareto, etc. Model applications, like network resources dimensioning, thee development of enhanced strategies for admission control, capacity planning, cost estimation, and pricing incentives, will also be explored.

**Author Contributions:** F.M.A. has implemented the model for the multi-service queue, conducted the experimental analysis, and did the paper writeup. I.U. designed the model and performed its validation and also assisted in results collection and paper writeup. S.A. conceived the overall idea and supervised this work. All authors contributed to this paper. All authors have read and agreed to the published version of the manuscript.

**Funding:** This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under Grant No. G:60-611-1441.

**Acknowledgments:** This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under Grant No. G:60–611–1441. The authors, therefore, acknowledge with thanks DSR for technical and financial support.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Tyagi, H.; Kumar, R. Cloud Computing for IoT. In *Internet of Things (IoT)*; Springer: Berlin, Germany, 2020; pp. 25–41.
2. Koulouzis, S.; Martin, P.; Zhou, H.; Hu, Y.; Wang, J.; Carval, T.; Grenier, B.; Heikkinen, J.; De Laat, C.; Zhao, Z. Time-critical data management in clouds: Challenges and a Dynamic Real-Time Infrastructure Planner (DRIP) solution. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5269. [CrossRef]
3. Tariq, A.; Pahl, A.; Nimmagadda, S.; Rozner, E.; Lanka, S. Sequoia: Enabling quality-of-service in serverless computing. In Proceedings of the 11th ACM Symposium on Cloud Computing, New York, NY, USA, 19–21 October 2020; pp. 311–327.
4. Ding, Z.; Wang, S.; Pan, M. QoS-Constrained Service Selection for Networked Microservices. *IEEE Access* **2020**, *8*, 39285–39299. [CrossRef]
5. Luo, S.; Yu, H.; Li, K.; Xing, H. Efficient file dissemination in data center networks with priority-based adaptive multicast. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1161–1175. [CrossRef]
6. Chen, J.; Du, C.; Xie, F.; Lin, B. Scheduling non-preemptive tasks with strict periods in multi-core real-time systems. *J. Syst. Archit.* **2018**, *90*, 72 – 84. [CrossRef]
7. Chaisiri, S.; Lee, B.; Niyato, D. Optimization of Resource Provisioning Cost in Cloud Computing. *IEEE Trans. Serv. Comput.* **2012**, *5*, 164–177. [CrossRef]
8. Bolze, R.; Cappello, F.; Caron, E.; Daydé, M.; Desprez, F.; Jeannot, E.; Jégou, Y.; Lanteri, S.; Leduc, J.; Melab, N.; et al. Grid'5000: a large scale and highly reconfigurable experimental grid testbed. *Int. J. High Perform. Comput. Appl.* **2006**, *20*, 481–494. [CrossRef]
9. Zheng, L.; Zhang, L. Modeling and performance analysis for IP traffic with multi-class QoS in VPN. In Proceedings of the MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No. 00CH37155), Los Angeles, CA, USA, 22–25 October 2000; Volume 1, pp. 330–334.
10. Tian, W. Analytical Models and Efficient Dimensioning Algorithms for Communication Systems In Randomly Changing Traffic Environments. Ph.D. Thesis, North Carolina State University, Raleigh, NC, USA, 2007.
11. Bekker, R. Queues with State-Dependent Rates. Ph.D. Thesis, Technische Universiteit Eindhoven, AZ Eindhoven, The Netherlands, 2005.
12. Ridley, A. Performance Analysis of a Multi-Class Preemptive Priority Call Center with Time-Varying Arrivals. Ph.D. Thesis, University of Maryland, College Park, MD, USA, 2004.
13. Snyder, P.M.; Stewart, W.J. An approximate numerical solution for multiclass preemptive priority queues with general service time distributions. In Proceedings of the 1985 ACM SIGMETRICS conference on Measurement and modeling of computer systems, New York, NY, USA, 26–29 August 1985; pp. 155–165.
14. Kumar, P.R. A tutorial on some new methods for performance evaluation of queueing networks. *IEEE J. Sel. Areas Commun.*, **1995**, *13*, 970–980. [CrossRef]
15. van der Heijdena, M.; van Hartena, A.; Sleptchenkob, A. Approximations for Markovian multi-class queues with preemptive priorities. *Elsevier Oper. Res. Lett.* **2003**, *32*, 273–282. [CrossRef]
16. Sleptchenko, A.; Harten, A.V.; Heijden, M.V.D. An Exact Solution for the State Probabilities of the Multi-Class, Multi-Server Queue with Preemptive Priorities. *Queueing Syst.* **2005**, *50*, 81–107. [CrossRef]
17. Ullah, I.; Munir, K. Performance prediction of a weighted capacity sharing scheme for grid bulk data transfers using a multi-service queue. In Proceedings of 7th the International Conference on Emerging Technologies, Islamabad, Pakistan, 5–6 September 2011; pp. 1–6.
18. Bonald, T.; Roberts, J. Performance modeling of elastic traffic in overload. In Proceedings of the International Conference on Measurement and Modeling of Computer Systems, Cambridge, MA, USA, 16–20 June 2001, pp. 342–343.
19. Berger, A.W.; Kogan, Y. Dimensioning Bandwidth for Elastic Traffic in High-Speed Data Networks. *IEEE/ACM Trans. Netw.* **2000**, *8*, 643–654. [CrossRef]
20. Phojanamongkolkij, N.; Cochran, J.K.; Fowler, J.W. Multi-Products Multi-Servers Bulk Service Queue with Threshold Service Size. In Proceedings of the International Conference on Semiconductor Manufacturing Operational Modeling and Simulation, San Francisco, CA, USA, 18–20 January 1999; pp. 153–156.
21. AlQahtani, S.A.; Mahmoud, A.S. Performance analysis of two throughput-based call admission control schemes for 3G WCDMA wireless networks supporting multiservices. *Comput. Commun.* **2008**, *31*, 49–57. [CrossRef]
22. Fodor, G.; Racz S., T.M. On Providing Blocking Probability- and Throughput Guarantees in a Multi-service Environment. *Commun. Syst.* **2002**, *15*, 257–285. [CrossRef]
23. Tolosana-Calasanz, R.; Banares, J.A.; Rana, O.F. Autonomic Streaming Pipeline for Scientific Workflows. *Concurr. Comput. Pract. Exp.* **2011**, *23*, 1868–1892. [CrossRef]
24. Kamath, S.; Singh, S.; Kumar, M.S. Multiclass queueing network modeling and traffic flow analysis for SDN-enabled mobile core networks with network slicing. *IEEE Access* **2019**, *8*, 417–430. [CrossRef]

25. Leahu, H.; Mandjes, M.; Oprescu, A.M. A numerical approach to stability of multiclass queueing networks. *IEEE Trans. Autom. Control* **2017**, *62*, 5478–5484. [CrossRef]
26. Ata, B.; Peng, X. An equilibrium analysis of a multiclass queue with endogenous abandonments in heavy traffic. *Oper. Res.* **2018**, *66*, 163–183. [CrossRef]
27. Puha, A.L.; Ward, A.R. Scheduling an overloaded multiclass many-server queue with impatient customers. In *Operations Research & Management Science in the Age of Analytics*; INFORMS: Catonsville, MD, USA, 2019; pp. 189–217.
28. Long, Z.; Shimkin, N.; Zhang, H.; Zhang, J. Dynamic Scheduling of Multiclass Many-Server Queues with Abandonment: The Generalized cμ/h Rule. *Oper. Res.* **2020**, *68*, 1218–1230. [CrossRef]
29. Wu, K.; Shen, Y. Pathwise stability of multiclass queueing networks. *Discrete Event Dyn. Syst.* **2020**, 1–19. [CrossRef]
30. Atar, R.; Lev-Ari, A. Workload-dependent dynamic priority for the multiclass queue with reneging. *Math. Oper. Res.* **2018**, *43*, 494–515. [CrossRef]
31. Raviv, L.O.; Leshem, A. Maximizing service reward for queues with deadlines. *IEEE/ACM Trans. Netw.* **2018**, *26*, 2296–2308. [CrossRef]
32. Snipas, M.; Valakevicius, E. Markov Model of Multi-Class, Multi-Server Queuing System with Priorities. *J. Commun. Comput.* **2010**, *7*, 1–3.
33. Rajah, K.; Ranka, S.; Xia, Y. Scheduling bulk file transfers with start and end times. *Comput. Netw.* **2008**, *52*, 1105–1122. [CrossRef]
34. Chen, B.B.; Primet, P.V.B. Scheduling deadline-constrained bulk data transfers to minimize network congestion. In Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), Rio de Janeiro, Brazil, 14–17 May 2007; pp. 52–58.
35. Munir, K.; Primet, P.V.B.; Welzl, M. Grid Network Dimensioning by Modeling the Deadline Constrained Bulk Data Transfers. In Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications, Seoul, Korea, 25–27 June 2009; pp. 52–58.
36. Stewart, W.J. *Probability, Markov chains, Queues, and Simulation*; Princeton University Press: Princeton, NJ, USA, 2009.