A novel approach for the performance bound of QoS-aware data networks under greedy CAC policy

Bahador Bakhshi[a,*], Siavash Khorsandi[a]

[a] *Computer Engineering and Information Technology Department, Amirkabir University of Technology, Hafez Avenue, Tehran, Iran*

Abstract

The performance of QoS-aware data networks is measured in terms of acceptance probability of traffic demands requiring an amount of end-to-end bandwidth, which is determined by Call Admission Control (CAC) policy and QoS routing algorithm. In this paper, we characterize the theoretical upper bound on the network performance under the greedy CAC policy. Whereas similar problems have been studied in teletraffic networks, the existing approaches consider a specific routing algorithm and obtain its performance. They cannot be used to find performance bound in data networks using complex dynamic QoS routing algorithms. We obtain the bound through modeling the network as a multidimensional Markov chain, which does not depend on a particular routing algorithm. Since this model is computational intensive, the *greediest online* routing algorithm is developed to estimate the performance bound with lower computational complexity through simulation. In *single rate* traffic condition, we show that resources allocated by this algorithm for each traffic class is approximately weighted max-min fair. Using this observation, we develop a novel technique to estimate the bound in which an individual loss system is considered per traffic class. Simulation results show that the technique provides an accurate estimation of the bound in a wide range of traffic and network parameter settings.

Keywords: Network performance bound, QoS routing, Blocking probability, Multidimensional Markov chain, Max-Min fairness

1. Introduction

Characterizing network performance has been the subject of many studies. Wide variety of performance metrics have been used in the literature, e.g., aggregate network throughput, end-to-end delay, and bandwidth acceptance ratio. These metrics can be classified into two broad categories. The first category is the metrics used for *elastic* flow, where network traffic does not have QoS requirements. Aggregate network throughput is an example of these metrics. The metrics in the second category are used in QoS-aware networks. In these networks, traffic demands have QoS requirements; each demand is admitted to

*Corresponding author

Email addresses: bbakhshi@aut.ac.ir (Bahador Bakhshi), khorsandi@aut.ac.ir (Siavash Khorsandi)

the network if its QoS requirements are met; otherwise, it is rejected. In these networks, performance measures directly depend on the acceptance probability of traffic demands. Call acceptance probability and bandwidth acceptance ratio are examples of the metrics used in QoS-aware networks. In this paper, we consider QoS-aware data networks, and assume that each traffic demand requires a fixed amount of bandwidth. Hence, each traffic demand is accepted if the network can provide the required bandwidth from demand's source node to its destination. In the remaining, we consider the *acceptance probability* of traffic demands requiring an amount of end-to-end bandwidth as the network performance measure.

Resource allocation scheme is the main factor affecting the achievable network performance. In the problem studied in this paper, *bandwidth constrained* routing algorithm is the resource allocation scheme since each demand requires an amount of end-to-end bandwidth. To accept a demand, this algorithm is responsible to find a path that has sufficient available bandwidth, which is called *feasible path*. In general, similar to other resource allocation algorithms, bandwidth constrained routing algorithms can be *off-line* or *online*. In off-line algorithms, the information about all traffic demands is known and given to the algorithm at the beginning. However, in online algorithms, it is assumed that this information is not available; traffic demands arrive one-by-one and there is not any information about future demands. In real networks, the bandwidth constrained routing algorithm needs to be online since in many applications, the exact information about future traffic demands is not known, and cannot be predicted precisely. In this paper, we also consider online bandwidth constrained algorithms, and characterize the achievable network performance by these algorithms. We seek to obtain a theoretical upper bound on the network performance rather than compute the acceptance probability given by a particular practical routing algorithm. Note that since we consider only the bandwidth requirement, in this paper, terms 'QoS routing' and 'bandwidth constrained routing' are interchangeable.

In addition to the bandwidth constrained routing algorithm, Call Admission Control (CAC) policy also influences the achievable acceptance probability. The admission policy is either *greedy* or *non-greedy* [1]. In the former policy, each demand is accepted *if and only if* there is a feasible path for it. However, in the latter, in spite of existence of a feasible path, the CAC policy may decide to reject a demand for some reasons, e.g., because the demand is very resource consuming. In this paper, we consider the greedy policy since first, it is widely used in both data and telecommunication networks (see the references in [1, 2]); second, most of existing non-greedy policies need information about future demands [1], which is not available in online scenarios; and third, non-greedy CACs may lead to unfair resource allocation [1].

In summary, in this paper, we study the problem of characterizing the performance of QoS-aware data networks, where traffic demands require a fixed amount of end-to-end bandwidth, and the network performance is measured in terms of the acceptance probability of the demands. Our goal is to obtain an upper bound on the performance of online bandwidth constrained routing algorithms under the greedy CAC policy. By the bound, we mean the maximum achievable acceptance probability.

This problem has already been studied in both teletraffic and data networks. In the teletraffic networks, since each call requires a predefined amount of band-

width, this problem is the fundamental issue to study the network performance. It has been studied in the context of *loss networks* [3]. In the data networks, this problem has been considered in design and performance evaluation of dynamic QoS routing algorithms, especially in MPLS networks [2, 4–6].

In loss networks, Erlang Fixed Point Approximation (EFPA) is widely used to estimate the blocking probability of demands (calls). The key step in this method is to compute the load on each link which is determined by routing algorithm. Since deriving analytical models for the complicated dynamic bandwidth constrained routing algorithms used in data networks is challenging, it is not straightforward to use this method in data networks. Moreover, EFPA method computes the blocking probability given by a specific routing algorithm, while we aim to obtain a theoretical performance bound and do not consider a particular routing algorithm.

In data networks, most of the previous work has focused on design of efficient bandwidth constrained routing algorithms to maximize the network performance. However, there has not been considerable research on analyzing the achievable network performance or finding an upper bound on the demand acceptance probability.

To sum up, there is not an efficient method to obtain performance bound of QoS-aware data networks. In this paper, we study this problem, and develop techniques to compute or estimate the bound under the greedy CAC policy. In this problem, we assume that there is a set of traffic classes, which are specified by source-destination pair and traffic parameters. Each demand belongs to a traffic class. Our contributions to the problem are as follows.

- A multidimensional Markov chain model is developed to represent the *maximum* number of admissible demands per traffic class under the greedy CAC policy. This model yields a closed-form formula to compute the performance bound.

- By assuming that flows are reroutable, we develop the greediest online bandwidth constrained routing algorithm. It is shown that this algorithm exactly visits the states which are represented by the multidimensional Markov chain. Hence, it provides a simulation-based technique to estimate the bound.

- The greediest online algorithm is analyzed from the fairness point of view. In *single rate* traffic, where all traffic classes require the same amount of bandwidth, it is shown that this algorithm yields to weighted max-min fair resource allocation among traffic classes.

- For single rate traffic, we develop a novel technique to estimate the performance bound that is based on the max-min fair feature of the greediest online algorithm. In this technique, an individual loss system is considered per traffic class whose capacity is proportional to the maximum possible flow of the class, and the load is scaled by the *sharing factor* of the class, which is defined according to the amount of the max-min fair flow allocated to the class.

The remainder of this paper is organized as follows. Related work is reviewed in Section 2. In Section 3, assumptions, system models and problem

formulation are presented. The Markov chain model is presented in Section 4. We develop the greediest online algorithm in Section 5. The fairness analysis of this algorithm and the max-min fairness based estimation technique are discussed in detail in Section 6. Computational complexity of these techniques are analyzed in Section 7. Simulation results to evaluate the proposed techniques are presented in Section 8. Finally, this paper is concluded in Section 9.

2. Related work

EFPA, also known as the reduced load approximation, is the method used for more than two decades to estimate call blocking probability in loss networks [3, 7]. Two main assumptions are made in this method: 1) blocking occurs independently from link to link, and 2) demands arrive to each link as a Poisson process. Under these assumptions, the blocking probability at each link is computed by the Erlang-B formula, where the load on the link is thinned due to blocking in other links along the route of the flows passing through the link. EFPA has been applied for various versions of loss networks, e.g., in single rate demands with fixed routing [3], multi-rate demands with fixed routing [8], sequential alternative routing [9], dynamic alternative routing [10], state-dependent routing in fully connected symmetric networks [11], fully connected asymmetric network [12], non-fully connected networks [13], and multi-rate demands with least loaded routing [14]. This method also has been extended to compute the blocking probability in hierarchical networks with hierarchical routing [15–17].

It was shown in [18] that the Poisson arrival assumption may not hold in some cases, which leads to inaccurate estimation of blocking probability obtained by EFPA. Assuming the identical holding time for all demands, the authors in [18] proposed an alternative method; instead of using the EFPA method directly for a given loss system, it is applied on a system with a priority structure which is derived from the original system.

Whereas the EFPA method has a well-developed theoretical foundation, it cannot be easily used in current data networks using complex dynamic (multi-path) routing algorithms. The key step in EFPA is to approximate the reduced load on each link, which is determined by routing algorithm. For some basic routing algorithms, e.g., fixed routing or sequential alternate routing, the effect of the routing algorithm on reduced load of links has been modeled through a set of nonlinear fixed-point equations in [8, 9, 14]. However, modeling the recently proposed dynamic bandwidth constrained routing algorithm such as MIRA [4], VFD [19], and IMIRA [20], which use complex routing metrics, is very difficult. Another limitation of EFPA to be used to approach the problem considered in this paper is that the method does not provide network performance bound. It considers a specific routing algorithm and computes reduced loads according to the decisions made by the algorithm; in other words, it computes the performance of the algorithm. However, in this paper, we aim to characterize the network performance bound under the greedy CAC policy rather than compute the performance of a particular routing algorithm.

In data networks, as mentioned, designing bandwidth constrained routing algorithms has been mainly studied [4–6, 21, 22]. Almost, all the proposed solutions evaluated their efficiency through simulation.

The problem of computing (estimating) the acceptance probability in QoS-aware data networks using dynamic QoS routing algorithms was studied in [2] and [23]. In [23], the authors focused on the precomputation perspective of QoS routing; in the first step, resources are allocated for each source-destination pair, and in the second phase, an adequate path from the predetermined paths is selected for each demand. They compute the acceptance probability in the case of proportional fair resource allocation. In this paper, we do not consider any particular precomputation scheme, and analyze network performance under the greedy CAC policy.

The most closely related work to this paper is [2], where the authors developed an optimization model that takes information about all demands, and finds the maximum number of admissible demands under the greedy CAC policy[1]. Since the optimization problem is a large Mixed Integer Programming (MIP) model, it is tractable only in small networks. They proposed an alternative method to obtain a *loose* estimation of the bound. In that method, the acceptance probability is computed by the multi-class Erlang-B formula whose capacity is a lower bound on the capacity of the multi-commodity minimum cut in the network. This method has a drawback. It yields very loose estimations of network performance in most cases. This issue is discussed in more detail in Section 8.5.

3. System model, problem formulation, and solution approach

3.1. System model and assumptions

The network is modeled as a directed graph $G = (V, E)$, where V is the set of vertices, and E is the set of directed edges. Each vertex $v \in V$ corresponds to a node in the network, and each edge $(u, v) \in E$ corresponds to the link from node u to node v. The (physical) capacity of link (u, v) is $c_{(u,v)}$ units of bandwidth. Set $C = \{c_{(u,v)}\ \forall (u, v) \in E\}$ denotes the capacity of all the links in the network. The notations used throughout this paper are summarized in Table 1.

In the problem considered in this paper, each traffic demand arrives at time t^s and requests b units of bandwidth from source node s to destination node d. If the demand is accepted (under the given CAC policy), it consumes network resources until time t^e, then leaves the network; i.e., the holding time of demand is $t^e - t^s$ units of time. Therefore, each traffic demand can be denoted by 5-tuple $\delta = (s, d, b, t^s, t^e)$. The set of all demands is Δ, which is sorted in ascending order of arrival times. Both arrival time t^s and holding time $t^e - t^s$ are random variables. In this paper, we assume that both these times are exponentially distributed; therefore, the demand arrival is a Poisson process.

Traffic demands are classified according to the source, destination, and stochastic parameters of traffic. Each traffic class is denoted by $\tau = (s, d, b, \lambda, \mu^{-1})$, where λ is the arrival rate, and μ^{-1} is the average holding time. The previous assumption implies that demands, which belong to traffic class τ, arrive according a Poisson process with mean λ demands per unit of time, and the holding time of the demands is exponentially distributed with mean μ^{-1} units of time.

[1] They modeled greedy CAC policy by adjusting the benefit associated to each demand.

Table 1: Notations

Notation	Description
u, v	Nodes in the network
V	Set of nodes
(u, v)	Directed link from u to v
E	Set of edges
$c_{(u,v)}$	Capacity of link (u, v)
C	Set of link capacities
$\delta = (s, d, b, t^s, t^e)$	Demand for bandwidth b from s to d during time t^s to t^e
Δ	Set of demands
$\tau = (s, d, b, \lambda, \mu^{-1})$	Traffic class from s to d with bandwidth b, arrival rate λ, and average holding time μ^{-1}
D	Set of traffic classes
ρ	Load of traffic class, $\rho = \lambda\mu^{-1}$
F_i	Flow of class τ_i
$f^i_{(u,v)}$	Flow of class τ_i on link (u, v)
$\boldsymbol{n} = (n_1, \ldots, n_{\lvert D\rvert})$	State of network
Ω	Set of feasible states
$p(\boldsymbol{n})$	Probability of state $\boldsymbol{n}$
$\boldsymbol{n}_0$	$\boldsymbol{n}_0 = (0, 0, \ldots, 0)$
$\boldsymbol{n}^i_0$	$\boldsymbol{n}^i_0 = (n_1, \ldots, n_{i-1}, 0, n_{i+1}, \ldots, n_{\lvert D\rvert})$
$F^*_{i,\boldsymbol{n}^i_0}$	Maximum flow for τ_i in state $\boldsymbol{n}^i_0$
$N_{i,\boldsymbol{n}^i_0}$	Maximum # of demands of τ_i in state $\boldsymbol{n}^i_0$
$\boldsymbol{n}^i$	$\boldsymbol{n}^i = (n_1, \ldots, n_{i-1}, N_{i,\boldsymbol{n}^i_0}, n_{i+1}, \ldots, n_{\lvert D\rvert})$
w	Weight of τ
W	Set of traffic classes' weights
z	Weighted max-min fair flow of τ
Z	Set of weighted max-min fair allocated flows

The load of traffic class τ is denoted by ρ which is $\rho = \lambda\mu^{-1}$. The set D contains all the traffic classes.

In addition to the assumption of exponential distribution of arrival and holding times, in Section 6, we assume that traffic is single rate; which implies that all traffic classes require *the same amount of bandwidth*. Moreover, we assume that flows are *splittable* that means the required bandwidth b can be divided and sent over multiple paths from source node to destination at any arbitrary granularity.

3.2. Problem formulation

In general, bandwidth provisioning problem in QoS-aware data network is as follows. A set Δ of demands and a network G with capacity C are given. Demands arrive one-by-one and the goal is to accept as much as possible demands under the given CAC policy subject to the *capacity constraint*. This constraint implies that the amount of flow on each link (u, v) cannot be greater than the link capacity $c_{(u,v)}$.

In more detail, upon arrival of a demand $\delta \in \Delta$ at time t^s, call admission controller decides whether to accept the demand or reject it. It *must reject* the demand if there is not enough free capacity to allocate the required bandwidth b

from s to d, i.e., when the capacity constraint is not satisfied. For this purpose, bandwidth constrained routing algorithm attempts to find a path that allocating the required bandwidth b along the path does not volatile the capacity constraint of any link in the network. If such a path exists, it is called a *feasible* path. In the case of existence of a feasible path, call admission controller *can* accept the demands that creates a flow from s to d at rate b until time t^e. At time t^e, the demand leaves the network and frees the allocated bandwidth. In this problem, the network performance is measured in terms of demand acceptance probability. Higher probability (rate) implies more efficient network resource utilization and better performance.

In this paper, we focus our attention on the greedy CAC policy and as mentioned, assume that flows are splittable. This assumption implies that instead of finding a single feasible path, bandwidth constrained routing algorithm can divide the required bandwidth b and allocate it along multiple feasible paths. Greedy CAC policy means that each demand is accepted *if and only if* there is feasible path(s) for it. Here, we consider online bandwidth constrained routing algorithms that implies to find feasible path(s) for a demand δ_i, these algorithms can only use the information of previous demands, i.e., $\{\delta_j \text{ s.t. } t_j^s \leq t_i^s\}$; in other words, they do not know any information about future demands.

Under these assumptions (splittable flows, greedy CAC policy, and online routing), our goal in this paper is to obtain a theoretical upper bound on the network performance, that is the maximum achievable acceptance probability through online bandwidth constrained routing algorithms under the greedy CAC policy.

3.3. Solution approach

The steps taken in the following sections to obtain the network performance bound are shown in Fig. 1. The problem studied in this paper, in some ways, is similar to the classical problem of finding rejection (blocking) probability in loss systems. Borrowing ideas from the queuing theory, at the first step, we model the network and its state as a multidimensional Markov chain in Section 4. The solution of the chain provides an upper bound on the achievable acceptance probability under the greedy CAC policy. However, constructing the Markov chain and finding the state probabilities are very computationally intensive.

At the second step, in Section 5, we develop a bandwidth constrained routing algorithm, which is the *greediest online* algorithm. It accepts each demand if and only if there is a feasible configuration of flow routes in the network. It is shown that the stochastic process corresponding to the number of demands accepted by the algorithm is, in fact, the multidimensional Markov chain. Therefore, the algorithm can be used to estimate the acceptance probability through simulation.

In Section 6.1, which is the third step, under the assumption of single rate traffic, by analyzing the behavior of the greediest online algorithm, we show that its resource allocation policy is weighted max-min fair. Finally, at the fourth step, which is discussed in Section 6.2, we consider a single loss system and restate the Erlang-B formula to show the role of weighted max-min fair resource allocation in computing the blocking probability of the system. Putting these analyses altogether, we propose a technique to estimate the acceptance probability achievable under the greedy CAC policy.

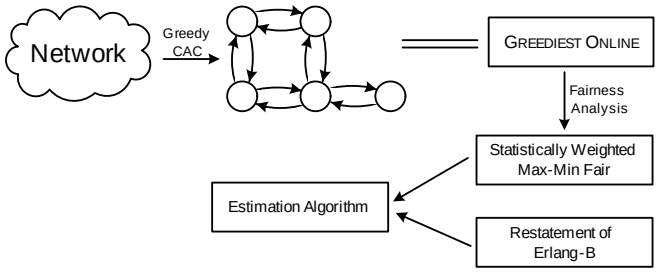


Figure 1: The steps of the proposed approach for the performance bound of QoS-aware data network under the greedy CAC policy

It is important to note that in the following sections, we develop algorithms to obtain a *theoretical* upper bound on the network performance. Most practical online QoS routing algorithms, e.g., [4–6, 21, 22, 24, 25], use greedy CAC policy and due to practical limitations, find a *single-path* route for each demand which is *fixed* during the holding time of the demand. In this paper, we also consider the greedy CAC policy; however, to find a theoretical upper bound, we relax the *fixed-single-path-route* constraint. In Section 4, we assume that flows are splittable and use multi-path routing. Moreover, in Section 5, in the greediest online algorithm, we assume that flows are reroutable. From optimization theory point of view, these relaxations imply that the results obtained in this paper are *theoretical upper bounds* on the maximum performance (acceptance rate) achievable by practical online QoS routing algorithms under the CAC policy.

4. Markov chain model

In this section, we derive an analytical formula for the network performance bound that is the *maximum* demand acceptance probability under the greedy CAC policy. For this purpose, the maximum number of admissible demands under this policy, disrespect of QoS routing algorithm is modeled as a multi-dimensional Markov chain. In this model, the state of the network at time t is specified by $\boldsymbol{n}(t) = \big(n_1(t), n_2(t), \ldots, n_{|D|}(t)\big)$, where $n_i(t)$ is the number of existing flows (accepted demands) from traffic class τ_i at time t. State $\boldsymbol{n}$ is a feasible state if there is a flow allocation in the network that provides $b_i n_i$ units of bandwidth for every traffic class τ_i while satisfies the capacity constraint. The set of all feasible states is denoted by Ω. There are two key observations about Ω and its relation to the greedy CAC policy. First, if $n_i > 0$ demands from traffic class τ_i are admissible, then obviously, $n_i - 1$ demands from the class must also be acceptable since it needs lesser resources. Formally, for each traffic class τ_i if $\boldsymbol{n} \in \Omega$ where $n_i > 0$, then $\boldsymbol{n} - e_i \in \Omega$, where e_i is defined as the $|D|$ dimensional vector with a '1' at the ith coordinate and '0' at the other components. This observation implies that Ω is a *convex set*. Second, we know that under the greedy CAC policy, demand $\delta = (s_i, d_i, b_i, t_i^s, t_i^e)$ from traffic class τ_i is accepted if and only if $\boldsymbol{n}(t_i^s) + e_i \in \Omega$. Hence, by definition, since Ω is a convex set, greedy CAC is a *coordinate convex* policy [26].

We need to find all feasible states $\boldsymbol{n} \in \Omega$ to compute the acceptance rate using the state probabilities of the chain. To do this, consider state $\boldsymbol{n}_0^i =$

$(n_1, \ldots, n_{i-1}, 0, n_{i+1}, \ldots, n_{|D|})$; let $N_{i,\boldsymbol{n}_0^i}$ denote the maximum number of admissible demands from traffic class τ_i when the network is in this state. Convexity of Ω implies that $\boldsymbol{n}^i + e_i \notin \Omega$ and $\{\boldsymbol{n'} = \boldsymbol{n}^i - je_i,\ 0 \leq j \leq N_{i,\boldsymbol{n}_0^i}\} \subset \Omega$ where $\boldsymbol{n}^i = (n_1, \ldots, n_{i-1}, N_{i,\boldsymbol{n}_0^i}, n_{i+1}, \ldots, n_{|D|})$.

Therefore, to compose Ω, we need to find $N_{i,\boldsymbol{n}_0^i}\ \forall \tau_i,\ \forall n_0^i$. It is obtained through solving an optimization problem. Let F_i denote the amount of the flow from s_i to d_i. Suppose that $F^*_{i,\boldsymbol{n}_0^i}$ is the maximum of F_i when the network is in state $\boldsymbol{n}_0^i$ wherein $n_j b_j$ units of bandwidth is guaranteed for all $\tau_j \in D \setminus \tau_i$. Clearly, we have $N_{i,\boldsymbol{n}_0^i} = \lfloor F^*_{i,\boldsymbol{n}_0^i}/b_i \rfloor$, where $\lfloor x \rfloor$ is the largest integer number smaller than x. The maximum flow $F^*_{i,\boldsymbol{n}_0^i}$ is obtained using the following optimization model.

Our goal is to find the maximum flow; so, the objective function is

$$\text{maximize } F_i. \tag{1}$$

Flow of each traffic class is specified by the well-known flow conservation constraint that implies for each traffic class τ_i, F_i amount of flow is sent out from the source node and received at the destination while it is not consumed or generated in intermediate nodes. More formally, the constraint is

$$\sum_{(u,v)\in E} f^j_{(u,v)} - \sum_{(v,u)\in E} f^j_{(v,u)} = \begin{cases} F_j, & \text{if } u = s_j \\ -F_j, & \text{if } u = d_j \\ 0, & \text{otherwise} \end{cases} \qquad \forall u \in V, \forall \tau_j \in D. \tag{2}$$

Where $f^j_{(u,v)}$ is the amount of flow from s_j to d_j which routed on link (u, v). Obviously, flow cannot be negative; so, we have

$$f^j_{(u,v)} \geq 0 \qquad \forall (u,v) \in E, \forall \tau_j \in D. \tag{3}$$

Moreover, note that $f^j_{(u,v)}$ is a continuous real variable; hence, in this model, flows are splittable, and we use multipath routing.

Since state $\boldsymbol{n}^i$ must be feasible, we need to reserve the required resources for the existing flows of traffic classes $\tau_j \in D \setminus \tau_i$, which is $n_j b_j$ units of bandwidth for class τ_j. Hence, we have

$$F_j = b_j n_j \qquad \forall \tau_j \in D \setminus \tau_i. \tag{4}$$

The capacity constraint needs to be satisfied to maintain feasibility of the flow allocations; formally,

$$\sum_{\tau_j \in D} f^j_{(u,v)} \leq c_{(u,v)} \qquad \forall (u,v) \in E. \tag{5}$$

The optimization model to find $F^*_{i,\boldsymbol{n}_0^i}$ is obtained by putting the aforementioned objective function and constraints altogether as follows.

Model:	MaxFlow$(G, C, D, \boldsymbol{n}_0^i, \tau_i)$
Objective:	(1)
Subject to:	(2)–(5).

The set Ω is composed using the MaxFlow model by the *recursive* StateSpace algorithm, which is depicted in Algorithm 1. The set of all feasible states of the multidimensional Markov chain is obtained by creating an empty set Ω and calling StateSpace$(G, C, D, \boldsymbol{n}_0, 1, \Omega)$, where $\boldsymbol{n}_0 = (0, 0, \ldots, 0)$.

Algorithm 1 : StateSpace$(G, C, D, \boldsymbol{n}, i, \Omega)$

```
1: F*_{i,n_0^i} ← MaxFlow(G, C, D, n_0^i, τ_i)
2: N_{i,n_0^i} ← ⌊F*_{i,n_0^i} / b_i⌋
3: if i = |D| then
4:    Ω ← Ω ∪ {n' = n^i − j e_i,  0 ≤ j ≤ N_{i,n_0^i}}
5: else
6:    for j = 0 to N_{i,n_0^i} do
7:       n' ← n + j e_i
8:       StateSpace(G, C, D, n', i + 1, Ω)
9:    end for
10: end if
```

In the multidimensional Markov chain with state space Ω, assuming that $\{\boldsymbol{n}, \boldsymbol{n} + e_i, \boldsymbol{n} - e_i\} \subset \Omega$, the transition rate from $\boldsymbol{n}$ to $\boldsymbol{n} + e_i$ is λ_i, and from state $\boldsymbol{n}$ to $\boldsymbol{n} - e_i$ is $n_i\mu_i$. Let $p(\boldsymbol{n})$ denote the probability of state $\boldsymbol{n}$. In [27], it is shown that if the CAC policy is coordinate convex, and the arrival and service processes are both memoryless, then the detailed balance equations are satisfied, $p(\boldsymbol{n} - e_i)\lambda_i = p(\boldsymbol{n})n_i\mu_i$, and the Markov chain has a product form solution. Hence, since the greedy CAC policy is coordinate convex, the arrival process is Poisson, and the holding times are exponentially distributed, we have

$$p(\boldsymbol{n}) = p(\boldsymbol{n}_0) \prod_{1 \le i \le |D|} \frac{\rho_i^{n_i}}{n_i!}, \tag{6}$$

where

$$p(\boldsymbol{n}_0) = \Big(\sum_{\boldsymbol{n} \in \Omega} \Big(\prod_{1 \le i \le |D|} \frac{\rho_i^{n_i}}{n_i!} \Big) \Big)^{-1}. \tag{7}$$

The acceptance probability is obtained from the state probabilities. For each state $\boldsymbol{n} \in \Omega$, we define blocked classes as $B(\boldsymbol{n}) = \{\tau_i \in D \text{ s.t. } \boldsymbol{n} + e_i \notin \Omega\}$. This is the set of traffic classes that a new demand from them is rejected if the network is in state $\boldsymbol{n}$. The rejection rate in state $\boldsymbol{n}$ is $r(\boldsymbol{n}) = \sum_{\tau_i \in B(\boldsymbol{n})} \lambda_i$. Therefore, the total acceptance probability is

$$P_A = 1 - \frac{1}{\Lambda} \sum_{\boldsymbol{n} \in \Omega} \big(p(\boldsymbol{n}) r(\boldsymbol{n}) \big), \tag{8}$$

where Λ is the total arrival rate, $\Lambda = \sum_{\tau_i \in D} \lambda_i$.

Equation (8) provides a closed-form formula to compute the acceptance probability. For a given G, C, and D, we can use the StateSpace algorithm to compose Ω, and then compute P_A by (8). Note that for two reasons, it is an upper bound on the network performance achievable through practical *online* QoS routing algorithms nuder the greedy CAC policy. First, the maximum number of admissible demands is used to construct the Markov chain while practical online

algorithms perhaps cannot accept the maximum number of demands because selected route for a demand by these algorithms may cause rejecting upcoming demands due to lack of information about future demands in online operation mode; this issue will be explained in more detail in the following section. Second, multipath routing and flow splitting is used to compose the chain while practical algorithms are single path; and as mentioned, multipath routing is the relaxation of the single-path routing constraint.

However, this approach is impractical due to very intensive computational requirements. To compute P_A, we have to enumerate *all* the states $\boldsymbol{n} \in \Omega$ that was shown is #P-Complete [28]. In the following sections, we develop other techniques to estimate the acceptance probability without enumerating all the feasible states.

5. Simulation-based estimation

Simulation is an alternative method to *estimate* demand acceptance probability. It consists of the following steps. First, for each traffic class $\tau_i \in D$, a set Δ_i of traffic demands is generated, $\Delta_i = \{\delta_j = (s_i, d_i, b_i, t_j^s, t_j^e)\}$, where the arrival rate of the demands is Poisson with mean λ_i, and the holding time $t_j^e - t_j^s$ is an exponential random variable with mean μ_i^{-1}. The set of all demands is $\Delta = \bigcup_{\tau_i \in D} \Delta_i$, which is sorted in ascending order of arrival time. Second, an online greedy bandwidth constrained routing algorithm is used to find feasible paths for the demands one-by-one. An estimation of the acceptance probability will be the number of accepted demands over the total number of demands.

In this technique, the used routing algorithm is an important issue. Various algorithms lead to different acceptance probabilities. Ability of algorithms to efficiently utilize network resources determines the achievable acceptance probability. Practical online bandwidth constrained routing algorithms may not be able to use all available network resources. These algorithms are not aware of future demands; therefore, the resource (path) allocation for a given demand may cause of rejection of many subsequent demands, that reduces the acceptance probability. For example, assume that two feasible paths are available for a given demand δ_i where the first path is a part of the only feasible path for a future demand δ_j. Since an online routing algorithm does not know that δ_j will arrive after δ_i, it may select the first path for δ_i. In this case, δ_j will be rejected in spite of existence a path allocation, wherein δ_i is routed through its second path, to accept both demands.

In general, due to significant overheads, practical online bandwidth constrained routing algorithms do not reroute flows. This causes that *route selection* for each demand affects acceptance probability of future demands. Obviously, this issue is resolved if it is assumed that routing algorithms can *reroute all existing flows*. Reconsider the previous example, and assume that the first path is selected for demand δ_i upon its arrival. When demand δ_j arrives, an online routing algorithm with rerouting capability can reroute δ_i to the second path that frees that first path to accept δ_j. An online bandwidth constrained routing with rerouting capability in developed as follows.

Upon arrival of a new demand, we consider all the existing flows and the new demand altogether, and attempt to find a feasible flow allocation for them. If such allocation exists, the new demand is accepted; otherwise, it is rejected.

More specifically, suppose that the network is in state $\boldsymbol{n}(t^s) = \big(n_1(t^s), ..., n_{|D|}(t^s)\big)$ at time t^s, which is the arrival time of demand $\delta = (s_i, d_i, b_i, t^s, t^e)$ of traffic class τ_i. The algorithm checks the feasibility of state $\boldsymbol{n'} = \boldsymbol{n}(t^s) + e_i$, and accepts the demand if $\boldsymbol{n'}$ is feasible; otherwise, it rejects δ. Feasibility of state $\boldsymbol{n'}$ is checked using the MaxFlow optimization model. For this purpose, we find $F^*_{i,\boldsymbol{n}^i_0} = \textsc{MaxFlow}(G, C, D, \boldsymbol{n}^i_0(t^s), \tau_i)$, where $\boldsymbol{n}^i_0(t) = \big(n_1(t), \ldots, n_{i-1}(t), 0, n_{i+1}(t), \ldots, n_{|D|}(t)\big)$. If $F^*_{i,\boldsymbol{n}^i_0} \geq (n_i(t^s) + 1)b_i$, the new demand is accepted since this inequality implies that $\boldsymbol{n'}$ is a feasible state. Note that transition from $\boldsymbol{n}(t^s)$ to $\boldsymbol{n'}$ may cause rerouting all the existing flows. This algorithm accepts each demand *if and only if* there is a feasible flow allocation, i.e., $\boldsymbol{n'}$ is feasible; hence, it implements the greedy CAC policy. Indeed, it is the *greediest online* algorithm, since the existence of a feasible flow allocation is investigated by the MaxFlow optimization model rather than a heuristic method.

It is worthwhile to mention that the greediest online algorithm yields an upper bound on the performance of practical online bandwidth constrained routing for two reasons. First, practical algorithms are usually single path while greediest online is a multi-path routing algorithm. It can accept a given demand by dividing it into multiple sub-flows and routing them even if there is not any single feasible path to accept the demand. Second, the greediest online algorithm can reroute all existing flows to find a feasible path for a new demand; however, practical algorithms usually are not allowed to reroute flows because of overhead and limitations in real networks. This leads to rejection of demands in spite of existing a routing to accept them, which is illustrated by the example at the beginning of this subsection.

Assume that the stochastic process $X(t)$ is the number of existing flows per traffic class at time t, which have been accepted using the greediest online algorithm. The following theorem shows the key attribute of the stochastic process.

Theorem 1. *Assume that demand arrival process is Poisson, and the holding times are exponentially distributed. First, the stochastic process $X(t)$ is Markovian. Second, it is, in fact, the multidimensional Markov chain with state space Ω, which is developed by the* StateSpace *algorithm.*

Proof. The proof is in Appendix A.1. □

This correspondence between the greediest online algorithm and the multidimensional Markov chain implies that we can use the algorithm to *estimate* the performance bound whose exact value is given by (8). Pseudo-code of the estimation algorithm is shown in Algorithm 2. In this algorithm, A and B are, respectively, the accepted demands set and working demands set (the set of existing flows and the new demand). Function TrafficClass returns the traffic class of a demand. Function GetN returns the number of existing flows of each traffic class. The main body of GreediestOnline is the greediest online algorithm. For each demand, GreediestOnline finds the maximum flow F_i from the source to the destination of the demand, and accepts the demand if the flow is sufficient. If demand is rejected, it is excluded from the working set in line 17. Moreover, if a demand leaves the network before the next demand arrives, it is also removed from the working set in line 8. This algorithm returns $|A|/|\Delta|$ as the estimation of the acceptance probability.

Algorithm 2 : GreediestOnline(G, C, D, Δ)

$A \leftarrow \{\}$, $B \leftarrow \{\}$
for $i = 1$ **to** $|\Delta|$ **do**
  $\delta_i \leftarrow \Delta[i]$
  $\tau_i \leftarrow$ TrafficClass(D, δ_i)
  $B \leftarrow B \cup \delta_i$
  for $\forall \delta_j \in B$ **do**
    if $t_j^e < t_i^s$ **then**
      $B \leftarrow B \setminus \delta_j$
    end if
  end for
  $\{\boldsymbol{n}_0^i, n_i\} \leftarrow$ GetN(B, D)
  $F^*_{i,\boldsymbol{n}_0^i} \leftarrow$ MaxFlow$(G, C, D, \boldsymbol{n}_0^i, \tau_i)$
  if $F^*_{i,\boldsymbol{n}_0^i} \geq (n_i + 1)b_i$ **then**
    $A \leftarrow A \cup \delta_i$
    $B \leftarrow B \cup \delta_i$
  else
    $B \leftarrow B \setminus \delta_i$
  end if
end for
return $|A|/|\Delta|$

The key advantage of using the GreediestOnline algorithm over the multidimensional Markov chain based analysis is that we do not need to enumerate *all* feasible states. This algorithm visits states according to demand arrivals and leaves. In fact, this algorithm only visits the *most probable* states instead of enumerating all the feasible states. This can significantly reduce the computational complexity needed to find the acceptance probability, especially in large networks with many traffic classes, where enumerating all the states is infeasible.

Whereas this algorithm is a valuable tool to estimate the acceptance probability *per scenario*, it is a simulation-based technique. The estimation error of this method is proportional to $|\Delta|^{-1}$. To achieve a reasonable confidence interval, we need to simulate quite large demand sets. This can be very time consuming because this algorithm needs to solve an optimization problem per demand arrival. In the following section, we develop an alternative estimation technique with lower computational complexity.

6. Weighted Max-Min Fairness based estimation

In this section, we analyze the greediest online algorithm, and show that the resource allocation strategy implemented by this algorithm can be approximated by the weighted max-min fair allocation scheme. Based on this observation, we propose a novel technique to estimate the upper bound of network performance.

6.1. Fairness analysis

The greediest online algorithm can be considered as a dynamic resource allocation scheme that, on average, allocates an amount of bandwidth to every

traffic class. Analysis of the allocated bandwidths helps us to understand the long-term behavior of the algorithm, and consequently, to estimate its provided acceptance probability. For the sake of simplicity of discussion, we start the analysis from a very special case, wherein the following assumptions are made.

A1. $b_i = b_j = \epsilon, \ \forall \tau_i, \tau_j \in D$.

A2. $\lambda_i = \lambda_j, \ \forall \tau_i, \tau_j \in D$.

A3. $\mu_i^{-1} = \mu_j^{-1} = M, \ \forall \tau_i, \tau_j \in D$, where $M \to \infty$.

A4. In each set of $|D|$ successive demands, there is a demand from every traffic class. This is possible due to assumption **A2**.

A5. Demand set is very large, $|\Delta| \to \infty$.

Let $Z = \{z_1, \ldots, z_{|D|}\}$ be the max-min fair bandwidth allocation among traffic classes, where z_i is the max-min flow rate of traffic class τ_i, which is the flow from s_i to d_i. Let y_i be the bandwidth allocated to class τ_i at the end of the GreediestOnline algorithm. Moreover, let $Y = \{y_1, \ldots, y_{|D|}\}$. The following theorem shows the relation between Z and Y.

Theorem 2. *For a set D of traffic classes and its corresponding demand set Δ that satisfy the assumptions **A1**–**A5**, if the* GreediestOnline *algorithm is used to accept the demands, we have $|z_i - y_i| < \epsilon \ \forall \tau_i \in D$.*

Proof. The proof is in Appendix A.2. □

An important fact is that most of these assumptions may not hold in practice. In the following, we analyze the effect of each assumption. If assumption **A4** does not hold, then instead of *exactly one* demand per class in each $|D|$ successive demands, there is *on average* one demand per class. In this case, the bandwidth allocation Y is *statistically* max-min fair, which implies $|z_i - \mathbb{E}[y_i]| < \epsilon$. If assumption **A2** is removed, and instated of it, we assume that λ_i is an integer number, then the assumption **A4** will be "there are λ_i demands from class τ_i in each Λ successive demands." In this case, the bandwidth allocation Y is *weighted* max-min fair, where the weight of τ_i, which denoted by w_i, is λ_i. Because per ϵ bandwidth for class τ_i, $\epsilon\lambda_j/\lambda_i$ amount of bandwidth is allocated for class τ_j. If instead of assumption **A3**, we assume that the holding time of each demand of traffic class τ_i is a random variable with mean $\mu_i^{-1} \ll \infty$, where μ_i^{-1} may not equal to μ_j^{-1}, then the allocated bandwidth for each class is freed over the time. The free resources are shared among traffic classes equally, due to assumption **A1** and **A2**. However, traffic classes with longer holding time acquire more bandwidth on average. In this case, the allocated bandwidths change over time; hence, y_i is considered as the *time average* of the allocated bandwidth for class τ_i by the GreediestOnline algorithm. If the aggregate offered load in terms of bandwidth, $\sum_{\tau_i \in D} \rho_i b_i$, is not large enough to fully utilize the network capacity, we have $y_i < z_i$ for some classes. However, if the network resources are completely utilized, the bandwidth allocation Y can be considered *statistically weighted* max-min fair, where the weight is proportional to the holding time. These analyses are summarized in Table 2, which yield the following proposition.

Table 2: Effect of the assumptions **A1**–**A5** on max-min fairness of the greediest online algorithm in the case of $\epsilon \to 0$

Assumptions					Fairness
A1	**A2**	**A3**	**A4**	**A5**	
√	√	√	√	√	Max-Min Fair
√	√	√	×	√	Statistically Max-Min Fair
√	×	√	√	√	Weighted Max-Min Fair, $w_i = \lambda_i$
√	√	×	√	√	Statistically Weighted Max-Min Fair, $w_i = \mu_i^{-1}$
√	×	×	×	√	Statistically Weighted Max-Min Fair, $w_i = \rho_i$

Proposition 3. *Under assumptions **A1** and **A5**, if the network capacity is fully utilized, then the long-term time average of bandwidth allocation by the* GreediestOnline *algorithm is statistically weighted max-min fair, where the weight of traffic class τ_i is ρ_i.*

The assumptions **A1** and **A5** are crucial to achieve max-min fairness by the greediest online algorithm. If assumption **A5** is removed, the network resources are not fully utilized; hence, the allocated bandwidth is less than the max-min fair flow rate. If assumption **A1** does not hold, there may be a large gap between Y and Z since different amounts of bandwidth are allocated for different classes per accepted demand.

6.2. Estimation technique

In this section, we develop a technique to estimate the upper bound of the acceptance probability; it is based on max-min fairness of the greediest online algorithm. As discussed in Section 6.1, assumption **A1** is needed to achieve max-min fairness by the algorithm. Therefore, in the following, we assume that traffic is *single rate*, which means all traffic classes require the same amount of bandwidth.

The major complexity to estimate acceptance probability is to consider the effect of different classes on each other. The idea in the proposed technique is that if the effect is modeled, we can consider an individual loss system per traffic class while taking the effect into account through adjusting the parameters of the systems. In the following, we model the effect using the max-min fairness feature of the greediest online algorithm.

For the ease of discussion, we start from a single loss system and restate the Erlang-B formula from fairness point of view. Consider a system with C servers. D types of customers use this system. Customers of type i arrive according to a Poisson process with mean λ_i and need a server. The holding time of each customer of type i is exponentially distributed with mean μ_i^{-1}. Since all the customers need the same number of servers, this system is, in fact, a loss system with capacity C and total offered load $\rho = \sum_{i=1}^{D} \rho_i$. Therefore, according to the Erlang-B formula, the blocking probability is

$$P_B = \frac{\rho^C}{C!} \left(\sum_{j=0}^{C} \frac{\rho^j}{j!} \right)^{-1}.$$

This probability can be rewritten as

$$P_B = \sum_{i=1}^{D} \frac{\lambda_i}{\Lambda} p_{b,i}, \tag{9}$$

where $\Lambda = \sum_{i=1}^{D} \lambda_i$, and

$$p_{b,i} = \frac{(\gamma_i \rho_i)^C}{C!} \left(\sum_{j=0}^{C} \frac{(\gamma_i \rho_i)^j}{j!} \right)^{-1}, \tag{10}$$

where $\gamma_i = \rho/\rho_i$. The key observation is the relation between γ_i and the weighted max-min fair resource (server) allocation in this system. It is easy to see that if weight $w_i = \rho_i$ is considered for each traffic class τ_i, the weighted max-min fair resource allocation is $z_i = C\rho_i/\rho$. Moreover, note that the *maximum possible* available resource for each traffic class is C servers. Therefore, γ_i is the maximum possible available resources for customer type i divided by the weighted max-min fair allocated resource to it. Hence, (10) can be interpreted as the blocking probability of a loss system with capacity C and load $\gamma_i \rho_i$. This discussion, in fact, proves the following lemma.

Lemma 4. *Blocking probability of a loss system with C servers and D types of customers, where all customers need one server, and the offered load by customer type i is ρ_i, is the weighted sum (9) of the blocking probabilities of individual loss systems considered per customer type. The system of customer type i has C servers and load $\gamma_i \rho_i$. Where, γ_i is the maximum number of available servers for type i in isolation divided by the weighted max-min fair number of allocated servers to the type.*

Lemma 4 is the idea behinds the proposed estimation technique. In a single loss system, where resource allocation is inherently weighted max-min fair, this lemma provides a method to compute blocking probability. Hence, if resource allocation in the network is weighted max-min fair, then we can use a similar method to *estimate* the acceptance probability. The key point is that as shown in Section 6.1, the bandwidth allocation by the greediest online algorithm is approximately weighted max-min fair. Therefore, this idea is applicable here, which is explicated in the following.

Let $F^*_{i,\boldsymbol{n}_0}$ denote the maximum flow from s_i to d_i when the traffic class is considered in isolation, i.e., network is in state $\boldsymbol{n}_0$. In the presence of other traffic classes, when the network is fully utilized, the greediest online algorithm allocates approximately z_i amount bandwidth to class τ_i instead of $F^*_{i,\boldsymbol{n}_0}$. Therefore, the effect of other classes on class τ_i can be considered as a reduction of available capacity from $F^*_{i,\boldsymbol{n}_0}$ to z_i. Using Lemma 4, we take the effect into account by scaling the offered load. In this case, γ_i is defined as $\gamma_i = F^*_{i,\boldsymbol{n}_0}/z_i$. We call γ_i the *sharing factor* of traffic class τ_i. It shows how traffic class τ_i shares its maximum possible available capacity with other classes. Similar to the lemma, to estimate acceptance probability, we consider an individual loss system for each traffic class whose capacity is proportional to the maximum flow $F^*_{i,\boldsymbol{n}_0}$ and the offered load of the class is scaled by the sharing factor γ_i.

Algorithm 3 shows the pseudo-code of the WMMFestimate algorithm to estimate the acceptance probability obtained by GreediestOnline in single-rate traffic condition that is the network performance bound in QoS-aware

data networks under the greedy CAC policy. In this algorithm, set $W = \{w_1, \ldots, w_{|D|}\}$ denotes the weights of traffic classes, where $w_i = \rho_i$. The weighted max-min fair flow allocation Z is obtained by the WeightedMaxMinFair algorithm, which is developed in Section 6.2.1. The maximum flow $F^*_{i,\boldsymbol{n}_0}$ is given by the MaxFlow model. Note that the capacity of the loss system for class τ_i is $N_{i,\boldsymbol{n}_0} \leftarrow \lfloor F^*_{i,\boldsymbol{n}_0}/b_i \rfloor$. This algorithm first finds $p_{b,i}$ $\forall \tau_i \in D$, then computes the acceptance probability according to the weighted sum (9).

Algorithm 3 : WMMFestimate(G, C, D)

$W \leftarrow \{w_i = \rho_i \ \forall \tau_i \in D\}$
$Z \leftarrow$ WeightedMaxMinFair(G, C, D, W)
for $\forall \tau_i \in D$ **do**
  $F^*_{i,\boldsymbol{n}_0} \leftarrow$ MaxFlow$(G, C, D, \boldsymbol{n}_0, \tau_i)$
  $\gamma_i \leftarrow F^*_{i,\boldsymbol{n}_0}/z_i$
  $N_{i,\boldsymbol{n}_0} \leftarrow \lfloor F^*_{i,\boldsymbol{n}_0}/b_i \rfloor$
  $p_{b,i} \leftarrow \dfrac{(\gamma_i \rho_i)^{N_{i,\boldsymbol{n}_0}}}{N_{i,\boldsymbol{n}_0}!} \left(\displaystyle\sum_{j=0}^{N_{i,\boldsymbol{n}_0}} \dfrac{(\gamma_i \rho_i)^j}{j!} \right)^{-1}$
end for
$P_B \leftarrow \sum_{\tau_i \in D} \frac{\lambda_i}{\Lambda} p_{b,i}$
return $1 - P_B$

6.2.1. Weighted max-min fair algorithm

The progressive filling algorithm [29] is the well-known approach to achieve max-min fair flow allocation among traffic classes. It is an iterative algorithm to compute the max-min flow rate when only *a single path* for each traffic class is given. This algorithm starts with all flow rates equal zero, and grows all the rates together equally until some links are saturated. The rate of the flows (traffic classes) passing through the saturated links cannot be increased anymore. These are called saturated classes. The algorithm fixes the rates of these saturated classes at the current value; and continues to increase rates of unsaturated classes until new saturated links and traffic classes are found, and so on. The algorithm terminates when all classes get saturated. In the problem studied in this paper, routing is not given; and we seek to find the weighted max-min fair flow allocation.

A few algorithms have been proposed to compute max-min fair flow allocation when routing is not given under both single-path and multi-path routing schemata (see the references in [30]). In this section, we use the idea of the progressive filling algorithm [29] to obtain the weighted max-min fair flow allocation. In the proposed algorithm, two optimization models are solved iteratively. These optimization problems perform the two basic operations in each iteration of the progressive filling algorithm. The first model equally grows the *normalized* flow rates of the unsaturated traffic classes, where the flow rates are normalized by the class weights w_i. The second model is used to find saturated classes in each iteration. Both these models are developed through modifications in the MaxFlow model. Two additional parameters σ_i and ϕ_i are used in these models. Saturation of class τ_i is denoted by σ_i, where

$$\sigma_i = \begin{cases} 1, & \text{if class } \tau_i \text{ is } \textbf{not} \text{ saturated} \\ 0, & \text{otherwise.} \end{cases}$$

As mentioned, in the progressive filling algorithm, allocated flow rate for each class is updated in each iteration of the algorithm. The current allocated flow rate for class τ_i in an iteration of the algorithms is denoted by ϕ_i.

The first model, that equally grows the normalized flow rates of unsaturated classes in each iteration of the progressive filling algorithm, is obtained by the following modifications in the MaxFlow model. The objective function is replaced by

$$\text{maximize } \beta, \tag{11}$$

where β is specified by

$$\frac{F_i}{w_i} \geq \beta\sigma_i \quad \forall\tau_i \in D. \tag{12}$$

This means that if class τ_i is not saturated, $\sigma_i = 1$, its normalized flow rate, F_i/w_i, must be at least β, which is the equal normalized rate for all the unsaturated classes. Note that if a traffic class is saturated, $\sigma_i = 0$, then (12) does not impose any constraint. Therefore, we need another constraint in this model to reserve the current allocated flow rate ϕ, which is

$$F_i \geq \phi_i \quad \forall\tau_i \in D. \tag{13}$$

This constraint allocates at least ϕ_i amount of flow for each traffic class τ_i disrespect of its saturation σ_i.

By these modifications in MaxFlow, we obtain the Fairness model to equally grow the normalized flow rates of unsaturated class, which are specified by set $\Sigma = \{\sigma_1, \ldots, \sigma_{|D|}\}$, whose current allocated flow rates and weights are given by $\Phi = \{\phi_1, \ldots, \phi_{|D|}\}$ and $W = \{w_1, \ldots, w_{|D|}\}$, respectively.

Model:	Fairness$(G, C, D, \Sigma, W, \Phi)$
Objective:	(11)
Subject to:	(2), (3), (5), (13), and (12).

The second required optimization model to implement the progressive filling algorithm finds unsaturated classes in each iteration. A traffic class τ_i is not saturated if we can allocate a flow rate F_i for it that is greater than its current flow rate ϕ_i while reserving other classes' allocated rates. In other words, if the following constraint where $\psi_i = 1$ is satisfied, then τ_i is not saturated.

$$F_i > \phi_i\psi_i \quad \forall\tau_i \in D. \tag{14}$$

Therefore, to examine saturation of a set of traffic classes which are specified by $\Psi = \{\psi_1, \ldots, \psi_{|D|}\}$, we need to investigate existence of a feasible solution for the following optimization model. If the Saturation model has a feasible solution for a given set Ψ, then none of traffic classes τ_i whose corresponding $\psi_i = 1$ is saturated.

Model:	Saturation(G, C, D, Φ, Ψ)
Objective:	No Objective Function
Subject to:	(2), (3), (5), (14), and (13).

The SATURATION and FAIRNESS models are solved iteratively in the WEIGHTEDMAXMINFAIR algorithm to obtain the weighted max-min fair rate allocation. Pseudo-code of the algorithm is shown in Algorithm 4. This algorithm at the beginning initializes Φ and Σ with proper values. Then, until there is an unsaturated class, in each iteration, grows rates of unsaturated classes in lines 4–9. New unsaturated classes are found in lines 10–19. Where, we create a temporary set Σ', then examine saturation of each traffic class using a proper set Ψ, and finally update Σ according to new saturated classes.

Algorithm 4 : WEIGHTEDMAXMINFAIR(G, C, D, W)

1: $\Sigma \leftarrow \{\sigma_i = 1, \forall \tau_i \in D\}$
2: $\Phi \leftarrow \{\phi_i = 0, \forall \tau_i \in D\}$
3: **while** $\exists \sigma_i \in \Sigma$ **s.t.** $\sigma_i = 1$ **do**
4: $\quad \beta \leftarrow$ FAIRNESS$(G, C, D, \Sigma, W, \Phi)$
5: $\quad$ **for** $\forall \tau_i \in D$ **do**
6: $\quad\quad$ **if** $\sigma_i = 1$ **then**
7: $\quad\quad\quad \phi_i \leftarrow \beta w_i$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad \Sigma' \leftarrow \{\sigma'_i = 0, \forall \tau_i \in D\}$
11: $\quad$ **for** $\forall \tau_i \in D$ **do**
12: $\quad\quad$ **if** $\sigma_i = 1$ **then**
13: $\quad\quad\quad \Psi = \{\psi_j |$ **if** $j = i$ **then** $\psi_j = 1$, **else** $\psi_j = 0\}$
14: $\quad\quad\quad$ **if** SATURATION(G, C, D, Φ, Ψ) has a feasible solution **then**
15: $\quad\quad\quad\quad \sigma'_i \leftarrow 1$
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: $\quad \Sigma \leftarrow \Sigma'$
20: **end while**
21: **return** Φ

7. Computational complexity analysis

In this section, we analyze the computational complexity of the different approaches discussed in this paper. All proposed algorithms (STATESPACE, GREEDIESTONLINE, WMMFESTIMATE, and WEIGHTEDMAXMINFAIR) need to solve at least a linear programming (LP) problem; therefore, the computational complexity of solving LP is a major factor in the running time of the algorithms. Karmarkar's algorithm solves LP problems in polynomial running time $O(n^{3.5}L^2)$, where n is the number of variables and L is the number of bits required to encode the coefficient in binary format [31]. The LP problems which are solved in these algorithms are instances of MAXFLOW or its variations. There are $|D||E|$ variables in the MAXFLOW model, which are $f^i_{(u,v)}$ $\forall \tau_i \in D$, $\forall (u, v) \in E$. Hence, its complexity is $O\big((|D||E|)^{3.5}L^2\big)$, where L is a fixed value depends on computing machine architecture.

In the following subsections, we first formally analyze the *worst case* computational complexity of each method; then discuss about average case complexity

of the algorithms and their comparison.

7.1. Complexity of multidimensional Markov chain

The Markov chain that is constructed by the StateSpace algorithm is a $|D|$-dimensional chain. In ith dimension, there are at most $N_{i,\boldsymbol{n}_0} = \lfloor F^*_{i,\boldsymbol{n}_0}/b_i \rfloor$ states. Therefore, the computational complexity of enumerating the chain, which is needed to compute $p(\boldsymbol{n}_0)$ in (7), is $O\Big(\prod_{1\le i\le |D|} N_{i,\boldsymbol{n}_0}\Big)$. Moreover, the computational complexity of StateSpace to construct the chain is $O\Big(\prod_{1\le i<|D|}\big(N_{i,\boldsymbol{n}_0} O(\textsc{MaxFlow})\big)\Big) = O\Big(\prod_{1\le i<|D|}\big(N_{i,\boldsymbol{n}_0}(|D||E|)^{3.5}L^2\big)\Big)$. In summary, the complexity of the multi-dimensional Markov chain based approach is

$$O\Big(\prod_{1\le i\le |D|} N_{i,\boldsymbol{n}_0} + \prod_{1\le i<|D|}\big(N_{i,\boldsymbol{n}_0}(|D||E|)^{3.5}L^2\big)\Big).$$

This complexity grows exponentially with the number of traffic classes and network capacity; as mentioned, in fact, it is #P-Complete. Therefore, this technique is applicable only in the case of small $|D|$ in low capacity networks, where $N_{i,\boldsymbol{n}_0}$ is very small.

7.2. Complexity of simulation based estimation

The GreediestOnline algorithm estimates the acceptance rate by simulating arrival of $|\Delta|$ demands. For each demand, its corresponding traffic class is found by TrafficClass with complexity $O(|D|)$, set B is updated whose maximum size is $O(|\Delta|)$, GetN finds the current number of flows per class which can be implemented with complexity $O(|D|)$, and an instance of MaxFlow is solved whose complexity is $O\big((|D||E|)^{3.5}L^2\big)$. Putting these complexities altogether yields that the computational complexity of simulation based method is

$$O\Big(|\Delta|\big(|D| + |\Delta| + (|D||E|)^{3.5}L^2\big)\Big) = O\big(|\Delta|^2 + |\Delta|(|D||E|)^{3.5}L^2\big).$$

7.3. Complexity of weighted max-min fairness based estimation

The complexity of this estimation technique is the running time of WMMFestimate which is $O(\textsc{WeightedMaxMinFair})+|D|O(\textsc{MaxFlow})+|D|N_{i,\boldsymbol{n}_0}$. The worst case complexity of WeightedMaxMinFair is follows. At each iteration of this algorithm, at least one traffic class is saturated; hence, the main loop of the algorithm runs at most $|D|$ times. In each iteration, an instance of Fairness is solved and saturation of unsaturated classes, whose maximum number is $|D|$, is examined through solving Saturation. In summary, the complexity of WightedMaxMinFair is $O\Big(|D|\big(O(\textsc{Fairness})+|D|O(\textsc{Saturation})+|D|\big)\Big) = O\Big(|D|\big(O(\textsc{MaxFlow})+|D|O(\textsc{MaxFlow})\big)\Big) = O\big(|D|^2 O(\textsc{MaxFlow})\big)$. Therefore, the computational complexity of the weighted max-min fairness based estimation is

$$O(\text{WMMFestimate}) = |D|^2 O(\textsc{MaxFlow}) + |D|O(\textsc{MaxFlow}) + |D|N_{i,\boldsymbol{n}_0}.$$

7.4. Average cases complexities

In this section, we provide an insight into the average case complexity of the simulation based and weighted max-min fairness based estimation techniques.[2] In the GreediestOnline algorithm, in practice, the number of active flows, $|B|$, is much less than the total number of demands, $|\Delta|$, because simulation is performed for a very large number of demands while each demand has a limited holding time. Therefore, the complexity of the simulation based method is $O\big(|\Delta|(|B| + (|D||E|)^{3.5}L^2)\big) = O\big(|\Delta|(|D||E|)^{3.5}L^2\big)$ that is the total computational complexity to solve $|\Delta|$ LP problems with $|D||E|$ variables.

In WMMFestimate, practically $|D|N_{i,\boldsymbol{n}_0} \ll |D|^2(|D||E|)^{3.5}L^2$; hence, we can ignore $|D|N_{i,\boldsymbol{n}_0}$. In the worst case analysis, we assumed that only one traffic class is saturated in each iteration of WeightedMaxMinFair; however, in practice, multiple classes may get saturated in an iteration; thus, the main loop in the algorithm runs less than $|D|$ times. For the sake of presentation of WeightedMaxMinFair, we examine saturation of traffic classes one-by-one in lines 11–18; however, it is possible to examine saturation of multiple classes at the same time or even examine saturation in a binary search fashion that reduces the number instances of Saturation needed to be solved. Moreover, we don't need to examine saturation of $|D|$ traffic classes in each iteration since some classes have been saturated in previous iterations. Therefore, the average case complexity of weighted max-min fairness based method is $O\big(|D|^{\epsilon}(|D||E|)^{3.5}L^2\big)$ where $0 < \epsilon \ll 2$. This is the total computational complexity of solving $|D|^{\epsilon}$ LP problems with $|D||E|$ variables.

Comparison of the average case complexities shows that WMMFestimate reduces the complexity of estimating network performance by $|\Delta|/|D|^{\epsilon}$ times that is significant speed up. Simulation results on the average case practical complexity of the algorithms are presented in Section 8.6, which also show this speed up.

8. Simulation results

In this section, we present the simulation results to evaluate the efficiency of the proposed techniques. All these methods (the Markov chain, GreediestOnline, and WMMFestimate) attempt to compute or estimate the network performance bound; therefore, *ideally*, we expect that they will provide the same result. However, in *practice*, due to the following reasons, there will be difference between estimations obtained from these approaches.

- As mentioned, accuracy of GreediestOnline depends on the number of simulated demands. Since, we cannot simulate it for a huge number of demands in large networks, there will be some errors in the bound obtained by GreediestOnline.
- Theorem 2 shows that the difference between Max-Min fair flow rates and actual allocated bandwidth in the network depends on the required bandwidth, b. Since $b \gg 0$ in our simulations, the actual bandwidth

[2] We don't consider multidimensional Markov chain here because its complexity is exponential.

Table 3: Topologies used in the simulations

Name	Placement	Area (m^2)	Node #	Link #
Grid	Grid	400×400	9	24
Rand-15	Random	600×600	15	72
Rand-25	Random	750×750	25	126
Rand-100	Random	1000×1000	100	656

allocation is not exactly max-min fair that leads to some errors in bound obtained by WMMFestimate.

In the following, we first compare the GreediestOnline algorithm to the Markov chain based analysis to measure the accuracy of the algorithm. Then, GreediestOnline, WMMFestimate and the multi-class Erlang-B based technique proposed in [2], which is named MCEB, are compared under various traffic patterns in different topologies. Then, through pathological examples, we explain why MCEB leads to a loose underestimate or overestimate of acceptance probability. Finally, results on the practical complexity of the algorithms are presented.

8.1. Simulation setup

The simulations were carried out in four topologies, which are shown in Table 3. In these topologies, there are two directional links (u, v) and (v, u) if the Euclidean distance between u and v is less than 200 m. The physical capacity of all the links is 100 units. The GreediestOnline algorithm was implemented in a flow-level event driven simulator developed in Java. The ZIB Optimization Suite [32] and CLP [33] were used to solve the optimization problems. Simulations were performed on an Intel Pentium IV 3.2 GHz machine with 2 Gigabytes RAM. In each experiment, a random set D of traffic classes and its corresponding demand set Δ were used. By random set D, we mean that all parameters of traffic classes were chosen randomly. For a given set D, to control the network load and find the performance bound versus it, arrival rate of all traffic classes was *equally* scaled. In the following results, we present acceptance probability versus the scale.

8.2. Accuracy of the simulation-based estimation

In this section, in order to measure the accuracy of the simulation-based technique, we compare the estimation of the acceptance probability obtained from the GreediestOnline algorithm to the probability computed by (8). The simulation results are shown in Table 4. The parameters of the traffic classes used in these simulations are shown in Appendix B, Table B.10. In these simulations, the maximum and average estimation errors are, respectively, 0.7% and 0.3% that shows GreediestOnline estimates the network performance bound accurately as it proved in Theorem 1. These simulations did not run in Rand-100 network and very few traffic classes were used since it is impractical to compose the Markov chain in large networks for a large number of traffic classes. As mentioned in Section 7.1, the number states grows exponentially by increasing the number traffic classes and network capacity, which is proportional to the network size. For example, in the Rand-15 and Rand-25 topologies,

Table 4: The acceptance probabilities obtained by the Markov chain based analysis and the simulation-based technique versus the scaled arrival rate

Topology	Scale	Markov	Simulation	Error %
Grid	1.0	0.992	0.991	0.1
	2.0	0.681	0.676	0.5
	3.0	0.466	0.461	0.5
	4.0	0.352	0.354	0.2
	5.0	0.282	0.278	0.4
Rand-15	1.0	0.954	0.951	0.3
	2.0	0.646	0.647	0.1
	3.0	0.475	0.473	0.2
	4.0	0.383	0.382	0.1
	5.0	0.323	0.321	0.2
Rand-25	0.8	0.999	1.000	0.1
	1.6	0.984	0.984	0.0
	2.4	0.877	0.884	0.7
	3.2	0.768	0.775	0.7
	4.0	0.690	0.687	0.3

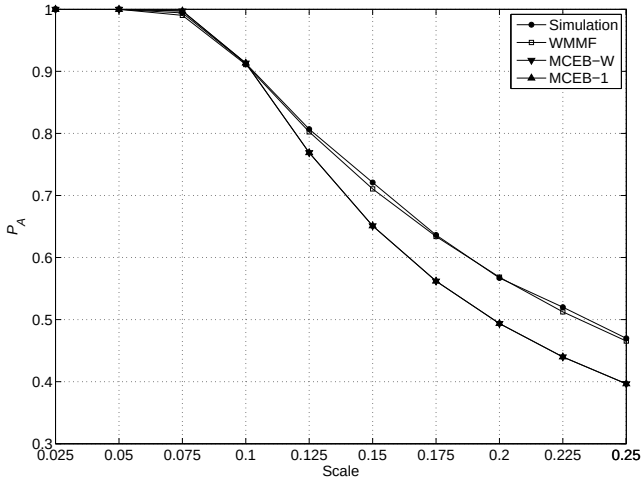
the number of states of the Markov chain is about 2,721,700 and 3,943,200, respectively.

Because of the accurate estimation of acceptance probability by GreediestOnline and the limitations of using the Markov chain, in the following sections, we use the GreediestOnline algorithm as the reference to measure the precision of the estimations provided by WMMFestimate and MCEB.

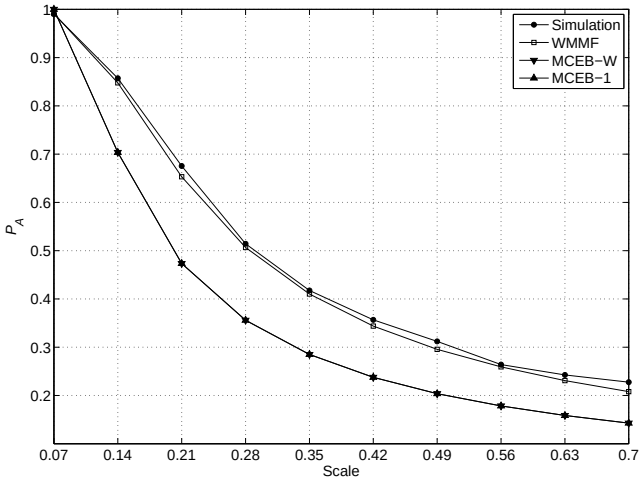
8.3. Homogeneous traffic

In this section, we consider homogeneous traffic, in which all traffic classes have the same arrival rate λ, average holding time μ^{-1}, and required bandwidth b. The traffic classes used for these simulations are shown in Table B.11. We evaluate the precision of the weighted max-min fairness based estimation by comparing it to the results obtained from the GreediestOnline algorithm. Moreover, the MCEB method was also simulated. MCEB considers flow f_i for each traffic class, then finds the maximum α such that simultaneously allocating bandwidth αf_i for every traffic class is feasible. The authors in [2] do not clarify how f_i is assigned. In these simulations, we use two schemes to assign f_i. The first one, which is named MCEB-1, considers the same $f_i = 1$ for all $\tau_i \in D$. The other, which is named MCEB-W, assigns $f_i = \lambda_i \mu_i^{-1} b_i$ for $\tau_i \in D$.

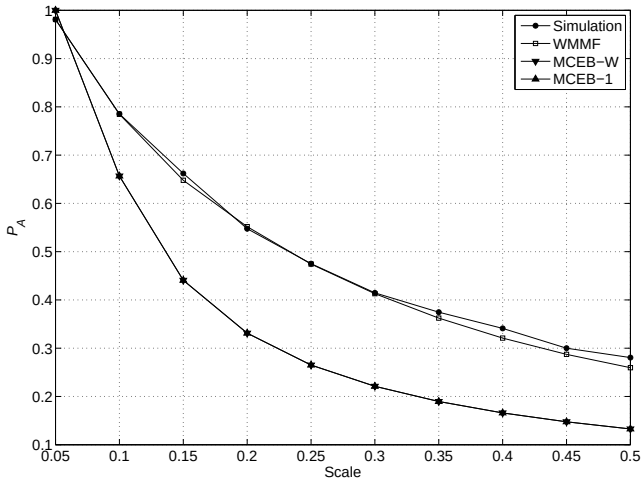
Acceptance probability versus the scale of arrival rate is depicted in Fig. 2. In these figures, "Simulation" and "WMMF" are, respectively, the acceptance probabilities obtained from GreediestOnline and WMMFestimate. Since traffic is homogeneous in these simulations, MCEB-1 and MCEB-W both yield to the same results. The maximum and average estimation errors, the difference between "Simulation" and "WMMF," are shown in Table 5. These results show that WMMFestimate provides a quite accurate estimation of acceptance probability regardless of the network topology, the number of traffic classes, and the scale of arrival rate. However, the MCEB method underestimates network performance bound which is due to incorrect estimation of network capacity by this method. We discuss about this problem in more detail in Section 8.5.



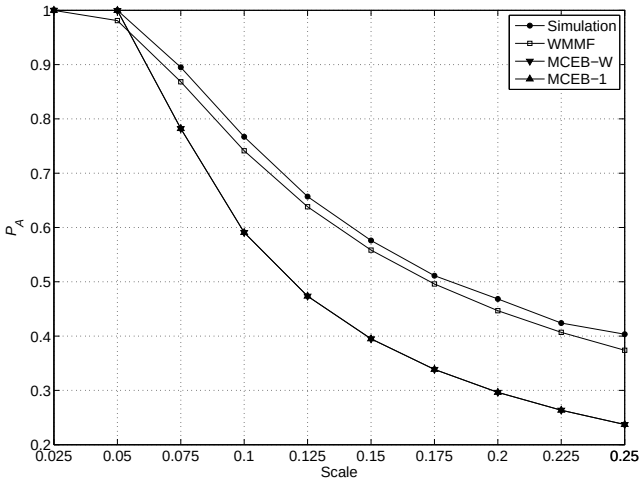
(a) Grid Topology



(b) Rand-15 Topology



(c) Rand-25 Topology



(d) Rand-100 Topology

Figure 2: Acceptance probability versus the scale of arrival rate under homogeneous traffic with the parameters shown in Table B.11

Table 5: The average and maximum estimation errors of the max-min fairness based technique in comparison to the simulation-based method corresponding to Fig. 2

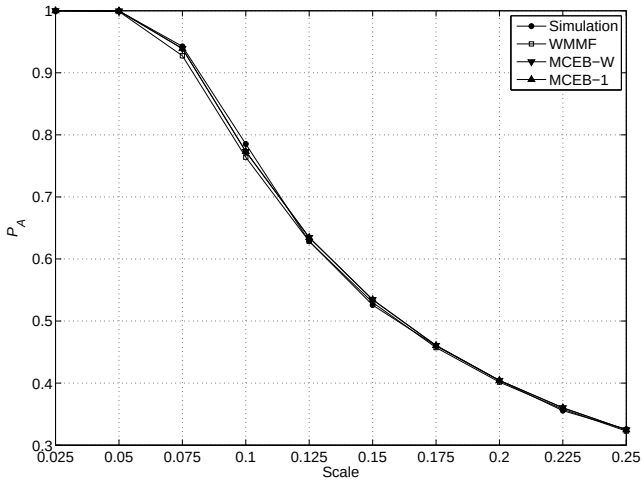
Topology	Average Err. %	Max. Err. %
Grid	1.27	2.38
Rand-15	1.07	2.34
Rand-25	0.62	1.21
Rand-100	1.91	2.92

8.4. Heterogeneous traffic

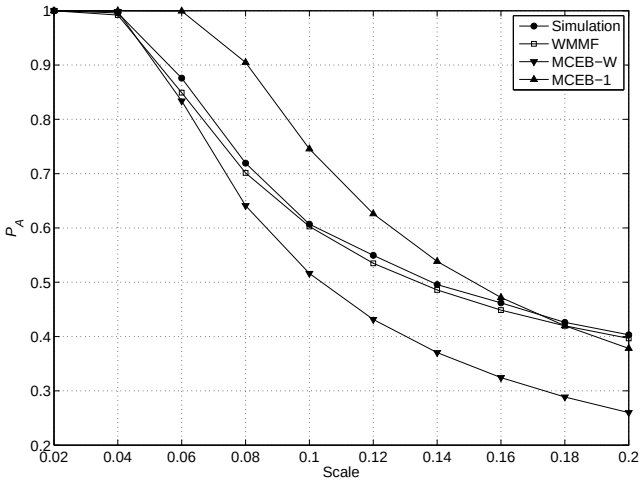
In this section, we evaluate the algorithms under heterogeneous traffic that implies that each traffic class τ_i has its own arrival rate λ_i and average holding time μ_i^{-1}. However, all the classes request the same bandwidth b. The parameters of the traffic classes used for these simulations are shown in Table B.12 and Table B.13. In these tables, $F^*_{\boldsymbol{n}_0}$ and z are the maximum flow and weighted max-min flow of traffic classes, respectively. The number of traffic classes used in these simulations is different from the numbers used in the previous section in order to see the effect of the number of traffic classes per topology.

The acceptance probabilities obtained by GreediestOnline, WMMFestimate and MCEB versus the scale of demand arrival rate are shown in Fig. 3. The maximum and average estimation errors of the weighted max-min fairness based technique are shown in Table 6. These results show that similar to the homogeneous traffic condition, the gap between WMMFestimate and GreediestOnline is quite small in all the network topologies with different numbers of traffic classes, and in a wide range of offered load.

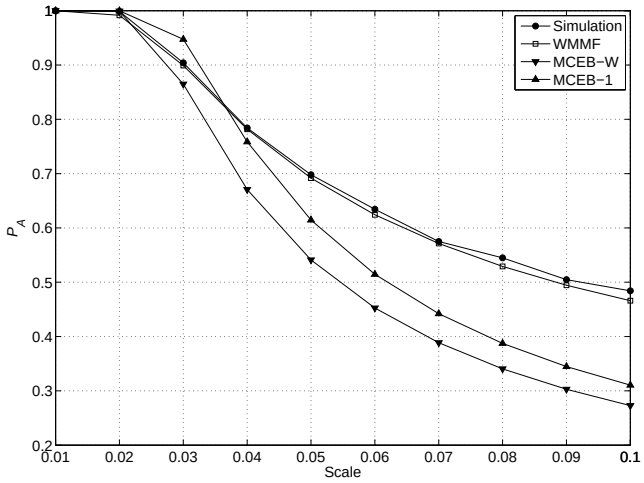
Except the Grid topology, MCEB provides a very loose bound of network performance. Interestingly, in the Grid topology, this method exactly estimates the acceptance probability. This is due to the location of the source-destination pairs in this simulation scenario. We explicate in Section 8.5 that MCEB assumes all traffic classes completely share the same bottleneck resource (the same min-cut in the network). This is the case in this scenario. Traffic classes $\tau_1 = (8, 1, 5, 18, 6)$, $\tau_2 = (5, 3, 20, 11, 6)$, and $\tau_3 = (2, 6, 10, 30, 6)$ were used in this simulation, which are shown in Fig. B.7. These classes share the same min-cut that is composed of links (2,1), (5,4), and (8,7) as shown in the figure. Since the complete resource sharing assumption holds in this scenario, it estimates acceptance probability exactly.



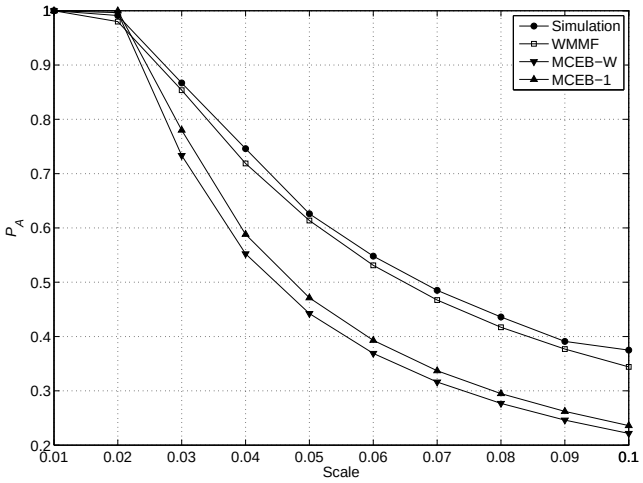
(a) Grid Topology



(b) Rand-15 Topology



(c) Rand-25 Topology



(d) Rand-100 Topology

Figure 3: Acceptance probability versus the scale of arrival rate under heterogeneous traffic with the parameters shown in Tables B.12 and B.13

Table 6: The average and maximum estimation errors of WMMFestimate in comparison to GreediestOnline in Fig. 3

Topology	Average Err. %	Max. Err. %
Grid	0.57	2.13
Rand-15	1.02	1.88
Rand-25	0.78	1.65
Rand-100	1.62	3.08

8.5. Pathological examples

Simulation results in the previous sections shows that the MCEB technique yields to a loose estimation of network performance. In this section, through two simple pathological examples, the shortcomings of the method are clarified.

A simple network topology is shown in Fig. 4. In this network, there are two traffic classes: $\tau_1 = (u_1, u_2, 1, 1, 1)$ and $\tau_2 = (u_2, u_3, 1, 1, 1)$. In the first case, we assume that both links have the same physical capacity, $c_{(u_1,u_2)} = c_{(u_2,u_3)} = 5$. Acceptance probabilities obtained by the multidimensional Markov chain, WMMFestimate, and MCEB, are depicted in Fig. 5. Note that since traffic is homogeneous, both MCEB-1 and MCEB-W provide the same results. As this figure shows, MCEB overestimates acceptance probability in this case. This is due to the sharing policy assumed in the method. MCEB models the whole links in the network as a single loss system that means it assumes *complete sharing* among all traffic classes. However, in this example, in fact, there is not any resource sharing among the traffic classes. This sharp contradiction between that assumption and this actual traffic pattern leads to the large gap between MCEB and Markov in this figure. The weighted max-min fairness based technique does not suffer the drawback since it considers a loss system *per traffic* class, and takes into account the resource sharing among classes using the *sharing factor* γ_i.

Now, let us consider a heterogeneous network. Assume that in Fig. 4, we have $c_{(u_1,u_2)} = 1$ and $c_{(u_2,u_3)} = 10$. Acceptance probabilities for the traffic classes τ_1 and τ_2 in the heterogeneous network are depicted in Fig. 6. The figure shows that MCEB gives a very loose underestimate of acceptance probability in this scenario. The reason is the lower bound on the capacity of the minimum multi-commodity cut which is used by MCEB as the capacity in the Erlang-B formula. As mentioned, MCEB considers flow f_i for each traffic class, and finds the maximum α such that simultaneously allocating bandwidth αf_i for every traffic class is feasible. Then, it considers $\sum_{\tau_i \in D} \alpha f_i$ as the capacity of the multi-class loss system and uses Erlang-B formula to find rejection probability. Indeed, this approach considers the *absolute fairness* strategy. The problem is that absolute fairness does not fully utilize network resources. For example in this figure, assuming $f_1 = f_2 = 1$, we have $\alpha = 1$, and hence, $\alpha f_2 = 1$ units of bandwidth is considered for τ_2 in spite of the fact that its maximum possible flow allocation is $c_{(u_2,u_3)} = 10$. Underutilization of network resources causes the underestimate of acceptance probability by MCEB. The weighted max-min fairness approach does not have this shortcoming because instead of absolute fairness, it considers max-min fairness, which leads to complete utilization of network resources.

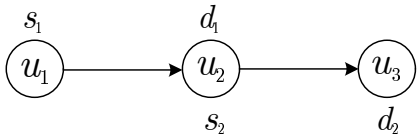


Figure 4: A simple topology to illustrate the drawbacks of the MCEB technique

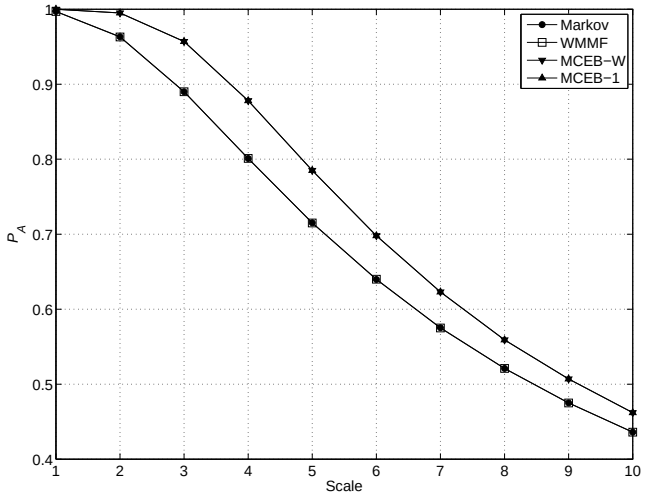


Figure 5: Acceptance probability versus the scale of arrival rate in the topology depicted in Fig. 4 with capacities $c_{(u_1,u_2)} = c_{(u_1,u_2)} = 5$

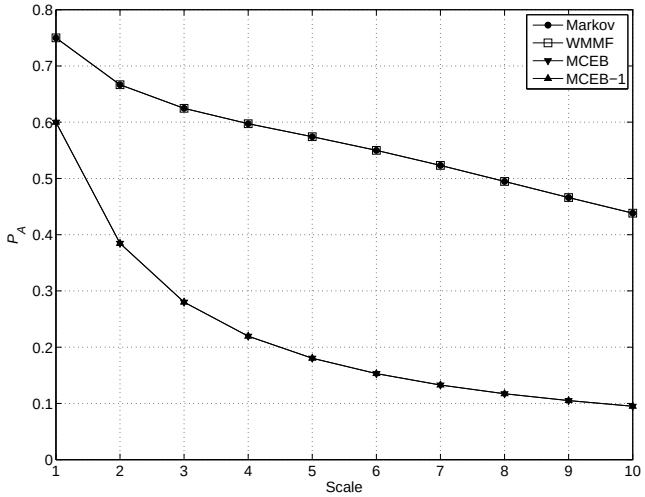


Figure 6: Acceptance probability versus the scale of arrival rate in the topology depicted in Fig. 4 with capacities $c_{(u_1,u_2)} = 1$ and $c_{(u_1,u_2)} = 10$

8.6. Practical computational complexity

In Section 7, we analyzed the worst case complexity of the algorithms. In this section, simulation results on the practical average case complexity of the methods are presented.

We first consider the average case running time of the GreediestOnline algorithm. This algorithm is developed in Java. Upon arrival of a demand, the Java code initializes the parameters of the MaxFlow model. An instance of MaxFlow using the parameters is created by `zimpl` [32], which is solved by `clp` [33]. Table 7 shows the average running time of `clp` to solve the instances of MaxFlow in each simulated scenarios. As mentioned in Section 7, the computational complexity to solve an instance of MaxFlow is $O\big((|D||E|)^{3.5}L^2\big)$ which is also justified by the results shown in Table 7. In this table, `clp` running time grows by increasing the number of traffic classes and network size. The average total running time (the sum of running time of Java code, `zimpl`, and `clp`) per demand is shown in Table 8. The actual running time of the simulation based method can be estimated using this table. For example, in the Rand-100 topology with 50 traffic classes, if simulation is performed for 5000 demands, it takes about 133 hours (5 days and 13 hours). These results show that using the GreediestOnline algorithm to estimate the network performance bound is impractical especially in large networks.[3]

The average case computational complexity of WMMFestimate is discussed in Section 7.4, where we mentioned that it is proportional to the number LP problems need to be solved in this algorithm. Table 9 shows these numbers and the running time of WMMFestimate in the simulated scenarios. In this table, columns Fairness, Saturation, and MaxFlow show the number of solved instances of their corresponding optimization models. The last column shows the running time of the algorithm. It must be noted that the average time to solve LP problems in WMMFestimate is less than the average time reported in Table 7; therefore, this algorithm estimate the network performance bound efficiently. If WMMFestimate and GreediestOnline are compared according to the number of required LP problems should be solved in these algorithms, Table 9 shows that WMMFestimate is significantly more efficient than GreediestOnline; e.g., if simulation is performed in Rand-100 with $|\Delta| = 5000$ for 10 scales, then 50000 LP problems are solved; while WMMFestimate needs solution of about 300 problems that implies about 160 times speed up.

Finally, it should be note that the MCEB method always solves one LP problem, which is an instance of the Fairness model. Therefore, its running time is much less than WMMFestimate; however, as discussed, its estimation of network performance may be quite inaccurate.

9. Conclusions and future work

In this paper, we have studied the problem of performance evaluation of QoS-aware data networks. In this problem, the network performance is measured in terms of acceptance probability of traffic demands requiring a fixed amount of

[3] In fact, our simulations in the Rand-100 network for different load scales took more than six weeks.

Table 7: Average running time to solve an instance of MaxFlow in the GreediestOnline algorithm

Topology	$\|D\|$	Traffic	Time (sec)
Grid	3	heterogeneous	0.0366
Grid	5	homogeneous	0.0393
Rand-15	5	heterogeneous	0.2618
Rand-15	15	homogeneous	0.9972
Rand-25	15	heterogeneous	1.0762
Rand-25	25	homogeneous	2.2531
Rand-100	50	heterogeneous	76.6010
Rand-100	50	homogeneous	74.2758

Table 8: Average running time of the GreediestOnline algorithm per demand

Topology	$\|D\|$	Traffic	Time (sec)
Grid	3	heterogeneous	0.4937
Rand-15	5	heterogeneous	0.93666
Rand-25	15	heterogeneous	1.5334
Rand-100	50	heterogeneous	95.7328

Table 9: The number of LP problems solved in WMMFestimate

Scenario		WeightedMaxMinFair		MaxFlow	Total	Runtime
Topology	$\|D\|$	Fairness	Saturation			(sec)
Grid	5	2	6	5	13	1.52
Grid	3	1	1	3	5	1.04
Rand-15	15	4	28	15	47	68.23
Rand-15	5	3	8	5	16	5.38
Rand-25	25	7	59	25	91	230.11
Rand-25	15	8	46	15	69	25.06
Rand-100	50	14	180	50	244	2321.41
Rand-100	50	15	240	50	305	2994.11

bandwidth. We obtained an upper bound on the maximum acceptance probability achievable through online bandwidth constrained routing algorithms under the greedy CAC policy. The bound is exactly computed by modeling the network stats as a multidimensional Markov chain. To tackle the computational complexity of composing and enumerating the Markov chain, we developed the greediest online algorithm to estimate the bound by simulation. It was shown that under single rate traffic load, the resource allocation by this algorithm is approximately weighted max-min fair. Based on this observation, a novel technique was developed to estimate the performance bound without enumerating the Markov chain or performing simulations. Extensive simulations showed that the proposed technique accurately estimates the bound under a wide range of traffic parameter settings in different networks.

In this paper, the weighted max-min fairness based estimation technique is developed for single-rate traffic, extending this method for multi-rate traffic is an open research problem. Multi-rate traffic can be modeled by multi-class Erlang-B formula, as we modeled single-rate traffic using the single-class Erlang-B formula; however, it yields to a multidimensional Markov chain which is computationally intensive. Moreover, estimating the performance of QoS routing algorithms in multi-hop wireless networks, where inference complicates the problem, in another future research direction.

Appendix A. Proofs

Appendix A.1. Proof of Theorem 1

Proof. Assume that at time t, $n_i(t)$ flows from traffic class τ_i are active in the network, which have been accepted by the greediest online algorithm. Hence, $\boldsymbol{n}(t) = \big(n_1(t), \ldots, n_{|D|}(t)\big)$ is the state of the network visited by the algorithm at time t. Assume $t_j < t_{j+1}$, and let

$$\Pr\Big(X(t_j) = \boldsymbol{n}(t_j) \;\Big|\; X(t_{j-1}) = \boldsymbol{n}(t_{j-1}), \ldots, X(t_1) = \boldsymbol{n}(t_1)\Big)$$

be the probability of being at state $\boldsymbol{n}(t_j)$ if the greediest online algorithm has visited states $\boldsymbol{n}(1)$, $\boldsymbol{n}(2)$, $\ldots$, and $\boldsymbol{n}(t_{j-1})$, sequentially.

Without loss of generality, assume that state $\boldsymbol{n}(t_{j-1})$ is visited by the algorithm and no demand leaves the network before t_j. At time t_j, a new demand from traffic class τ_i arrives. The algorithm checks the feasibility of state $\boldsymbol{n}(t_{j-1}) + e_i$, and accepts the demand if the state is feasible that means state $\boldsymbol{n}(t_j) = \boldsymbol{n}(t_{j-1}) + e_i$ is visited, $X(t_j) = \boldsymbol{n}(t_j)$. Assuming that demand arrival is Poisson and holding time is exponentially distributed, visiting state $\boldsymbol{n}(t_j)$ does not depend on the states visited before t_{j-1} since the algorithm can reroute all the existing flows; in other words, the route selections have been performed in states $\boldsymbol{n}(t_{j'})$ $1 \leq j' \leq j-1$ do not influence acceptance of the new demand. Therefore, we have

$$\begin{aligned}\Pr\Big(X(t_j) = \boldsymbol{n}(t_j) \;\Big|\; X(t_{j-1}) = \boldsymbol{n}(t_{j-1}), \ldots, X(t_1) = \boldsymbol{n}(t_1)\Big)& \\ = \Pr\Big(X(t_j) = \boldsymbol{n}(t_j) \;\Big|\; X(t_{j-1}) = \boldsymbol{n}(t_{j-1})\Big),&\end{aligned}$$

that shows the Markovian property of the stochastic process $X(t)$.

For the second part of the theorem, we use the following observations. First, transitions between the states visited by the algorithm are triggered when a demand arrives or leaves the network. Therefore, assuming states $\boldsymbol{n'}$ and $\boldsymbol{n} = \boldsymbol{n'} + e_i$ are feasible, the transition rate from $\boldsymbol{n'}$ to $\boldsymbol{n}$ is λ_i, and from $\boldsymbol{n}$ to $\boldsymbol{n'}$ is $n_i\mu_i$. Note that these are the transition rates of the multidimensional Markov chain as discussed in Section 4. Second, the greediest online algorithm uses the MaxFlow model to accept demands. This model is also used in the StateSpace algorithm to compose the state space of the Markov chain. Therefore, if state $\boldsymbol{n}$ is visited by the algorithm, it implies that the state is feasible; in other words, $\boldsymbol{n} \in \Omega$. Hence, the state space of $X(t)$ is equal to Ω, and since the transition rates are the same, the stochastic process $X(t)$ is the multidimensional Markov chain. □

Appendix A.2. Proof of Theorem 2

The set $\{\delta_{1+i|D|}, \delta_{2+i|D|}, \ldots, \delta_{|D|+i|D|}\} \subset \Delta$ of successive demands is called ith round of the greediest online algorithm. By $\boldsymbol{n} = \boldsymbol{n'} + \mathbf{1}$, we mean $\exists D' \subseteq D$ s.t. $n_i = n'_i + 1\ \forall \tau_i \in D'$ and $n_j = n'_j\ \forall \tau_j \in D \setminus D'$. We use the following facts about the greediest online algorithm to prove the theorem.

F1. Suppose states $\boldsymbol{n}(1)$, $\boldsymbol{n}(2)$, ... are visited by the algorithm, we have $\boldsymbol{n}(i+1) = \boldsymbol{n}(i) + \mathbf{1}$. Because due to assumption **A3** no demand leaves the network, hence the number of existing flows are always increasing.

F2. Assume that for the first time, a demand of traffic class τ_i is rejected in jth round. No demand will be accepted from the class in the future rounds. Because due to the greedy CAC policy, the rejection in jth round implies that sufficient resources are not available for the class at this round, and since no demand leaves the network, assumption **A3**, no resource will be freed for the class in the future rounds. In this case, round j is called the *blocking* round of class τ_i.

F3. If traffic class τ_i is blocked in jth round, then $y_i = (j-1)\epsilon$ and vice versa. Because due to assumptions **A1** and **A4**, at each round at most ϵ units of bandwidth is allocated for each traffic class, and according to fact **F2** no bandwidth is allocated for the class after round $j-1$.

F4. If $\boldsymbol{n}$ is the last state visited by the algorithm, then $\boldsymbol{n} + e_i \notin \Omega\ \forall \tau_i \in D$. Because the CAC policy is greedy, and according to assumption **A5**, there is a sufficiently large number of demands; hence, if a traffic class τ_i exists such that $\boldsymbol{n'} = \boldsymbol{n} + e_i \in \Omega$, then there is a demand from the class that triggers visiting the state; hence, the algorithm visits it, and $\boldsymbol{n}$ cannot be the last state. This fact implies that at the end of simulation, the network is completely saturated and all resources are fully utilized. Moreover, it implies that each traffic class τ_i is blocked in (n_i+1)th round.

F5. Let $\boldsymbol{N}(1) = \{\boldsymbol{n} \in \Omega \text{ s.t. } n_i = k_i, n_j = k_j\}$, $\boldsymbol{N}(2) = \{\boldsymbol{n} \in \Omega \text{ s.t. } n_i = k_i + 1, n_j = k_j + 1\}$, and $\boldsymbol{N}(3) = \{\boldsymbol{n} \in \Omega \text{ s.t. } n_i = k_i + 2\}$. Fact **F1** implies that if the algorithm visits a state $\boldsymbol{n}(2) \in \boldsymbol{N}(2)$, then it must also visit a state $\boldsymbol{n}(1) \in \boldsymbol{N}(1)$ before visiting $\boldsymbol{n}(2)$ because $\boldsymbol{n}(2) = \boldsymbol{n}(1) + \mathbf{1}$. Similarly, if the algorithm visits a state $\boldsymbol{n}(3) \in \boldsymbol{N}(3)$, it must visit a state $\boldsymbol{n}(1) \in \boldsymbol{N}(1)$ because $\boldsymbol{n}(3) = (\boldsymbol{n}(1) + \mathbf{1}) + \mathbf{1}$.

By definition [29], bandwidth allocation Z is max-min fair if and only if for any other allocation Z', if a τ_i exists such that $z'_i > z_i$, then there must exist τ_j such that $z'_j < z_j$ and $z_j \leq z_i$.

Proof. We prove the theorem by contradiction. For the first case, assume that $\exists \tau_i \in D$ s.t. $y_i \geq z_i + \epsilon$. Due to max-min fairness of Z, there is $\tau_j \in D$ such that $y_j < z_j$ and $z_j \leq z_i$. Let $y_i = k_i\epsilon$, $y_j = k_j\epsilon$, $z_i = \overline{k}_i\epsilon$, and $z_j = \overline{k}_j\epsilon$. Note that due to fact **F3**, k_i and k_j are integer numbers; in fact, class τ_i and τ_j are blocked in $k_i + 1$ and $k_j + 1$ rounds, respectively.

Without loss of generality, we assume that $y_i = z_i + \epsilon$. Hence, $k_i = \overline{k}_i + 1$. By the max-min fairness of Z, we have $y_j < z_j \leq z_i$. If we assume $z_j = z_i$, then we have $y_j < z_j = z_i = y_i - \epsilon$. Therefore, $k_j \leq k_i - 1$. Without loss of generality, assume $k_j = k_i - 2$. In summary, we have $\overline{k}_i = \overline{k}_j = k_i - 1 = k_j + 1$, and all these numbers are integer.

Let us define $\boldsymbol{N}(1) = \{\boldsymbol{n} \in \Omega \text{ s.t. } n_i = k_j, n_j = k_j\}$. $\boldsymbol{N}(2) = \{\boldsymbol{n} \in \Omega \text{ s.t. } n_i = \overline{k}_i = k_j + 1, n_j = \overline{k}_j = k_j + 1\}$, and $\boldsymbol{N}(3) = \{\boldsymbol{n} \in \Omega \text{ s.t. } n_i = k_i = k_j + 2, n_j = k_j\}$. Since bandwidth allocation Y is given by the greediest online algorithm, the algorithm visits a state $\boldsymbol{n}(3) \in \boldsymbol{N}(3)$. Fact **F5** implies that it must visit state $\boldsymbol{n}(1) \in \boldsymbol{N}(1)$ before $\boldsymbol{n}(3)$. Suppose that state $\boldsymbol{n}(1)$ is visited at time t, consider a round starting from this time. In this round there is a demand from τ_i and a demand from τ_j because of the assumption **A4**. Both demands are accepted by the algorithm because $\boldsymbol{n}(2) = (\frac{z_1}{\epsilon}, \ldots, n_i = \overline{k}_i = k_j + 1, \ldots, n_j = \overline{k}_j = k_j + 1, \ldots, \frac{z_{|D|}}{\epsilon}) \in \boldsymbol{N}(2)$ is feasible due to the feasibility of bandwidth allocation Z. However according to **F1**, visiting $\boldsymbol{n}(2)$ implies that the algorithm does not visit $\boldsymbol{n}(3)$ because $\boldsymbol{n}(3) \neq \boldsymbol{n}(2) + \boldsymbol{1}$. Hence, bandwidth allocation Y cannot be produced by the algorithm.

For the second case, assume that $\exists \tau_i$ s.t. $y_i < z_i - \epsilon$. Let $\boldsymbol{n} = (n_1, \ldots, n_{|D|})$ where $n_i = y_i/\epsilon$, and $\overline{\boldsymbol{n}} = (\overline{n}_1, \ldots, \overline{n}_{|D|})$ where $\overline{n}_i = \lfloor z_i/\epsilon \rfloor$. Fact **F3** and **F4** imply that $\boldsymbol{n}$ is the last state visited by the algorithm. If $\nexists \tau_j$ s.t. $y_j \geq z_j + \epsilon$, we can assume $y_j = z_j$ $\forall \tau_j \in D \setminus \tau_i$ without loss of generality. Hence, we have $\overline{\boldsymbol{n}} = \boldsymbol{n} + e_i$. Since Z is a feasible bandwidth allocation, we have $\overline{\boldsymbol{n}} \in \Omega$. However, in this case, fact **F4** implies that $\boldsymbol{n}$ cannot be the last state visited by the algorithm; therefore, bandwidth allocation Y cannot be produced by the algorithm. If $\exists \tau_j$ s.t. $y_j \geq z_j + \epsilon$, due to the first case, we know that Y is not obtained by the greediest online algorithm. □

Appendix B. Simulation Traffic

References

[1] L. Narayanan and Y. Saintillan. Fair and efficient call routing and admission control algorithms. In *IEEE Global Telecommunications Conference*, volume 2, pages 1527–1534, 2002.

[2] Antonio Capone, Luigi Fratta, and Fabio Martignon. Dynamic online QoS routing schemes: Performance and bounds. *Computer Networks*, 50(7):966–981, 2006.

[3] F.P. Kelly. Loss networks. *The annals of applied probability*, pages 319–378, 1991.

Table B.10: Traffic classes used in the simulations in Section 8.2

Topology	λ	μ^{-1}	b
Grid	1.0	3.0	6.0
	2.0	8.0	4.0
	4.0	4.0	8.0
Rand-15	1.0	3.0	6.0
	2.0	8.0	4.0
	4.0	4.0	8.0
	6.0	10.0	5.0
Rand-25	3.1	1.5	15.0
	1.0	0.1	20.0
	1.5	3.0	23.0
	2.0	5.0	11.0
	5.0	1.0	27.0

Table B.11: The parameters of the homogeneous traffic classes used in the simulations in Section 8.3

Topology	λ	μ^{-1}	b	$\lvert D\rvert$
Grid	1	15	5	5
Rand-15	10	10	10	15
Rand-25	10	10	10	25
Rand-100	12	20	10	50

[4] K. Kar, M. Kodialam, and TV Lakshman. Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications. *IEEE Journal on Selected Areas in Communications*, 18(12):2566–2579, 2002.

[5] Subhash Suri, Marcel Waldvogel, Daniel Bauer, and Priyank Ramesh Warkhede. Profile-based routing and traffic engineering. *Computer Communications*, 26(4):351–365, 2003.

[6] Fabio Ricciato and Ugo Monaco. Routing demands with time-varying bandwidth profiles on a MPLS network. *Computer Networks*, 47(1):47–61, 2005.

[7] D. Mitra, J.A. Morrison, and K.G. Ramakrishnan. ATM network design and optimization: A multirate loss network framework. *IEEE/ACM Transactions on Networking*, 4(4):531–543, 1996.

[8] S.-P. Chung and K.W. Ross. Reduced load approximations for multirate loss networks. *IEEE Transactions on Communications*, 41(8):1222–1231, August 1993.

[9] Albert G. Greenberg and R. Srikant. Computational techniques for accurate performance evaluation of multirate, multihop communication networks. *IEEE/ACM Transactions on Networking*, 5:266–277, 1997.

[10] F.P. Kelly. Routing and capacity allocation in networks with trunk reservation. *Mathematics of operations research*, 15(4):771–793, 1990.

[11] D. Mitra, R.J. Gibbens, and B.D. Huang. State-dependent routing on symmetric loss networks with trunk reservations. I. *IEEE Transactions on Communications*, 41(2):400 –411, February 1993.

Table B.12: Heterogeneous traffic classes used in the simulations in Section 8.4

Topology	λ	μ^{-1}	b	$F^*_{\boldsymbol{n}_0}$	z
Grid	5	18	6	200	44.2
	20	11	6	300	108.1
	10	30	6	200	147.5
Rand-15	5	10	5	400	400.0
	31	15	5	300	121.4
	10	1	5	300	200.0
	15	30	5	200	117.5
	20	50	5	300	261.1
Rand-25	6	10	7	700	217.8
	9	13	7	300	282.2
	14	17	7	400	49.7
	19	21	7	200	76.8
	12	13	7	300	300.0
	09	20	7	400	37.6
	16	14	7	200	43.1
	18	16	7	200	100.0
	09	11	7	300	200.0
	18	18	7	300	62.3
	13	27	7	200	67.5
	7	16	7	400	23.4
	16	17	7	400	56.8
	9	29	7	300	50.2
	13	27	7	200	200.0

[12] Shun-Ping Chung, Arik Kashper, and Keith W. Ross. Computing approximate blocking probabilities for large loss networks with state-dependent routing. *IEEE/ACM Transactions on Networking*, 1:105–115, February 1993.

[13] A. Girard. *Routing and dimensioning in circuit-switched networks*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1990.

[14] Mingyan Liu and J.S. Baras. Fixed point approximation for multirate multihop loss networks with state-dependent routing. *IEEE/ACM Transactions on Networking*, 12(2):361 – 374, 2004.

[15] M. Liu and J.S. Baras. Performance analysis using a hierarchical loss network model. In *IEEE Global Telecommunications Conference*, volume 3, pages 1793 –1797, 2000.

[16] Ben-Jye Chang and Ren-Hung Hwang. Performance evaluation model for adaptive hierarchical loss networks. In *IEEE Global Telecommunications Conference*, volume 2, pages 1875 – 1879, 2002.

[17] Ben-Jye Chang and Ren-Hung Hwang. Performance analysis for hierarchical multirate loss networks. *IEEE/ACM Transactions on Networking*, 12:187–199, February 2004.

[18] Eric W.M. Wong, Andrew Zalesky, Zvi Rosberg, and Moshe Zukerman. A new method for approximating blocking probability in overflow loss networks. *Computer Networks*, 51(11):2958 – 2975, 2007.

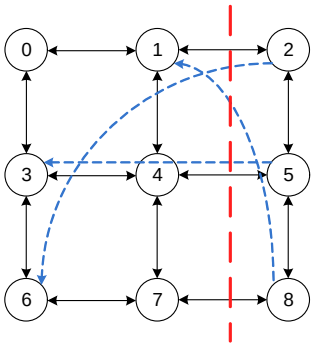


Figure B.7: Traffic classes used in the Grid topology in Section 8.4, and corresponding min-cut

[19] A. Capone, L. Fratta, and F. Martignon. Virtual flow deviation: Dynamic routing of bandwidth guaranteed connections. In *International Workshop on Quality of Service in Multiservice IP Networks*, pages 592–605, 2003.

[20] K. Hendling, G. Franzl, B. Statovci-Halimi, and A. Halimi. IMRA–A Fast and Non-greedy Interference Minimizing On-Line Routing Algorithm for Bandwidth Guaranteed Flows. *High Speed Networks and Multimedia Communications*, pages 336–347, 2004.

[21] A. Capone, L. Fratta, and F. Martignon. Dynamic routing of bandwidth guaranteed connections in MPLS networks. *International Journal of Wireless and Optical Communications*, 1:75–86, 2003.

[22] B. Wang, X. Su, and C.L.P. Chen. A new bandwidth guaranteed routing algorithm for MPLS traffic engineering. In *IEEE International Conference on Communications*, volume 2, pages 1001–1005, 2002.

[23] C.H. Wang and H. Luh. Blocking Probabilities of Multiple Classes in IP Networks with QoS Routing. *Advances in Queueing Theory and Network Applications*, pages 281–290, 2009.

[24] R.A. Guerin, A. Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. In *IEEE GLOBECOM, 1997*, 1997.

[25] Zheng Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, sep. 1996.

[26] K.W. Ross and P.J. Hancock. *Multiservice loss models for broadband telecommunication networks*. Springer, 1995.

[27] J. Kaufman. Blocking in a shared resource environment. *IEEE Transactions on Communications*, 29(10):1474–1481, 2002.

[28] G.M. Louth. *Stochastic networks Complexity, dependence and routing*. PhD thesis, Cambridge University, 1990.

[29] D.P. Bertsekas, R.G. Gallager, and P. Humblet. *Data networks*. Prentice-hall Englewood Cliffs, NJ, 1987.

[30] D. Nace and M. Pióro. Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial. *IEEE Communications Surveys & Tutorials*, 10(4):5–17, 2008.

[31] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *16th ACM symposium on Theory of computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. ACM.

[32] ZIB. http://zibopt.zib.de/ZIBopt.shtml.

[33] CLP. http://www.coin-or.org/Clp/index.html.

Table B.13: Heterogeneous traffic classes used in the simulations in Section 8.4

Topology	λ	μ^{-1}	b	$F^*_{\boldsymbol{n}_0}$	z
Rand-100	12	26	8	400	72.2
	13	22	8	200	66.2
	12	26	8	400	55.4
	15	21	8	200	56
	10	23	8	700	87.9
	12	23	8	500	63.9
	10	24	8	700	55.6
	10	25	8	1200	666.7
	12	23	8	200	61.2
	15	28	8	200	74.7
	14	25	8	100	100
	13	29	8	200	87.3
	13	25	8	600	118.6
	13	24	8	600	55.4
	15	22	8	400	76.4
	10	23	8	200	51
	12	26	8	500	72.2
	15	21	8	400	69.9
	12	26	8	200	200
	10	22	8	300	187.3
	15	27	8	500	71.9
	14	28	8	600	69.6
	11	23	8	400	44.9
	15	23	8	600	79.9
	15	25	8	500	86.8
	10	20	8	700	533.3
	13	23	8	500	69.2
	15	26	8	300	86.5
	13	20	8	500	60.2
	15	26	8	200	69.3
	10	23	8	600	87.9
	15	26	8	600	69.2
	13	30	8	200	112.7
	15	26	8	700	90.3
	14	30	8	600	74.5
	13	27	8	400	62.3
	12	21	8	900	58.3
	13	22	8	600	50.8
	15	24	8	500	83.3
	13	23	8	600	69.2
	10	29	8	600	67.1
	11	25	8	700	100.4
	13	20	8	600	46.1
	13	24	8	700	151.4
	13	25	8	500	139.2
	15	29	8	600	166.3
	10	28	8	100	62.1
	13	29	8	400	162.2
	13	29	8	400	87.3
	10	27	8	300	103.2