# STOCHASTIC DYNAMIC PROGRAMMING AND STOCHASTIC FLUID FLOW MODELS IN THE DESIGN AND ANALYSIS OF WEB-SERVER FARMS

A Dissertation

by

PIYUSH GOEL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2009

Major Subject: Industrial Engineering

# STOCHASTIC DYNAMIC PROGRAMMING AND STOCHASTIC FLUID FLOW MODELS IN THE DESIGN AND ANALYSIS OF WEB-SERVER FARMS

A Dissertation

by

PIYUSH GOEL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

| | |
|---|---|
| Chair of Committee, | Natarajan Gautam |
| Committee Members, | Richard M. Feldman |
| | Gary M. Gaukler |
| | Jean-Francois Chamberland |
| Head of Department, | Brett A. Peters |

August 2009

Major Subject: Industrial Engineering

# ABSTRACT

Stochastic Dynamic Programming and Stochastic Fluid-Flow Models in the Design and Analysis of Web-Server Farms. (August 2009)

Piyush Goel, B.Tech., Indian Institute of Technology, Bombay

Chair of Advisory Committee: Dr. Natarajan Gautam

A Web-server farm is a specialized facility designed specifically for housing Web servers catering to one or more Internet facing Web sites. In this dissertation, stochastic dynamic programming technique is used to obtain the optimal admission-control policy with different classes of customers, and stochastic fluid-flow models are used to compute the performance measures in the network. The two types of network traffic considered in this research are streaming (guaranteed bandwidth per connection) and elastic (shares available bandwidth equally among connections).

We first obtain the optimal admission control policy using stochastic dynamic programming, in which, based on the number of requests of each type being served, a decision is made whether to allow or deny service to an incoming request. In this subproblem, we consider a fixed bandwidth capacity server, which allocates the requested bandwidth to the streaming requests and divides all of the remaining bandwidth equally among all of the elastic requests. The performance metric of interest in this case will be the blocking probability of streaming traffic, which will be computed in order to be able to provide Quality of Service (QoS) guarantees.

Next, we obtain bounds on the expected waiting time in the system for elastic requests that enter the system. This will be done at the server level in such a way that the total available bandwidth for the requests is constant. Trace data will be converted to an ON-OFF source and fluid-flow models will be used for this analysis.

The results are compared with both the mean waiting time obtained by simulating real data, and the expected waiting time obtained using traditional queueing models.

Finally, we consider the network of servers and routers within the Web farm where data from servers flows and merges before getting transmitted to the requesting users via the Internet. We compute the waiting time of the elastic requests at intermediate and edge nodes by obtaining the distribution of the outflow of the upstream node. This outflow distribution is obtained by using a methodology based on minimizing the deviations from the constituent inflows. This analysis also helps us to compute waiting times at different bandwidth capacities, and hence obtain a suitable bandwidth to promise or satisfy the QoS guarantees.

This research helps in obtaining performance measures for different traffic classes at a Web-server farm so as to be able to promise or provide QoS guarantees; while at the same time helping in utilizing the resources of the server farms efficiently, thereby reducing the operational costs and increasing energy savings.

*To My Parents...*

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards Dr. Natarajan Gautam for being my advisor during my pursuit of this doctoral degree. I am really grateful for the constant support and supervision he has provided me throughout the years I worked under his supervision.

I am so blessed, for the tremendous sacrifices that my parents, Smt. Vijaya Laxmi Goel and Shri Harish C. Goel, have made to make it possible for me to be where I am. They are my champions, and I hope that they are as proud of me as I am of them. I would also like to thank my dear sister, Ms. Shweta Goel, who has constantly cheered and supported me and my parents through all the good and the tough times.

I am very fortunate to have come across some truly wonderful people during the course of my studies. These are very special people in my life, and I want them to know that I truly appreciate each and every moment of their company, all the support they have given me and all their thoughts for me. I would also like to convey my deepest acknowledgements to all my friends and colleagues, both at Pennsylvania State University and at Texas A&M University, for the continued support and friendship they have given me.

Finally, I would like to thank Dr. Jean-Francois Chamberland, Dr. Richard M. Feldman, and Dr. Gary M. Gaukler for serving as members of my advising committee and providing valuable suggestions and comments to make this a better dissertation.

# TABLE OF CONTENTS

Page

CHAPTER

# LIST OF TABLES

## LIST OF FIGURES

FIGURE Page

# CHAPTER I

# INTRODUCTION

## I.1. Background and Motivation

A Web-server farm is a group of computers acting as servers and housed together in a single location. This could either be a Web site hosted on more than one server, or an Internet service provider (ISP) that provides Web hosting services using multiple servers. Server farms are typically co-located with network switches and/or routers which enable communication between different parts of the cluster and the users of the cluster. Communication within the farm is often based on networks running the IP protocol suite.

To deal with the explosive increase in Internet traffic over the past few years, the only feasible solution is to have a server cluster, rather than one high-performance server. Besides the advantage of scalability, the cluster also offers redundancy and reliability, by providing a safety against failure of any individual server. Performace of Web-servers is critical for the end-to-end performance of sites which service a high volume of requests [1, 2]. As a result, modelling and analysis of Web-servers and server-farms has received significant attention [3, 4].

A very simplified version of a typical Web farm can be represented as shown in Fig. 1. There can be multiple layers of routers/switches within the communication network of the server farm itself, with a link to connect to the outside world.

Designing and managing a Web farm is a very challenging task. It is a significant problem when both the revenue and performance aspects are considered. It entails minimizing the cost of operation while providing the quality of service (QoS)

This dissertation follows the style and format of *IEEE/ACM Transactions on Networking*

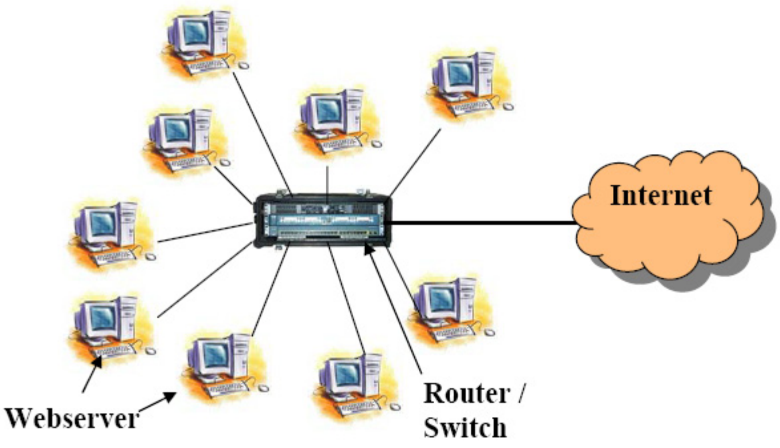

Fig. 1. Typical Web-Server Farm, with a Router Connecting to the Servers and the Internet

expected by the users. This provides a tradeoff between having the maximum possible bandwidth for the servers and routers to provide the best possible service, and minimizing the level of service provided in an effort to control costs. However, once a server farm is in place, efficiently managing the resources can provide a significant boost in service with little increase in costs.

Many Web-server farms serve multiple service sites from a common hardware base. With the rising paradigm of energy and cost savings, these type of Web farms have gained a lot of popularity because they provide a huge potential for financial savings. It is well known that Internet traffic varies dynamically over multiple time scales, and so estimation of workload in such cases is itself a topic of research [5]. Moreover, in the case of shared Web farms, traffic may come from all over the world, which makes the traffic loads dynamic and unpredictable. As a result, static allocation of resources is usually inefficient and risky: while over-provisioning can lead to under-utilization of resources, under-provisioning can lead to possible violation of

QoS guarantees, resulting in loss of customers.

With this in mind, it becomes imperative that different applications be allocated resources based both on their requirements and on the QoS criteria that need to be satisfied. Since the total bandwidth available to the server is limited, we need to make decisions as to which requests are to be accepted and served and how much bandwidth is to be allocated. If we couple in the revenue aspect of the problem as well, we are looking at an admission-control problem for the traffic. Because we have QoS guarantees to be satisfied, we also need to obtain the performance measures of the traffic entering the system.

In this research work, we will work with the given constraints of bandwidth at a Web farm. We will obtain the optimal policies to maximize the revenue, and then compute the performance metrics for different classes of customers in order to be able to evaluate or provide QoS guarantees. The overall objective is to regulate the number of customers coming into the system, sometimes rejecting service even when the capacity exists, in order to maximize the revenue and optimize the level of service that can be provided. This forms a trade-off between blocking customers, and having the maximum performance level for the admitted customers. Coming up with an admission-control policy using the stochastic dynamic programming technique would maximize the profits from the Web-server farm, and then, using different methodologies, we will compute the performance measures for the system.

## I.2. Problem Definition

In this dissertation, a Web-server farm catering to different types of requests is considered. Specifically, we consider two main request classes: streaming and elastic. Consider a Web farm where a given server caters to only one class of request. This assumption is reasonable because the type of files stored on a server can be assumed to be similar. That is, some servers store only video files, some only data files, etc. In order to serve a request, the appropriate file is transmitted from the server using the assigned bandwidth.

The requests in the streaming class require a fixed bandwidth, commonly observed in audio and video requests. Given a fixed maximum capacity available to a server for transmitting the data, only a finite number of such requests can be served simultaneously. This leads to blocking of such requests when a request arrives, but there is no remaining capacity available to transmit the requested data.

Elastic type requests do not necessarily have a bandwidth constraint. Data files are examples of these kind of requests. The performance measure associated with these requests is typically time based, such as the waiting time in the system. Average workload in the system is another measure that is used to ascertain the resource load and utilization. Given a fixed maximum bandwidth available to the server for transmitting the data, any amount of bandwidth can be allocated to each of the elastic requests being served, so that they sum to the capacity. Typically, scheduling logics such as processor sharing, round robin, or even first-in-first-out are used to cater to such requests.

As shown in Fig. 1, the Web farm usually has a router/switch as a link to the Internet. At this point, the files transmitted by all of the servers are received and bandwidth needs to be allocated to all these requests. The total bandwidth available for transmitting is fixed. Now, the streaming requests need to be allocated the bandwidth specified by the request, and hence, take away a fixed chunk from this capacity. All of the elastic requests are served using the remaining capacity. This means that the available bandwidth for elastic requests varies with the total number of requests being served by the system, both streaming and elastic. Because there is potentially no minimum bandwidth requirement for elastic requests, there can be any number of elastic requests being served simultaneously.

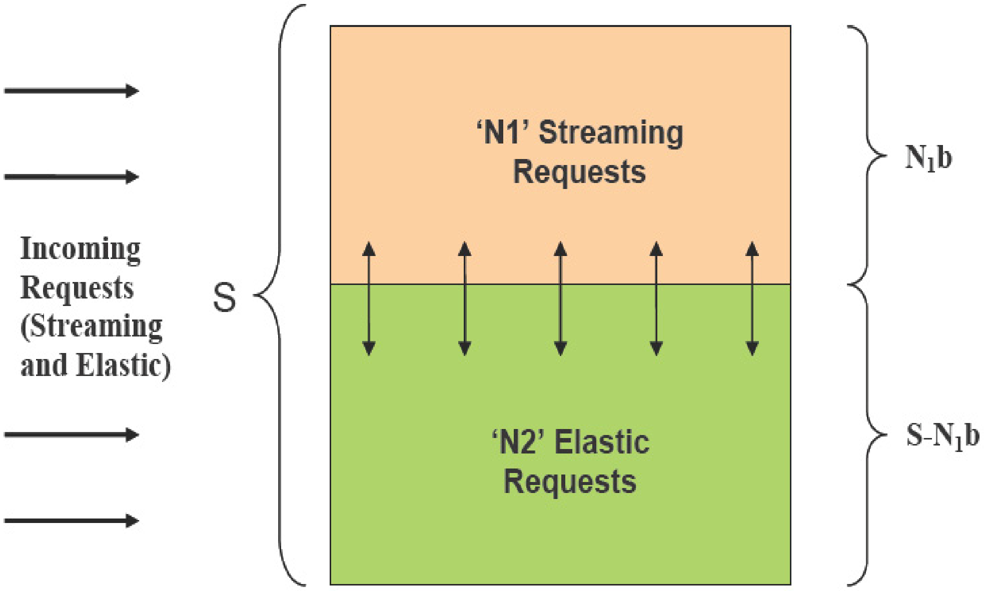

Fig. 2. Bandwidth Allocation in the Server with Two Classes of Requests

In this research, in order to analyze the above-mentioned system, we perform the analysis at three different levels. First, we make a decision as to which requests to serve in order to use the resources efficiently. When either a streaming or an elastic request is served, a file is transmitted from the server to the user. All of the servers in the system are connected to the Internet through a router. Conventional wisdom is that performance limitation in Web farms lie at the edge of the network, i.e., last mile connectivety to users [6]. We will consider total bandwidth available to this router as a contiguous block, which can be allocated dynamically to either type of requests. Once the bandwidth has been allocated to the streaming requests in the system, all of the remaining bandwidth will be used by the elastic requests. This also means that as and when requests enter and leave the system, the bandwidth available to each of the elastic requests changes (Fig. 2). Thus, if the server capacity is $S$ bytes/s, bandwidth required by each streaming class is $b$ bytes/s, and there are $N_1$ and $N_2$ requests of the streaming and elastic class, respectively, the available bandwidth for each elastic class is $\{(S - bN_1)/N_2\}$ bytes/s.

If we let as many streaming customers as the capacity allows, into the server, the bandwidth available to the elastic requests will dwindle drastically, which would have an adverse effect on the performance measures of the elastic class. On the other hand, however, if we do not allow some of the streaming customers, we would be blocking and causing loss of customers even before the system has reached its capacity. Hence, we will have to optimize our decision by considering both types of customers. That is, we will have to compute an admission-control policy for both types of requests so as to maximize the revenue of the system. This policy will consider the customers already in the system and decide if the incoming customer should be admitted or not, depending upon its class. We will use the stochastic dynamic programming technique to achieve the above-mentioned objectives, and the procedure will be demonstrated in Chapter II.

The elastic requests are typically served by different servers in the Web farm. The second task will be to compute the performance measures of these kinds of requests and be able to design the system to satisfy the QoS requirements. For this undertaking, we will first analyze one of these servers, catering to elastic requests. This server has a fixed capacity and will operate using a work-conserving policy– typically processor sharing. Because of this, we don't need to specify the scheduling methodology it uses. The analysis of such a system is demonstrated in Chapter III.

Data from all of these servers catering to elastic requests passes through the router before being transmitted via Internet. This could be a bottleneck node, as data from all the servers is received here and transmitted using the limited-capacity bandwidth. In order to be able to provide a QoS guarantee to an elastic request, one also needs to analyze this node. At this point, data from different servers gets aggregated, and thus, we will need to use some kind of superpositioning principle in order to obtain the performance measure at the conglomeration. We propose a simple and fast methodology to achieve this goal, and we present it in Chapter IV.

The above mentioned methods of analysis will help us in obtaining the performance measures for different types of requests. In case of streaming requests, we will use the performance measure of interest as the blocking probability. This will essentially represent a measure of the fraction of arriving customers not being allowed to enter the system. In case of elastic requests, we will be interested in obtaining the blocking probability as well as the time spent in the system. We need to evaluate this value in order to project the system performance under varying load conditions. Due to the characteristics of network traffic, we chose to use the fluid-flow model for this analysis. This requires converting the discrete arrivals into an ON-OFF source and computing the distributions from the trace data. This fluid-flow model, and obtaining distribution using the trace data, will also be discussed in forthcoming chapters.

## I.3. Organization of the Dissertation

This research work is organized as follows. A brief description of the problem being dealt with has been presented in Section I.2. The problem is then broken down into two major parts: obtaining the admission-control policy and obtaining the performance measures for the Web server. Each of these problems will be addressed in forthcoming chapters, with each problem involving its own challenges, related literature, solution methodology, and conclusions. In order to obtain the admission-control policy, we will use the stochastic dynamic programming approach,. We define this problem in detail and then present the solution methodology in Chapter II. Chapter III describes the methodology being used to compute the waiting time in the system for incoming elastic requests. This chapter, however, deals only with the case of a single inflow. This methodology is expanded upon and the performance metrics are obtained for the aggregate elastic flow in Chapter IV. Finally, we conclude the dissertation in Chapter V by summarizing the findings and presenting some future research ideas.

# CHAPTER II

# DYNAMIC RESOURCE ALLOCATION AND ADMISSION-CONTROL IN WEB-SERVER FARMS

## II.1. Introduction

With a tremendous increase in popularity over the past couple of decades, Internet has become an important medium for hosting businesses and other e-commerce services. With the World Wide Web becoming more and more prevalent, the Web-server farms that host third-party Web applications and services are receiving a lot of attention.

Web farms serve multiple service sites from a common hardware base. In this era of efficiency, it can be a challenging task to allocate resources to maximize revenue while satisfying QoS guarantees of various companies that the Web-server farm might be catering.

The Web sites a server farm may be hosting can comprise different types of traffic. The two most popular and broad classifications of traffic are streaming and elastic. As mentioned in Section I.2, streaming class has fixed-bandwidth requirements, and usually it is associated with a loss-based QoS requirement. In contrast, elastic class can utilize however much bandwidth is allocated to it, and usually has a delay-based QoS requirement attached to it.

Most Web pages on the Internet today have both streaming and elastic kinds of information that need to be delivered to end users. Hence, in order to enhance the customer experience, it is necessary to give importance to both kinds of traffic and have good response performance in both types of requests. In order to do this, shared resources need to be optimized for both types of traffic simultaneously which improves the overall resource utilization as well.

The final router that provides connection to the Internet receives both kinds of files and has to make the best use of available bandwidth to transmit them. The total bandwidth available to serve these two kinds of requests is limited by the capacity, and in order to cater to both classes of requests effectively bandwidth needs to be allocated to the two classes optimally. Thus, to satisfy the QoS and to provide the best service, it is necessary that the available resources be optimized to perform at maximum efficiency.

It is well known that Internet traffic varies dynamically over multiple time scales, and so the estimation of such workloads is itself a topic of research [5]. As a result, static allocation of resources, where a block of bandwidth is allocated to the streaming class, and the remaining is utilized by the elastic class, is both inefficient and risky. While at some point during the day a given allocation may be over-provisioning and leading to under-utilization of resources for a class, at other times during the same day it may be under-provisioning for the same class, leading to possible violation of QoS guarantees and resulting in customer dissatisfaction and loss of customers. The only way to approach this issue is to dynamically allocate the resources amongst the competing applications and run the system efficiently.

In this research, we will develop a strategy to maximize revenue by choosing when to accept or reject an incoming request. Our secondary goal would be to provide the best possible QoS for different classes of requests, and evaluate the QoS guarantees that could be offered. Most of the present literature in this area focuses on static allocation policies. The main drawback of this kind of policy, as discussed earlier, is that the elastic traffic is unable to utilize any remaining bandwidth if there is not enough streaming traffic on the server. The same is true for the streaming traffic, as an incoming streaming request will be turned down if there are enough streaming requests in the system even if there is little or no elastic traffic.

This calls for dynamic allocation of bandwidth among the two classes, depending upon the number of requests of each type being served, in order to improve the efficiency of the system and increase the revenue. In this research work, we will consider a Web-server farm catering to these two types of requests. The requests will be satisfied from the same block of bandwidth, with a fixed maximum capacity. We will formulate the problem as a Markov Decision Process (MDP) and obtain an admission-control policy for the requests dependent upon the current number of requests being served of each type, so as to maximize revenue and, at the same time, satisfy the QoS requirements of the two classes.

The rest of the chapter is organized as follows. In Section II.2, we present a brief review of related literature. Following this, we define the problem at hand in Section II.3. We then formulate the problem as an MDP in Section II.4. We also obtain the optimal admission-control policy analytically after obtaining some structural properties in this section. Numerical examples of the problem are then presented, where we compute the optimal policy and compare it with some of the existing policies in the literature. This numerical analysis is presented in Section II.6. Finally, we present the conclusions Section II.7.

## II.2. Literature Review

The system being considered in this research work is essentially a combination of two types of problems traditionally seen in the literature: *loss network* for the streaming requests, in which the performance measure of interest is generally the *loss* or *blocking* probability; and *delay network* for elastic class, in which the performance index of interest would be the *delay*.

Most of the literature, especially on analytical models for multi-class requests, falls under one of these two categories. Loss networks have been fairly well studied in the literature, but research focusing on loss networks considers only the objective of the streaming traffic, while completely ignoring the elastic traffic. Several articles deal with the stochastic knapsack problem or $G/G/C/C$ queueing models to optimize for streaming traffic and assume that any unused bandwidth will be consumed by the elastic traffic. Ross [7] has provided a collection of results on the evaluation of broadband telecommunication networks, providing a rigorous treatment of call admission and congestion control in ATM networks. Many researchers, such as Kaufman [8] have tried to develop analytical loss models to compute the blocking probability in case of shared resource environments. Application examples of this modeling paradigm include the work by Dziong and Mason [9], who have tried to use the framework of cooperative game theory for the analysis and synthesis of call admission strategies in broadband networks.

As mentioned earlier, these models do not consider the QoS requirements of the elastic traffic. On the other hand, however, research focusing on delay networks assume that a fixed portion of bandwidth is allocated for these requests, ignoring the bandwidth that may be available to them by virtue of the streaming requests not using it. Chandra et al. [10] developed an analytical model that considers dynamic resource allocation in servers that use generalized processor sharing. It considers Web data traffic in which the QoS requirement is mean response time. Some researchers (such as Abdelzaher et al. [11]) have focused on designing adaptive systems for Web servers based on a control theoretical approach, while others (such as Lu et al. [12]) designed adaptive systems that can react to workload changes in the context of storage systems. Chase et al. [13] have tried to obtain optimization techniques for reducing energy consumption in data centers, whereas Massoulié and Roberts [14] concern themselves with the design of a distributed algorithms for sharing network bandwidth among contending flows, using fairness notions.

Hence, most of the loss models do not incorporate the elastic requests coming into the system, whereas the analysis based on delay networks typically ignores any bandwidth unused by the streaming requests. Nonetheless, there have been some studies that consider both kinds of requests into the server. Fodor et al. [15] have analyzed a system with three classes of requests: rigid, streaming and elastic, and have used the classical multi-rate methodology for the performance analysis of admission-control-based service. They have applied it to a single link, demonstrating the trade-off between blocking probability and throughput. Mahabhashyam and Gautam [16] have also considered a shared data center and have developed an analytical model using the Matrix Geometric Method to obtain the performance measures of the system. However, unlike this research work, they consider a static admission-control policy, which is independent of the current state of the system. They obtain a threshold on the streaming class requests being served; any further streaming class arrivals above this would be rejected. This would make it a special case of the problem we are analyzing.

Dynamic programming technique has been widely used in developing admission-control and resource-allocation policies in various applications, including high-speed networks. Altman [17] has presented a comprehensive survey of related works until that point in time. Bhatnagar and Reddy [18] have studied a problem of admission control of packets in communication networks in the continuous-time queueing framework. However, in recent times, the neuro-dynamic programming technique, which uses neural networks approximations to solve multi-dimensional problems, has become more popular. This technique has also been applied to wireless communication networks [19], ATM networks [20], etc. Senouci et al. [21] have considered the call admission-control problem in a multimedia cellular network that handles several classes of traffic with different resource requirements, and they have proposed algorithms to earn higher revenues.

## II.3. Problem Description

The objective in this research work is to obtain the optimal admission-control policy given that all of the bandwidth will be shared by the two classes of requests. We will use stochastic dynamic programming approach to obtain the admission-control policy for the system, while also trying to maximize the revenue.

We consider a system with these two classes of requests: streaming and elastic, which will henceforth be referred as Class-1 and Class-2, respectively. The requests are served by the Web server by transmitting the requested data. Let the bandwidth capacity of the Web server be $S$.

Let each of the Class-1 traffic requests require a constant bandwidth of $b$, while Class-2 traffic shares all of the leftover bandwidth from Class-1 traffic. Hence, for example, if there are $n_1$ Class-1 and $n_2$ Class-2 requests being served by the system, all of the Class-1 requests would be allocated bandwidth $b$ each, whereas the Class-2 request would be allocated $((S - n_1 b)/n_2)$ each for processing (as shown in Fig. 2). Now, as $n_1$ and $n_2$ changes over time, notice that the processing rate for Class-2 would also change. We assume no minimum or maximum bandwidth allocation for either of the classes, except where limited by the capacity of the Web server.

In this system, we will assume that all of the requests that are accepted are processed simultaneously. This is essentially the widely known processor-sharing discipline in case of the elastic requests. Each streaming request, of course, uses the bandwidth allocated to it, and does not interact with other requests. It is assumed that the rejected requests are lost.

We define the state of the system as the number of requests being served of each type. Given the state of the system (i.e., $n_1$ and $n_2$ values), if the system receives a request, we need to decide whether to accept it and provide service or not. Through this decision process, we would try to make sure that we are allowing as many Class-1 customers into the system, while not impeding the QoS provided to the Class-2 customers. Such a set of decisions would constitute the decision policy, which would be the admission-control policy. Our objective is to come up with such a policy in order to maximize the profit of the system, given the parameters for each class such as the acceptance reward, holding cost, arrival rate, and expected workload.

For elastic traffic, it is generally assumed that admitting a new demand and reducing the throughput of the system is better than rejecting the new request: the utility of a request as a function of its throughput is assumed to be positive and strictly concave everywhere so that overall utility increases as more flows are admitted [22]. With this in mind, we would allow all of the elastic requests into the system and exercise admission-control only on the Class-1 requests.

## II.4. Analytical Solution

In this section, we formulate the problem described in Section II.3 as an MDP problem, and we obtain the optimal objective function. We then solve the problem by establishing some structural policies for this function, and we obtain the optimal admission-control policy analytically.

### II.4.1. Formulation

Let the arrival rate of requests for Class-$i$ be according to Poisson processes with rate $\lambda_i$. Let the customers of Class-1 remain in the system (and block bandwidth $b$) for an exponentially distributed time with rate $\mu_1$. As for the Class-2 customers, due to the nature of the system, their processing rate might change with every arrival or departure from the system. Hence, rather than defining their holding time distribution, we assume that each brings in a workload exponentially distributed with mean $1/\mu_2$ bytes.

Now, let each request of Class-$i$ bring a reward of $R_i$, received only if the request is allowed to enter the system and be served. Moreover, let $C_i$ be its holding cost, per unit time.

Our main objective here is to derive a decision policy, such that, given the state of the system, we can decide whether or not to accept a Class-1 arrival. We define the state of the system as a two-dimensional vector representing the number of requests being served of both classes: $(n_1, n_2)$. The decision space at a Class-1 arrival consists of two actions: $A_s = \{0, 1\}$, where 0 represents rejecting, and 1 represents accepting the incoming request. The only action associated with the arrival of a Class-2 request or any departure is to allow it.

In order to use stochastic dynamic programming to obtain the optimal admission-control policy, we use the well-known uniformization technique (see [23, 24]). We have

$$\lambda_1 + \lambda_2 + M\mu_1 + S\mu_2 + \alpha = 1 \ , \tag{2.1}$$

where $M = \lfloor S/b \rfloor$ is the maximum number of Class-1 customers that can be present in the system (i.e., $n_1 \leq M$), and $\alpha$ is the uniformization parameter. Note that if there are $M$ Class-1 customers in the system, a Class-1 arrival must be rejected. Since there is no minimum bandwidth requirement for the Class-2 customers, we can have any number ($\geq 0$) of these in the system. The uniformization technique introduces phantom arrivals and departures into the system. In our system, we would stay in the original state at rate $(1 - \lambda_1 - \lambda_2 - \mu_1 n_1 - (S - bn_1)\mu_2)$.

In order to write the optimality equation, we need to compute the future profit functions corresponding to different transition events. We will make use of event-based dynamic programming, as introduced in [25], in order to formulate the MDP. According to this, we have an *event operator* corresponding to every possible event in the system. This operator maps the set $F$ of all real-valued functions to the state variable $x$ into itself. Thus, for example, for arrivals of Class-1 customers we have the corresponding arrival operator $T_{A_1}$ defined as

$$T_{A_1}f(x) = \max\{R_1 + f(x + e_1), f(x)\} \tag{2.2}$$

for $f \in F$, where $e_i$ is the $i^{th}$ unity vector. $T_{A_1}$ may be interpreted as the optimal value function for a one-stage problem in which one must decide to accept or reject a Class-1 arrival, after which a terminal state-dependent revenue is earned according to the function $f$. For this problem, we choose event operators by associating one with every event in the system: arrivals and departures of Class-1 and Class-2 requests. Note that there is no decision to be made in case of the departures and arrival of a Class-2 request.

Now, we define the value function $V_n$ of this problem as $V_{n+1} = TV_n$ and $V_0(x) = C(x)$. The fact that $\lambda_1 + \lambda_2 + M\mu_1 + S\mu_2 < 1$ for $\alpha > 0$ ensures that $T$, which is a linear combination of the event operators, is a contraction operator. Following, we show some structural properties for $V_n$. Since these properties hold for each $V_n$, they will also hold true for the limiting optimal policy.

### II.4.2. Submodularity

First, we show that the value function is submodular in nature. This property, along with others, would be very helpful in obtaining the structure of the optimal decision policy.

In order to establish the structural properties, it is sufficient to show that certain properties of the function defined on the state space are preserved under the action of the value-iteration operator, $T$ (see [26]). In particular, a function is said to be submodular if

$$V_n(x+e_1)+V_n(x+e_2)\geq V_n(x)+V_n(x+e_1+e_2)\ . \tag{2.3}$$

Below, we will show that this property holds and is preserved under the value-iteration operator at hand. This proof is very similar to that presented in [27]. We have

$$\begin{aligned}V_{t+1}(n_1,n_2) &= -C_1n_1-C_2n_2+\lambda_1\max\{R_1+V_t(n_1+1,n_2),V_t(n_1,n_2)\}\\ &\quad+\lambda_2(R_2+V_t(n_1,n_2+1))+\mu_1n_1V_t((n_1-1)^+,n_2)+(S-bn_1)\mu_2V_t(n_1,(n_2-1)^+)\\ &\quad+(1-\lambda_1-\lambda_2-\mu_1n_1-(S-bn_1)\mu_2)V_t(n_1,n_2)\ ,\end{aligned} \tag{2.4}$$

where $(a)^+\equiv\max\{a,0\}$.

It is clear that $V_0$ satisfies the property. We will show that if the above inequality is satisfied by $f$, it gets preserved by each of the event operators. This proves the inequality because it is closed under linear combinations.

We show the proof for the operator $T_{A_1}$ here, since the proof involving others are much simpler and follow easily due to their not having a decision involved. Let $a_1$ be the maximizing action in $T_{A_1}f(y)$ for $y=x$ and $a_2$ be the maximizing action for $y=x+e_1+e_2$, where actions 0 and 1 refer to rejecting or accepting a request, respectively. Consider the case in which $a_1=1$ and $a_2=0$. Then,

$$\begin{aligned}&T_{A_1}f(x)+T_{A_1}f(x+e_1+e_2)\\ &\qquad=f(x+e_1)+R_1+f(x+e_1+e_2)\\ &\qquad\leq T_{A_1}f(x+e_1)+T_{A_1}f(x+e_2)\ .\end{aligned}$$

The inequality is true because $T_{A_1}f(x+e_1) \geq f(x+e_1)$ and $T_{A_1}f(x+e_2) \geq R_1 + f(x+e_1+e_2)$, as a direct consequence of Equation (2.2). Similarly, for the case in which $a_1 = 0$ and $a_2 = 1$, we have

$$\begin{aligned} & T_{A_1}f(x) + T_{A_1}f(x+e_1+e_2) \\ &\qquad = f(x) + R_1 + f(x+2e_1+e_2) \\ &\qquad \leq R_1 + f(x+2e_1) + f(x+e_2) \\ &\qquad \leq T_{A_1}f(x+e_1) + T_{A_1}f(x+e_2)\ . \end{aligned}$$

Here, the first inequality follows from Equation (2.3) and the second from Equation (2.2). In case $a_1 = a_2$, the result can similarly be established as follows.

$a_1 = a_2 = 0$:

$$\begin{aligned} & T_{A_1}f(x) + T_{A_1}f(x+e_1+e_2) \\ &\qquad = f(x) + f(x+e_1+e_2) \\ &\qquad \leq f(x+e_1) + f(x+e_2) \\ &\qquad \leq T_{A_1}f(x+e_1) + T_{A_1}f(x+e_2)\ . \end{aligned}$$

$a_1 = a_2 = 1$:

$$\begin{aligned} & T_{A_1}f(x) + T_{A_1}f(x+e_1+e_2) \\ &\qquad = f(x) + R_1 + f(x+2e_1+e_2) + R_1 \\ &\qquad \leq f(x+e_2) + f(x+2e_1) + 2R_1 \\ &\qquad = (f(x+2e_1) + R_1) + (f(x+e_2) + R_1) \\ &\qquad \leq T_{A_1}f(x+e_1) + T_{A_1}f(x+e_2)\ . \end{aligned}$$

Hence, we can say that submodularity is preserved under $T_{A_1}$. Similarly, it can easily be shown that the property holds for other event operators, and hence the value function is submodular.

### II.4.3. Monotonicity

We would call the value function to be monotonic if it satisfies the following condition for all $x$ and $n$:

$$V_n(x) \geq V_n(x + e_1) \tag{2.5}$$

Note that we are proving the monotonicity only with respect to Class-1 customers. It could however be proven with respect to Class-2 customers as well. Again, we use $V_0 = C$, and the property obviously holds for $V_0$. We use the same procedure as before to show that the property is preserved under each event operator. Following, we demonstrate the case for $T_{A_1}$. The proof for other event operators follows similarly.

Let $a_1$ be the maximizing action in $T_{A_1}f(y)$ for $y = x$ and $a_2$ be the maximizing action at $y = x + e_1$. Then, we have the following cases.

$a_1 = a_2 = 0$:

$$\begin{aligned} T_{A_1}f(x + e_1) &= f(x + e_1) \\ &\leq f(x) \\ &= T_{A_1}f(x) \ . \end{aligned}$$

$a_1 = 1, a_2 = 1$:

$$\begin{aligned} T_{A_1}f(x + e_1) &= R_1 + f(x + 2e_1) \\ &\leq R_1 + f(x + e_1) \\ &= T_{A_1}f(x) \ . \end{aligned}$$

$a_1 = 1, a_2 = 0$:

$$\begin{aligned} T_{A_1} f(x+e_1) &= f(x+e_1) \\ &\le R_1 + f(x+e_1) \\ &= T_{A_1} f(x) \ . \end{aligned}$$

$a_1 = 0, a_2 = 1$:

$$\begin{aligned} T_{A_1} f(x+e_1) &= R_1 + f(x+2e_1) \\ &\le f(x) \\ &= T_{A_1} f(x) \ . \end{aligned}$$

Thus, the property is preserved under the event operator $T_{A_1}$. The proof with other operators can be constructed similarly and easily. Hence, as the property is preserved under each individual event operator, it would be preserved under their linear combination $T$. Therefore, the property holds for the value function at hand.

### II.4.4. Concavity

The value function for the given problem would be said to be concave if the following property holds:

$$\begin{aligned} 2V_n(x+e_1) &\ge V_n(x) + V_n(x+2e_1) \\ 2V_n(x+e_2) &\ge V_n(x) + V_n(x+2e_2) \ . \end{aligned} \tag{2.6}$$

It is obvious that the property holds for $V_0 = C$. Again, we would follow the same process as used to prove submodularity to obtain the result. That is, if the property is satisfied by $f$ and is preserved by each event operator, we would have proven the inequality. We first show the case for $T_{A_1}$ here.

Let $a_1$ be the maximizing action at $T_{A_1}f(y)$ for $y = x$, $a_2$ be the maximizing action for $y = x + e_1$ and $a_3$ be the maximizing action at $y = x + 2e_1$. Then, we have the following cases.

$a_1 = a_2 = a_3 = 0$:

$$\begin{aligned} T_{A_1}f(x) + T_{A_1}f(x+2e_1) &= f(x) + f(x+2e_1) \\ &\leq 2f(x+e_1) \\ &= 2T_{A_1}f(x+e_1) \ . \end{aligned}$$

$a_1 = a_2 = a_3 = 1$:

$$\begin{aligned} T_{A_1}f(x) + T_{A_1}f(x+2e_1) &= f(x+e_1) + 2R_1 + f(x+3e_1) \\ &\leq 2R_1 + 2f(x+2e_1) \\ &= 2T_{A_1}f(x+e_1) \ . \end{aligned}$$

$a_1 = 1, a_3 = 0$:

$$\begin{aligned} T_{A_1}f(x) + T_{A_1}f(x+2e_1) &= f(x+e_1) + R_1 + f(x+2e_1) \\ &\leq 2f(x+e_1) + 2R_1 \\ &\leq 2T_{A_1}f(x+e_1) \ . \end{aligned}$$

$a_1 = 0, a_3 = 1$:

$$\begin{aligned} T_{A_1}f(x) + T_{A_1}f(x+2e_1) &= f(x) + R_1 + f(x+3e_1) \\ &\leq f(x) + f(x+2e_1) + 2R_1 \\ &\leq 2f(x+e_1) + 2R_1 \\ &\leq 2T_{A_1}f(x+e_1) \ . \end{aligned}$$

Note that since the value function is monotonic with respect to $e_1$, if $a_1 = a_3$, we do not allow for $a_2$ to take a different value. Hence, from above, we can say that concavity is preserved under $T_{A_1}$. Similarly, we can show that the property is conserved under other event operators and hence, the property holds for the value function.

### II.4.5. Optimal Policy

Intuitively, it is expected that it would be less profitable to accept requests because the number of requests already in the system increases. This can be seen from Equation (2.3), by rewriting the equation as follows [28]:

$$V_n(x + e_1 + e_2) - V_n(x + e_1) \leq V_n(x + e_2) - V_n(x)$$

$$V_n(x + e_1 + e_2) - V_n(x + e_2) \leq V_n(x + e_1) - V_n(x)$$

We now prove that the optimal admission-control policy is of a switching curve type. Using the concavity property given in Equation 2.6, we have

$$V_n(x + e_1) - V_n(x) + R_1 \geq V_n(x + 2e_1) - V_n(x + e_1) + R_1 \ .$$

Thus, if $R_1 + V_n(x + 2e_1) - V_n(x + e_1) \geq 0$, i.e., admission is optimal in state $x + e_1$, then we have $R_1 + V_n(x + e_1) - V_n(x) \geq 0$, i.e., admission is optimal in state $x$. Similarly, we can show that if rejection is optimal in state $x$, then so is rejecting the request in state $x + e_1$. Similar results can be obtained for the other dimension with respect to Class-2 arrivals as well.

We can state this optimal switching curve policy as follows: for every fixed value of $n_2$ there is a threshold level $L(n_2)$, such that it is optimal to admit customers of Class-1 if and only if $n_1 \leq L(n_2)$.

This admission-control policy can be represented as shown in Fig. 3. Here, if the

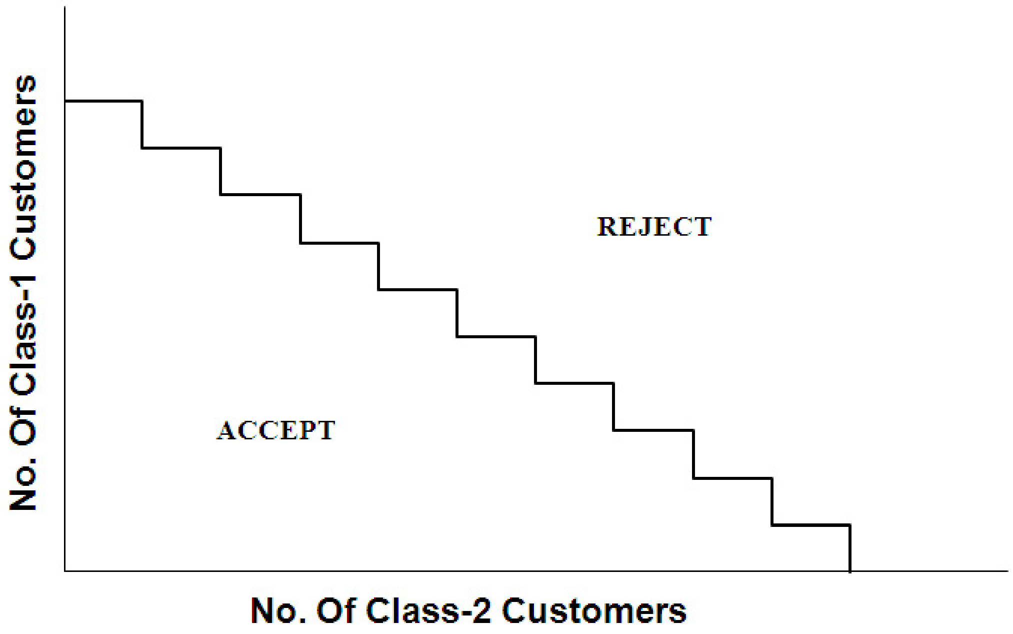

Fig. 3. Switching Curve Representing the Form of the Optimal Admission-Control Policy

state of the system lies in the region marked “Accept”, an incoming Class-1 requested is admitted to the system and served. Otherwise, the request is rejected and lost.

Hence, we obtain the optimal objective function as

$$\begin{aligned}V(x) = V(n_1, n_2) = &-c_1 n_1 - c_2 n_2 + \lambda_1 \max\{R_1 + V(n_1 + 1, n_2), V(n_1, n_2)\} \\ &+ \lambda_2(R_2 + V(n_1, n_2 + 1)) + \mu_1 n_1 V((n_1 - 1)^+, n_2) \\ &+ (S - bn_1)\mu_2 V(n_1, (n_2 - 1)^+) + (1 - \lambda_1 - \lambda_2 - \mu_1 n_1 - (S - bn_1)\mu_2)V(n_1, n_2)\ .\end{aligned} \tag{2.7}$$

### II.4.6. Blocking Probability

Whenever we have a system in which it is not possible to provide service to every incoming customer, one of the performance metrics of interest is the blocking probability: the fraction of customers that will be turned down because of unavailable capacity. In the process of determining the optimal admission-control policy for Class-1 customers, we also need to compute the fraction of customers for which we will not be able to provide service for. This would essentially be the sum of steady-state probabilities of all states where we would reject an incoming request.

## II.5. Value-Iteration and Neighborhood Search

One of the popular techniques of solving MDP problems is using the value-iteration algorithm (see [29]). We have obtained the value function (given in Equation (2.7)) for the problem at hand. Hence, we can use the value-iteration procedure to obtain the optimal admission-control policy for any given parameters. However, note that the value-iteration procedure makes no presumptions about the form of the optimal policy. In other words, while solving the numerical problem, we would not be using the information that the optimal policy is of the form of a switching curve.

It is also well known that the value-iteration algorithm does not lend itself well to large-scale MDPs, mainly due to its slow convergence rate. Although there are various methods described in the literature that try to overcome this shortcoming, for our given problem, we can use the fact that we know the structure of the optimal policy to obtain the numerical solution, which cuts down significantly on the time required to solve the problem.

We will use a simple neighborhood search to obtain the optimal admission-control policy with the given parameters. Following, we describe the neighborhood search algorithm we used. We start with an initial solution, which could be intelligently 'guessed', or could be just a naïve solution such as rejecting all Class-1 requests. If we define 'neighboring' states as the ones with only one different decision in the decision policy. For example, the only feasible neighbor to a 'reject-all' policy, given the structure of a switching curve, is to accept the request when there are no Class-2 requests in the system. In general, neighbors of a given feasible decision policy are shown in Fig. 4, where a neighboring policy is obtained by picking just one set of dotted line at a time, along with the solid lines that mark the original policy.

For each set of neighbors, we evaluate the policies and obtain the best neighborhood solution. If this is better than the incumbent solution, we make this the incumbent solution and find neighbors for it. Otherwise, we stop the procedure and declare the incumbent solution as the optimal solution.

In order to compare different neighborhood solutions, we need to evaluate the expected reward from each such solution and compare them. We make use of the fact that arrival times for both classes of requests are modeled using exponential distribution. Because the arrival and service rates are known, given the decision policy, we can generate the rate transition diagram, as shown in Fig. 5. Note that in this diagram, rates are shown when we would accept all incoming arrivals. If we were

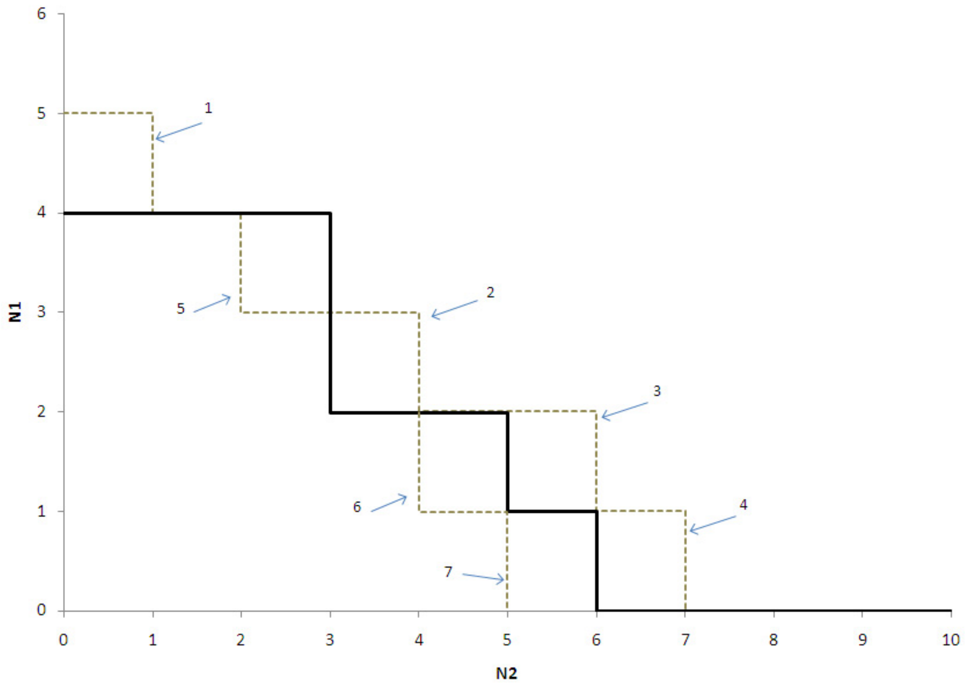

Fig. 4. Picking the Neighbors in the Neighborhood Search Algorithm

to reject a Class-1 arrival in state $(n_1, n_2)$, the transition rate to state $(n_1 + 1, n_2)$ would be 0. Also note that the number of states is limited to $M$ in the Class-1 dimension, but could go to infinity in the other dimension. Using this rate diagram, we can obtain the state transition probability matrix $(Q_D)$.

Using continuous time Markov chain (CTMC) analysis, we can compute the steady-state probability of each of the states by solving for $\pi_D \cdot Q_D = 0$ and $\sum_{i \in S} \pi_D^i = 1$ simultaneously. Expected reward and holding cost can be computed for each state using the $Q_D$ and $\pi_D$ matrices. Thus, we obtain the expected profit for the decision policy, which can later be compared to that of another policy.

Performance of this neighborhood search can be improved significantly by using a 'good' initial solution. One could use some clever heuristic to come up with a

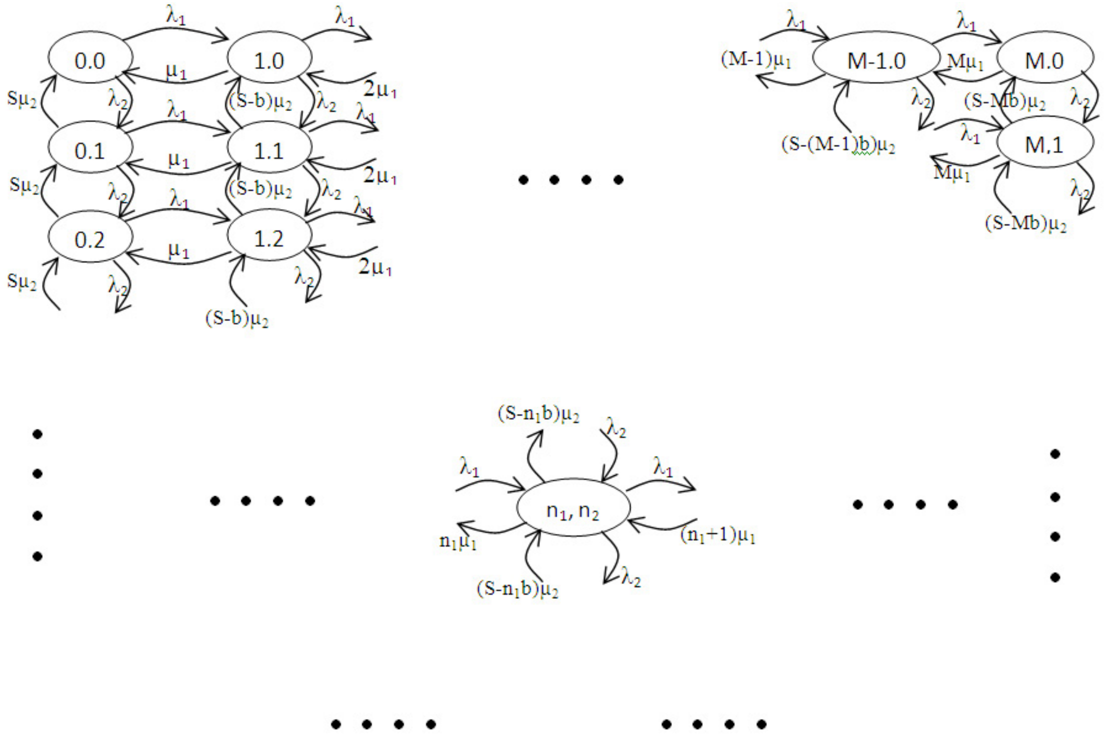

Fig. 5. Transition Rate Diagram for the CTMC

good starting point, and then obtain the optimal solution in just a few iterations. However, our focus here is to emphasize that one could use the structure of the policy and a clever technique and come up with a solution much faster than with the value-iteration algorithm, be it a neighborhood search algorithm, or some other technique. We found that even our method yields results much quicker than the value-iteration algorithm, especially for problems reasonably large in size.

## II.6. Numerical Analysis

We use some numerical examples to illustrate the problem we are trying to solve, and we demonstrate the methodology being used. We use both methods (value-iteration and neighborhood search) to obtain the optimal admission-control policy for a small example problem. We offer the following hypothetical example.

Consider a server with the output bandwidth capacity ($S$) of 2,560 kbps. Consider exponentially distributed arrivals of both types of requests, streaming and elastic, at the rate of 5 requests per second. We assume that each of the Class-1 (streaming) requests requires 256 kbps of dedicated bandwidth. Then, there can be a maximum of $M = \lfloor 2,560/256 \rfloor = 10$ Class-1 requests that can be served simultaneously. Of course, there can be any number of Class-2 requests in the system, because we would not reject any such request. However, note that in this particular case, if we were to accept 10 Class-1 customers, there would be no bandwidth available for Class-2 customers. Let the processing rate of each Class-1 request be 1 per sec, the reward associated with accepting each such request be $R_1 = \$60$, and the holding cost be $C_1 = \$1/\text{sec}$. Class-2 (elastic) requests use all of the bandwidth left over from serving the Class-1 requests in the system. Let each of these requests bring in an exponentially distributed workload with a mean of 0.1. That is, if there were a total of $n_1$ Class-1 and $n_2$ Class-2 requests in the system, the mean processing time for a Class-2 request would be $n_2\mu_2/(S - n_1)b$, where $1/\mu_2 = 0.1$.

Let the reward obtained for each such request served be $R_2 = \$25$ and associated holding costs be $C_2 = \$0.5/\text{sec}$. Then, using these parameters and the value function written in Equation (2.7), we can use the value-iteration procedure to obtain the optimal admission-control policy. This solution is depicted in Fig. 6.

The optimal solution can be summarized as follows: admit a Class-1 customer if

- $n_2 < 1$ and $n_1 < 6$, or
- $1 \leq n_2 < 2$ and $n_1 < 4$, or
- $2 \leq n_2 < 3$ and $n_1 < 3$, or
- $3 \leq n_2 < 4$ and $n_1 < 2$, or

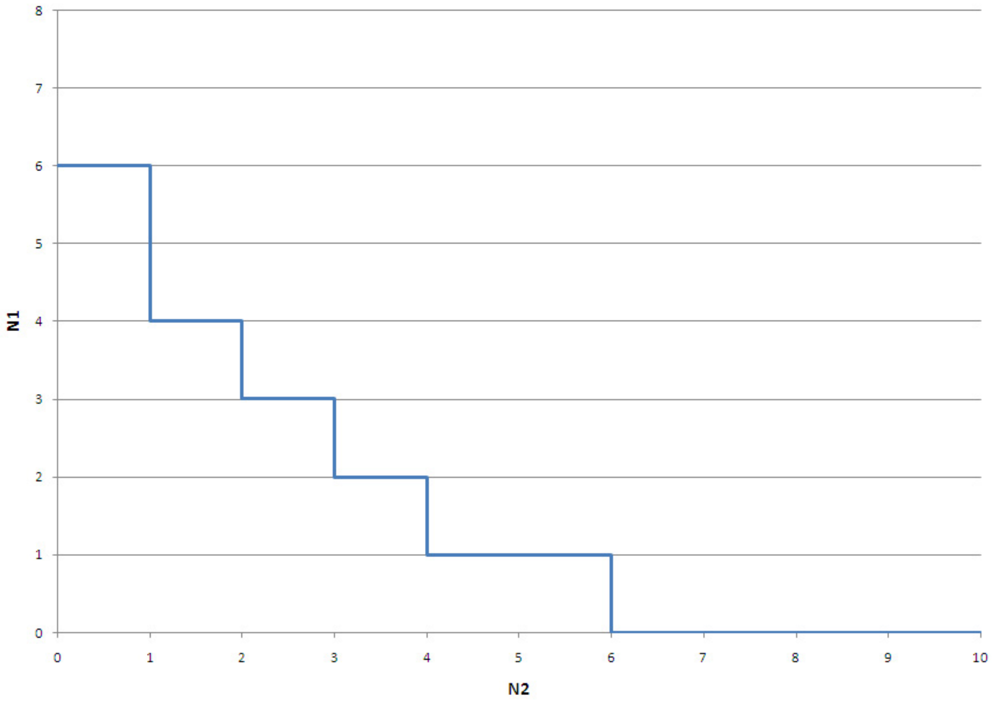

Fig. 6. Optimal Admission-Control Policy for the Small Numerical Example

- $4 \leq n_2 < 6$ and $n_1 < 1$;

reject otherwise.

As can be seen from Fig. 6, the optimal admission-control policy is indeed in the form of a switching curve. We also use the neighborhood search algorithm described in Section II.5 to obtain the optimal policy, and not surprisingly, we obtain the same policy as the optimal policy using this method as well. We used the naïve solution, rejecting all Class-1 arrivals, as our starting solution in this case.

The steady-state probability of being in a state for the system are obtained as

$$P = \begin{bmatrix} 0.00492 & 0.00256 & 0.00050 & 0.00001 & 0.00000 \\ 0.02700 & 0.01289 & 0.00281 & 0.00001 & 0.00000 \\ 0.07344 & 0.03248 & 0.00799 & 0.00002 & 0.00000 \\ 0.13211 & 0.05474 & 0.00154 & 0.00004 & 0.00000 \\ 0.17668 & 0.06907 & 0.00229 & 0.00008 & 0.00000 \\ 0.18186 & 0.06569 & 0.00260 & 0.00010 & 0.00000 \\ 0.10603 & 0.05028 & 0.00231 & 0.00011 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \end{bmatrix}.$$

Note that the number of columns would actually go to infinity towards the right, but all these values would be equal to zero. Using this, we obtain the blocking probability for this optimal policy as $(0.10603 + 0.05028 + 0.00231 + 0.00011) = 0.15873$. That is, about 15.9% of incoming Class-1 customers will be denied service.

We could compare this solution to a static allocation policy to see the difference in performance. A static allocation policy, in this case, would be to allow an incoming Class-1 customer if the number of Class-1 requests being served is less than or equal to a fixed number, say $m(\leq M)$. In this static allocation policy, we would still be allocating all of the bandwidth not being used by Class-1 customers to the Class-2 customers. Thus, even this policy would clearly outperform a policy in which the Web server is divided into two blocks, each serving a separate class of requests.

This comparison is shown in Fig. 7. The graph represents the expected profit,

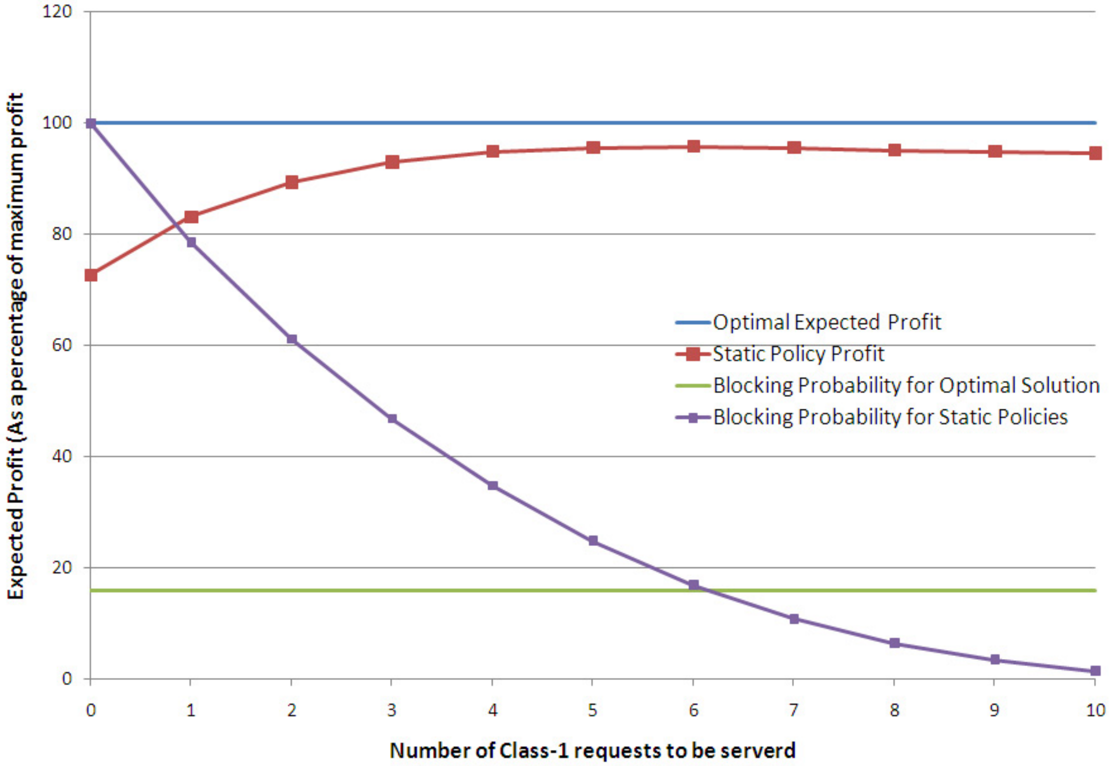

Fig. 7. Comparison of the Static Allocation Policy with the Optimal Admission-Control Policy

as a percentage of the maximum profit, for each value of $m$ (note that the value corresponding to $n_1 = 0$ represents the 'reject-all' policy). Also plotted on the same axis are the blocking probabilities for the two policies.

The static policies behave as one would expect. The expected profit increases as more and more Class-1 customers are allowed to enter the system, up to a threshold. Once we reach $n_1 = 6$, accepting any more Class-1 customers actually decreases the expected profit. Hence, if we were limited to a static policy, we would pick this as the best policy, with a corresponding blocking probability equal to 16.92%.

The optimal switching curve solution obviously performs better than any of the static policies. The expected profit at the optimal solution is over 5% higher, with a blocking probability lower by more than 1 percentage point. Hence, we are performing much better than the best static policy, both in terms of the revenue, as well as in terms of the performance metric.

This experiment corroborates our method, and shows that much better results can be obtained by using the switching curve policy than a static policy. Since the methodology we proposed is also much faster, it can be used to solve large problems, wherein the number of states can be prohibitively large to solve using the value-iteration method. In order to demonstrate that this methodology can be used to solve a problem with a large number of states, we consider a problem identical to the one we just solved, except with a bigger server capacity. We would let the number of possible states be of the order of $10,000$.

Consider a server with maximum bandwidth of $25,600$ kbps. With each Class-1 request still requiring 256 kbps, maximum number of Class-1 requests that can be server concurrently is 100. Assuming that rest of the parameters are exactly the same as in the previous example, the optimal admission-control policy is obtained as shown in Fig. 8.

We see that, again, the optimal policy performs much better than the best static policy. The improvement in the revenue is about 9.2% with a reduction in blocking probability of about 2 percentage points.

## II.7. Conclusions

With different types of requests arriving in a Web-server farm, resource allocation for these classes becomes very important in order to operate efficiently. It is known that simply allocating a fixed amount of bandwidth to different classes can be very inefficient. Static allocation policies, under which the remaining bandwidth from

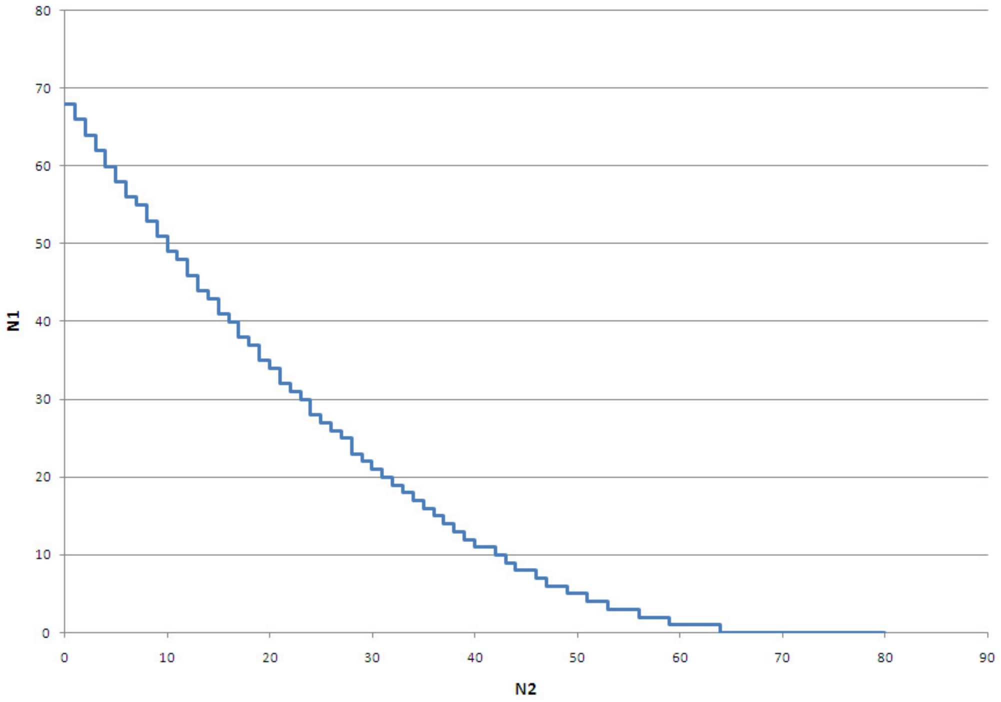

Fig. 8. Optimal Admission-Control Policy for the Larger Problem

serving one kind of requests is used to serve the other kind would be better, but would still fare worse than the optimal solution. For the Web server to operate optimally, one needs to use a dynamic resource allocation, based not only on the load on the server of different types of requests, but also on the arrival and service distributions, rewards, and holding costs.

We established some structural properties, such as concavity and submodularity of the value function for the given problem. Using these properties, we were able to analytically obtain the optimal admission-control policy. This decision policy takes the form of a switching curve, as shown in Fig. 3. Because we know the structure of the optimal admission-control policy, we do not need to use the value-iteration algorithm to obtain the optimal policy, but can use simple and clever heuristic methods, such as a neighborhood search algorithm, to obtain the optimal solution. This would provide not only the ability to obtain the optimal solution much faster, one would also be able to solve problems with a large state-space.

We also computed the performance metric, the blocking probability, of the streaming class requests under the optimal policy. It was observed that if the optimal policy of the form of a switching curve is used, not only would we be improving the resource utilization of the server, we would also be providing better QoS.

# CHAPTER III

# PERFORMANCE ANALYSIS OF ELASTIC TRAFFIC USING STOCHASTIC FLUID-FLOW MODELS

## III.1. Background and Introduction

As described in Section I.2, we have elastic requests coming into the system which will be served by transmitting data through the Web-server farm network. In this chapter, we consider a small piece of this problem: we look at a single server that is transmitting data. This server would have a fixed link capacity to transmit the data through. We are interested in characterizing this traffic flow and obtaining the performance metrics at the server level.

Research area of computing the performance measures in case of high-speed packet computer-communication networks received a lot of attention during the 80's and 90's [30]. Despite the significant amount of research addressing Internet traffic models, there is not yet widespread agreement about the characteristics of backbone Internet traffic [31]. With networks become more complex and QoS becoming more challenging, it is more important than ever to be able to obtain various performance metrics. Bounds on these performance measures have been computed using various techniques by researchers. These approximations and bounds are useful in different scenarios, such as design or control of a communication system, resource allocation, or to compute the projected QoS.

However, only a few of the design and control strategies based on these performance analyses have been implemented in practice. One of the reasons is that the theoretical networking results are largely probabilistic and are based on assumed models of traffic. Therefore, there is a divide in the literature in terms of theoretical and practical networking. In this chapter, we address this shortcoming by using measured data and to fit appropriate models for performance analysis.

One of the main stumbling blocks for using measurement data for performance analysis is the fact that traffic patterns are inherently bursty, resulting in high correlations and large variations. It is well documented and accepted that traffic in computer-communication networks exhibit long-range dependence (LRD). Whether one looks at connection-level, request-level, or packet-level traffic, the arrivals are not independent. They show a high degree of correlation over a very large number of arrivals, as shown by Sahinoglu and Tekinay [32]. Statistical analyses of high time-resolution traffic measurements have also provided convincing evidence that actual traffic data represents self-similarity, or fractal characteristics, as reported by several researchers, including Leland et al. [33], Paxson and Floyd [34], and Willinger et al. [35]. For this reason, various types of “self-similar” models to characterize this LRD traffic have been used.

While these self-similar models do a fantastic job of characterizing traffic, they are typically difficult to use in performance analysis. For example, when self-similar traffic is fed into a queue to be processed, it is usually, except for some special cases, not easy to analytically obtain the required performance metrics of the system as closed-form algebraic expressions.

However, as we move into the next phase of high-speed network services, in which users demand excellent QoS for their applications, it is critical to be able to characterize, predict, and obtain system performance metrics in computer communication networks. Several researchers, such as Frost and Melamad [36] and Jagerman et al. [37], have tried to obtain these system parameters using queueing theory and Markovian assumptions about traffic arrival patterns and exponential assumptions about resource-holding requirements. However, as mentioned earlier, modern high-speed networks result in packet traffic that is generally much more complex and bursty. Moreover, these approximations tend to give performance measures that only underestimate the expected waiting time, workload, etc.

Hence, on the one hand self-similar models cannot easily and directly be used to obtain performance metrics, while on the other hand, queueing approximations based on $G/G/1$ models would result in severe underestimation of the performance metrics. Several other stochastic models have come up over the years, such as fluid-flow models [38], the Markov modulated Poisson process [39], the batched Markovian arrival process (BMAP) models [40], etc. These models have maintained the analytical tractability of queueing and performance problems. However, the resulting models have hardly been judged by how well they fit actual traffic data in a statistical sense [41].

Therefore we need a methodology that not only can characterize the network traffic nicely but at the same time can be useful in acquiring performance metrics. In this research, we propose to use the stochastic fluid-flow models to obtain the performance measures, while using the parameters for the fluid flow obtained from trace data collected from different Web-servers across the United States.

The motivation for considering fluid models, especially ON-OFF sources, is because frequently traffic arrives in bursts, causing dependence as well as poor estimates of performance. Since the seminal article by Anick, Mitra and Sondhi [38], there have been several papers that obtain performance metrics for queues with first- (and/or second-) order fluid traffic. In this chapter, we propose to leverage upon these results to address the question of whether fluid-flow models adequately capture high-speed network traffic characteristics and effectively approximate performance measures seen in real traces. In the process, we will take real trace data, which happen to be discrete in nature, and fit fluid-flow models for them. This will require obtaining the parameters for the distributions used in the fluid-flow model using the trace data.

The remainder of the chapter is organized as follows. In Section III.2, we present a brief review of the related work done by researchers in analysis of fluid models and other methods of obtaining performance measures in network traffic. In order to employ different methods to perform the analysis, we have used a number of data sets. Section III.3 describes these data sets by providing details about the data contained and the source of the data. In Section III.4, we present the methodologies that have been adapted to model and analyze the system. These methodologies have then been used on the traces, and the process of computation of numerical results is presented in Section III.5. The system has also been modeled as a Markov-modulated fluid source. This analysis is presented in Section III.6. The numerical results of all the different analysis methods are presented in Section III.7. Finally, we present the findings in Section III.8.

## III.2. Related Literature: Network Traffic and Modeling

The first step in the performance analysis of communication networks is modeling traffic. Self-similarity concepts have been used to model communication systems ever since the seminal article by Mandelbrot [42]. Since then, researchers have been debating the issue of traditional mathematical modeling (based on Markov processes) vs. unconventional fractal modeling (based on LRD), even in different areas of application (see Klemeš [43], Liebovitch [44]).

The property of self-similarity, or LRD, in LAN traffic was first reported by Leland et al. [33]. Since then, researchers have discovered that it is also exhibited in other kinds of network traffic, such as WWW traffic [45], Wide Area Networks [46, 34], and TCP and UDP traffic [47].

It has been found that computing performance measures of the traffic, in case self-similarity is exhibited, is very difficult, because it makes the problem analytically intractable. Moreover, using simpler models, such as the Poisson approximation, has been shown to be inadequate [48]. These authors have demonstrated, by means of synthesized traffic from a Poisson model, a fractal model, and the Internet traffic, at different orders of magnitude, that fractal models are required in order to be able to characterize the traffic appropriately. Gelenbe et al. [49] have also shown that loss probabilities obtained by simulating the Internet traffic using first-order renewal approximation (Poisson), second-order renewal approximation (Pareto), and actual bursty traffic, yield very different results.

The importance or irrelevance of capturing the self-similarity has been a debated topic, and Sahinoglu and Tekinay [32] have provided a list of several studies. They mention that as an effect of self-similarity, the buffer sizes needed at the switches and multiplexers are larger than those predicted by traditional queueing analysis and simulations. They also go on to say that self-similarity introduces new complexities into optimization of network performance and makes the task of providing QoS, together with high utilization, difficult.

Other researchers, such as Partridge [50] and Ramaswami [51] have also demonstrated that simple queueing models that do not capture the self-similarity are far from adequate in representing the network traffic, and hence they are not nearly good enough to obtain the performance measures.

This has led to researchers coming up with various methods and approximations in order to evaluate performance measures of systems that exhibit the self-similarity characteristic. A bibliographical guide to modeling and analysis of self-similar traffic has been presented by Willenger et al. [30]. It lists most of the network traffic studies of the time in the areas of data analysis, statistical inference, mathematical modeling, queueing, and performance analysis.

An important step in obtaining the performance measures is to use an appropriate methodology. The most common methodology to model the network traffic is to use a heavy-tailed distribution, such as the Pareto distribution, and to deploy the Hurst parameter, as demonstrated by Addie et al. [52] and Willinger and Paxson [48].

Other methodologies applied to related systems can also be seen in the literature. Gautam and Seshadri [53] have concentrated on models based on queueing network analysis to obtain the performance measures of an e-commerce system. They have demonstrated the impact of self-similarity on performance degradation and have shown that it is possible to come up with reasonable approximations for waiting time in the system by decomposing the traffic into bursty and non-bursty components. In a later work by the same authors [54], the potentially high impact of self-similarity has been clearly demonstrated, and it has been shown that the traditional techniques are inadequate for predicting the network performance.

Another concept, that of effective bandwidth, has also been well established to satisfy the QoS requirements in the case of queueing models [55, 56, 57]. This methodology to analyze the buffer content process is based upon the exponential approximation of the tail probabilities. Although this technique is widely used for large buffer sizes, it may not be appropriate for low buffer sizes. Researchers have modified this methodology to redress the shortcomings, including Elwalid et al. [58] and Elwalid and Mitra [59].

Other approaches have been developed in order to avoid the approximations of the simple queueing models, such as deriving upper and lower bounds for the tail of buffer content process in steady-state with a Markov additive input in the stationary regime in the $G/G/1$ queue [60, 61]. Fluid-flow models have also been used to characterize and analyze the communications systems.

Fluid models with exponential ON and OFF times are intensively studied, starting with the pioneering work by Anick et al. [38]. Elwalid and Mitra [62] have given models and analytical techniques for a composite system of access regulation that uses the Asynchronous Transfer Mode (ATM), using a Markov-modulated fluid source, to allow for the bursty characteristic to be modeled accurately. Yang and Tsang [63] have proposed an approach for estimating the cell loss probability in an ATM multiplexer loaded with ON-OFF sources, using Markov-modulated deterministic process (MMDP) to approximate the arrival process and use queueing theory.

Other works dealing with the fluid-flow models include that of Liu et al. [64], who have developed a framework for computing upper and lower bounds on backlog, queue length, and response time, of an exponential form, for a large class of single-resource systems with Markov additive inputs. They have presented the bounds in the context of queueing theory and have also presented the numerical comparison with other bounds. Aalto [65] has looked at storage models where input rate and demand are modulated by a Markov jump process, such as a multiplexer loaded by exponential ON-OFF sources. They have shown that the output process is modulated by another Markov jump process, which turns out to be a modification of the $G/G/1$ process.

Palmowski and Rolski [66] have developed exponential bounds for the distribution of buffer-content process, in the case of simple fluid models, whose input traffic is modulated by a continuous time Markov chain (CTMC). The same authors then studied the fluid models with general ON times and OFF times and derived the exponential upper and lower bounds for the tail of the steady-state distribution of the buffer content [67]. Later, Gautam et al. [68] generalized the results in Palmowski and Rolski [67] to obtain exponential bounds for a large class of single-resource system fed by multiplexing semi-Markov processes in continuous time and a more general input case. They have considered an infinite sized buffer fluid model with a constant output capacity buffer.

Although several researchers have alluded to the fact that fluid models can be used to characterize bursty traffic, to the best of our knowledge no one has explicitly taken traces, modeled them using fluids, and compared the accuracy of performance predictions. For that we use trace data, collected from Web-servers, to obtain the performance measures of a buffered queue, which we describe next.

## III.3. Trace Data

In this research, we have used data collected from Web-servers for our experimentation. For the purpose of the analysis, we have used seven different data sets. The raw data obtained are in the form of a trace, which includes the arrival times and the size of the file requested from the server. The traces have been obtained from the IRCache Web site (*http://www.ircache.net*), a Web caching project originally funded by the National Science Foundation (grants NCR-9616602 and NCR-9521745), and the National Laboratory for Applied Network Research. They provide sanitized cache access logs for academic and research purposes. These traces are essentially log files of a large number of requests collected from different servers from across the United States. For the traces used in the analysis, location of the Web-server for the traces, along with some parameters defining the trace capture, are given in Table I.

We assign numbers to these traces to facilitate easier reference to a particular dataset. Note that Trace 4, which was obtained from Paxson and Floyd [34], and not from the IRCache Web site, is over a decade older than the rest.

These traces contained information about the inter-arrival time and the file sizes of each of the requests. By looking at the arrival times in the data, it was found that

**TABLE I** Web-Server Location of the Trace Data

| Trace | Web-server Location | Log Time | Duration (Hrs) |
|---|---|---|---|
| 1 | Pittsburgh, PA | Oct '04 | 09.6 |
| 2 | New York, NY | Oct '04 | 12.2 |
| 3 | Urbana-Champaign, IL | Feb '05 | 08.6 |
| 4 | Berkeley, CA | Jan '94 | 01.0 |
| 5 | Research Triangle Park, NC | May '06 | 12.6 |
| 6 | San Jose, CA | May '06 | 10.1 |
| 7 | Palo Alto, California | Jun '06 | 07.8 |

all of the traces could be modeled using piece-wise constant arrival rates. That is, the mean interarrival rate was independent of time in each such piece. As an example, consider Trace 5 shown in Fig. 9.

The data in this trace can be approximated using three piecewise constant arrival rates. Zooming in on the near-constant inter-arrival rate between arrivals 400,000 to 1,000,000, we get Fig. 10. This data can suitably be approximated with a mean interarrival rate that is independent of time.

It was also observed that the number of arrivals during a piecewise constant phase is large enough to warrant a steady-state analysis (or quasi-steady-state analysis) rather than a transient analysis. With this in mind, we will, for the rest of the chapter, consider constant mean inter-arrival rates and only perform steady-state analysis. Note that the arrival process is still stochastic in nature. The data sets have been summarized in Table II, which provides the mean and standard deviation of the inter-arrival times and the packet sizes, along with total number of arrivals in

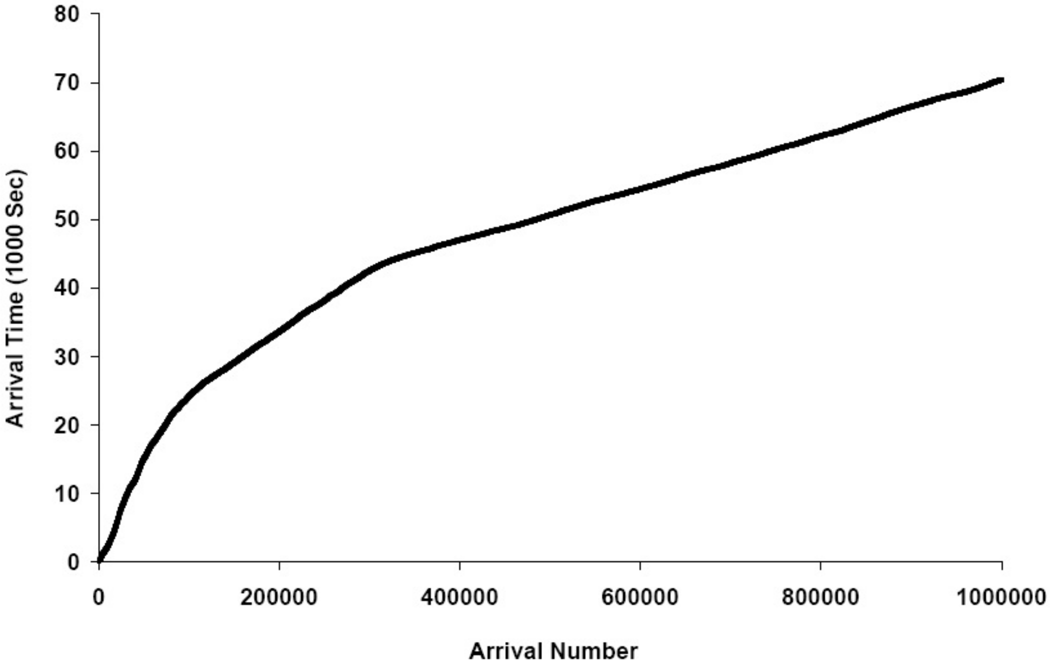

Fig. 9. Piece-Wise Linear Arrival Times of Trace 5

the trace.

Once we have selected the linear piece from the trace data, we perform the analysis as described in the next section. The numerical results obtained from such an analysis are presented in Section III.5.

## III.4. Analysis

Consider an abstract scenario of a buffer in which information arrives, waits, gets processed, and then, transmitted. Now, if processing starts only after the entire file has arrived, the workload profile in the system will look as shown in Fig. 11. However, if the processing starts as soon as information starts arriving, the workload in the system could be represented by the figure on page 49.

The objective in such scenarios is to obtain performance measures, such as the

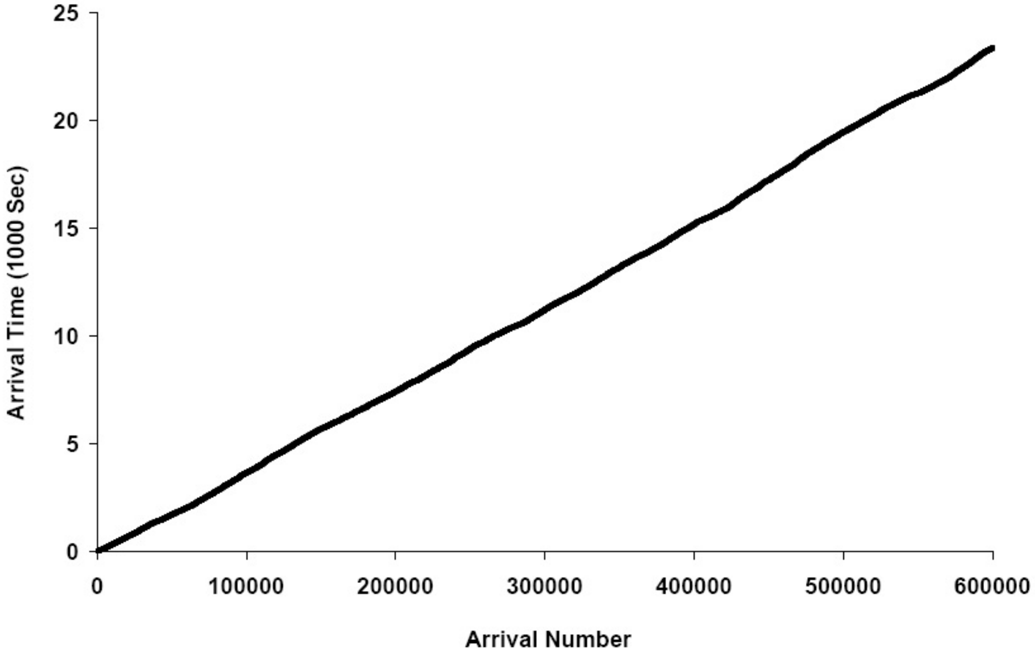

Fig. 10. Time-Independent Inter-Arrival Times Used for Analysis of Trace 5

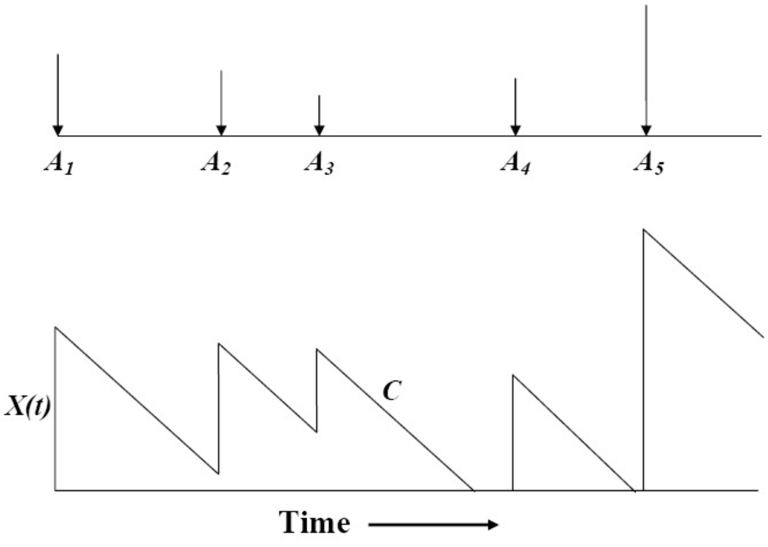

Fig. 11. Workload in the System in the Discrete Case

**TABLE II** Summary of Trace Data

| Trace | Inter-Arrival Time (sec) | | Packet Size (MB) | | Total |
|---|---|---|---|---|---|
| No. | Mean | Std. Dev. | Mean | Std. Dev. | Requests |
| 1 | 0.139 | 0.216 | 16.048 | 76.125 | 250,000 |
| 2 | 0.155 | 0.206 | 14.287 | 151.86 | 290,000 |
| 3 | 0.062 | 0.090 | 5.698 | 160.556 | 500,000 |
| 4 | 0.004 | 0.006 | 3.412 | 7.463 | 870,000 |
| 5 | 0.039 | 0.063 | 9.490 | 299.840 | 600,000 |
| 6 | 0.122 | 0.144 | 20.844 | 410.130 | 300,000 |
| 7 | 0.440 | 0.651 | 11.215 | 895.534 | 65,000 |

mean waiting time in the queue. There are a number of analytical and numerical approaches that can be used to compute performance measures such as average waiting time, when arrival and file size characteristics are given. We will analyze each of the traces at three different traffic intensity values: 60%, 80%, and 90%, in order to observe the effect of changing traffic intensity on performance measures. In this section, we will define the problem and then describe the methodologies used in order to compute the performance measure.

### III.4.1. Problem Description

In the system considered, when a request comes in for a particular file of a given size, the server processes or transfers that file. Since the processing speed of the server is assumed to be constant (this we believe is reasonable for the objective of this study), the processing time for a file is linearly proportional to its size. If these, potentially large files, are downloaded one after the other and they pass through a buffer with output capacity $C$ (Fig. 12), then we are interested in obtaining the average time for a file to pass through the buffer, or in other words, the mean waiting time of this response process. The maximum rate at which this data is flowing into the buffer can be assumed to be $R$, the channel capacity.

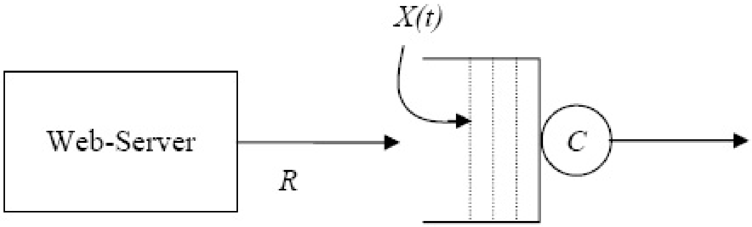

Fig. 12. Data Arriving, Being Processed and Leaving a Buffer

If we assume that the processing starts after the entire file has arrived to the buffer, the workload profile will look as shown in Fig. 11. The jump in the figure represents the discrete time point at which the file has completed arriving in the system. Now, consider the system in Fig. 11 with processing speed of $C$ bytes/s. Let the buffer capacity be infinite, and let $A_i$ be the time of arrival of the $i^{th}$ request, of size (in bytes) $S_i$. Furthermore, let $X(t)$ be the amount of workload (in bytes) in the system at time $t$. This workload in the system is shown in Fig. 11. The timeline in the top part of the figure shows the time points of the discrete arrivals, with the length of arrows being proportional to the size of request. The slope of each of the lines reducing the workload in the system is equal to the processing rate, $C$. The workload at time $t$ in such a system can be written as the following expression:

$$X(t) = \left( \sum_{i:A_i \in (t_0, t)} S_i \right) - (t - t_0)C \ ,$$

where $t_0$ is the last time the system was empty and there was an arrival. In the next section we will provide analytical expressions for the expected waiting time.

However, instead of the case considered in Fig. 11, it can be argued that the file starts being transmitted as soon as it arrives. In this case, the workload profile will look similar to the one shown in Fig. 13. Here, we can assume the inflow to be in

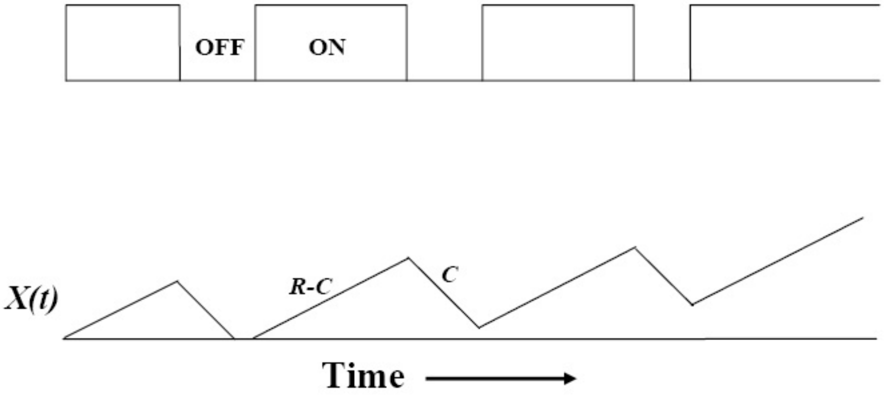

Fig. 13. Workload in the System in the Fluid Arrivals Case

the form of an ON-OFF source feeding a buffer with a constant output rate $C$, as shown in Fig. 14. Each ON may represent one or more files flowing into the system. Let the channel capacity of the input source be $R$ bytes/s. That is, the fluid enters the system either at the rate of $R$ bytes/s, or not at all. In this case, the workload ($X(t)$) in the system will look similar to the plot shown in Fig. 13. Here, the slope of the lines increasing the workload in the system is $(R - C)$, and the slope of the lines reducing the workload is the same as before, $C$. Let $t_0$ be the last time the system

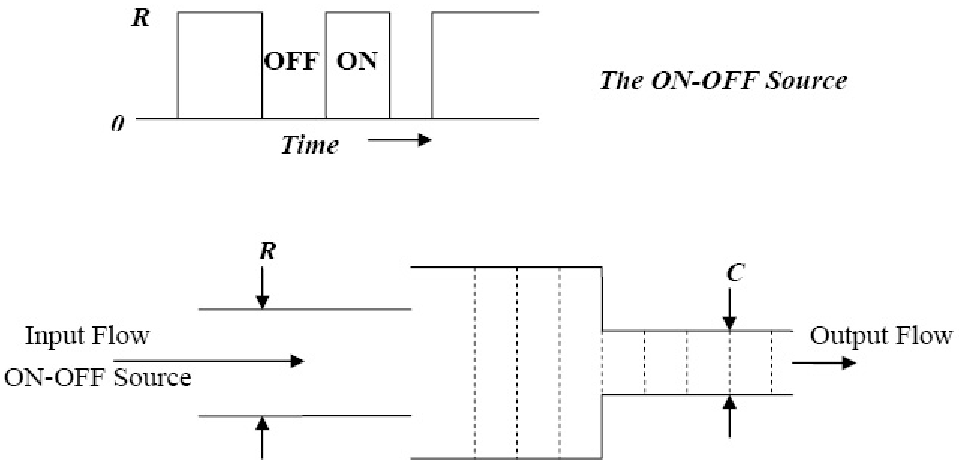

Fig. 14. A Buffer Representing the Inflow and Outflow of Fluid in the system

was empty and an ON time started. Then, if $U_i$ and $D_i$ are the ON time and the OFF time of the $i$th ON and OFF, respectively, and $U_N$ and $D_N$ are the last time an ON and OFF period started in the time period $(t_0, t)$, workload in the system is given as:

$$
\begin{aligned}
X(t) &= \sum_{i:D_i \in (t_0,t)} R(D_i - U_i) - (t - t_0)C \\
&\qquad\qquad\qquad\qquad \text{if } (U_N < D_N) \\
&= \sum_{i:D_i \in (t_0,t)} R(D_i - U_i) + R(t - U_N) - (t - t_0)C \\
&\qquad\qquad\qquad\qquad \text{if } (U_N > D_N) \ .
\end{aligned}
$$

That is, for each OFF time in the time period, we sum the amount of fluid accumulated in the last ON ($R(D_i - U_i)$) and subtract the total amount of fluid that left the system ($(t - t_0)C$). In case the system is in ON state at time $t$ (i.e., $U_N > D_N$), we add the amount of fluid during the last ON time ($R(t - U_N)$). The amount of fluid in the system represents the remaining workload. We use the methodology developed in the area of fluid-flows to analyze this system and obtain the expected waiting time. The waiting time so obtained will be independent of the scheduling principle (FCFS, LIFO etc.).

### III.4.2. Software Simulation

Before describing the methodologies and computing the performance measures, we describe the basis against which these obtained performance measures will be compared. In order to compute the actual waiting time in the queue for the trace data, we will use a software simulation. Here, we use the actual data in the trace and pass the files through a buffer to obtain the mean waiting time.

The traces have the arrival times and the file size for each of the requests processed. Using this data, a simulation can be performed, in which a request comes to the system at its specified arrival time, joins the queue for the server, gets processed in a time proportional to its file size, and leaves the system. We assume a first-come-first-serve scheduling policy for the server, which is a work-conserving policy, and the results so obtained should hold for any other work-conserving scheduling discipline as well. This method of simulating the trace will give us the actual waiting time a request would have to experience in the system. The average number of bytes in the system in the long run can be computed as the average workload or the buffer content in the system. To keep consistency with the other methods of analysis, the processing rate for the data should be chosen in such a way so as to obtain the appropriate traffic intensity. This can be done by choosing an appropriate processing rate, $C$, such that given the mean inter-arrival time and the mean file size, we have the desired traffic intensity. From now on, we will refer to these results obtained from the simulation as the actual waiting times.

Now that we have the actual waiting times from the data, the next steps are to analyze the system using discrete and fluid-flow methodologies to compare the results.

### III.4.3. $G/G/1$ Approximation

One of the techniques to obtain long-run expected waiting time in a queue is to use $G/G/1$ approximation from queueing theory. In this scenario, we assume that processing of the files starts only after the whole file has arrived, or that the file arrives as a bulk at a discrete time point. The workload in such a system would be as shown in Fig. 11. Renewal arrival times and independent and identically distributed (iid) service times are assumed in this approximation. Hence, This assumption is not congruent with the LRD property of the inter-arrival times.

Using the data from the Web-trace, we can compute the sample mean and sample variance for the inter-arrival times and the service times, which would be the estimators of mean and variance. Now, let $\rho$ be the traffic intensity in the system, which is a measure of the average occupancy of the server and can be computed as the ratio of mean arrival rate and the mean service rate. Also, let $C_a$ and $C_s$ be the coefficient of variation (ratio of standard deviation to mean) of the inter-arrival times and the service times, respectively. Then, the expected waiting time for a file in the queue, using the $G/G/1$ approximation, is given by (from [69] and using Little's law [70]):

$$W = \frac{1}{\lambda}\left[\frac{\rho^2(1+C_s^2)}{2-\rho+\rho^2 C_s^2}\frac{\rho(2-\rho)C_a^2+\rho^2 C_s^2}{2(1-\rho)}\right] \tag{3.1}$$

where $1/\lambda$ is the mean inter-arrival time.

### III.4.4. Fluid-Flow Model

The information flow into the system has been considered as discrete in the case of the $G/G/1$ approach. In this section, we treat the arrivals as fluid. As described earlier, we will model this system as an ON-OFF source, with input link speed as $R$ bytes/s, and with a processing speed of $C$ bytes/s. It should be noted that a single ON time may represent a confluence of more than one file. Processing of data starts as soon as the file starts arriving (Fig. 13).

Let the mean ON time be $\tau_U$ and the mean OFF time be $\tau_D$. Then, the sever utilization in this kind of an ON-OFF source can be obtained as $R\tau_U/\{C(\tau_U+\tau_D)\}$. Assume that the ON times and the OFF times are independent of each other, and that each is identically distributed. Let $U(\cdot)$ be the cumulative density function (CDF) of the ON times, and let $D(\cdot)$ be the CDF of the OFF times. Given a distribution, these CDFs can be obtained from the data that will be described in Section III.5. Now, the Laplace-Stieltjes transform (LST) of a non-negative function $F(x)$ is given by:

$$\tilde{F}(s) = \int_{x=0}^{\infty} e^{-sx} dF(x). \tag{3.2}$$

Using the above equation, the LSTs of ON times and OFF times can be computed as $\tilde{U}(\cdot)$ and $\tilde{D}(\cdot)$, respectively.

Using the theory of large deviations, we can obtain the distribution of the buffer contents in the following manner (details in [68]). Let $X(t)$ be the amount of fluid (in bytes) in a buffer of infinite capacity, at time $t$. Now, in order to compute the expected waiting time in the queue, we first need to evaluate the amount of fluid in the buffer, which is essentially the remaining workload in the system. Hence, if we can obtain expressions to evaluate this quantity, we will be able to characterize the expected waiting time. To obtain the limiting distribution of $X(t)$, define $\eta$ as the smallest real-positive solution to

$$\tilde{U}(-\eta(R-C))\ \tilde{D}(\eta C)\ =\ 1. \tag{3.3}$$

Then, we have bounds for the distribution of $X(t)$ in the limit as $t \to \infty$ (for any $x \in [0, \infty)$) as

$$K_* e^{-\eta x} \leq \lim_{t\to\infty} \Pr\{X(t) > x\} \leq K^* e^{-\eta x}, \tag{3.4}$$

where

$$K^* = \frac{\left(\tilde{U}(-\eta(R-C)) - 1\right) R}{(\tau_U + \tau_D)\, C(R-C)\eta \inf_x \left\{ \frac{\int_x^\infty e^{\eta(R-C)(y-x)} dU(y)}{1-U(x)} \right\}}$$

and

$$K_* = \frac{\left(\tilde{U}(-\eta(R-C)) - 1\right) R}{(\tau_U + \tau_D)\, C(R-C)\eta \sup_x \left\{ \frac{\int_x^\infty e^{\eta(R-C)(y-x)} dU(y)}{1-U(x)} \right\}}.$$

In order to evaluate these bounds, the first step is to compute the CDFs of the ON times and that of the OFF times. Since we have the ON times and OFF times from the trace data, we will have to fit a distribution here by computing its parameters. Then, we can use the technique mentioned above to compute the bounds.

If we assume that an arriving file sees a time-averaged amount of fluid in the system, we can compute the bounds on the expected waiting time for the arriving file as:

$$K_*/\eta C \leq W \leq K^*/\eta C. \tag{3.5}$$

It should be noted that the bounds presented here are valid for traffic that is fluid and satisfies the underlying distributions. That is, these are bounds on expected waiting times of the fluid approximation of actual data. In our analysis, iid distributions have been fitted on trace data and this data is approximated as an ON-OFF source. Hence, in our analysis, these bounds may be violated and might as well be treated as upper-level and lower-level approximations for the expected waiting time of the actual data.

One of the open research questions is when an ON-OFF source with ON time CDF $U(\cdot)$ and OFF time CDF $D(\cdot)$ inputs fluid into an infinite buffer at rate $R$ and with an output rate $C$ is: whether $\lim_{t\to\infty} \Pr\{X(t) > x\}$ in Equation (3.4) is closer to the upper bound, lower bound, or somewhere in between. We aim to address that problem as a byproduct of this work.

## III.5. Computing the Performance Measure

In this section, we present the process to take the trace data mentioned in Section III.3 and compute the waiting times, which is our performance metric of choice, using the methodologies described in Section III.4. It should be noted that the calculations are valid for any work-conserving system, such as first-come-first-serve, random-order-of-processing, or processor sharing. In the case of the fluid model, we are essentially looking only at the work remaining in the system. Hence, the scheduling discipline becomes irrelevant, as long as it is a work-conserving system. Once we have the numerical results, we would be able to compare the different methods with the actual waiting times and against each other.

In order to compute the performance measures using any of the methodologies, we first need to ascertain the values of the inflow rate, $R$, and the processing rate, $C$, from the trace data so as to have a desired traffic intensity of $\rho$. This is done by setting the processing rate as $C = \lambda/(\rho \cdot \mu)$, where $\lambda$ is the mean inter-arrival rate and $1/\mu$ is the mean file size. Now, to obtain the value of $R$, we compare the workload profile in the system for different ratios $R/C$ to the workload profile in the discrete case. It is clear that as $R/C$ increases, the system will behave more and more like a discrete system. For the analysis, we want to choose a low $R/C$, while not causing significant deviation in the workload profile. It was found that the value $R/C = 1.5$ serves very well for all the traces, and hence, $R = 1.5C$ is used. The workload profile comparison for one such case is provided in Fig. 15.

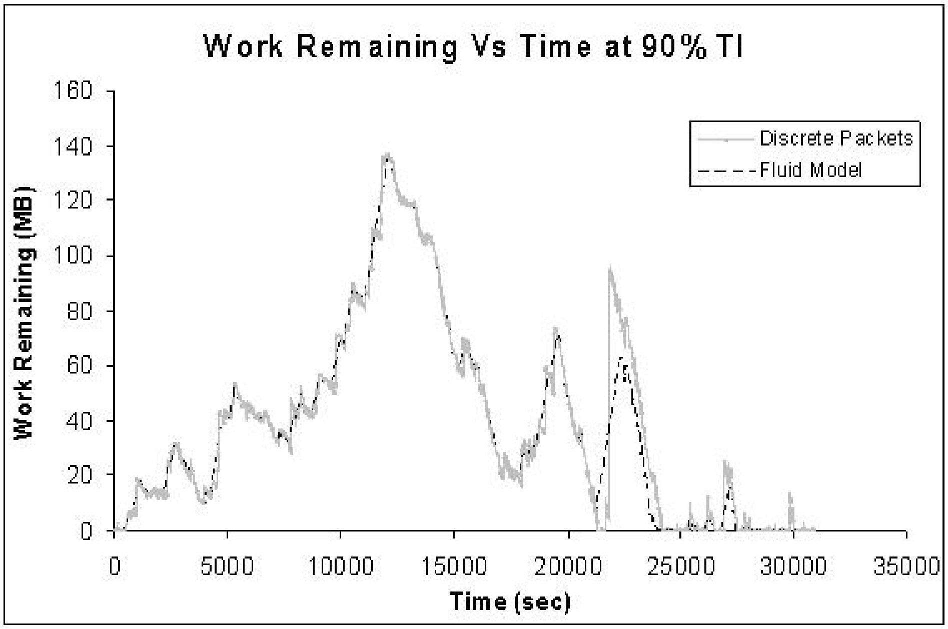

Fig. 15. Comparison of Work Remaining in the System in Discrete and Fluid Case ($R/C = 1.5$) for Trace 7

Using the values of $R$ and $C$ obtained for different traces at different traffic intensities, we can compute the $G/G/1$ approximation for the expected waiting time in the queue, using Equation (3.1).

To compute the performance measure of the system using the fluid-flow model, we first obtain the ON time and OFF times from the data. A single ON could consist of more than one files and can represent a burst of data. Distributions now need to be fitted to the ON times and the OFF times. We can choose from an array of distributions to model these, such as Pareto, exponential, etc. As demonstrated by Feldmann and Whitt [71], it is possible to use a phase-type distribution, such as the hyper-exponential distribution, to model such a system and analyze the performance model. They also proved that completely monotone probability density functions (PDF's), which represent the case many times for long-tailed distributions, can be approximated arbitrarily closely by hyper-exponential PDF's.

In this work, based on the performance shown in fitting the data, we chose two-phase hyper-exponential (2P-HE) distribution to model the ON times of the traffic. The other advantage of using the hyper-exponential distribution over other distributions, such as Pareto, is that we can perform exact analysis. OFF times of the traffic are approximated using the exponential distribution. There are various ways to fit a distribution to a given data set, and one of the easiest and most popular methods is by matching moments. The number of unknown parameters in the specified distribution defines the number of moments that need to be equated.

Using this methodology, we will be able to obtain the parameters for the 2P-HE distribution for the ON times and exponential distribution for the OFF times. The CDF of a 2P-HE distribution (ON times) can be represented as:

$$U(x) = 1 - pe^{-\alpha_1 x} - (1-p)e^{-\alpha_2 x} \quad x \geq 0 \ ,$$

with parameters $p$, $\alpha_1$ and $\alpha_2$. The CDF of the exponential distribution (OFF times) is given as:

$$D(x) = 1 - e^{-\lambda x} \quad x \geq 0.$$

Once we have obtained the CDFs of both the ON times and the OFF times, we can compute the bounds on the expected waiting time in the system using the methodology described in Section III.4.4. This would involve computing the value of $\eta$ using Equation (3.3), and then using this value of $\eta$ in Equation (3.4), to obtain the bounds on the desired performance measure. The details of performing this analysis and a numerical example for one of the traces is provided in Section III.5.1.

Exponential distribution does not approximate the ON times very well, and hence, if used, it could lead to error in the estimation of the performance measure. However, one great advantage of using an exponential distribution for the approximation is that we only need to obtain one parameter for the CDF, which can easily be computed by comparing the mean to the first moment. Moreover, if we use exponential distribution to approximate both the ON times and the OFF times, the bounds given in Equation (3.4) converge to

$$K_* = K^* = \frac{R\tau_U}{C(\tau_U + \tau_D)}. \tag{3.6}$$

Hence, rather than getting bounds as in the case of the hyper-exponential distribution, we obtain the approximation for expected waiting time as $W = K^*/\eta C$. Therefore, we will evaluate this approximation as well and observe how far off we would be in terms of the performance measures by using this distribution.

We have approximated both the ON times and the OFF times with exponential family distributions. This gives us an additional option of obtaining the expected waiting time in the system using exact analysis, by considering the arrival process modulated by a CTMC. The details of this analysis are provided in Section III.6.

### III.5.1. Computing the Fluid-Flow Model Bounds

Here, we will demonstrate how to compute the parameters for the fluid-flow model, by using Trace 3 as an example. The computations are for traffic intensity of 80%.

In order to obtain these parameters for the ON times, we evaluate the first three moments of the 2P-HE distribution analytically and compare them with the moments obtained from the actual data. Using the CDF, we can compute its three moments

$(m_1, m_2, m_3)$ as:

$$m_i = \frac{i!\, p}{\alpha_1^i} + \frac{i!(1-p)}{\alpha_2^i}.$$

The three moments obtained from the data were found to be: 0.07614, 9.2097 and 5110.5980. Therefore $p = 0.999867$, $\alpha_1 = 19.4990$, and $\alpha_2 = 0.0054$, and hence, we have

$$U(x) = 1 - 0.999867e^{-19.4990x} - 0.000133e^{-0.0054x} \ .$$

To obtain the parameter of the CDF of the OFF times (Exponential, CDF: $1 - e^{-\beta x}$), we just need to equate the mean OFF time obtained from the data to $1/\beta$. For this data set, we get $1/\beta = 15.0089$.

Now that we have the CDFs of both the ON times and the OFF times, the value of $\eta$ can be computed using Equation (3.3). When we use the 2P-HE and the exponential distributions for the ON times and the OFF times respectively, this turns out to be a cubic equation. One of the roots of this equation would be zero. Moreover, for this system, both of the other roots would be positive. We will pick the smaller positive root of this equation as the value of $\eta$. In this particular case, the roots of the equation were obtained as 0, 0.007528 and 23.9923. Hence, we choose $\eta = 0.007528$.

For other analyzed traces, the numerical values of $p$, $\alpha_1$, $\alpha_2$, $\beta$ and $\eta C$ for the 90% traffic intensity case are shown in Table III.

Now that we have the $\eta$ value, we need to obtain the bounds on the waiting times. Notice from Equation (3.4) that the denominator for both of the bounds requires us to compute a supremum and infemum of the same expression. However, the integral is a non-decreasing function of $x$ for the distributions we are using. The graph of this expression as a function of $x$, for the example at hand is shown in Fig. 16. Here, we can obtain the infemum at $x = 0$ and supremum at $x = \infty$ as 1.00023 and 3.2965, respectively. Because we have all the other values in the expression, we compute the

**TABLE III** Fluid Approximation Values of the Trace Data at 90% Traffic Intensity

| Trace | $p$ | $\alpha_1$ | $\alpha_2$ | $\beta$ | $\eta C$ |
|---|---|---|---|---|---|
| 1 | 0.998236 | 4.33054 | 0.015278 | 4.3345 | 0.015212 |
| 2 | 0.999713 | 5.40856 | 0.003896 | 5.8029 | 0.004200 |
| 3 | 0.999839 | 16.8072 | 0.003841 | 14.955 | 0.003459 |
| 4 | 0.999975 | 160.390 | 0.011283 | 177.05 | 0.012586 |
| 5 | 0.999884 | 28.7201 | 0.003851 | 23.079 | 0.003217 |
| 6 | 0.999674 | 10.7731 | 0.003398 | 7.9515 | 0.002682 |
| 7 | 0.999607 | 63.5095 | 0.000494 | 1.8510 | 0.000251 |

numerical values of $K_*$ and $K^*$ as 0.425 and 1.400, respectively.

Since these $K_*$ and $K^*$ values can give us the probability of the workload in the system being greater than any given value, we can obtain the bounds on the expected waiting time in the system as $K_*/\eta C$ and $K^*/\eta C$. Computing these values, in this case, we obtain $56.42 \leq W \leq 185.88$.

We notice that the $p$ values in Table III are all very close to 1. This implies that the contribution of the first term is very large as compared with the second term when evaluating the workload in the system. However, the $\alpha_1$ values are greater than the $\alpha_2$ values by orders of magnitude; thus, giving some significance to the contribution of the second term.

If we use the exponential distribution to approximate both the ON times and the OFF times, the bounds given in Equation (3.4) converge to the same value, given in Equation (3.6). For the case of Trace 3 at 80% traffic intensity, this value was computed to be 0.800, giving the expectation of the waiting time in the system to be

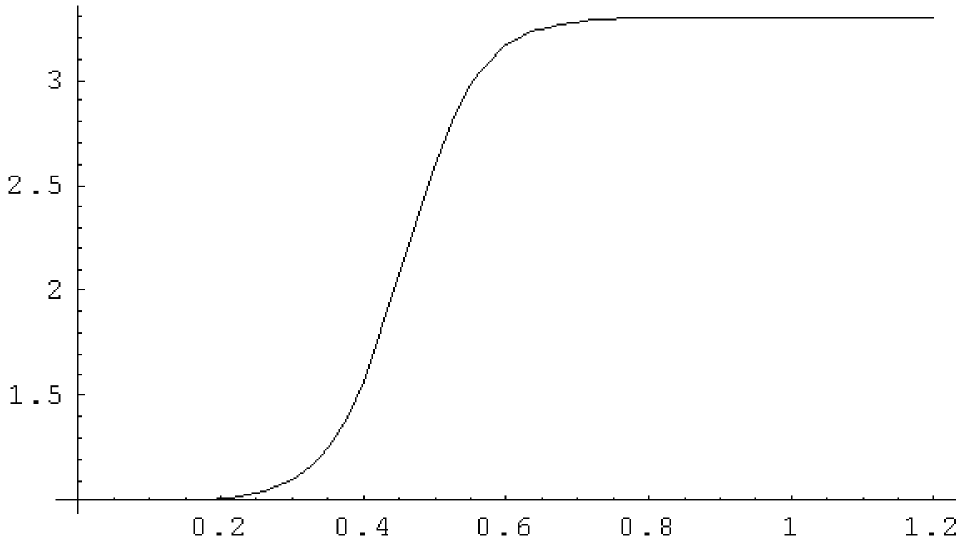

Fig. 16. Graph Representing the Function for Which Obtaining the Infemum and Supremum Is Desired

106.27.

### III.6. CTMC Analysis

We have approximated the ON times as 2P-HE distribution. A hyper-exponential ($H_k$) distribution is a mixture of $k$ exponential distributions for some $k$. That is, the CDF has the form

$$H(x) = 1 - \sum_{i=1}^{k} p_i e^{-\lambda_i x} \ , \tag{3.7}$$

where $p_i \geq 0$ for all $i$ and $p_1 + p_2 + \ldots + p_k = 1$. The OFF times have been approximated as an exponential distribution. Because the ON and OFF times are based on exponential family, we could do exact buffer content analysis using CTMC for the source (Markov modulated fluid source). In this section, we will obtain the expected waiting time in the system by performing this analysis, which is described next for a generic CTMC with finite state space.

In a Markovian fluid-flow model, the drift matrix is defined as $D = \bar{R} - CI$, where $\bar{R} = \text{diag}[r_{ii}]$ defines the rate matrix and $I$ the identity matrix. Here, $r_{ii} = r(i)$ is the rate of inflow when CTMC is in state $i$ and $C$ is the constant buffer output. For the case of exponential family of ON-OFF sources, $r(i)$ is $R$ or 0 if state $i$ corresponds to ON or OFF, respectively. A generator matrix $Q = [q_{ij}]$ defines the transition rates between different states of the system. Using the drift matrix $D$ and the infinitesimal generator matrix $Q$, we can evaluate the distribution of the limiting probability of the remaining workload in the buffer. This can be used to get the expected waiting time in the system, which can be compared with the waiting times evaluated using other methods.

In order to perform this analysis, we use the methodology initially proposed by Anick et al. [38] and then further developed by Elwalid and Mitra [62, 72]. The methodology has also been explained in [73]. Let $X(t)$ be the amount of fluid in the buffer at time $t$. We will consider an infinite capacity buffer and so, stability is guaranteed if the mean traffic arrival rate is less than the buffer output rate, $C$. Let the traffic be generated by an irreducible CTMC on state space $S = \{1, 2, \ldots, M\}$. Also, let $M_+$ be the cardinality of the set of states with positive drift (i.e. $i : r(i) - C \geq 0$), and $M_-$ be the number of states with negative drift. In Equation (3.4), we obtained bounds for $\Pr\{X(t) \leq x\}$ as $t \to \infty$. Here, for the special case of exponential family ON-OFF source, we obtain the exact expression for this limiting probability as:

$$\lim_{t \to \infty} \Pr\{X(t) \leq x\} = \sum_{i=1}^{M_+ + M_-} a_i \exp\{e_i x\} \phi_i, \tag{3.8}$$

where, $e_i$ and $\phi_i$ denote the eigenvalues and eigenvectors of the matrix $[QD^{-1}]$ respectively. Moreover,

$$\begin{aligned} Re(e_1) \leq Re(e_2) \leq \ldots \leq Re(e_{M_+}) &< Re(e_{M_+ + 1}) \\ &= 0 < Re(e_{M_+ + 2}) \leq \ldots \leq Re(e_{M_+ + M_-}). \end{aligned}$$

The only unknowns in Equation (3.8) are the $a_i$ values. These values can be computed using the boundary conditions. For a stable system with infinite buffer, boundary conditions are:

$$\begin{aligned} a_j &= 0 && \text{if } Re(e_j) > 0, \\ a_{M_+ +1} &= 1, \\ \sum_{i=1}^{M_+ +1} a_i \phi_{ij} &= 0 && \text{if } j \in S_+. \end{aligned} \tag{3.9}$$

Thus, we can evaluate $\lim_{t\to\infty} \Pr\{X(t) \le x\}$ for a given $x$. Again, assuming that an arriving file sees a time-averaged amount of fluid in the system, we can compute the expected waiting time in the system for the file as

$$W = \frac{1}{C} \int_0^\infty \left(1 - \lim_{t\to\infty} \Pr\{X(t) \le x\}\right) dx. \tag{3.10}$$

### III.6.1. Numerical Example for CTMC Analysis

The distribution of the ON times and the OFF times chosen are from the exponential family, which has allowed us to perform CTMC analysis and obtain the mean waiting time in the system. Following, we present a numerical example, using Trace 2, at 90% traffic intensity. Other traces were analyzed in a similar way.

Let the channel output rate $C$ be 1 unit. We have chosen the inflow rate to be 1.5 times the output channel capacity. Hence, the rate matrix can be given as:

$$\bar{R} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{bmatrix},$$

where the first state corresponds to the OFF state and the second and third states correspond to the two phases of the ON state. The drift matrix and the infinitesimal generator matrix for this trace can be given as:

$$D = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}, \text{ and}$$

$$Q = \begin{bmatrix} -5.8029 & 5.8013 & 0.0017 \\ 5.4086 & -5.4086 & 0 \\ 0.0039 & 0 & -0.0039 \end{bmatrix}.$$

Next, we obtain the eigenvalues ($e$) and the left-eigenvectors ($\phi_i$) of $QD^{-1}$. The following is what we obtain:

$$e = -5.0178, -0.0042, 0,$$

$$\phi_1 = \begin{bmatrix} -0.4471 & -0.8945 & 0.0003 \end{bmatrix},$$
$$\phi_2 = \begin{bmatrix} 0.5763 & 0.6184 & 0.5342 \end{bmatrix},$$
$$\phi_3 = \begin{bmatrix} -0.6547 & -0.7022 & 0.2799 \end{bmatrix}.$$

Now, we need to compute the $a_i$ values. In this particular trace, using the boundary conditions given in Equation (3.9), we have:

$$-0.8945a_1 + 0.6184a_2 - 0.7022a_3 = 0,$$
$$0.0003a_1 + 0.5342a_2 - 0.2799a_3 = 0,$$
$$-(0.6547 + 0.7022 + 0.2799)a_3 = 1.$$

Solving the above equations, the values of $a_i$ can be obtained as $a_1 = 0.2583$, $a_2 = -0.3202$, and $a_3 = -0.6110$. Thus, we have obtained all of the required parameters for the CDF, and using Equation (3.8) we obtain:

$$\lim_{t\to\infty} \Pr\{X(t) \leq x\} = 1 - 0.3464e^{-5.0178x} - 0.5536e^{-0.0042x}$$

$$\Rightarrow W = 0.3464/5.0178 + 0.5536/0.0042 = 131.88.$$

Hence, we can compute the expected workload in the system from the above CDF as 131.88, which would be equal to the expected waiting time if the processing rate is 1 unit.

## III.7. Results

We have obtained several approximations for the expected waiting time in the system in the three previous sections. In this section we compare the approximation schemes using the different traces at traffic intensities of 60%, 80%, and 90%. We first summarize the various schemes and, for quick reference, present them in Table IV, including notation used in figures as well as future tables.

Essentially, there are three types of techniques: actual waiting times observed by running traces through a simulator, discrete queueing model using $G/G/1$ results, and fluid models. The fluid models provide four different approximations. Our objective is to determine whether fluid models are suitable, and also to provide guidelines for choosing the most appropriate fluid model.

For that, we begin by summarizing the mean waiting time using the various methods in Table IV for all the traces. Obervations were recorded for all of the traces, and are presented in Fig. 17–Fig. 23.

Note that the plots corresponding to L-Bound and CTMC are running virtually on top of each other for all of the traces, making it difficult to tell them apart in some cases. The critical observation to make here is that U-Bound produces a significantly

**TABLE IV** List of Methodologies and Legend for Graphs and Tables

| Name in Fig./Table | Type | Model Input |
|---|---|---|
| Actual | Simulation | Traces fed directly |
| $G/G/1$ | Discrete | Moments of arrival and service times in Equation (3.1) |
| L-Bound | Fluid | CDF of ON and OFF times for LHS of Equation (3.5) |
| U-Bound | Fluid | CDF of ON and OFF times for RHS of Equation (3.5) |
| CTMC | Fluid | CTMC $Q$ and $R$ matrices to derive Equation (3.10) |
| Exp | Fluid | Mean ON and OFF times for Equation (3.6) |

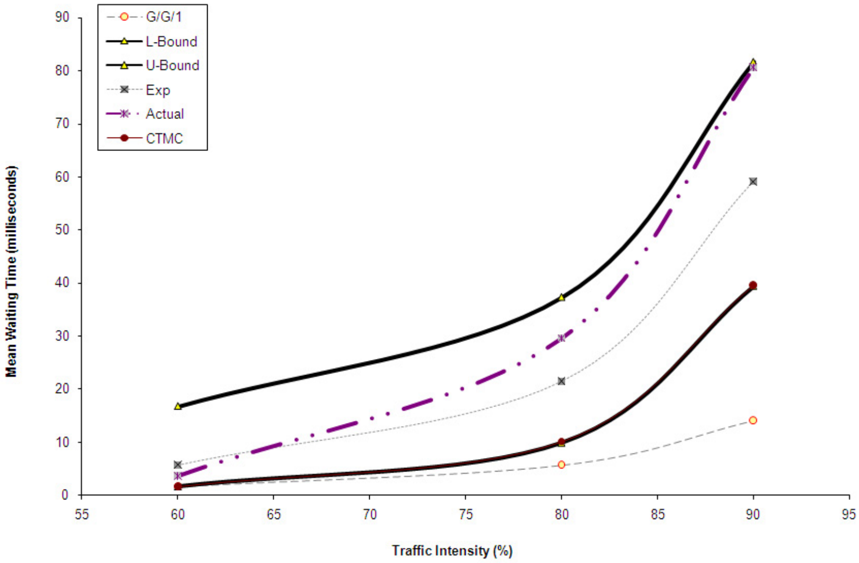

Fig. 17. Different Bounds and Approximations at Different Traffic Intensities for Trace–1

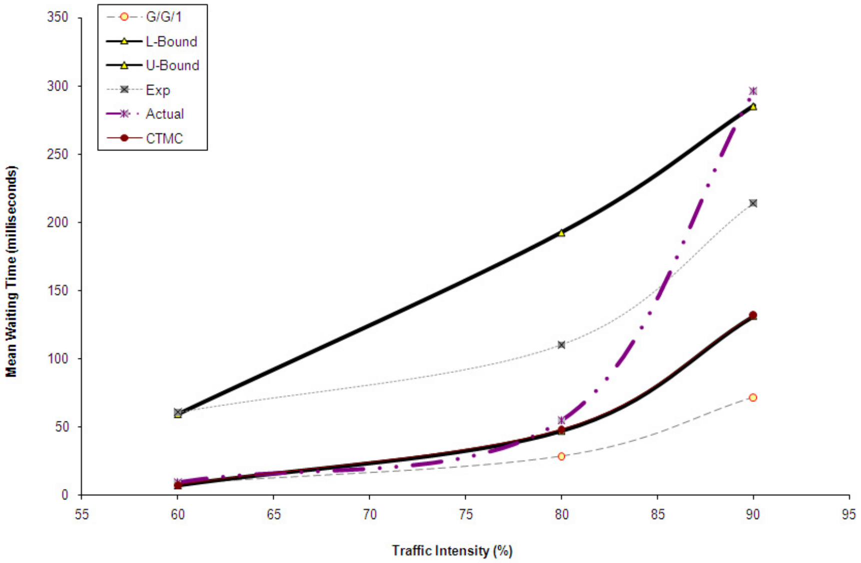

Fig. 18. Different Bounds and Approximations at Different Traffic Intensities for Trace–2

higher mean waiting time compared with the remaining four that are all close to each other, especially at low and medium traffic intensities. The margin of separation is very low at higher traffic intensities, which prompts us to study the high traffic intensity separate from the low and medium ones.

### III.7.1. High Traffic Intensity

Based on the results from Fig. 17, Fig. 18, Fig. 19, and Fig. 22, at 90% traffic intensity, the actual waiting times ("Actual") are much closer to the fluid upper bound ("U-Bound") as compared with the other methods used. This can also be observed in Table V, which shows the results at 90% traffic intensity. For the rest of the traces, as shown in Fig. 20, Fig. 21, and Fig. 23, lower bound comes out to be a much better approximation. Also note that in six out of these seven cases, traditional $G/G/1$

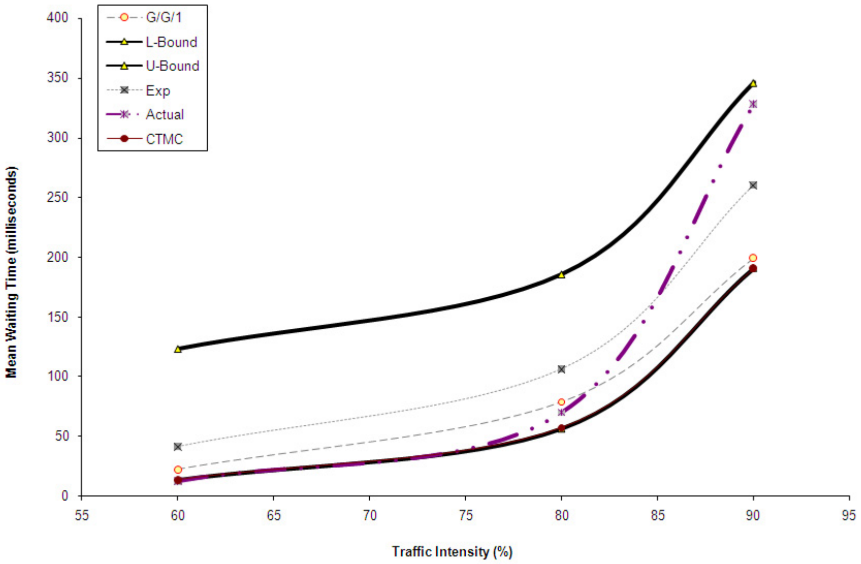

Fig. 19. Different Bounds and Approximations at Different Traffic Intensities for Trace–3

approximation severely under-estimates waiting time.

With the advent of “green computing”, in order to save energy, it is envisioned that computer systems will run at high traffic intensities. Thus it is imperative that we have excellent performance models under those conditions, and U-bound seems to be the better choice for that.

It appears from some of these results that at a traffic intensity higher than 90%, “Actual” waiting time might be higher than the “U-Bound”. As mentioned in Section III.4.4, these fluid-bounds may be violated in our analysis since we have fitted distributions to trace data in order to obtain these bounds, and have assumed the ON and OFF times to be iid. Hence, this is not a contradiction.

It can be observed from Table V that the actual waiting time in the system (“Actual”) was found to be lower than the lower bound obtained using the fluid-flow

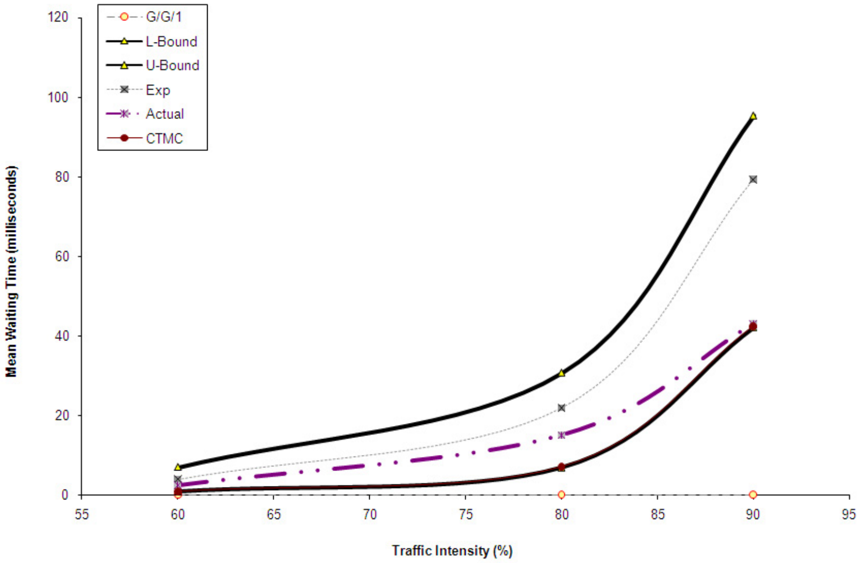

Fig. 20. Different Bounds and Approximations at Different Traffic Intensities for Trace–4

**TABLE V** Different Bounds and Approximations of Waiting Times at 90% Traffic Intensity for Different Traces

| Trace | $G/G/1$ | L-Bound | CTMC | Exp | Actual | U-Bound |
|---|---|---|---|---|---|---|
| 1 | 14.10 | 39.48 | 39.572 | 59.16 | 80.7 | 81.61 |
| 2 | 72.02 | 131.66 | 131.92 | 214.27 | 296.30 | 285.50 |
| 3 | 199.19 | 190.70 | 190.73 | 260.17 | 328.2 | 346.09 |
| 4 | 0.113 | 42.22 | 42.51 | 79.46 | 43.11 | 95.38 |
| 5 | 158.08 | 217.22 | 218.20 | 279.76 | 177.81 | 373.02 |
| 6 | 191.19 | 268.81 | 269.05 | 335.57 | 420.16 | 445.75 |
| 7 | 11367 | 3567.2 | 3568.03 | 3585.7 | 1927.4 | 4780.6 |

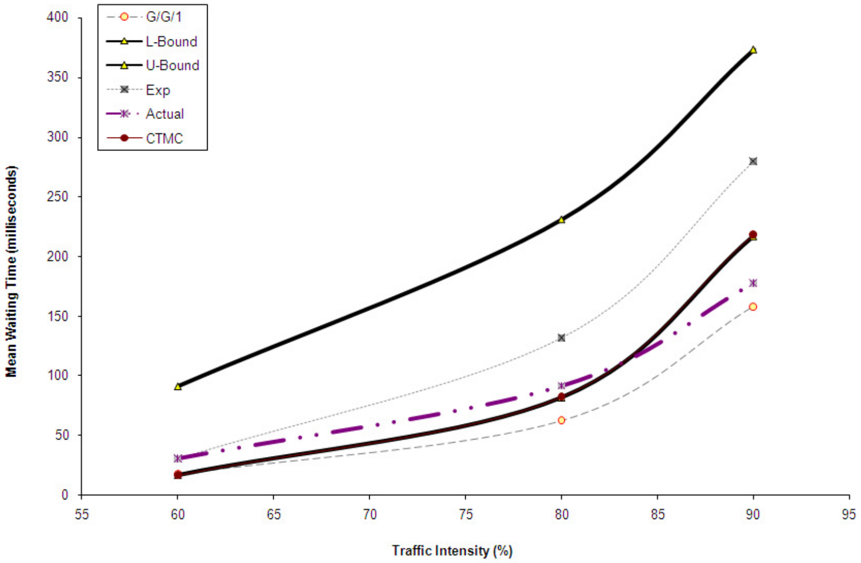

Fig. 21. Different Bounds and Approximations at Different Traffic Intensities for Trace–5

model ("L-Bound") in two cases. Upon investigation, we found that these two traces have the highest $C_s/C_a$ ratio (Table VI). Notice that the $C_a$ values are close to each other for all the traces, but the $C_s$ values for these two traces are extremely high. This implies that there is a significantly large coefficient of variation in the service times, or the file sizes resulting in very heavy tails. Extremely large variances coupled with high traffic intensity make model parameter estimation extremely unreliable. A more robust parameter estimation technique (perhaps with more data points) would be required in those cases.

### III.7.2. Medium to Low Traffic Intensity

We can observe from the results for various traces that as the traffic intensity eases up, the actual waiting time ("Actual") draws towards "$G/G/1$", "L-Bound" and

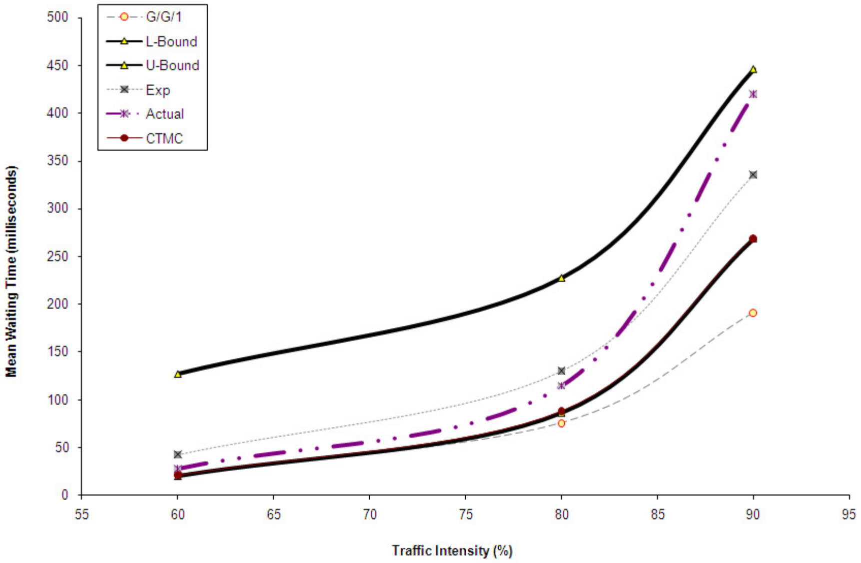

Fig. 22. Different Bounds and Approximations at Different Traffic Intensities for Trace–6

**TABLE VI** Comparison of $C_a/C_s$ Values of Different Traces

| Trace | $C_a$ | $C_s$ | $C_s/C_a$ |
|---|---|---|---|
| 1 | 1.555 | 4.743 | 3.051 |
| 2 | 1.331 | 10.629 | 7.988 |
| 3 | 1.458 | 28.174 | 19.326 |
| 4 | 1.398 | 2.180 | 1.560 |
| 5 | 1.618 | 31.596 | 19.525 |
| 6 | 1.187 | 19.676 | 16.581 |
| 7 | 1.479 | 79.845 | 53.984 |

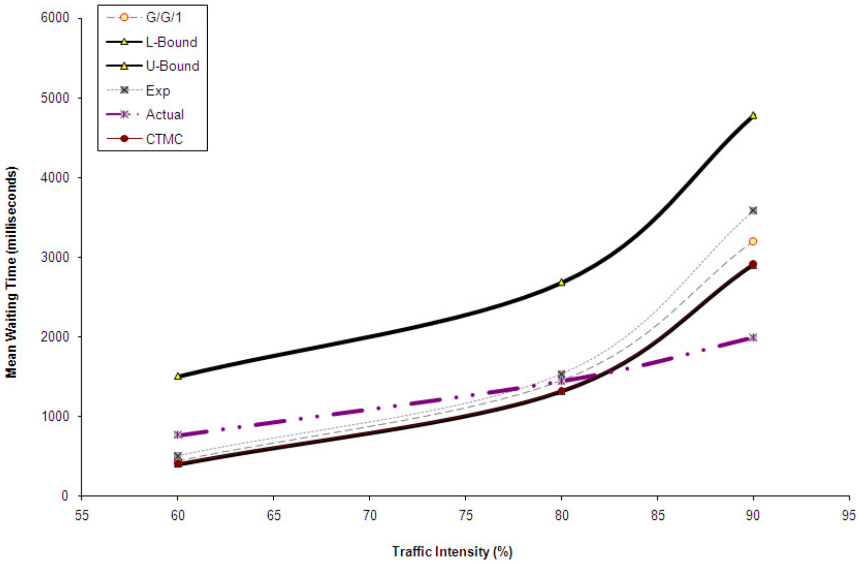

Fig. 23. Different Bounds and Approximations at Different Traffic Intensities for Trace–7

**TABLE VII** Approximations at 80% Traffic Intensity

| Trace | $G/G/1$ | L-Bound | CTMC | Exp | Actual | U-Bound |
|---|---|---|---|---|---|---|
| 1 | 5.7 | 9.894 | 10.01 | 21.488 | 29.62 | 37.291 |
| 2 | 28.563 | 47.25 | 47.37 | 110.32 | 55.03 | 192.83 |
| 3 | 78.75 | 56.42 | 56.73 | 106.27 | 70.1 | 185.88 |
| 4 | 0.048 | 6.960 | 6.964 | 21.95 | 15.08 | 30.76 |
| 5 | 62.5 | 81.94 | 82.00 | 131.9 | 91.46 | 230.83 |
| 6 | 75.6 | 86.43 | 86.56 | 130.04 | 114.51 | 227.41 |
| 7 | 4492.1 | 1489.7 | 1489.8 | 1533.7 | 1445 | 2683.2 |

**TABLE VIII** Approximations at 60% Traffic Intensity

| Trace | $G/G/1$ | L-Bound | CTMC | Exp | Actual | U-Bound |
|---|---|---|---|---|---|---|
| 1 | 1.648 | 1.62 | 1.658 | 5.7 | 3.639 | 16.74 |
| 2 | 8.053 | 7.152 | 7.983 | 60.6 | 9.42 | 59.54 |
| 3 | 22.20 | 13.31 | 13.377 | 41.2 | 12.44 | 123.4 |
| 4 | 0.014 | 0.864 | 0.872 | 3.92 | 2.41 | 7.04 |
| 5 | 17.60 | 16.77 | 16.8248 | 30.37 | 30.43 | 91.07 |
| 6 | 21.26 | 20.04 | 20.309 | 42.35 | 27.36 | 126.87 |
| 7 | 1263.5 | 553.78 | 553.809 | 502.1 | 765.9 | 1504.6 |

"CTMC". Although the marginal distributions of the fluid ON and OFF times fit the data well, the successive ON and OFF times appeared to still be dependent in high traffic intensities (hence "Actual" waiting time was much higher than "CTMC"). In contrast, in the case of lower traffic intensities, the dependence in arrivals is captured by the fluid model that essentially clubs bursts of arrivals into a single ON time, and successive ON and OFF times are independent.

One of the important things to observe from these results that the expected waiting times computed using "CTMC" are extremely close to "L-Bound". This was found to be the case for all of the traces and at all traffic intensities. In order to explain this, recall that for the traces we used, "CTMC", "U-Bound" and "L-Bound" use hyperexponential ON and exponential OFF times. Therefore, performance measures for "CTMC" must lie between "L-Bound" and "U-bound". The most dominant term of "CTMC" is indeed exactly equal to "L-Bound" and the other terms contribute insignificantly to make "CTMC" just a little larger. One of the take-aways is that the “L-Bound” is a very good approximation of a fluid system that has a truly iid ON and OFF time traffic source.

In tables VII and VIII approximations for all traces are listed at traffic intensity of 80% and 60% respectively. Notice that for the 80% case, CTMC and L-Bound perform better than $G/G/1$ for a majority of the cases; but that for the 60% case, $G/G/1$ is better. However, notice that the $G/G/1$ results have a tremendous variability with respect to actual, unlike CTMC and L-Bound (which appear to be much more robust). In low traffic intensities, if mean estimate needs to be accurate, then $G/G/1$ should be used, and if confidence interval of estimate needs to be minimized, then choose CTMC (or L-Bound). Whereas, if it is critical to be conservative (under medium and low traffic intensities), then the ideal choice is “Exp”. In addition, for “Exp”, all we need is the mean ON times and mean OFF times of the fluid source.

### III.8. Findings and Discussion

Addressing the main objective of this research, it is evident that fluid models capture the effect of burstiness nicely and also provide expressions for various performance measures including average waiting time. Hence fluid models are suitable for traffic characterization. In addition, one can obtain workload distributions (not just averages) using $\Pr\{X(t) > x\}$. In fact, it was only for the sake of simulations and $G/G/1$ analysis we picked average waiting time as the measure of performance.

The following are the summary of findings based on our numerical experiments:

1. Fluid ON-OFF times generated from the trace data can be approximated well using the exponential family distributions. This implies that Markov modulated fluid sources can be effectively used for traffic models.

2. The results from CTMC analysis and lower bound are close. This illustrates

**TABLE IX** Choosing a Fluid Model When Traffic Exhibts LRD

| **Name** | **Ideally suited when** |
|---|---|
| U-Bound | Traffic intensity is high |
| L-Bound | Traffic intensify is medium and non-exponential ON/OFF |
| CTMC | Exponential family ON-OFF; medium traffic intensity |
| Exp | Only mean ON-OFF can be obtained; or conservative estimate needed in medium/low traffic intensity |

that $\Pr\{X(t) > x\}$ is indeed closer to the lower bound $K_* e^{-\eta x}$ in Equation (3.4) than the upper bound.

3. The fluid upper bound is found to be a conservative estimate for all the traces at lower traffic intensities. At high traffic intensities, it continues to be a conservative estimate for some traces, and a reasonably good approximation for others. In fact, for majority of cases, the actual expected waiting time in the system is closer to the fluid upper bound than any other estimate at higher traffic intensities.

4. Among the approximations considered, exponential ON-OFF is the best choice for combining conservativeness and accuracy in medium to low traffic intensities. Moreover, it only requires the mean ON and OFF times for the computations.

5. At medium and low traffic intensities, the CTMC fluid model and fluid lower bound produce more robust estimates than $G/G/1$ approximations, although the estimates in low traffic intensities are not as good.

As mentioned earlier in Section III.2, the fluid-flow models have been analyzed by many researchers, and these bounds have been well established. However, not many practitioners evaluate the performance measures of communication networks using these models. Leveraging on the well-established theoretical results, we have shown that these methods do indeed model the system in question very well. In this chapter, we have shown that fluid models certainly do an excellent job in handling LRD traffic and lend themselves well to obtaining closed-form performance metrics, including probability distributions. In closing, refer to Table IX for recommendations on the most appropriate fluid model for various scenarios.

# CHAPTER IV

# MULTIPLEXING NETWORK TRAFFIC

## IV.1. Introduction

A characteristic feature of modern switches and routers is that typically a large number of flows are multiplexed. The number of such flows can run to the thousands in the case of an Internet Protocol (IP) router. The same characteristic is exhibited by servers construing a Web-server farm, where thousands of requests are served and the data flows through a limited number of routers. Developing accurate models for this kind of traffic aggregation will provide a basis for efficient multiplexing schemes that optimally utilize the network resources, such as bandwidth, buffer, etc., and provide the best possible QoS.

Researchers have been trying to characterize the aggregated traffic and develop models for this multiplexed traffic considering various applications, such as in case of ATM multiplexers [58] and wireless packet data networks [74]. These kinds of models can also be very useful when designing or characterizing the flow in case of Web-server farms, where there are a large number of servers and the traffic from these connects to the outside world through a very small number of routers/switches. A representation of such a Web farm is shown in Fig. 24. In designing and operating such centers, it becomes critical to be able to obtain the performance measures to provide the required QoS, and this analysis is only possible in case of a multi-layered network when one is able to characterize the aggregate flow from the first and subsequent layers of nodes.

A major challenge for designing and controlling high-speed communication networks is to develop methods for analyzing more realistic source-traffic models that

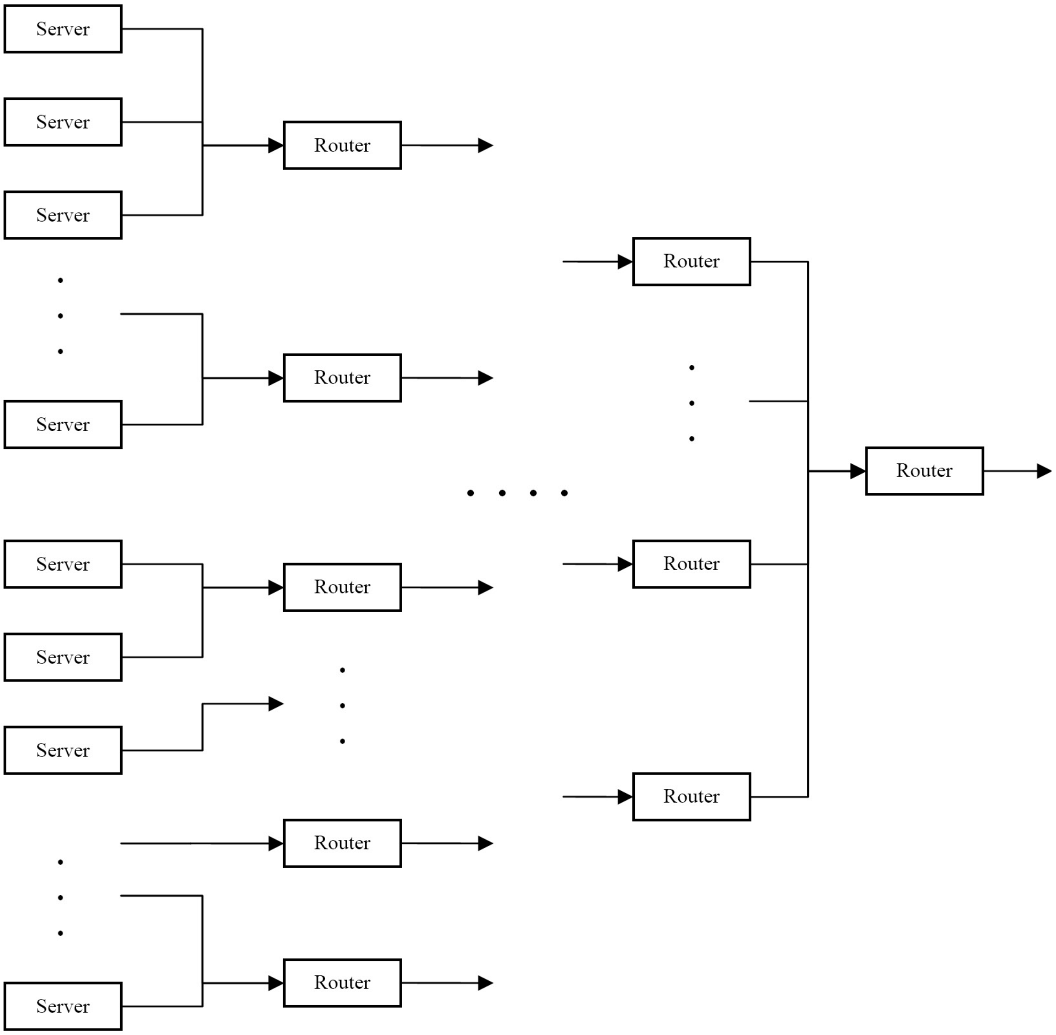

Fig. 24. Network Representation Demonstrating the Merging of Flows at Different Layers of Routers

are consistent with recent traffic measurements. It has been observed that the aggregated Internet traffic flow have similar patterns in different time scales, referred to as self-similarity [45]. Network traffic has been shown to be quite complex, exhibiting phenomena such as heavy-tailed probability distributions and long-range dependence [34]. Heavy-tailed distributions are known to cause self-similarity in models of (asymptotically largely) aggregated traffic [75]. These properties make modeling such traffic very challenging and the computation of analytical expressions for different performance measures extremely difficult. For example, in an M/G/1 queueing system, the average delay is proportional to the variance of the service time. In case of heavy-tailed ON-OFF sources, the service time variance could be immeasurably large, which equates to a huge value for the average delay. Furthermore, for ON-OFF sources with heavy-tailed ON times, the moment generation function of the service time is very large, meaning that the Chernoff bound can not be used to analyze the queue delay performance.

Researchers have developed different kinds of analytical, numerical, and simulation models in order to represent network flow, and a brief review of such models will be presented in Section IV.2. In this research work, we take a lead from stochastic fluid-flow models and represent data as fluid flowing into and out of the system. Stochastic fluid-flow models have been used for a long time to describe networks of nodes that provide service to traffic of some sort that flows among them, and it is widely accepted that network flow can be approximated reasonably well using such models [62]. Such models are commonly used to approximate discrete state systems of traffic flows, in which the number of states rapidly becomes prohibitively large as the complexity of the system increases.

In this research work, we take data from a real trace and model it as a source alternating between bursts (the ON state) and silences (the OFF state). This source then feeds into a feed-forward network (Fig. 24), where other ON-OFF sources are also contributing to the buffer content of a FIFO server that is being emptied at a constant rate. Willenger et al. [75] have proven that that superposition of a number of ON-OFF traffic sources, with heavy tailed ON and OFF periods, results in a self-similar aggregate traffic. These results are confirmed by Park et al. [76], who show that an individual data-traffic source can be modeled as an ON-OFF source with heavy-tailed ON and OFF periods. Hence, the output flow from these servers can also be modeled as ON-OFF flows. Our objective is to characterize the aggregate flow of these sources superpositioned on top of one another. We present a simple and fast methodology to characterize this aggregate flow that can be used to solve various design and analysis problems concerning such network flows. These problems include computing the performance metrics of the network given the structure and channel capacities of the network, as well as computing the desired channel capacities given the QoS requirements.

We will concentrate on elastic traffic here, which is usually associated with a delay-based QoS requirement. In case of Web farms catering to such requests, the bottleneck is usually the router where multiple servers send the data to be transmitted via the Internet [6]. This makes modeling these routers very important and obtaining the remaining workload or the waiting time approximations for these files critical.

We allow the ON and OFF times associated with the flows to have heavy-tailed distributions and compute the buffer content (workload in the system) when such data gets multiplexed. We need to obtain not only the workload at different nodes, but also the distribution of the output flow, since this flow would serve as inflow to the subsequent node. This would help in solving design or control problems relating to the aggregate traffic flow, such as acquiring the optimal bandwidth for a link. Aggregate flow will also be characterized as ON-OFF flow, and we will obtain the distribution of this, given the inflow parameters.

The remainder of the chapter is organized as follows. We present a brief review of related literature, presenting the work done by researchers on similar problems, in Section IV.2. This is followed by the problem description, presented in Section IV.3, and proposed methodology in Section IV.4, along with method used to validate the solutions. Numerical results of the analysis are presented in Section IV.5, where we demonstrate using several examples the applicability of the proposed methodology. Finally, we present the conclusions in Section IV.6.

## IV.2. Literature Review

Acquiring performance measures for a high-speed network for its design and operation has been an active area of research for some time. One of the biggest challenges in dealing with such systems is the associated long-range dependence in traffic [34]. This long-range dependence in traffic leads to self-similarity [75], which generally leads to underestimation of the performance metrics of the system [32]. In the case of networks, where a large number of individual flows get merged to form aggregate flows, the problem gets exaggerated.

Researchers have analyzed network flow in many different ways. Some have modeled the flows as renewal sources, which lead to tractable problems and good approximations using analytical techniques. Researchers have been able to compute the overflow probability in case of a stationary buffer [38] or transient buffer [77]. However, Sriram and Whitt [78] have established that the aggregate arrival process resulting from the superposition of many independent renewal processes in not nearly renewal. This is essentially because the instantaneous arrival rate in the aggregate process at any time is a function of the number of active incoming streams, which fluctuates substantially.

Several other researchers, such as Cáceres [79], Crovella and Bestavros [45], Leland et al. [33], Paxson and Floyd [34], and Feldmann [80] have also shown that network traffic exhibits heavy-tailed probability distributions and is quite complex to model. Thus, results obtained using approximations that model the arrivals as renewal source cannot be used for the aggregate flows. This has led to researchers considering different kinds of models and approximations to obtain the performance parameters of aggregate traffic. Aggregating traffic into a shared link also yields to unexpected end-to-end delay. This additional delay may add up to the already critical end-to end delay in case of delay-sensitive flows, and may degrade the QoS provided.

Hassan and Garcia [81] have pointed out the inadequacy of the Hurst exponent in describing burstiness when the underlying protocol is TCP. They have suggested a model that performs well when idle periods are very long compared with activity periods–that is, when the traffic intensity is relatively low. However, they have conceded that in case a large number of connections are needed, the proposed algorithm does not perform very well because a considerable number of equivalent processes need to be executed.

For a slightly different application, Hassan et al. [82] have shown that VoIP traffic can be approximated very well by exponential law under light traffic intensities. They have shown that under these conditions, the MMPP-2 process regenerates the correlation structure of aggregate VoIP traffic quite well. However, they have conceded that this approximation is not suitable under heavy traffic intensities, since heavy loads amplify the effect of time-dependence. Moreover, this model limits the possibility to achieve analytical QoS parameters evaluation when queue networks are considered.

Some authors have considered aggregation of flows in order to deal with resource allocation control protocols, such as Ravindran and Gong [83]. Aggregation is a common approach to address the scalability issue in resource allocation, since it leads to less demand on the resources at the aggregation point. Their contribution is in the idea of casting the routing subsystems of multicast networks with RSVP-style resource allocation functions. They have developed a mechanism to generate a composite flow from a set of flows and vice versa.

The end-to-end reservation approach has been considered in some cases to model the network traffic. This approach works well in case of inelastic traffic, where there is a fixed bandwidth requirement, such as in a telephone network, video streaming etc. Researchers have also tried to obtain the optimal amount of bandwidth required given the QoS criteria in case of inelastic flows [84]. However, these approaches do not perform very well with *elastic* traffic flows because the overheads are too high for short-lived connections. Hence, aggregation of flows is essential in order to make the resource allocation scalable. According to Wang and Basu [85], for these kinds of short-lived flows, network resources should be allocated to the aggregate flows rather than the individual flows.

Aalto [65] has shown that the aggregated departure process in case where input rate is modulated by a Markov jump process can be modeled using another Markov jump process. Shah-Heydari and Le-Ngoc [86] have also suggested that the superposition of a number of homogeneous ON-OFF sources can be approximated by a Markov modulated Poisson process (MMPP). Now, if we approximate the ON times and the OFF times as a 2P-HE distribution, we will need three parameters to define each. Hence, we will be able to capture the properties of MMPP-2 process.

Aalto [87] has considered a fluid model for a multiplexer with the goal of charaterizing the output process. They have considered independent and heterogeneous ON-OFF sources and shown that similar to the input sources, the aggregate flow behaves as an ON-OFF channel as well. They further characterize the ON times of the output using a normalized load, and have characterized the distribution of the ON-times of the output source using Laplace transforms.

Once we have obtained the parameters for the distribution defining the aggregate flow, we can obtain the performance measures or solve a control/design problem by using analytical techniques such as in stochastic fluid-flow models. However, we want to validate that the performance metrics obtained using this aggregate flow accurately represent the sum of the individual flows. For this purpose, we will build a simulation.

The simulation of a traffic source consists of building a generating model able to create requests with arrival times and file size fitting the dynamical statistical behavior of real traffic. Boots and Mandjes [88] have pointed out that aggregate network traffic flow is not only difficult to model, but also that researchers have had trouble simulating such a system and have come up with clever techniques in order to accomplish this. They have performed a characterization study in order to obtain aggregate traffic model under different traffic intensities. These models become very difficult to execute however, when thousands of connections are considered.

Hence, in this research work, we would build aggregate traffic models, which would include finding a good approximation of the arrival process of multiple connections, and will represent a good trade-off between accuracy and simulation efficiency.

## IV.3. Problem Definition

In this research work, the objective is to characterize the network traffic flow resulting from superpositioning of a number of such flows and obtaining the marginal distribution of the buffer content at the same time. The flows are modeled as fluids, where each incoming flow is modeled as an ON-OFF source.

Consider a $K$-stage feed-forward queueing system as shown in Fig. 24. In the $k^{th}$ stage, fluid from $N_{jk}$ sources is input into the $k^{th}$ stage buffer $j_k$. That is, for each first-stage buffer $j_1$, there are $N_{j1}$ external sources of flow, whereas for every other stage in the system, there are no external sources. Hence, the flow coming into a subsequent buffer $\{j_k : k > 1\}$ comprises only of the flows coming from the previous stage buffers.

Let the ON and OFF times representing all external input sources be given. Being network flows, we assume these to be self-similar. In this research work, we use ON and OFF times derived from trace data. With these, the distribution parameters of these flows can be obtained and used in the analysis.

Now consider buffer $j_k$ of stage $k$. Source $i_{jk}$ (for $i_{jk} = 1, \dots, N_{jk}$) is an ON-OFF source. When source $i_{jk}$ is ON, fluid is generated at rate $R_{ijk}$ kbps and no fluid is generated when the source is OFF. Buffer $j_k$ is of infinite size and has an output channel capacity of $C_{jk}$ kbps. Since this output channel is the input flow for a subsequent stage buffer, we would have a one-to-one mapping (defined by the network structure) of all $R_{lm\{k+1\}}$ and $C_{jk}$ $(k < K)$, where $R_{lm\{k+1\}}$ represents some input flow $l$ of buffer $m$ of stage $k + 1$.

The output from all the $k$-stage buffers is multiplexed into $k + 1^{th}$ stage infinite-sized buffers. Hence, the output from all the buffers is poured eventually into the final-stage buffer, with output channel capacity $C_{1K}$. A discriminating feature of this model is the assumption that if there is a link of capacity $C$ kbps, fluid flows in it either at rate $C$ or 0, which is typical in practice for any channel. Therefore all the flows in the system are essentially ON-OFF in nature.

For $j_k = 1, \dots, J_k$ let $X_{jk}(t)$ be the amount of fluid in buffer $j_k$ at time $t$. The objective of the performance analysis is to obtain the limiting buffer content distribution

$$\lim_{t\to\infty} \Pr\{X_{jk}(t) > x\} \tag{4.1}$$

for any finite $x$. The performance measures can be used in several optimization instances, such as designing which sources should be clubbed into which buffer, capacity planning, designing the number of first-stage buffers, computing optimal buffer size, etc. We will illustrate a few examples in Section IV.5.

In order to obtain the performance measure at a particular buffer, the input flows will have to be characterized at that buffer. As described in Section IV.2, in general, it is very difficult to obtain the characteristics or parameters of aggregate flow analytically. Moreover, as shown by Alagoz [89], as the number of input sources increases, the existing approximation models also start to underestimate the variability, and as a result, the loss probabilities, of the aggregate flow.

We will try to overcome this shortcoming by using an optimization technique to obtain the parameters of the aggregate flow. Here, irrespective of the number of input flows, we would have only one constraint–flow balance–and unknown parameters characterizing the output flow, which are dependent only upon the distribution chosen, and not the number of input flows. Hence, with the increase in the number of input flows, although we would have to perform more calculations to obtain the input flow and to compute deviations in the objective, the amount of time taken to solve the optimization would largely remain unchanged. Details of the optimization, including the objective function, are provided in Section IV.4.

## IV.4. Modeling and Analysis

In this section, we will solve the optimization problem described earlier. We will model the problem presented in the previous section, and demonstrate the solution methodology in Section IV.4.1. This would be followed by validation methodology using simulation in Section IV.4.2 and a comparison against effective bandwidth method in Section IV.4.3.

### IV.4.1. Optimization

We will model input sources and the aggregate output flow as ON-OFF sources and characterize them using 2P-HE distributions. The CDF of a 2P-HE distribution is given as:

$$F(x) = 1 - pe^{-\alpha_1 x} - (1-p)e^{-\alpha_2 x} \qquad\qquad x \geq 0 \; , \tag{4.2}$$

with parameters $p$, $\alpha_1$ and $\alpha_2$. Hence, to characterize a flow, we would need to obtain six parameters, three each for ON and OFF times.

Given the input flows, parameters defining its distribution can be obtained using the moment-matching technique. The $i^{th}$ moment of the 2P-HE distribution can be written as:

$$m_i = i! \left[ \frac{p}{\alpha_1^i} + \frac{1-p}{\alpha_2^i} \right] \; . \tag{4.3}$$

The first three moments of ON and OFF times can be compared in order to compute the parameters defining the distribution.

Now, consider the $j_k^{th}$ buffered queue of stage $k$, as described in Section IV.3. We have $N_{jk}$ ON-OFF fluid sources feeding this queue. Parameters defining the $i_{jk}$th source are represented as $p_{ijk}$, $\alpha_{ijk1}$, $\alpha_{ijk2}$, $q$, $\beta_{ijk1}$ and $\beta_{ijk2}$. For simplicity, we drop the subscripts $j$ and $k$ for the remainder of the chapter. Thus, the $i^{th}$ source feeding this queue has the channel capacity of $R_i$, ON times parameters $p_i$, $\alpha_{i1}$ and $\alpha_{i2}$, and OFF time parameters $q_i$, $\beta_{i1}$ and $\beta_{i2}$. The output channel capacity of this queue is $C$ kbps (Fig. 25). We denote the ON and OFF times parameters of the output flow as

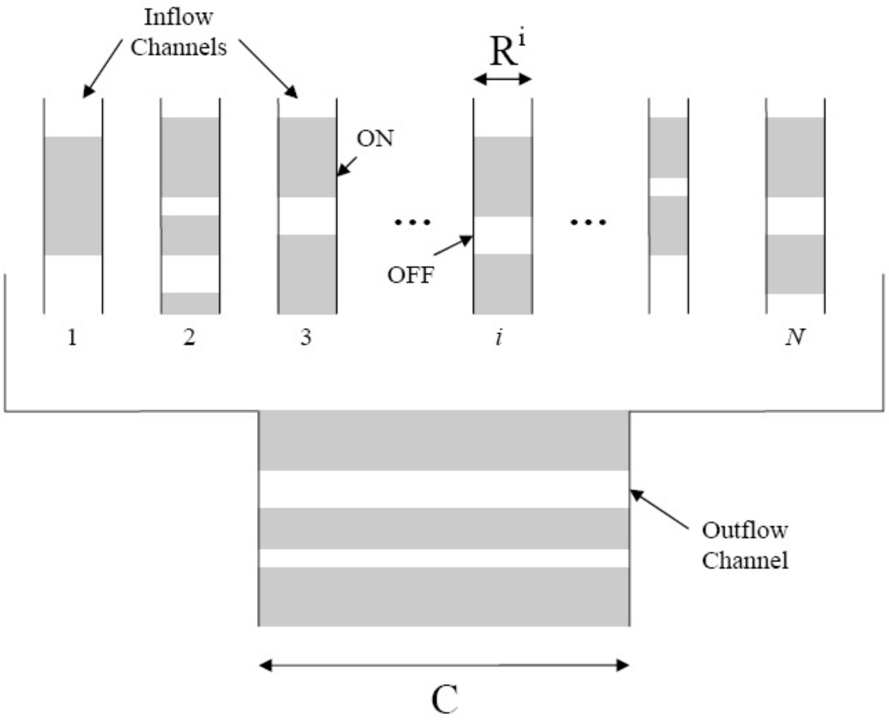

Fig. 25. Buffered Queue with a Number of Arrival Channels

$p_o$, $\alpha_{o1}$ and $\alpha_{o2}$ and $q_o$, $\beta_{o1}$ and $\beta_{o2}$, respectively.

We use optimization to obtain the parameters for the output flow. In order to compute the expected inflow for the flow-balance equation, we need to obtain the steady-state probabilities of different states of the inflow channel. These can be obtained by using continuous time Markov chain (CTMC) analysis, as we are using exponential family distributions. The generator matrix for an incoming flow $i$ can be written as shown in Equation (4.4). Here, the first two states correspond to the two phases of the OFF times and the last two correspond to the ON time states.

$$Q = \begin{bmatrix} -\beta_{i1} & 0 & p_i\beta_{i1} & (1-p_i)\beta_{i1} \\ 0 & -\beta_{i2} & p_i\beta_{i2} & (1-p_i)\beta_{i2} \\ q_i\alpha_{i1} & (1-q_i)\alpha_{i1} & -\alpha_{i1} & 0 \\ q_i\alpha_{i2} & (1-q_i)\alpha_{i2} & 0 & -\alpha_{i2} \end{bmatrix} \tag{4.4}$$

Solving for the steady-state probabilities, we obtain:

$$\begin{aligned} \pi_{i1} &= \frac{q_i\alpha_{i1}\alpha_{i2}\beta_{i2}}{\beta_{i1}\beta_{i2}(p_i\alpha_{i2}+\alpha_{i1}(1-p_i))+\alpha_{i1}\alpha_{i2}(\beta_{i1}(1-q_i)+q_i\beta_{i2})} \\ \pi_{i2} &= \frac{(1-q_i)\alpha_{i1}\alpha_{i2}\beta_{i1}}{\beta_{i1}\beta_{i2}(p_i\alpha_{i2}+\alpha_{i1}(1-p_i))+\alpha_{i1}\alpha_{i2}(\beta_{i1}(1-q_i)+q_i\beta_{i2})} \\ \pi_{i3} &= \frac{p_i\alpha_{i2}\beta_{i1}\beta_{i2}}{\beta_{i1}\beta_{i2}(p_i\alpha_{i2}+\alpha_{i1}(1-p_i))+\alpha_{i1}\alpha_{i2}(\beta_{i1}(1-q_i)+q_i\beta_{i2})} \\ \pi_{i4} &= \frac{(1-p_i)\alpha_{i1}\beta_{i1}\beta_{i2}}{\beta_{i1}\beta_{i2}(p_i\alpha_{i2}+\alpha_{i1}(1-p_i))+\alpha_{i1}\alpha_{i2}(\beta_{i1}(1-q_i)+q_i\beta_{i2})} \end{aligned} \tag{4.5}$$

Because the fluid comes into the queue only in the last two states, net expected inflow can be obtained as $\sum_{i=1}^{N} R_i(\pi_{i3}+\pi_{i4})$. Thus, we can write the total inflow as:

$$I = \sum_{i=1}^{N}\left[\frac{R_i(p_i\alpha_{i2}+(1-p_i)\alpha_{i1})\beta_{i1}\beta_{i2}}{\beta_{i1}\beta_{i2}(p_i\alpha_{i2}+\alpha_{i1}(1-p_i))+\alpha_{i1}\alpha_{i2}(\beta_{i1}(1-q_i)+q_i\beta_{i2})}\right]. \tag{4.6}$$

Net outflow from the buffered queue is essentially the fraction of time system is in ON state, multiplied by the rate of flow. This is of course true for a stable system, where all the fluid coming in must leave the system as some point. Now, the mean of the 2P-HE distribution defined in Equation (4.2) is given as $\mathbb{E}(x) = [p\alpha_2 + (1-p)\alpha_1]/\alpha_1\alpha_2$. Using this, outflow from the system can be obtained as

$$O = C\frac{p_o\alpha_{o2}+(1-p_o)\alpha_{o1}\beta_{o1}\beta_{o2}}{\beta_{o1}\beta_{o2}(p_o\alpha_{o2}+(1-p_o)\alpha_{o1})+\alpha_{o1}\alpha_{o2}((1-q_o)\beta_{o1}+q_o\beta_{o2})}. \tag{4.7}$$

If the output channel capacity is small as compared with the amount of inflow, buffer content may become very large and cause the queue to be unstable. In such a case, even if the output is flowing at a constant rate $C$, it will be less than the inflow. That is,

$$\sum_{i=1}^{N} R_i(\pi_{i3} + \pi_{i4}) > C \ .$$

Hence, for a stable system, we need to have

$$C \geq \sum_{i=1}^{N} \left[ \frac{R_i(p_i\alpha_{i2} + (1-p_i)\alpha_{i1})\beta_{i1}\beta_{i2}}{\beta_{i1}\beta_{i2}(p_i\alpha_{i2} + \alpha_{i1}(1-p_i)) + \alpha_{i1}\alpha_{i2}(\beta_{i1}(1-q_i) + q_i\beta_{i2})} \right] . \tag{4.8}$$

Equation (4.8) is what we would reference as the stability equation. Using this equation and given inflow parameters, the minimum value of output channel capacity $C$ can be computed below which, the system would become unstable because long-run buffer content would tend to infinity.

The objective function used for optimization is to minimize the sum of squared percent deviations of the aggregate flow parameters with respect to the input flow parameters. The idea behind this is that the aggregate flow must look statistically similar to the input flows. Since we are using hyper-exponential distributions to characterize the flows, the individual flows are in different phases with defined probabilities. If there was only one input flow, the aggregate flow parameters would be exactly same as the inflow parameters. Thus, we would expect the parameters defining the aggregate flows to be close the parameters of the individual flows. We would compute this value $D$, which we will represent as:

$$D = \min_{\{p_o,\alpha_{o1},\alpha_{o2},q_o,\beta_{o1},\beta_{o2}\}} \left[ \left\{ \frac{\sum_{i=1}^{N} p_i - Np_o}{\sum_{i=1}^{N} p_i} \right\}^2 + \left\{ \frac{\sum_{i=1}^{N} q_i - Nq_o}{\sum_{i=1}^{N} q_i} \right\}^2 + \left\{ \frac{\sum_{i=1}^{N} \alpha_{i1} - N\alpha_{o1}}{\sum_{i=1}^{N} \alpha_{i1}} \right\}^2 \right.$$
$$\left. + \left\{ \frac{\sum_{i=1}^{N} \alpha_{i2} - N\alpha_{o2}}{\sum_{i=1}^{N} \alpha_{i2}} \right\}^2 + \left\{ \frac{\sum_{i=1}^{N} \beta_{i1} - N\beta_{o1}}{\sum_{i=1}^{N} \beta_{i1}} \right\}^2 + \left\{ \frac{\sum_{i=1}^{N} \beta_{i2} - N\beta_{o2}}{\sum_{i=1}^{N} \beta_{i2}} \right\}^2 \right] \tag{4.9}$$

In order to solve the optimization problem, we will also need to impose probability constraints ($0 \leq p_o, q_o \leq 1$) and the non-negativity constraints ($\alpha_{o1}, \alpha_{o2}, \beta_{o1}, \beta_{o2} \geq 0$).

Solving this optimization problem gives us the parameters of distribution defining aggregate flow from the queued buffer. One such problem would have to be solved at every node in the system for which output flow parameters need to be computed. Nonetheless, since we are modeling the output parameters using the same family of distribution as the input parameters (2P-HE), solving the next layer of the problem does not involve any additional difficulty. This way, we can obtain the distribution defining all of the flows in the system and, hence, the outflow.

Once we have obtained the outflow parameters at a particular node, we can obtain the performance measure using the fluid-flow model. Details of this analysis are given in [68]. We will use the upper bound on the workload in the system as the performance metric of interest, which has been shown in Chapter III to be a good and conservative estimate.

We obtain $\eta$ as the smallest real positive solution to the equation:

$$\tilde{U}(-\eta(R-C))\;\tilde{D}(\eta C)\;=\;1\;, \tag{4.10}$$

where $\tilde{U}$ and $\tilde{D}$ represent the Laplace-Steiltjes transforms of distributions defining ON times and OFF times of the aggregate flow, respectively, $R$ is the sum of input channel capacities, and $C$ is output channel capacity. For 2P-HE ON and OFF times, this equation turns out to be a fourth-degree equation, with one root being equal to zero. Then, we obtain the upper bound on the workload in the system as

$$\frac{K^*}{\eta} = \frac{\left(\tilde{U}(-\eta(R-C))-1\right)R}{\left(\tau_U+\tau_D\right)C(R-C)\eta^2\inf_x\left\{\frac{\int_x^\infty e^{\eta(R-C)(y-x)}dU(y)}{1-U(x)}\right\}}. \tag{4.11}$$

### IV.4.2. Validation

Once we have obtained the parameters representing the aggregate output flow, we need to validate that the flow represented by these parameters indeed captures the characteristics of the aggregate flow. In order to do this, we will perform a simulation and compare the expected workload with the case in which all of the individual flows are passed through a single buffered queue. In the simulation, we will need to model the outflow from the buffer. As mentioned in Section IV.1, [75] and [76] have shown that aggregate flow from a number of heavy-tailed ON-OFF traffic sources results in a self-similar aggregate traffic, which can be modeled as an ON-OFF source with heavy-tailed ON and OFF periods. Hence, we will model the output flow from these servers also as ON-OFF flow.

As the first scenario, we will generate traffic using the CDF of the inflows. Separate ON-OFF times will be generated for each of the inflows, based on their distribution, and then combined. This combined flow will have the flow rate equal to the sum of capacities of channels that are in the ON state at any given time. Hence, this could vary between zero and the sum of inflow capacities, in discrete intervals, changing at discrete time instances. This combined flow will be passed through a buffered queue with a constant output capacity, and the average workload in the system will be computed. It is possible that the number of input flows in the ON state are small so that the total input flow rate is less than the outflow rate. We will make sure that the outflow is still ON-OFF by setting a small minimum ON time associated with the outflow. Thus, the outflow might be OFF while there is still some fluid (albeit infinitesimal) in the buffer. This simulation will be repeated multiple times, to obtain a reasonably large number of estimates of the average workload. The mean of these values will be obtained as the expected workload in the system.

As the second scenario, traffic will be generated using parameters representing the aggregate outflow. This will be treated as an ON-OFF flow, with channel capacity equal to the sum of underlying input flow capacities. This flow will then be fed into the same queue with the constant output capacity. In this case, the output channel capacity will be less than the input capacity, and hence, the output flow would be

ON-OFF if the input flow is such. Using this simulation, we compute the average workload in the system. Again, multiple estimates will be obtained to deal with the variability while generating the flow. The mean of average workload values so obtained will be taken as the expected workload in the system.

A comparison of these two values of the workload obtained will give us a measure of the accuracy of our model. If these two values are comparable, we can conclude that the aggregate flow does represent the flow obtained by superpositioning of the input flows for workload assessment.

### IV.4.3. Comparing Against Effective-Bandwidth

One of the popular methods for computing the expected workload in the system is to use effective bandwidths. We would like to compare the performance of our model against this technique, by using the simulation results as the baseline.

Since we are defining the ON-OFF sources using hyperexponential distributions, it is possible for us to model the system as a CTMC. We will use the technique presented in [55] in order to compute the effective bandwidth of a source. Let $\{Z(t), t \geq 0\}$ be an irreducible, finite state CTMC with generator matrix $Q$. When the CTMC is in state $i$, the source generates fluid at rate $r(i)$. Let $R = \text{diag}[r_{ii}]$, where $r_{ii} = r(i)$, and let $e(M)$ denote the largest real-eigenvalue of a square matrix $M$. Then, the effective bandwidth of the source can be given as

$$eb(\upsilon) = \frac{1}{\upsilon} e(Q + \upsilon R). \tag{4.12}$$

When we have multiple independent sources feeding into a buffer, we can compute the sum of the effective bandwidths of these sources as

$$\sum_{k=1}^{K} eb_k(\upsilon) = \sum_{i=1}^{K} \frac{1}{\upsilon} e(Q_i + \upsilon R_i), \tag{4.13}$$

where there are $K$ sources, $1, \ldots, K$. If the output channel capacity is $C$, we obtain $\upsilon$ by solving

$$\sum_{k=1}^{K} eb_k(\upsilon) = C. \tag{4.14}$$

Then, we have

$$\Pr(X > x) \approx e^{-x\upsilon} \tag{4.15}$$

using which, we can obtain the expected workload in the system as $W = 1/\upsilon$. Hence, this value obtained would be the expected workload in the system when $K$ flows merge into the system with an output channel capacity of $C$. As the effective bandwidth does not change when a flow is passed through a buffer, we obtain the effective bandwidth of the output flow as the sum of the input flow effective bandwidths.

## IV.5. Numerical Results

For the purpose of numerical analysis, we obtain Web traces from the IRCache Web site, as mentioned in Section III.3. These traces have the arrival times of the requests and the sizes of files transmitted from the server. Multiple traces were obtained to facilitate the analysis.

We treat the data being transmitted as fluid. It is assumed that the arrival time of the request is the time the requested file starts transmitting from the server. Given channel capacity, ON times and OFF times for a trace can be computed. 2P-HE distribution can then be fit on this data to obtain the parameters using the moment-matching technique. This can be done by computing the first three moments numerically from the data and equating them with those listed in Equation (4.3).

The rate at which a file gets transmitted from the server defines the traffic intensity. Given a trace and channel capacity, the traffic intensity can be computed and parameters can be obtained for the defining distribution. This distribution is set for the traffic intensity, and traffic can be generated from this distribution for any channel capacity.

We will demonstrate the process of computing the parameters for the inflow and outflow using a simple example, presented in Section IV.5.1. This will be followed by analysis of more general and complex networks, with different objectives, in later sections.

### IV.5.1. Demonstrative Example

Consider a simplified version of the Web farm presented in Fig. 24, where we have only one layer of nodes (Fig. 26). In this case, there are a number of servers generating traffic that are connected to the Internet by the means of a single router. Consider the following test cases in which the primary goal is to obtain the aggregate flow parameters.

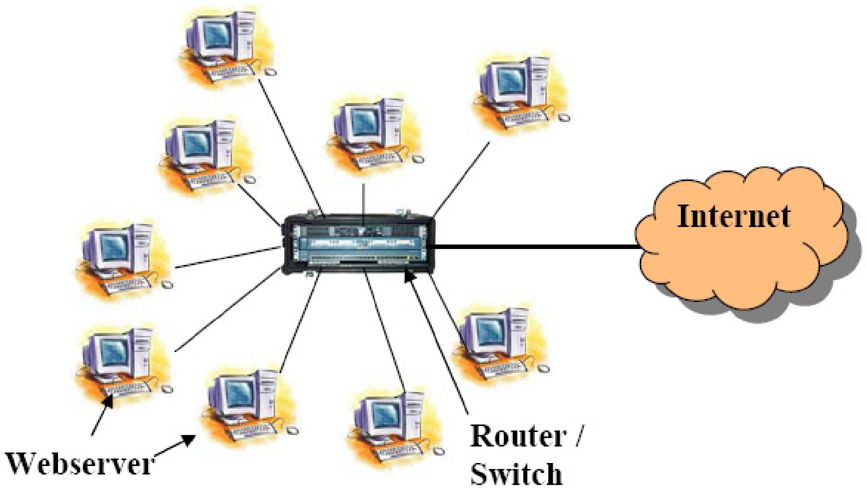

Fig. 26. Single-Layer Network with a Switch Connecting a Number of Servers to the Internet

**TABLE X** Obtaining Output Parameters for a Small Experiment

| | $p$ | $\alpha_1$ | $\alpha_2$ | $q$ | $\beta_1$ | $\beta_2$ |
|---|---|---|---|---|---|---|
| In-1 | 0.9999100 | 53.682 | 0.002826 | 0.61833 | 290.29 | 92.811 |
| In-2 | 0.9996143 | 43.129 | 0.000991 | 0.84467 | 31.136 | 8.3139 |
| In-3 | 0.9998523 | 74.532 | 0.000149 | 0.60878 | 13.618 | 5.0062 |
| Out | 0.9945041 | 57.638 | 0.001321 | 0.99979 | 110.81 | 0.0004 |

### IV.5.1.1. Case 1

Consider for this example three flows coming in with same traffic intensity of 90%. We use three different traces and obtain the parameters for the 2P-HE distribution, defining these flows using the technique described earlier in Section IV.5. The parameters of the distribution defining these flows are given in Table X, referenced by a serial number. For simplicity, let the channel capacities of these flows be the same: 100 units. Hence, we have three independent sources with the same traffic intensity and channel capacity pouring data into a common infinite-sized buffer. Let the output channel capacity be equal to 300 units, the sum of the input channel capacities.

In order to compute the parameters of the outflow, we use the optimization technique, with the objective function and the constraints as defined in Section IV.4. A computer program was written in MATLAB to perform this optimization and obtain the output parameters. These were obtained as presented in Table X, as 'Out'. The steady-state probabilities for the different states in the outflow were computed as the following:

$$\pi^o = \begin{bmatrix} 0.0019 & 0.1044 & 0.0037 & 0.8900 \end{bmatrix} .$$

Traffic intensity of the output flow is about 90%, the same as the input traffic intensities, since the sum of the input capacities is the same as the output channel capacity. Also, note that none of the non-negativity constraints (associated with parameters defining the aggregate flow distribution) are binding at the optimal solution.

#### IV.5.1.2. Case 2

If we reduce the output channel capacity, we expect to see an increase in the output traffic intensity. However, note that if the output channel capacity is too low, the problem may become infeasible. This is because the system will become unstable, as the rate of inflows will exceed the output channel capacity, as demonstrated by Equation (4.8).

In this case, we use the same example as that in Case 1, but we reduce the output capacity to 280. The results of this analysis are presented in Table XI as 'O-280'. Because the input flows are exactly the same as in the previous case, only the parameters of the output flow are presented here.

**TABLE XI** Obtaining Aggregate-Flow Parameters with Different Output Channel Capacities

| | $p$ | $\alpha_1$ | $\alpha_2$ | $q$ | $\beta_1$ | $\beta_2$ |
|---|---|---|---|---|---|---|
| O-280 | 0.9994173 | 78.921 | 0.001318 | 0.78970 | 134.00 | 14.724 |
| O-320 | 0.9999204 | 56.206 | 0.001329 | 0.68732 | 111.45 | 35.266 |

The steady-state probabilities obtained for this experiment are:

$$\pi^o = \left[ \begin{array}{cccc} 0.0124 & 0.0301 & 0.0267 & 0.9309 \end{array} \right].$$

This gives a resultant traffic intensity of 95.76%. As expected, this is an increase over the traffic intensity of 90% obtained in Case 1. A further reduction in outflow channel capacity to 260 units renders the problem infeasible. This is validated by Equation (4.8), solving which, we obtain the minimum output channel capacity for the system to be stable as 268.1.

This method can also be used when the output capacity is greater than the sum of the input capacities. As an example, we use output capacity as 320 units, and the output parameters are listed in Table XI as 'O-320'. In this case, we obtain the resultant traffic intensity of 83.78% and the steady-state probabilities as

$$\pi^o = \begin{bmatrix} 0.0665 & 0.0956 & 0.1919 & 0.6459 \end{bmatrix}.$$

### IV.5.2. Large Network

In the previous section, we demonstrated by means of a small example the calculation of the aggregate flow parameters. In this section, we use the methodology for the case in which a large number of flows merge to form aggregate flow. We will compute not only the aggregate flow parameters but also the expected workload in the system and validate the results using simulation and effective bandwidth approximations.

We still keep the simple structure of the network from the previous example (Fig. 26). Consider the network with 100 input sources, each with the mean traffic intensity of 95%. Let the 2P-HE distribution defining each of these input flows be as given in Table XII. These parameters were obtained from a real trace using the technique described towards the beginning of Section IV.5.

#### IV.5.2.1. Computing Expected Workload

We first compute the aggregate flow parameters and the expected workload in the system. For this example, let the input channel capacity be 100 units each, and let the output channel capacity be slightly less than the sum of input capacities: 9,500 units.

**TABLE XII** Obtaining Output Parameters with 100 Input, and 1 Output Channel

| | $p$ | $\alpha_1$ | $\alpha_2$ | $q$ | $\beta_1$ | $\beta_2$ |
|---|---|---|---|---|---|---|
| In | 0.9998763 | 73.865 | 0.00176 | 0.59709 | 295.49 | 91.562 |
| Out | 0.9996125 | 90.278 | 0.00177 | 0.74164 | 270.62 | 103.80 |

Then, the parameters obtained using optimization for the resultant aggregate flows can be computed by solving the optimization problem described in Section IV.4.1. The values so obtained are presented in Table XII. The traffic intensity of the aggregate flow can be computed from this as 97.77%, and the steady-state probabilities are given by:

$$\pi^o = \begin{bmatrix} 0.0116 & 0.0106 & 0.0470 & 0.9307 \end{bmatrix} .$$

Now, we will demonstrate the computation of the upper bound on the expected remaining workload in the system using the fluid-flow model. We first solve Equation (4.10) and obtain $\eta$ as the smallest real, positive solution. If we use the output channel capacity of $C = 9,500$, Equation (4.10) can be rewritten as:

$$\left(1 + \frac{0.00012368 \times 500\eta}{0.00176 - 500\eta} + \frac{0.99987632 \times 500\eta}{73.865 - 500\eta}\right) \times$$
$$\left(1 - \frac{0.40291 \times 9500\eta}{91.562 + 9500\eta} - \frac{0.59709 \times 9500\eta}{295.49 + 9500\eta}\right) = 1 \ .$$

We obtain the roots of this equation as $-0.0191$, 0, $1.239 \times 10^{-6}$, and 0.1260. The smallest positive root, $1.23936 \times 10^{-6}$, is picked as the value of $\eta$.

To obtain the upper bound on the waiting time, we solve Equation (4.11), and for this we need to obtain infemum of the integral in the denominator, which is a non-decreasing function of $x$. The graph of this expression as a function of $x$, for the example at hand, is shown in Fig. 27. From this graph, the the value of the infemum can be obtained as 1.00 at $x = 0$. This gives us $K^* = 1.432$.

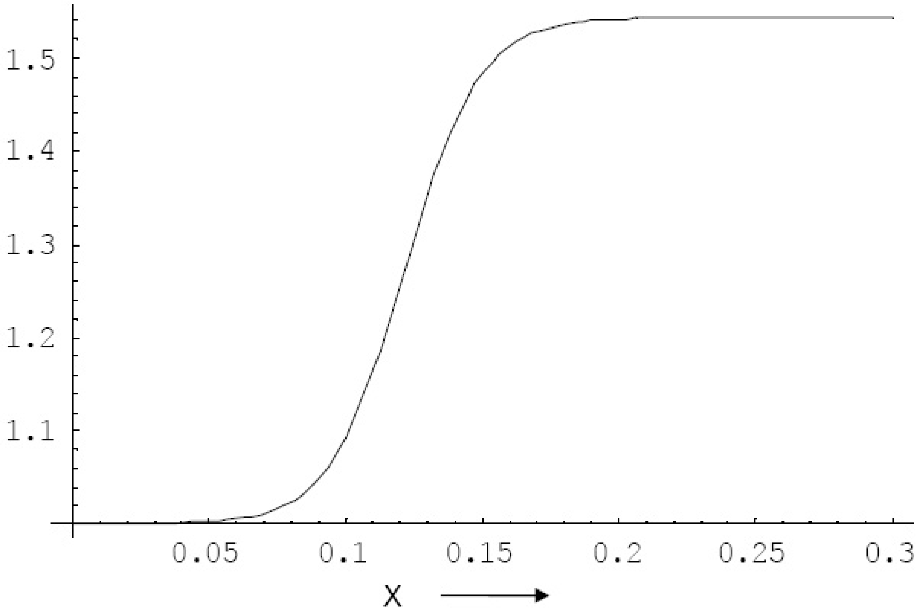

Fig. 27. Function for which Obtaining Infemum is Required, in Order to Compute $K^*$

Since the $K^*$ value can give us the probability of the workload in the system being greater than any given value, we can obtain the upper bound on the expected workload in the system as $K^*/\eta$. Computing this, in this case, we obtain $W \leq 1.156 \times 10^6$, or the upper bound on the workload in the system is $1.156 \times 10^6$.

### IV.5.2.2. Validation

As the first step in validation, we want to check whether the output parameters obtained sufficiently characterize the aggregate flow. We use a simple simulation for this purpose, as described in Section IV.4.2. The remaining workload when the input flows are fed individually was obtained as $4.824 \times 10^5$, whereas that for the aggregate flow was determined to be $4.652 \times 10^5$, a difference of about 4%. Hence, the simulation shows that the aggregate flow reflects the performance metrics of the superposition of the individual flows.

Using effective bandwidth approximation is a well-known methodology to obtain the expected workload in a system. We will demonstrate the use of this methodology in our analysis here, and compare it to the workload obtained using simulation. The solution methodology has been described in Section IV.4.3.

Using the inflow parameters given in Table XII, we use Equation (4.4) to write the generator matrix $Q$ as

$$Q = \begin{bmatrix} -295.49 & 0 & 295.453 & 0.037 \\ 0 & -91.562 & 91.551 & 0.011 \\ 44.104 & 29.761 & -73.865 & 0 \\ 0.00105 & 0.00071 & 0 & -0.00176 \end{bmatrix}.$$

Since each of the input channel capacities is 100 units, the rate matrix $R$ of each flow can be written as

$$R = diag\left\{ \begin{matrix} 0 & 0 & 100 & 100 \end{matrix} \right\}.$$

We would have the same effective bandwidth for each of the 100 inflows. Then, solving Equation (4.12) and Equation (4.14), we obtain $\upsilon = 1.24 \times 10^{-6}$ for each flow. Thus, we obtain the expected workload in the system as $W = 1/1.24 \times 10^{-6} = 8.071 \times 10^5$.

We have computed the expected workload in the system using the proposed methodology as $1.156 \times 10^6$, whereas the effective bandwidth approximation yields the value as $8.071 \times 10^5$. That is, the workload predicted by our method is about 40% higher than that predicted by effective bandwidth approximation. Simulation results validate that the workload computed using the input flows is about the same as that computed using the aggregate flow, suggesting that the aggregate flow sufficiently represents the super-position of the input flows.

### IV.5.2.3. Obtaining Output Bandwidth

In this example, we have assumed that input and output channel capacities are given. Now, we consider the related design problem, in which the output channel capacity is to be computed in order to satisfy some QoS requirement. It is possible that some or all of the bandwidth capacity can be purchased/sold/reallocated by the network administrators.

In this case, the output channel capacity $C$ from the router is not known. However, the maximum expected workload for the flow is set. There could also be an upper bound set on the expected traffic intensity of the aggregate flow as well, if it is to flow to some other internal node(s). In this case, we can solve the optimization problem for different values of $C$ and obtain the required parameter using interpolation or extrapolation.

This time, we consider having 1,000 input streams with parameters given in Table XII, with input channel capacity of 100 units each. Assume that the maximum expected workload is set at $1 \times 10^7$. Moreover, let the maximum expected output traffic intensity be set at 98%. In order to solve this problem, we will have to use an iterative approach, in which we will obtain the expected workload for a given $C$, and pick the one that satisfied the constraints.

In order to obtain the workload, given the aggregate flow parameters, we would use the fluid-model upper bounds, as mentioned in Section IV.4.1. We would use the methodology similar to the calculation presented in Section IV.5.2.1. We compute the aggregate flow parameters and remaining workload for different output channel capacities, and the plot is presented in Fig. 28. The system quickly becomes unstable for lower values of $C$, and the waiting time goes down to zero as the channel capacity $C$ is increased. Using this figure, we obtain the value of $C$ for which the workload is

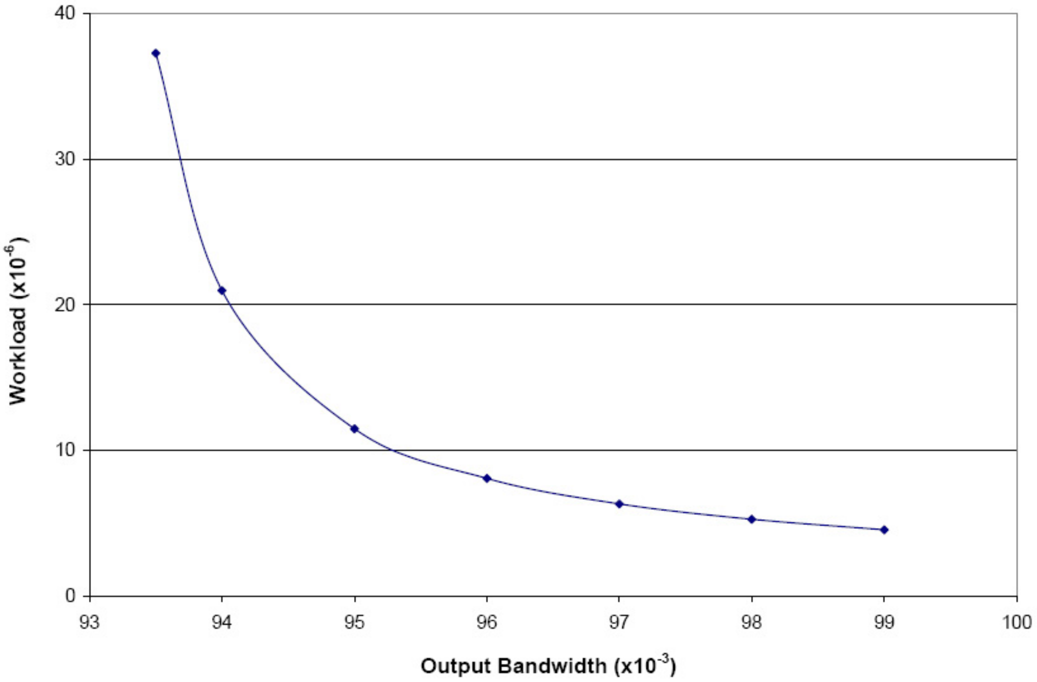

Fig. 28. Remaining Workload as a Function of Output Channel Capacity

less than the QoS requirement as $9.533 \times 10^4$. The traffic intensity for the aggregate flow is computed as 97.44%, which is less than 98%. Hence, this also satisfies the QoS requirement, and the output channel capacity of $9.533 \times 10^4$ can be picked as a solution to this example problem.

### IV.5.3. Multiple Layers of Nodes

Now, we consider the case with multiple layers of routers in the network, as opposed to the single layer that we have analyzed so far. Most real networks are represented more closely by Fig. 24, which represents an in-tree sort of network. Here, a relatively small number of flows merge at each node. Output flows from these nodes get merged at another node, within the system domain itself. This merging of network traffic can occur at two or more levels before the traffic gets transmitted outside.

#### IV.5.3.1. Computing Expected Workload

If the objective is to obtain the expected workload at the bottleneck node and to characterize the output flow, we can use the proposed methodology by solving for one node at a time. Based on the example presented in Section IV.5.2, the output flow parameters of a node can be computed if the input flow parameters and the channel capacities are known. Hence, starting with the upstream nodes, we can work our way to the final node that connects to the outside world, which usually is the bottleneck node. Once we have the input flow parameters for this node, the expected workload and the aggregate flow parameters can be computed.

#### IV.5.3.2. Design with Multiple Layers of Nodes

We now consider a design problem related to the problem in Section IV.5.3. Although generally, edge nodes in a network are the bottleneck, non-access nodes can also be the hot-spots in many cases. Moreover, there is no easy way to obtain the critical path amongst these non-access nodes [6].

Consider a case in which the capacities of some of the internal links need to be determined. This could be the case when setting up a new system or when reallocating bandwidth when new QoS requirements are presented. The idea is to use only as little bandwidth as necessary, leaving the rest for other potential applications.

Consider the scenario shown in Fig. 29. This system has four nodes at the input level, where traffic from three input streams merge at each node. The output from these four nodes merge at another node, from which, the traffic is transmitted outside. Hence, there are a total of 12 input streams, and the aggregate data from these is transmitted outside via a two-layer, in-tree network. The small numbers in the example are considered for demonstration; a real network might consist of a large

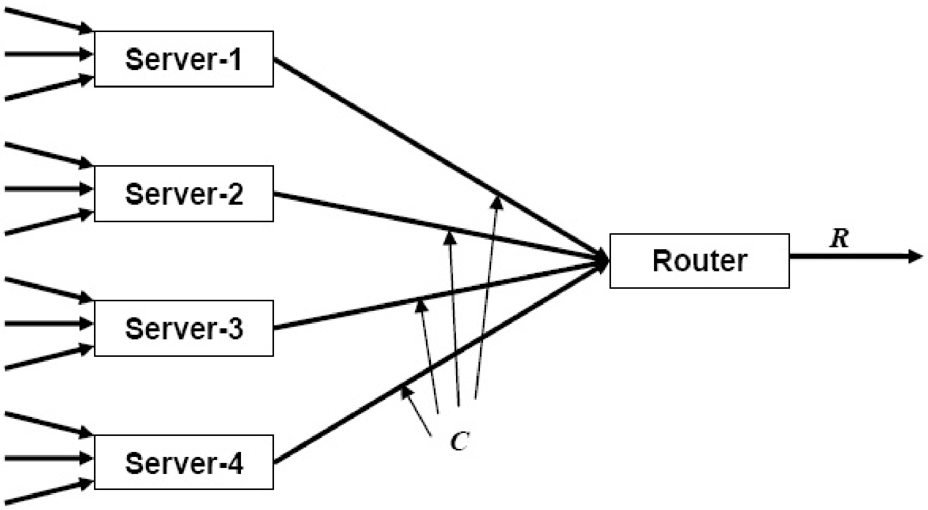

Fig. 29. A Small Multi-Layer Network Where Internal Link Capacities Need To Be Determined

number of input streams and layers. However, the methodology used to solve the problem would essentially remain unchanged.

For the given network, the bandwidth of incoming flows and that of the final outgoing link are known, whereas the capacities of the internal links need to be determined. We will let all these links have the same capacity, and they are marked with a bandwidth of $C$ in Fig. 29.

The parameters of the distributions defining the input flows are given in Table XIII. We assume for now that all of the input flows follow the same distribution. These parameters were obtained from a real trace with a traffic intensity of 80%. Let all of these input flows have the same channel capacity of 100 units each. If we know the output capacity of each node, we can perform the optimization (as presented in Section IV.4.1) to obtain the output flow distribution and the expected workload. In this case however, we can obtain the aggregate flow parameters for a given value of $C$ and compute the expected workload. Then, given the QoS upper limit on the expected workload, we can obtain the value of channel capacity $C$ by plotting expected workload against the unknown capacities.

**TABLE XIII** Input Flow Parameters for the Multi-Layer Network

| $p$ | $\alpha_1$ | $\alpha_2$ | $q$ | $\beta_1$ | $\beta_2$ |
|---|---|---|---|---|---|
| 0.99994953 | 73.70559 | 0.0046258 | 0.605761 | 301.3551 | 95.926 |

In this example, we assume that the output channel capacity $R = 1,000$ units, and that the upper bound on the expected waiting time at the output node equal to 100 units. We have chosen this node because usually, this is the bottleneck node for the system. Note that as the channel capacities for internal links increase, expected waiting time for this node will increase. Hence, in this example, we will come up with the maximum channel capacity: anything less would be acceptable for the given QoS criterion. Now, using different values of $C$, we obtain the expected waiting time at this node. These values are plotted in Fig. 30. From this plot, we obtain the required value of the bandwidth of internal links as approximately $C = 255.8$. Note that for $C$ values of less than 250, the waiting time would be zero because the output channel capacity would be greater than the sum of all the input channel capacities.

## IV.6. Conclusions

Aggregate traffic models are essential in performance evaluation studies on large-scale networks. In this research work, we have proposed a methodology to characterize the aggregate flow when a number of input sources, each with a potentially different channel capacity, merges into a buffer/router.

In this research work, we modeled the discrete trace data as fluid ON-OFF source. Multiple such flows were passed through a buffered queue and the aggregate flow

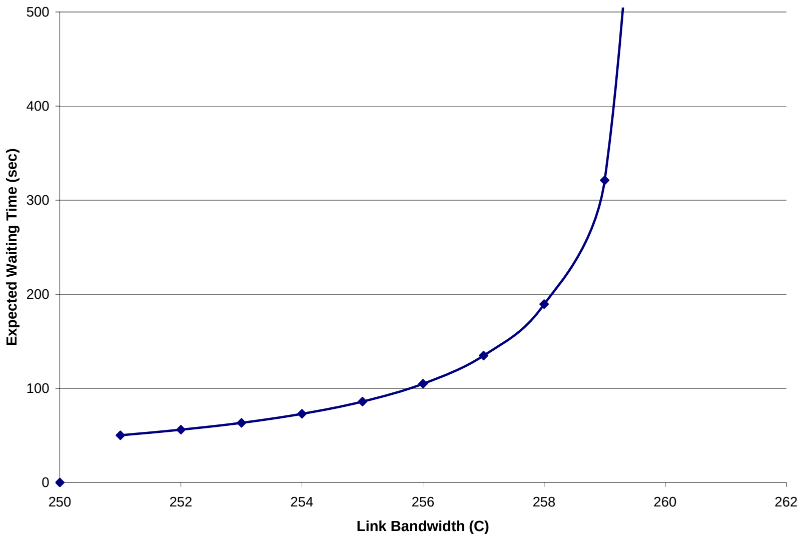

Fig. 30. Expected Workload as a Function of Internal Link Capacity

was also characterized as an ON-OFF flow. This not only helps in obtaining the performance metrics at the node where the flows merge, but also in evaluating a multi-layered network where such a multiplexing may be happening at different levels. The methodology we propose can be used in the analysis of a given network to assess the QoS guarantees, and can easily be extended and used while designing a network, to compute the link capacities.

The results obtained using the proposed methodology were validated by the means of a simulation. The waiting times were also computed using effective bandwidth approximations.

We have demonstrated the ease of use of the proposed methodology by means of several numerical examples. The performance measure at different levels of the network was computed, helping to evaluate the bandwidth requirements for the internal links as well. That is, we could use this methodology to obtain the required internal and external channel capacities in a multi-layer network, in order to use the available resources effectively, and promise and evaluate the QoS requirements.

# CHAPTER V

# CONCLUSIONS AND FUTURE RESEARCH

In this research work, we have considered a typical Web-server farm and analyzed it for resource allocation and performance measures. We consider streaming and elastic requests coming into the system, decide which ones to server, and then evaluate the QoS the system would be providing. In order to evaluate the performance measures, we have taken trace data and converted it into a fluid ON-OFF source, before analyzing it. The problem is broken into three parts. All of these sub-problems present separate challenges, and have been dealt with in different chapters. We present in this chapter the conclusions and contributions from these parts, followed by some possible future directions.

## V.1. Summary and Contributions

In the first part (Chapter II), we solve the admission-control problem for the Web-server to decide which of the requests to be allowed into the system. Given the capacity (bandwidth) constraints of a Web farm, it is generally not possible to serve all of the incoming requests without sacrificing the quality of service. The contributions from this part of the dissertation are:

1. We were able to prove some structural properties, such as submodularily and concavity, of the value function defining the problem. This led us to analytically prove that the optimal admission-control policy would be of the form of a switching curve.

2. Using this structure of the optimal policy, we have proposed a simple method to obtain the optimal policy without using value-iteration algorithm. This helps

in obtaining the solution faster and provides the ability to solve large problems.

3. Given the optimal admission-control policy, we obtained the QoS metric (blocking probability) for the streaming requests arriving at the system and showed that the optimal policy not only provided increased revenue, but also improved customer service.

Elastic network traffic entering the system is very difficult to characterize and obtain the performance measures of, as it is known to be bursty and self-similar in nature. In Chapter III, we used real trace data and demonstrated how stochastic fluid-flow models can be used to compute the bounds on the performance measures. The contributions from this part are following:

1. We used trace data and converted it into a fluid ON-OFF source. It was shown that the bursty nature of self-similar network traffic is captured well by the stochastic fluid-flow models, especially using exponential family distributions.

2. We showed that the $G/G/1$ approximation severely underestimates the expected waiting time in the system, especially at high traffic intensities. In contrast, the upper bound obtained using the fluid-flow model was more conservative and much better approximations of the performance metric.

3. At medium to low traffic intensities, we have shown that exponential ON-OFF is a very good choice for combining conservativeness and accuracy. This is significant because of the ease of computing the exponential approximation.

As this traffic flows through the communication network within the Web-server farm, it passes through various routers/switches, where traffic from other servers merges to it. This may happen at multiple levels before data is relayed to the user. We have considered the problem of obtaining performance measures in this network, where traffic from several sources merge. Following are the contributions from this part:

1. We have proposed a methodology to characterize the aggregate flow coming out of a buffered queue of network traffic. The aggregate flow was modeled using the same distributions as the input traffic, which helps in extending the method for cases with several layers of nodes. The methodology was validated using trace data and we have shown that this model yields good characterizations.

2. Using the proposed methodology, we were able to compute the performance measure at nodes where traffic from a multitude of traffic sources merge. This helps in evaluating QoS guarantees.

3. We were also able to obtain bandwidth requirements for various links within the network in order to be able to satisfy QoS requirements. This would help in designing a system, where it is easy to reallocate bandwidth to critical links and utilize the available resources efficiently.

## V.2. Future Directions

The models and solution methodologies presented in this dissertation can be extended to consider generalizations and complexities as mentioned below.

1. *Admission control for elastic class*: We have obtained the admission control policy in Chapter II only for the streaming class requests. Obtaining this kind of control for elastic requests might lead to better service for the customers being allowed to enter the system. It will be an interesting problem to obtain the optimal admission control policy, when control is exercised on both classes of customers.

2. *Consider other classes of network traffic*: Throughout this research, we have considered streaming and elastic classes, with loss and delay QoS parameters, respectively. This work can be extended to incorporate different classes of traffic which involves different QoS metrics. For example, traffic involving playback-type applications could be considered that have jitter based QoS requirements, in which, the packet needs to arrive before playback point, but the application can tolerate some loss.

3. *Incorporating failures*: In the server farm we considered, it was assumed that full capacity of any link is always available. However, this may not always be the case. Different facets of incorporating failure of links can be considered, such as rerouting traffic through active links, or obtaining a robust admission control policy that would have minimum impact on QoS.

4. *Splitting traffic in various links*: We considered the problem where traffic from various nodes merges in a buffered queue and characterized the aggregate traffic. It is possible that at some nodes, traffic is routed to more than one links. Characterizing traffic under such conditions would help in otaining performance metrics in such complex networks that do not have the in-tree structure.

5. *Using different traffic data*: In our analysis of the elastic traffic, we have used Web server trace data and modelled it as fluid. It will be interesting to see how the fluid-flow models perform when different traffic data, such as that in mobile networks, is used.

6. *Performing hardware experiments*: We obtained the aggregate flow parameters in Chapter IV for ON-OFF sources. This aggregate flow characterization exercise could be carried out in sync with hardware experiments, where the actual

flow data within such a network was measured and compared against the values obtained using the methodology we have proposed.

# REFERENCES

[1] A. Iyengar, E. MacNair, and T. Nguyen, “An analysis of web server performance,” in *Proc. IEEE GLOBECOM ’97*, Phoenix, AZ, 1997, pp. 1943–1947.

[2] W. van der Weij, S. Bhulai, and R. van der Mei, “Dynamic thread assignment in web server performance optimization,” *Performance Evaluation*, vol. 66, no. 6, pp. 301 – 310, 2009.

[3] J. Yang, D. Jin, Y. Li, K. Hielscher, and R. German, “Modeling and simulation of performance analysis for a cluster-based web server,” *Simulation Modeling Practice and Theory*, vol. 14, no. 2, pp. 188 – 200, 2006.

[4] V. Gupta, M. H. Balter, K. Sigman, and W. Whitt, “Analysis of join-the-shortest-queue routing for web server farms,” *Performance Evaluation*, vol. 64, no. 9-12, pp. 1062 – 1081, 2007.

[5] R. Shumway and S. Stoffer, *Time Series Analysis and Its Applications.* New York: Springer, 2000.

[6] A. Akella, S. Seshan, and A. Shaikh, “An empirical evaluation of wide-area internet bottlenecks,” in *IMC ’03: Proc. 3rd ACM SIGCOMM Conference on Internet Measurement*, New York, 2003, pp. 101–114.

[7] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks.* New York: Springer-Verlag, 1995.

[8] J. S. Kaufman, “Blocking in shared resource environment,” *IEEE Transactions on Communications*, vol. 29, no. 10, pp. 1481–1494, 1981.

[9] Z. Dziong and L. G. Mason, “Call admission and routing in multi-service loss networks,” *IEEE Transactions on Communications*, vol. 42, no. 3, pp. 2011–2022, 1994.

[10] A. Chandra, W. Gong, and P. Shenoy, “Dynamic resource allocation for shared data centers using online measurements,” in *Proc. Eleventh IEEE/ACM International Workshop on Quality and Service*, Monterey, CA, June 2003, pp. 381–400.

[11] T. Abdelzaher, K. G. Shin, and N. Bhatti, “Performance guarantees for web-server end-systems: A control-theoretical approach,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 80–96, January 2002.

[12] C. Lu, G. Alvarez, and J. Wilkes, “Aqueduct: Online data migration with performance guarantees,” in *Proc. Conference on File and Storage Technologies*, Berkeley, CA, January 2002, pp. 219–230.

[13] J. Chase, D. Anderson, P. Thakkar, A. Vahdat, and R. Doyle, “Managing energy and server resources in hosting centers,” in *Proc. Eighteenth ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001, pp. 103–116.

[14] L. Massoulié and J. Roberts, “Bandwidth sharing: Objectives and algorithms,” *IEEE Transactions on Networking*, vol. 10, no. 3, pp. 320–328, 2002.

[15] G. Fodor, S. Rácz, and M. Telek, “On providing blocking probability- and throughput guarantees in a multi-service environment,” *International Journal of Communication Systems*, vol. 15, pp. 257–285, 2002.

[16] S. R. Mahabhashyam and N. Gautam, “Dynamic resource allocation of shared data centers supporting multiclass requests,” in *Proc. International Conference on Autonomic Computing*, New York, 2004, pp. 222–229.

[17] E. Altman, “Applications of Markov decision processes in communication networks : A survey,” in *Handbook of Markov Decision Processes: Methods and Applications* (E. A. Feinberg and A. Shwartz, eds.), pp. 489–535, Dordrecht, Netherlands: Kluwer, 2001.

[18] S. Bhatnagar and I. B. Reddy, “Optimal threshold policies for admission control in communication networks via discrete parameter stochastic optimization,” *Telecommunication Systems*, vol. 29, no. 1, pp. 9–31, 2005.

[19] S. Singh and D. Bertsekas, “Reinforcement learning for dynamic channel allocation in cellular telephone systems,” in *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), vol. 9, pp. 974–980, Cambridge, MA: The MIT Press, 1997.

[20] P. Marbach and J. Tsitsiklis, “A neuro-dynamic programming approach to call admission control in integrated service networks: The single link case,” Tech. Rep. LIDS-P-2402, Laboratory for Information and Decision Systems, MIT, November 1997.

[21] S.-M. Senouci, A.-L. Beylot, and G. Pujolle, “Call admission control in cellular networks: A reinforcement learning solution,” *International Journal of Network Management*, vol. 14, no. 2, pp. 89–103, 2004.

[22] S. Shenker, “Fundamental design issues for the future internet,” *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 7, pp. 1176–1188, 1995.

[23] S. A. Lippman, “Applying a new device in the optimization of exponential queueing systems,” *Operations Research*, vol. 23, pp. 687–710, 1975.

[24] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* New York: John Wiley and Sons, 1994.

[25] G. M. Koole, "Structural results for the control of queueing systems using event-based dynamic programming," *Queueing Systems*, vol. 30, pp. 323–339, 1998.

[26] E. Porteus, "Conditions for characterizing the structure of optimal strategies in infinite-horizon dynamic programs," *Journal of Optimization Theory and Applications*, vol. 36, pp. 419–432, 1982.

[27] E. Altman, T. Jimenez, and G. Koole, "On optimal call admission control," in *Proc. 37th IEEE Conference on Decision and Control*, Tampa, FL, December 1998, pp. 569–574.

[28] E. L. Ormeci, A. Burnetas, and J. van der Wal, "Admission policies for a two class loss system," *Stochastic Models*, vol. 17, no. 4, pp. 513–539, 2001.

[29] R. Bellman, "A Markov decision process," *Journal of Mathematics and Mechanics*, vol. 6, pp. 679–684, 1957.

[30] W. Willinger, M. S. Takku, and A. Erramilli, "A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks," in *Stochastic Networks: Theory and Applications* (F. P. Kelly, S. Zachary, and I. Ziedins, eds.), pp. 339–366, Oxford: Oxford University Press, 1996.

[31] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning," in *IMW '02: Proc. 2nd ACM SIGCOMM Workshop on Internet Measurement*, New York, 2002, pp. 91–92.

[32] Z. Sahinoglu and S. Tekinay, “On multimedia networks: Self-similar traffic and network performance,” *IEEE Communications Magazine*, vol. 37, no. 1, pp. 48–52, 1999.

[33] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, “On the self-similar nature of ethernet traffic,” in *ACM SIGCOMM* (D. P. Sidhu, ed.), San Francisco, CA, 1993, pp. 183–193.

[34] V. Paxson and S. Floyd, “Wide area traffic: The failure of Poisson modeling,” *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[35] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, “Self-similarity through high-variability: Statistical analysis of ethernet LAN traffic at the source level,” *ACM SIGCOMM Computer Communications Review*, vol. 25, pp. 100–113, October 1995.

[36] V. Frost and B. Melamad, “Traffic modeling for telecommunications networks,” *IEEE Communications Magazine*, vol. 32, pp. 70–80, 1994.

[37] D. L. Jagerman, B. Melamad, and W. Willinger, “Stochastic modeling of traffic processes,” in *Frontiers in Queueing: Models and Applications in Science and Engineering*, pp. 271–320, Boca Raton, FL: CRC Press Inc., 1997.

[38] D. Anick, D. Mitra, and M. M. Sondhi, “Stochastic theory of a data handling system with multiple sources,” *Bell Systems Technical Journal*, vol. 61, pp. 1871–1894, 1982.

[39] H. Heffes and D. M. Lucantoni, “A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance,” *IEEE Journal on Selected Areas in Communications*, vol. 4, pp. 856–868, 1986.

[40] D. M. Lucantoni, “The BMAP/G/1 queue,” in *Models and Techniques for Performance Evaluation of Computer and Communication Systems* (L. Donatiello and R. Nelson, eds.), pp. 330–358, New York: Springer-Verlag, 1993.

[41] Å. Arvidsson and R. Harris, “Performance comparison of models of individual and merged bursty traffics,” in *Proc. Tenth Nordic Teletraffic Seminar*, Århus, Denmark, 1992, pp. 185–192.

[42] B. B. Mandelbrot, “Self-similar error clusters in communication systems and the concept of conditional systems and the concept of conditional stationarity,” *IEEE Transactions on Communications Technology*, vol. 13, pp. 71–90, 1965.

[43] V. Klemeš, “The hurst phenomenon: A puzzle?” *Water Resources Research*, vol. 10, pp. 675–688, 1974.

[44] L. S. Liebovitch, “Testing fractal and Markov models of ion channels kinetics,” *Biophysics Journal*, vol. 55, pp. 373–377, 1989.

[45] M. E. Crovella and A. Bestavros, “Self-similarity in world wide web traffic: Evidence an possible causes,” *IEEE/ACM Transaction on Networking*, vol. 5, no. 6, pp. 835–846, 1997.

[46] M. T. Lucas, D. E. Wrege, B. J. Dempsey, and A. C. Weaver, “Statistical characterization of wide-area IP traffic,” in *Proc. Sixth International Conference on Computer Communications and Networks*, Las Vegas, NV, 1997, pp. 442–447.

[47] T. Kushida, “The traffic measurement and the empirical studies for the internet,” in *Global Telecommunications Conference*, vol. 2, Sydney, Australia, 1998, pp. 1142–1147.

[48] W. Willinger and V. Paxson, “Where mathematics meets the internet,” *Notices of the American Mathematical Society*, vol. 45, no. 8, pp. 961–970, 1998.

[49] E. Gelenbe, V. Srinivasan, S. Seshadri, and N. Gautam, “Optimal policies for ATM cell scheduling and rejection,” *Telecommunication Systems*, vol. 18, no. 4, pp. 331–358, 2001.

[50] C. Partridge, “The end of simple traffic models,” *IEEE Network, Editor’s Note*, vol. 7, p. 3, September 1993.

[51] V. Ramaswami, “Traffic performance modeling for packet communications: Whence where and whither,” keynote address in *Proc. 3rd Australian Teletraffic Seminar*, Adelaide, Australia, 1988.

[52] R. G. Addie, M. Zukerman, and T. D. Neame, “Broadband traffic modeling: Simple solutions to hard problems,” *IEEE Communications Magazine*, vol. 36, pp. 88–95, August 1998.

[53] N. Gautam and S. Seshadri, “Approximations for system performance under self-similar traffic,” in *3rd International Conference on Telecommunications and Electronic Commerce*, Dallas, TX, November 2000, pp. 239–250.

[54] N. Gautam and S. Seshadri, “Performance analysis for e-business: Impact of long range dependence,” *Electronic Commerce Research*, vol. 2, no. 3, pp. 233–253, 2002.

[55] A. I. Elwalid and D. Mitra, “Effective bandwidth of general Markovian traffic sources and admission control of high-speed networks,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 329–343, 1993.

[56] G. L. Choudhury, D. M. Lucantoni, and W. Whitt, “On the effectiveness of effective bandwidths for admission control in ATM networks,” in *Proc. ITC-14, Elsevier Science*, North Holland, 1994, pp. 411–420.

[57] W. Whitt, “Tail probabilities with statistical multiplexing and effective bandwidth for multiclass queues,” *Telecommunication Systems*, vol. 2, pp. 71–107, 1993.

[58] A. I. Elwalid, D. Heyman, T. V. Lakshman, D. Mitra, and A. Weiss, “Fundamental bounds and approximations for ATM multiplexers with applications to video teleconferencing,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1004–1016, 1995.

[59] A. I. Elwalid and D. Mitra, “Analysis, approximations and admission control of a multi-service multiplexing system with priorities,” *INFOCOM*, vol. 2, pp. 463–472, 1995.

[60] S. M. Ross, “Bounds on the delay distribution in $G/G/1$ queues,” *Journal of Applied Probability*, vol. 11, pp. 417–421, 1974.

[61] D. Artiges and P. Nain, “Upper and lower bounds for the multiplexing of multiclass Markovian on/off sources,” *Performance Evaluation*, vol. 27 and 28, pp. 673–698, 1996.

[62] A. I. Elwalid and D. Mitra, “Analysis and design of rate-based congestion control of high speed networks, part I: Stochastic fluid models, access regulation,” *Queueing Systems - Theory and Applications*, vol. 9, pp. 29–64, 1991.

[63] T. Yang and D. H. K. Tsang, “A novel approach to estimating cell loss prob-

ability in an ATM multiplexer loaded with homogeneous on-off sources," *IEEE Transactions on Communications*, vol. 43, no. 1, pp. 117–126, 1995.

[64] Z. Liu, P. Nain, and D. Towsley, "Exponential bounds with application to call admission," *Journal of the Association for Computing Machinery*, vol. 44, pp. 366–394, 1997.

[65] S. Aalto, "Characterization of the output rate process for a Markovian storage model," *Journal of Applied Probability*, vol. 35, no. 1, pp. 184–199, 1998.

[66] Z. Palmowski and T. Rolski, "A note on martingale inequalities for fluid models," *Statistics and Probability Letters*, vol. 31, no. 1, pp. 13–21, 1996.

[67] Z. Palmowski and T. Rolski, "The superposition of alternating on-off flows and a fluid model," *The Annals of Applied Probability*, vol. 8, no. 2, pp. 524–540, 1998.

[68] N. Gautam, V. Kulkarni, Z. Palmowski, and T. Rolski, "Bounds for fluid models driven by semi-Markov inputs," *Probability in the Engineering and Information Sciences*, vol. 18, no. 4, pp. 429–475, 1999.

[69] J. A. Buzacott and J. G. Shanthikumar, *Stochastic Models of Manufacturing Systems.* Englewood Cliffs, NJ: Prentice Hall, 1993.

[70] J. D. C. Little, "A proof of the queueing formula: $L = \lambda W$," *Operations Research*, vol. 9, pp. 383–387, 1961.

[71] A. Feldmann and W. Whitt, "Fitting mixtures of exponentials to long-tail distributions to analyze network performance models," *Performance Evaluation*, vol. 31, no. 3-4, pp. 245–279, 1998.

[72] A. I. Elwalid and D. Mitra, "Fluid models for the analysis and design of statistical multiplexing with loss priorities on multiple classes of bursty traffic," *IEEE Transactions on Communications*, vol. 42, no. 11, pp. 2989–3002, 1992.

[73] V. G. Kulkarni, "Fluid models for single buffer systems," in *Frontiers in Queueing: Models and Applications in Science and Engineering*, , pp. 321–338, Boca Raton, FL: CRC Press Inc., 1997.

[74] S. Teymori and W. Zhuang, "Queue analysis and multiplexing of heavy-tailed traffic in wireless packet data networks," *Mobile Networks and Applications*, vol. 12, pp. 31–41, February 2007.

[75] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: Statistical analysis of ethernet LAN traffic at the source level," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 71–86, 1997.

[76] K. Park, G. Kimy, and M. Crovellaz, "On the relationship between file sizes, transport protocols, and self-similar network traffic," in *Proc. 4th International Conference on Network Protocols*, Columbus, OH, 1996, pp. 171–180.

[77] M. Mandjes and A. Riddler, "A large deviations analysis of the transient of a queue with many Markov fluid inputs: Approximations and fast simulation," *ACM Transactions on Modeling and Computer Simulations*, vol. 12, no. 1, pp. 1–26, 2002.

[78] K. Sriram and W. Whitt, "Characterizing superposition arrival processes in packet multiplexers for voice and data," *IEEE Journal on Selected Areas in Communication*, vol. 4, pp. 833–846, September 1986.

[79] R. Cáceres, P. G. Danzig, S. Jamin, and D. J. Mitzel, "Characteristics of

wide-area TCP/IP conversations," *Computer Communications Review*, vol. 21, pp. 101–112, 1991.

[80] A. Feldmann, "Modeling characteristics of TCP connections," Tech. Rep., AT&T Laboratories, Florham Park, NJ, 1996.

[81] H. Hassan and J.-M. Garcia, "Aggregate modeling for TCP sessions," in *Proc. 2nd ACM International Workshop on Wireless Multimedia Networking and Performance Modeling*, New York, October 2006, pp. 73–78.

[82] H. Hassan, J.-M. Garcia, and C. Bochstal, "Aggregate traffic models for VoIP applications," in *International Conference on Digital Telecommunications*, Washington, DC, 2006, p. 70.

[83] K. Ravindran and T. J. Gong, "Resource allocation control protocols for multicast data transport," in *Sixth International Conference on Computer Communications and Networks*, Las Vegas, NV, 1997, pp. 182–188.

[84] B. Budiardjo, B. A. A. Nazief, and D. Hartanto, "Delay based bandwidth allocation in aggregate traffic," in *Communications and Computer Networks* (M. H. Hamza, ed.), Cambridge, MA, 2002, pp. 200–205.

[85] Z. Wang and A. Basu, "Resource allocation for elastic traffic: Architecture and mechanisms," in *Network Operations and Management Symposium*, Honolulu, HI, IEEE/IFIP, 2000, pp. 157–170.

[86] S. Shah-Heydari and T. Le-Ngoc, "MMPP modeling of aggregated ATM traffic," in *IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 1, Waterloo, ON, Canada, 1998, pp. 129–132.

[87] S. Aalto, "Output of a multiplexer loaded by heterogeneous ON-OFF sources," *Stochastic Models*, vol. 14, no. 4, pp. 993–1005, 1998.

[88] N. K. Boots and M. Mandjes, "Fast simulation of a queue fed by a superposition of many (heavy-tailed) sources," *Probability in the Engineering and Informational Sciences*, vol. 16, no. 2, pp. 205–232, 2002.

[89] F. Alagoz, "Approximations on the aggregate MPEG video traffic and their impact on admission control," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 10, no. 1, pp. 73–84, 2002.

# VITA

Piyush Goel received his Bachelor of Technology degree in civil engineering from Indian Institute of Technology, Bombay, Mumbai, India, in June 2000. After working with Infosys Technologies Ltd. for a short period of time, he initially joined the Department of Engineering Science and Mechanics, and later changed to the Industrial and Manufacturing Engineering Department, at the Pennsylvania State University. He transferred to Texas A&M University the fall of 2005 to pursue a Doctor Of Philosophy degree in industrial engineering. He received the degree in August of 2009. His research interests include Markovian Processes, Applied Probability, Probabilistic Optimization and Queueing Theory.

Piyush Goel may be reached at Department of Industrial and Systems Engineering, c/o Dr. Natarajan Gautam, Texas A&M University, College Station, TX 77843-3131. His email address is pg@tamu.edu.