Statistical Characterization for Per-Hop QoS*

Mohamed El Gendy, Abhijit Bose, Haining Wang, and Kang G. Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
{mgendy,abose,hxw,kgshin}@eecs.umich.edu

Abstract

The Differentiated Services (DiffServ) architecture is designed to provide scalable network-level Quality of Service (QoS) via service differentiation at intermediate nodes of a network (called *Per-Hop Behaviors* (PHBs)). Per-hop QoS is measured in terms of throughput, delay, jitter, and loss rate experienced by traffic crossing a PHB. In this paper, we use a statistical approach that is based on experiments on a real network testbed to characterize the per-hop QoS of a given PHB. Specifically, we employ a full factorial statistical design of experiments to study the effects of different PHB configurations and input traffic scenarios on per-hop QoS. We use Analysis of Variance (ANOVA) to identify the input and PHB configuration parameters that have the most significant influence on per-hop QoS. Then, multiple regression analysis is applied to construct models for the per-hop QoS with respect to these parameters. The overall approach is shown to be effective and capable of characterizing any given PHB, within the ranges of the experiments, and for construction of functional relationships for the PHB output parameters. We are also able to identify the operational differences between different realizations of a given PHB. The approach in this paper forms a "fundamental" step towards achieving predictable end-to-end QoS when applying statistical QoS control at intermediate nodes.

1 Introduction

Providing Quality of Service (QoS) in large-scale IP networks is the main motivation behind the Differentiated Services (DiffServ) architecture [1, 2]. In this architecture, packets entering a DiffServ-enabled network are marked with different DiffServ Code Points (DSCPs), and based on these markings, they are subject to classification, traffic conditioning (such as metering, shaping, and policing), as well as to a small set of packet forwarding techniques called *Per-Hop Behaviors* (PHBs). The DiffServ architecture achieves scalability by performing traffic conditioning and per-flow management at edge routers only, and by applying PHBs to traffic aggregates at core routers. Neither per-flow states nor signaling are required at core routers. This, in effect, converts the large number of flows at the edge into a small set of *Behavior Aggregates* (BAs) or services across the DiffServ network.

The IETF DiffServ Working Group has been standardizing the building blocks of the DiffServ architecture, and a set of PHBs have been proposed in [1, 3, 4]. One of these PHBs, The Expedited Forwarding (EF), is a building block for low loss, low delay, and low jitter services. On routers equipped with an EF PHB, incoming packets marked with EF DSCP are expected to encounter short or empty queues. The Assured Forwarding (AF) PHB is proposed to offer different levels of forwarding assurance in terms of packet loss. Under AF, each

*The work reported in this paper was supported in part by Samsung Electronics and the ONR under Grant N00014-99-0465.

packet is marked with drop precedence according to a specific marking rule or algorithm. At core routers, if the network is congested, the packets with higher precedence are preferentially dropped before packets with lower precedence.

Service differentiation is achieved by allocating different amounts of network resources, such as link bandwidth and buffers, to different types of traffic traversing the DiffServ-enabled routers. The output of this service differentiation at each router in terms of throughput, delay, jitter, and loss of the output traffic, is called *per-hop QoS*. The per-hop QoS can be determined if the resource allocation is well controlled. Given the per-hop QoS for every node along an end-to-end path, the end-to-end QoS perceived by users can be calculated. This is similar to *Integrated Services* [5], which uses RSVP signaling to estimate such end-to-end QoS on a hop-by-hop basis.

A PHB is the key building block of the DiffServ architecture and is realized by a variety of traffic management components, such as queues, schedulers, buffer management, policer, and filters. These components must be properly designed and implemented so that the per-hop QoS guarantees provided by the PHB can be achieved. However, it is a challenging task to assemble these building blocks together, given the complex nature of the interactions among the various components. Network architects often have to estimate per-hop QoS when designing a network or making policy decisions for supporting specific types of IP-based services, e.g., Voice-over-IP or Video-on-Demand. It is also becoming more important for network operators to be able to adjust the configuration parameters of intermediate nodes within their networks to address the needs of specific services being offered, and to respond to dynamic changes of the input traffic. Characterizing a network node accurately is, therefore, an important research problem to be solved in the field of QoS. The main goal of our study is to demonstrate that careful statistical analysis coupled with an experimental framework can effectively characterize any DiffServ PHB, and extract a statistical model that can predict the per-hop QoS for this PHB.

We use a generic quantitative approach for characterizing the per-hop QoS for any PHB realization under a wide range of input traffic and configuration parameters. We first abstract the DiffServ PHBs into sets of inputs, configuration and output parameters. These parameters can be controlled and/or measured via experiments. Then, we design a set of full factorial [6] experiments to measure the effects of the inputs and configuration parameters on the output of a PHB. By performing Analysis of Variance (ANOVA) and regression analysis of the data from these experimental measurements, we can deduce functional relationships of the per-hop QoS parameters and construct a statistical model of the PHB. Our approach has the advantage of capturing all the effects in the PHB rather than ignoring some of the effects as commonly found in assumptions made by traditional analysis methods. Once the PHBs are characterized and the operational limits for these PHBs are determined, they can be concatenated together to build edge-to-edge Per-Domain Behaviors (PDBs). The results thus obtained demonstrate the effectiveness of our approach in capturing the functional dependencies of per-hop QoS on the input and configuration parameters. In addition, they point out the differences in performance guarantees provided by different PHB implementations. The potential benefits of our approach are:

1. Facilitating the control and optimization of the PHB performance. This is done by identifying the most important factors that control a certain PHB and by extracting the relationships between these factors and the output of the PHB (per-hop QoS);

2. In addition to being able to control the PHB, one can use the statistical models of the PHBs, derived in this study, in a per-hop admission control mechanism that contributes to the end-to-end admission control decision.

The remainder of the paper is organized as follows. Section 2 presents our strategy for characterizing a given PHB. Section 3 describes the experimental framework we use. Sample results and analysis of our study are presented in Section 4. In Section 5, we discuss some of the related work and previous studies of DiffServ PHBs and services. We conclude the paper with Section 6, and discuss extensions of our approach to studying larger-scale QoS problems and related protocol-level mappings.

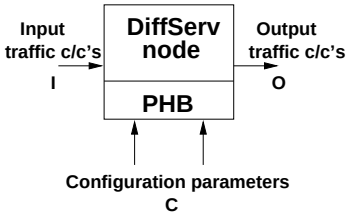


Figure 1: Model of per-hop QoS

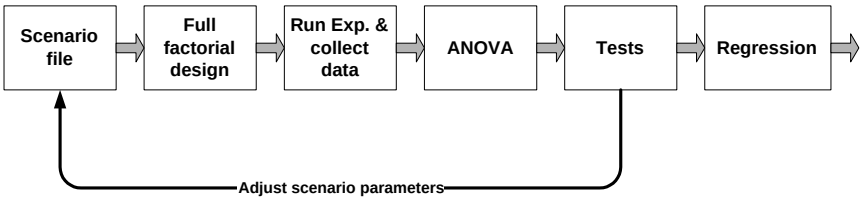


Figure 2: Steps of the statistical approach

2 Approach to Statistical Characterization

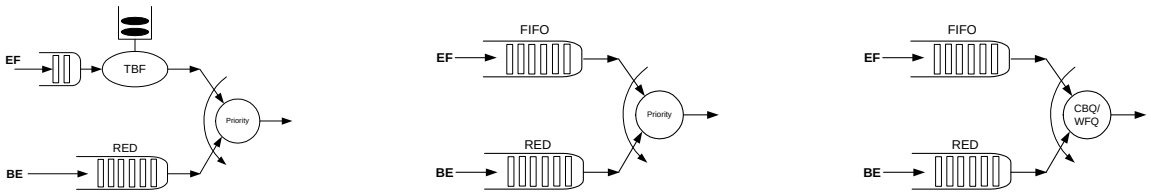
This section describes the approach we have taken to statistically characterize per-hop QoS. As shown in Figure 1, the central idea is to abstract a given PHB as a system that consists of input (I) and output (O) traffic characteristics, and a set of configuration parameters (C). The details of the internal implementation of per-hop QoS are not considered explicitly, but their effects on the overall performance (i.e., the output traffic characteristics) are modeled. We apply statistical analysis methods to identify the most influential parameters affecting the per-hop QoS within the range and domain of our experiments and we estimate their relationships to the output. In the context of our experiments, we refer to O as the per-hop QoS attributes experienced by traffic crossing a DiffServ PHB. The basic steps of our approach are shown in Figure 2, and can be summarized as follows.

Identification of the parameters in I and C that account for most of the effects on O : In experimental design, the parameters of I and C are termed as *input factors* and the attributes of O as *response variables*. The values each factor takes are called *levels*. The most important factors are identified by calculating the percentages of the total output variation caused by the factors and their interactions. We use the Analysis of Variance (ANOVA) in this step.

Construction of appropriate models of the response variables based on the most important factors : This step requires careful validation of the observed data so that the basic assumptions about the models hold for the experimental data. We use regression analysis to find such models after performing suitable transformations of the factors and response variables.

Our approach can also be used to evaluate alternative design choices. The IETF DiffServ Working Group does not specify how to design traffic control elements, such as queues, schedulers and filters for any of the proposed PHBs. One can, therefore, construct a PHB using a variety of traffic conditioning blocks as long as it conforms to the traffic handling guidelines specified in the DiffServ standard. Figures 3(A), 3(B), and 3(C) show different choices for EF and BE PHBs that share a single physical link. While these different implementations are expected to provide similar qualitative behavior of the aggregated output traffic, the functional relationships among the parameters are different due to the internal construction of the traffic handling blocks. We demonstrate this point further through our experiments. Next, we explain why we repeat some of the experiments by adjusting the parameters as shown in Figure 2.

For certain experiments, it is useful to start with a reduced number of levels for the full set of input factors. Once the most significant factors are identified, the number of levels corresponding to these factors are increased with the others fixed. This allows us to construct a higher-order regression model in the second stage of the analysis. This repetition of factorial analysis is shown as the "Adjust scenario parameters" step in Figure 2. Note that our approach and framework for data collection, designing the required set of experiments, and statistical analysis, are general and mostly automated. This is a departure from most recent experimental studies of



(A) An edge-based EF PHB (EF-EDGE)

(B) A core-based EF PHB (EF-CORE)

(C) A CBQ-based EF PHB (EF-CBQ)

Figure 3: Three different realizations for EF PHB sharing the a link with best-effort (BE)

Factor	Symbol
Assured traffic average rate	a_r
Assured traffic peak rate	a_p
Assured traffic burst size	a_b
Assured traffic packet size	a_{pkt}
Assured traffic number of flows	a_n
Assured traffic type of traffic	a_t
Best effort traffic rate ratio	R_{ab}
Best effort traffic burst size	b_b
Best effort traffic packet size	b_{pkt}
Best effort traffic number of flows	b_n
Best effort traffic type of traffic	b_t
Number of input interfaces to the PHB node	NI

Table 1: Symbolic representation of the factors in I

DiffServ EF and AF PHBs for multimedia traffic where a set of experiments were performed on a given testbed using a fixed configuration for traffic forwarding [7, 8].

2.1 Factors and Factor Levels

As mentioned above, the parameters in I and C affect the output response (O) of a PHB. Therefore, we need to determine which parameter should be considered as factors and at what levels. The parameters of I may include both marked and unmarked input traffic.[1] We use the Dual Leaky Bucket (DLB) representation for the input traffic, as shown in Table 1. For the best-effort traffic, the ratio R_{ab} of assured traffic rate to best-effort traffic rate is more meaningful than its absolute value. Since most QoS schemes do not provide any guarantee for best-effort traffic, we are primarily interested in the performance of assured traffic. Note that some of the parameters of I depend on the PHB node itself, such as the number of input interfaces.

The set of configuration parameters (C) usually consists of parameters such as queue length, drop probability, and scheduling parameters. Choosing the parameters of C requires either knowledge of the traffic control components of the node, or the use of vendor-supplied specifications. Alternatively, the definitions of PHBs in the IETF standard RFCs and Internet drafts can be used for this purpose. We use routers based on the open-source Linux operating system and, therefore, we can access all the configuration parameters as well as their implementation details. The Linux traffic control module provides a flexible way to realize various PHBs with the help of a number of queueing disciplines and traffic conditioning modules. The response variables in O used in this study are throughput (BW), per-hop delay (D), per-hop jitter (J), and loss rate (L).

An important issue is the choice of the levels for the factors in designing a set of experiments, which covers the entire range of the expected performance of a PHB. In some cases, intuition can reveal the relationships be-

[1] Marked traffic is also called assured, and unmarked traffic is often best-effort.

tween the factors and the response variables. For example, if the configured service rate of the PHB forwarding engine is higher than the total input traffic rate, the output throughput converges to the input rate. On the other hand, if one considers delay (as part of O), it is not obvious *a priori* as to how it will be affected by the relative difference between the input and configured rates. Our approach provides a generic solution for this issue, in which polynomial models are derived to represent such complex relationships. It may not be possible to cover all possible ranges and various modes of operation of a specific PHB in this manner. However, the experiments should capture the expected ranges of operation for the node.[2]

A *full factorial design* utilizes every possible combination of all the factors [6] at all levels. If we have k factors, with the i-th factor having n_i levels, and each experiment is repeated r times, then the total number of experiments to perform will be $\prod_{i=1}^{k} n_i \times r$. One of the drawbacks of the full factorial analysis is, therefore, the number of experiments growing exponentially with the number of factors and their levels. Moreover, in the context of network measurements, the total duration of an experiment can be prohibitively long, and often taking several days. To reduce the number and the execution time of experiments, we use a combination of *factor clustering* and *iterative experimental design* techniques. In factor clustering, the input and configuration parameters having similar effect on the output, are grouped together. This is similar to [9] in which the authors clustered ten congestion and flow control algorithms in TCP Vegas into three groups, according to the three phases of the TCP protocol. The iterative design technique is used to investigate three distinct types of network provisioning (see Section 3). Nevertheless, even with both experimental reduction techniques used, the number of experiments can still be large. To efficiently handle a large volume of experimental data, and to study all possible interactions, we automate the entire process of traffic generation, configuration of intermediate per-hop traffic handling mechanisms, trace collection and cataloging into an integrated framework.

2.2 Statistical Analysis

We now briefly describe the statistical methods we used, namely, ANOVA and regression analysis. A more detailed description of these methods can be found in [6, 10]. For any three factors (i.e., $k = 3$) denoted as A, B, and C with levels a, b, and c, and with r repetitions of each experiment, the response variable y can be written as a linear combination of the main effects and their interactions:

$$\begin{aligned} y_{ijkl} &= \mu + \alpha_i + \beta_j + \varepsilon_k + \gamma_{ABij} + \gamma_{ACik} + \gamma_{BCjk} + \gamma_{ABCijk} + e_{ijkl} \\ & i = 1, \ldots, a; \quad j = 1, \ldots, b; \quad k = 1, \ldots, c; \quad l = 1, \ldots, r \end{aligned} \tag{1}$$

where

y_{ijkl} = response in the l-th repetition of experiment with factors A, B, and C at levels i, j, and k, respectively.
μ = mean response = $\bar{y}_{...}$
α_i = effect of factor A at level i = $\bar{y}_{i..} - \mu$
$\bar{y}_{...}$ = average response at the i-th level of A over all levels of other factors and repetitions.
β_j = effect of factor B at level j = $\bar{y}_{j..} - \mu$
γ_{ABij} = effect of the interaction between A and B at levels i and j = $\bar{y}_{ij..} - \alpha_i - \beta_j - \mu$
γ_{ABCijk} = effect of the interaction between A, B, and C at levels i, j, and k
= $\bar{y}_{ijk.} - \gamma_{ABij} - \gamma_{BCjk} - \gamma_{ACik} - \alpha_i - \beta_j - \varepsilon_k - \mu$
e_{ijkl} = error in the l-th repetition at levels i, j, and k,

and so on. Squaring both sides of the model in Eq. (1), and summing over all values of responses (cross-product terms cancel out) we get:

$$\sum_{ijkl} y_{ijkl}^2 = abcr\mu^2 + bcr\sum_i \alpha_i^2 + acr\sum_j \beta_j^2 + abr\sum_k \varepsilon_k^2 + cr\sum_{ij} \gamma_{ij}^2 + br\sum_{ik} \gamma_{ik}^2 + ar\sum_{jk} \gamma_{jk}^2$$

[2] Given by a network administrator.

$$+r\sum_{ijk}\gamma_{ijk}^2+\sum_{ijkl}e_{ijkl}^2$$

which can be written as:

$$SSY = SS0+SSA+SSB+SSC+SSAB+SSAC+SSBC+SSABC+SSE$$

The total variation of y, denoted as the sum of square total or SST, is then:

$$SST=\sum_{ijkl}(y_{ijkl}-\mu)^2 = SSY-SS0.$$

The error in the k-th repetition is $e_{ijkl}=y_{ijkl}-\bar{y}_{jk}$, and the sum of squared errors (SSE) is equal to:

$$SSE=\sum_{ijk}e_{ijkl}^2 = SST-SSA-SSB-SSC-SSAB-SSAC-SSBC-SSABC.$$

The percentages of variation can be calculated as $100\times(\frac{SSA}{SST})$ for the effect of factor A, $100\times(\frac{SSAB}{SST})$ for the interaction between A and B, and so on. From these percentages of variations, we can identify the most important factors. The factors with small or negligible contributions to the total variation of the output can be removed from the model. Using ANOVA also allows us to calculate the mean square error (MSE) and compare it with the mean square of the effect of each factor to determine the significance of these effects against the experimental errors. This is called the *F-test* in ANOVA and usually leads to the same conclusion if we compare the percentage of variations of the factors with those of errors.

The above linear model used in ANOVA is based on the following assumptions [6]: (1) the effects of the input factors and the errors are additive, (2) errors are identical and independent, normally distributed random variables, and (3) errors have a constant standard deviation. Therefore, an important step after the ANOVA analysis is to validate the model by inspecting the results. This can be done using several "visual tests": (1) the scatter plot of the residuals (errors), e_{ijkl}, versus the predicted response should not demonstrate any trend and (2) the normal quantile-quantile (Q-Q) plot of residuals should be approximately linear (after removing the outliers). The ANOVA method itself does not make any assumption about the nature of the statistical relationship between the input factors and the response variables [10].

Once the most important factors have been identified, we use regression models [10] to capture possible relationships between each output response variable and the most significant input factors. Basically, we use a variant of the multiple linear regression model called the *polynomial regression* for this purpose. The justification for this is that any continuous function can be expanded into piecewise polynomials given enough number of terms. We use a number of simple transformations [6] such as inverse, logarithmic, and square root, to capture non-linearity in these relationships and convert them into linear ones. However, more complex transformations [6, 10] can be used for complex models. We choose the transformation that best satisfies the visual tests, minimizes the error percentage in ANOVA, and maximizes the coefficient of determination (R^2) in the regression model. We also calculate the confidence intervals for the mean response as well as the extracted parameter estimates of the regression model. These results are presented in Section 4.

3 Experimental Framework

An automated framework is necessary to expedite our experiments due to the complexity of the tasks involved, such as the experimental scenarios setup, network elements configuration, traffic generation, and collection and analysis of very large data sets. To the best of our knowledge, our framework is the first to integrate all the essential components for large-scale network analysis. Since networks are inherently complex with many different types of equipment, protocols, traffic sources and QoS architectures, the framework has been designed to meet the following requirements:

- The framework components have to be distributed — the placement of the components depend on their functionalities and domain of operation;
- The framework have to be scalable, i.e., it can be used across many hosts and network elements in a given network;
- The design of the middleware[3] have to be modular and flexible so that the components can be added/removed (e.g., with commercial or open-source routers) easily, without requiring major changes to the underlying protocols and measurement probes;
- The framework components have to be coordinated and controlled using message passing from a Controller agent (to be described next) and internal messaging among the components themselves.

3.1 Framework Components

Figure 4 illustrates the framework components used in our study and their locations. Each component in this framework essentially builds an abstraction for a particular service, and the components communicate with each other by exchanging messages. Such an approach allows us to incorporate new devices and replace any of the underlying software without changing the overall architecture. We can characterize other QoS frameworks, such as MPLS, by simply replacing the DiffServ-specific parameters in C with appropriate MPLS-specific parameters. The experimental framework components are described as follows:

Traffic Generation Agent is a modified version of Iperf [11], and it can generate both TCP and UDP traffic as constant bit rate (CBR), ON/OFF (fixed, exponential, and Pareto), and video traffic from trace files (MPEG, H.261 and H.263). UDP traffic can be optionally policed with a built-in leaky bucket so that the output traffic follows a specific average rate r and burst b. Parameters, such as peak rate, packet sizes and the duration of each flow, are controllable within the traffic generator agent. The agent has built-in routines for measuring *throughput*, *one-way delay*, *jitter*, and *loss rate*. In case of UDP traffic, these measurements are enabled through the use of a RTP-like header that includes sequence numbers and timestamps. The end-to-end jitter is calculated in the same way as the RTP protocol does, $J = J + (|D(i-1,i)| - J)/16$, where $D(i-1,i)$ is the end-to-end delay variation between packets i and $(i-1)$.

Controller and Remote Agents are set of distributed agents are placed in the network, typically one at each host or router, that are controlled remotely through a *Controller* agent to execute and keep track of the experiment steps. The Controller agent resides on host H, as shown in Figure 4, and sends periodic messages to the remote agents at the other hosts and routers according to the scenario of each experiment. It reads in a scenario file, that defines the parameters (factors) and their values (levels), and run the experiments accordingly.

Network Configuration Agent is a domain-level agent, responsible for configuring the traffic conditioning elements in a given network domain. It receives information from the Controller agent about each test to be performed for a set of experiments, and sends messages to each of the routers under its administrative control with instructions for the set of traffic control (router configuration) parameters to be installed for this test. For a DiffServ PHB, these parameters are often the configuration parameters for setting up appropriate queuing disciplines, buffer sizes, traffic classes and filters.

Router Configuration Agents are placed on the individual routers in the network that participate in the experiments, and are based on a set of APIs called DiffAgent. These APIs are used to configure the traffic control blocks in each router. Currently, our implementation is based on the traffic control (tc) APIs in

[3] PHB configuration middleware.

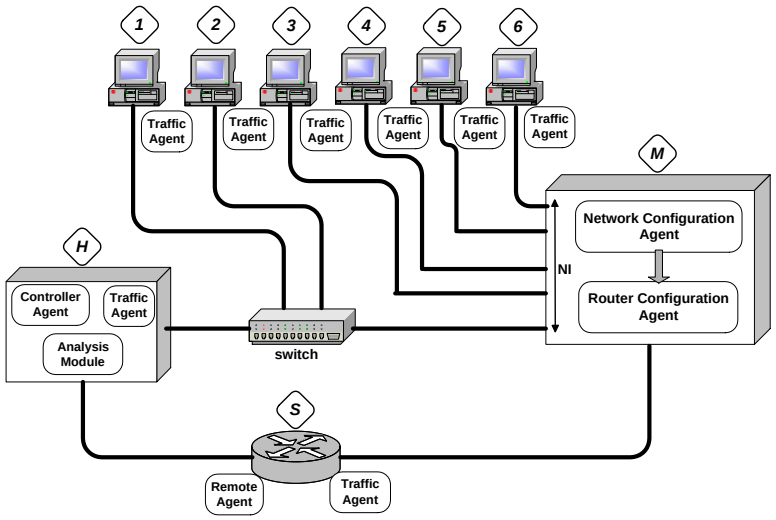


Figure 4: Framework network

the Linux kernel [12]. Upon receipt of a configuration message from its Network Configuration agent, the DiffAgent issues a Netlink socket call to the kernel with the appropriate instructions for setting up queuing disciplines, classifiers and traffic classes. This design allows inclusion of routers based on operating systems other than Linux, such as Cisco IOS, in which case, the DiffAgent maps the incoming messages to OS-specific calls before transmitting them to the router.

Analysis Module performs ANOVA, model validation tests and polynomial regression on the experimental output data. Once a set of experiments have been completed, the Controller agent invokes this module. The outcome of the module is the set of functional relationships between factors and response variables.

3.2 Network Setup and Testbed Configurations

Our network testbed consists of Linux-based software routers and end-hosts. The traffic conditioning and handling mechanisms built into the Linux kernels [12] give us two advantages: (i) a wide variety of standard traffic management components are supported, such as Token Bucket Filters (TBF), Weighted Fair Queuing (WFQ), and Priority Queuing (PQ) schedulers to name a few; and (ii) the flexibility of fine-grained specification of individual PHB components allows us to study in great detail the functional relationships between the factors and the response variables.

Figure 4 shows the network testbed used in our study. A ring topology is used so that the one-way delay can be measured without sophisticated time synchronization techniques such as GPS. Since the objective of our experiments is to characterize per-hop QoS of a single DiffServ PHB, we need only one router (*M*) implementing the PHB. We are primarily interested in finding out how a single EF or AF flow, which we refer to as the *designated flow*, is influenced by PHB configuration parameters, other traffic in the same aggregate, number of interfaces, etc. This *designated flow* is always generated from host *H*. In order to build the ring network, we add a second router *S* (i) to forward outgoing packets from *M* back to host *H* using *iptables* in Linux, and (ii) to act as a destination for the background traffic. The variable number of input interfaces (*NI*) at *M* is considered as one of the input factors. Since we do not perform admission control, router *M* performs PHB-related traffic conditioning at its egress interface. Hosts 1 through 6 are used to generate background traffic (both assured and best-effort) that share the links between routers *M* and *S* along with the *designated flow*. The experimental

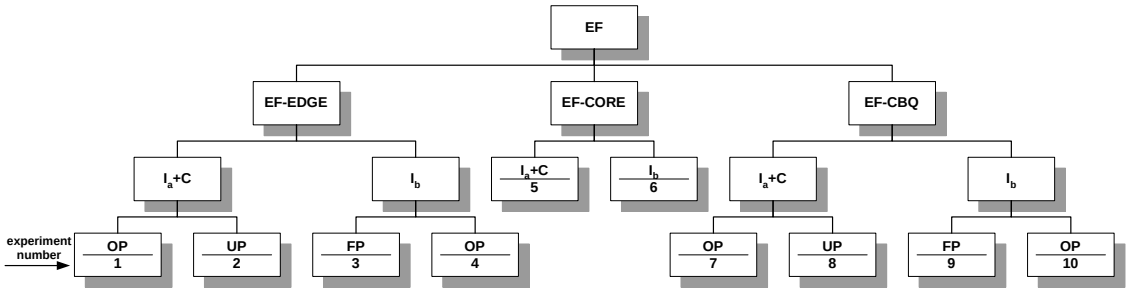


Figure 5: Experiments scenarios for EF PHB

module in the Controller agent can generate any experiment plan involving one or multiple background flows of particular types depending on the experiment scenario.

3.3 Design of Experiments

As discussed earlier, we use clusters of input factors to reduce the parameter space and duration of the experiments. The I set can be partitioned into two subsets: the assured traffic factors (I_a), and the best-effort traffic factors (I_b). Along with these, we also choose a few parameters from the configuration set (C). If the size of C is large (depending on the particular choice of traffic components used), one may also decide to partition this set.

One can envision three different operating modes for a given PHB configuration with respect to the assured input traffic: The PHB can be *over-provisioned* (OP), *fully-provisioned* (FP), or *under-provisioned* (UP). In the OP mode, the total assured input traffic rate is less than the configured rate of the PHB. It is larger than the configured PHB rate for the UP mode. In the FP mode, they are nearly equal. Although we have only investigated static modes, the modes of a PHB are dynamic in practice, depending on the current level of traffic passing through it and the configuration of this PHB. These three modes can be further investigated for different scenarios of input traffic type and the degree of flow aggregation. Figure 5 illustrates the organization of the experiment scenarios for the EF PHB.

In the first scenario, we investigate the interaction between sets I_a and C in the absence of any best-effort traffic. This experiment is performed for the two modes of OP and UP. The purpose of this scenario is to identify the most important factors in sets I_a and C that affect the output per-hop QoS. This is our base scenario for the assured traffic.

Next, we target a specific type of assured traffic, e.g., set I_a to certain constant values and vary the factors in I_b around these constant values. The factors in the best-effort traffic (such as rate, burst size, packet size and number of flows) range in values that include both higher and lower levels than their corresponding parameters in the assured traffic. The purpose is to investigate the effect of best-effort traffic on the assured traffic within a given range. This scenario is investigated for OP and FP modes only, since the per-hop QoS of an already-overloaded PHB is not going to change significantly due to changes in the best-effort traffic.

In addition to these two scenarios, we analyze different realizations of a given PHB to investigate how the choice of different traffic conditioning elements affect the overall performance of a PHB. Different choices of schedulers, classes and filters give rise to different functional configurations and therefore, constitute different sets of parameters in C. For the EF PHB, we consider three different configurations:

- *EF-EDGE*: an edge router configuration consisting of a TBF served by a priority scheduler as shown in Figure 3(A). Table 2 lists the parameters of C for this realization.
- *EF-CORE*: a core router configuration consisting of a single priority scheduler as shown in Figure 3(B).

Factor	Symbol
Token bucket rate	ef_r
Bucket size	ef_b
Maximum Transfer Unit	ef_{mtu}

Table 2: "C" set for an EF PHB at edge router

Factor	Symbol
EF service rate	ef_r
Burst size	ef_b
Average packet size	ef_{avpkt}

Table 3: "C" set for an EF PHB based on CBQ

Factor	Symbol
Minimum threshold	min_{th}
Maximum threshold	max_{th}
Drop probability	$prob$

Table 4: "C" set for the AF PHB

The only parameter in C for this realization is the length (ef_{limit}) of the FIFO queue in the priority scheduler. The service rate for the PHB queue is the same as the link speed.

- *EF-CBQ*: a CBQ-based configuration as shown in Figure 3(C). Table 3 lists the parameters of C for this realization.

We also repeat similar performance analysis for the AF PHB. We use a multi-color RED (or GRED) queue for each AF class served by a CBQ scheduler. Table 4 lists the C set for the AF PHB. It includes the parameters for the AF11 RED virtual queue only. We choose the AF PHB as another instance of a DiffServ PHB to demonstrate the generality of our measurement framework and analysis approach. We assume that the multiple PHBs configured at a single node do not affect the performance of each other. Such isolation can be achieved in practice by suitable choice of traffic handling blocks and forwarding policy.

4 Results and Analysis

The results from our experiments are presented in the same order of the scenarios described in Section 3.3 and the chart in Figure 5. We first present results for the three different implementations of the EF PHB, followed by a single configuration of the AF PHB. Unless otherwise mentioned, each experiment is repeated five times (i.e., $r = 5$) and each traffic trace is collected for 20 seconds. We use a rest period of three seconds in-between successive runs so that the network is empty of packets from a previous run. Depending on the experiment scenario, the input traffic enters router M from one or multiple input interfaces (i.e., $NI \geq 1$).

4.1 Characterizing EF PHB

For each of the EF PHB realizations, we group the results into two categories: the interaction between sets I_a and C, and the effect of background traffic (I_b) on the fixed assured traffic (I_a). The input factors (I_a and C) used in each experiment scenario are listed in Tables 13, 14, and 15 in Appendix A for EF-EDGE, EF-CORE, and EF-CBQ, respectively.

4.1.1 Over-Provisioned (OP) EF-EDGE PHB without Background Traffic

This scenario corresponds to experiment 1 in Figure 5. The PHB is over-provisioned (OP), i.e., the throughput of the assured traffic is less than the configured rate for the PHB.

The effects of significant factors and their significant interactions[4] are identified by using ANOVA and listed in Table 5. The transformation applied to each response variable, satisfying the basic assumptions of the linear ANOVA model, is indicated in the second column. The mean, standard deviation (SD), and 90% confidence interval (CI) are listed in Table 6. The loss (L) in this experiment is basically zero (no loss) since this is an over-provisioned case.

[4]We neglect any factor or interaction with a percentage of variation less than 2%.

Response Variable	Transformation	a_r	a_{pkt}	a_n	(a_r,a_n)	(a_r,a_{pkt})	Error
BW	linear	48.55%	$\approx$ 0%	34.23%	17.21%	$\approx$ 0%	$\approx$ 0%
D	linear	$\approx$ 0%	83.15%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	13.53%
$1/J$	inverse	4.12%	12.52%	46.58%	$\approx$ 0%	2.88%	25.34%

Table 5: ANOVA results for experiment 1

Response Variable	Mean	SD of Mean	90% CI for Mean
BW	948.81 Kbps	0.125	(948.60,949.015)
D	0.262199 msec	0.001088	(0.260409,0.263989)
$1/J$	311.21	3.74	(305.05,317.37)

Table 6: Mean, standard deviation (SD), and 90% confidence interval (CI) for the response variables in experiment 1

To verify the linearity of the ANOVA model, the visual tests for jitter (J), as an example, are shown in Figures 6 and 7. After the inverse transformation, there is no trend in the residual versus predicted response scatter plot. Moreover, the errors are normal since the normal Q-Q plot is almost linear. The tests for the throughput and the delay show linearity as well.

The polynomial regression models for the response variables are calculated, and the parameter estimates for the BW model are listed in Figure 8. The surface plot corresponding to this model is shown in Figure 9. The coefficient of determination (R^2) is 96%. Eq. (2) represents the model for jitter, with the corresponding value of $R^2 = 84\%$. Note that we use the same transformation from ANOVA while performing the regression analysis. We also normalize the input factors by their maximum values to provide a scaled version of the model equations.

$$\begin{aligned} 1/J \;=\; & 620.62 + 1343.58a_r - 1024.12a_{pkt} - 2212.85a_n - 1586.15a_r^2 - 231.39a_r a_{pkt} \\ & -574.60a_r a_n + 913.48a_{pkt}^2 + 717.55a_{pkt} a_n + 3485.72a_n^2 + 281.31a_r^3 + 583.23a_r^2 a_{pkt} \\ & +700.89a_r^2 a_n - 133.74a_r a_{pkt}^2 - 249.64a_r a_{pkt} a_n - 51.79a_r a_n^2 - 268.81a_{pkt}^3 \\ & -307.76a_{pkt}^2 a_n - 106.48a_{pkt} a_n^2 - 1768.39a_n^3 \end{aligned} \quad (2)$$

As shown in Table 5, the throughput (BW) depends mostly on the sending rate of the assured traffic (a_r) and number of EF flows (a_n). The reason for the dependency of BW on a_n is that, in this experiment, we divide the total EF rate (a_r) specified in Table 13, by the number of flows (a_n) to get an equal share for each EF flow. Because we only measure a single designated EF flow, the resulting throughput depends on the number of flows. The per-hop delay is mainly affected by the EF packet size. On the other hand, jitter is affected mostly by a_n and a_{pkt}, but little by a_r. This follows our intuition about jitter: the packets from the monitored flow have to wait in the queue because of other EF flows sharing the same queue, as well as due to the relative difference of their packet sizes. Larger values of a_n and a_{pkt} result in larger jitter. This can be seen in the regression results as well, since the corresponding coefficients are negative in the model equation given that an inverse transformation is applied to jitter. The statistical analysis also provides us with confidence intervals for each of the parameter estimates in the regression equation. The confidence intervals determine the accuracy of the models extracted, and therefore constitute an important part of model checking. However, we omit these values due to space limitation.

4.1.2 Under-Provisioned (UP) EF-EDGE PHB without Background Traffic

In this scenario (experiment 2), the EF PHB is under-provisioned (UP), i.e., the throughput of the assured traffic is greater than the configured rate for the PHB. Here we present the results for loss (L) only. The effects of

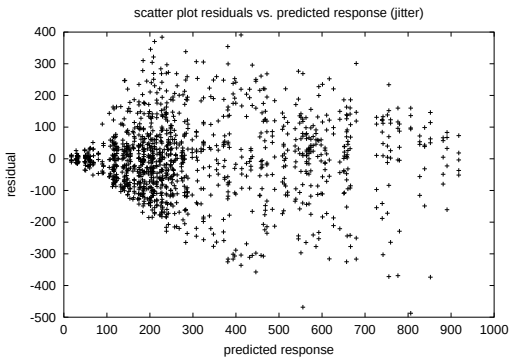


Figure 6: Residuals versus predicted response for jitter – experiment 1

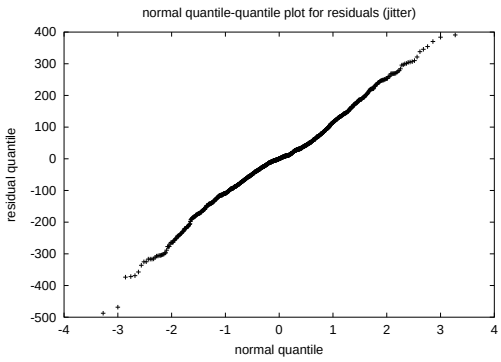


Figure 7: Normal Q-Q plot for jitter residuals – experiment 1

BW	Parameter
Intercept	299733.48
a_r	3246957.34
a_n	-2318542.4
a_r^2	-32371.95
a_n^2	2223351.57
$a_r\ a_n$	-2815325.9

Figure 8: Parameters for *BW*

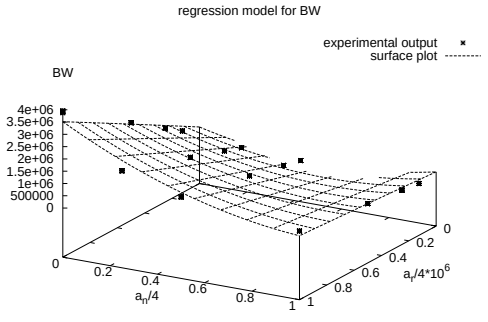


Figure 9: Response service for *BW*

signifi cant factors and their signifi cant interactions are listed in Table 7. The transformation used is indicated in the second column as before. The visual tests for L are shown in Figures 10 and 11. The polynomial regression model for loss is given in Eq. (3) with $R^2 = 74\%$. The model for loss clearly shows that by increasing the sending rate, the loss increases (due to the negative sign of a_r and the inverse of L). A similar effect was observed with the number of flows (a_n). Note that intuitively, the loss can be reduced by increasing the EF service rate (ef_r). The positive coeffi cient of ef_r in Eq. (3) clearly indicates this.

$$\begin{aligned} 1/L \;=\; & 0.06 - 0.63a_r - 0.2a_n + 0.63ef_r + 1.48a_r^2 + 0.79a_r a_n - 2.05a_r ef_r + 0.3a_n^2 \\ & -0.69a_n ef_r + 0.43ef_r^2 - 1.14a_r^3 - 0.73a_r^2 a_n + 2.37a_r^2 ef_r - 0.42a_r a_n^2 + 1.23a_r a_n ef_r \\ & -1.69a_r ef_r^2 - 0.18a_n^3 + 0.46a_n^2 ef_r - 0.57a_n ef_r^2 + 0.57ef_r^3 \end{aligned} \quad (3)$$

4.1.3 Over-Provisioned (OP) EF-EDGE PHB with Background Traffic

Next, we investigate the effect of background (e.g., best-effort) traffi c on the assured traffi c for the EF PHB. As shown in Table 8, the throughput (*BW*) of the EF traffi c is affected heavily by the number of background traffi c flows sharing the link with it. Since the PHB is over-provisioned, the effect of the ratio R_{ab} or the size of the background traffi c itself is not dramatic. As a result of over-provisioning too, the percentage of variation due to experimental error is high. This is because of the weak interaction between the inputs and the output *BW*. The

Response Variable	Transformation	a_r	a_n	ef_r	(a_r,a_n)	(a_r,ef_r)	(a_n,ef_r)	(a_r,a_n,ef_r)	Error
$1/L$	inverse	6.06%	20.81%	6.63%	15.1%	8.9%	16.21%	25%	$\approx 0\%$

Table 7: ANOVA results for experiment 2

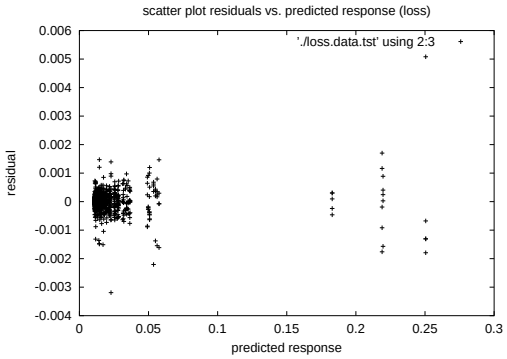


Figure 10: Residuals versus predicted response for loss – experiment 2

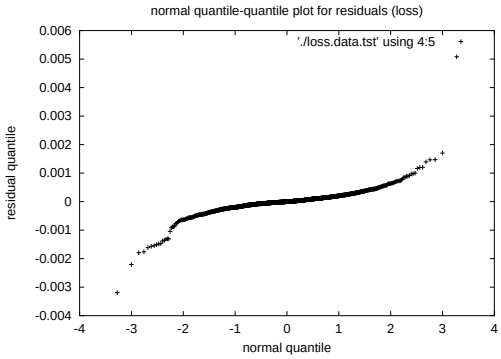


Figure 11: Normal Q-Q plot for loss residuals – experiment 2

results also indicate that the priority scheduler does a good job in isolating the EF traffic from the background flows. There are no significant EF losses for the same reason above. Later, we will compare these results with the results of Section 4.1.6 that employs CBQ, instead of priority scheduling. Both delay and jitter values show very little experimental errors due to repetition, and strongly depend on the background traffic characteristics, such as the packet size, number of background flows, and their interactions. Note that a linear transformation fits jitter well with respect to the background traffic parameters, where a logarithmic transformation is appropriate for delay. This is different from what we found in Section 4.1.1 with EF-EDGE, where jitter is inversely related to the EF traffic parameters. Similar results are observed in Sections 4.1.4, 4.1.5, and 4.1.6 (to be shown).

4.1.4 EF-CORE PHB without Background Traffic

The purpose of experiment 5 is to demonstrate how different implementations of a specific PHB can differ in their extracted input/output relationships. We present the results for jitter (J) as an example. The most important factors affecting J are listed in Table 9. Note that there is no effect due to a_r, unlike the scenario in Section 4.1.1 using EF-EDGE PHB. The visual tests are shown in Figures 12 and 13 to test the validity of the ANOVA model. The jitter model is given in Eq. (4) with a coefficient of determination (R^2) of 64%. The response surface is shown in Figure 14. The inverse relationship also holds here with respect to the EF traffic parameters as mentioned earlier.

$$1/J = 727.74 - 810.85a_n - 748.17a_{pkt} + 425.13a_n^2 + 322.99a_{pkt}^2 + 189.18a_{pkt}a_n \quad (4)$$

The negative coefficients for both a_n and a_{pkt} in the above model indicate that an increase in either of them will increase the value of jitter. A very important result can be deduced accordingly — the larger the number of flows in an EF traffic, the larger the jitter.

Response Variable	Transformation	b_{pkt}	b_n	R_{ab}	(b_{pkt},b_n)	(b_{pkt},R_{ab})	(b_n,R_{ab})	(b_{pkt},b_n,R_{ab})	Error
BW	linear	$\approx 0\%$	36.6%	2.65%	$\approx 0\%$	$\approx 0\%$	2.87%	4.4%	38.76%
$\log(D)$	logarithmic	3.28%	8.98%	36.3%	3.54%	10.01%	27.07%	10.69%	$\approx 0\%$
J	linear	3.36%	7.5%	39.14%	3.7%	9.72%	12.13%	22.77%	1.4%

Table 8: ANOVA results for experiment 4

Response Variable	Transformation	a_{pkt}	a_n	(a_{pkt},a_n)	Error
$1/J$	inverse	29.37%	18.27%	6.97%	31.78%

Table 9: ANOVA results for experiment 5

4.1.5 Over-Provisioned (OP) EF-CBQ PHB without Background Traffic

In this experiment scenario, we evaluate the EF-CBQ implementation, corresponding to experiment 7 in Figure 5.

Table 10 shows the results from the ANOVA analysis. The throughput (BW) has a square-root relationship with the input EF traffic parameters, which is different from the previous two implementations. The delay (D) model is also different. Another important difference is that the jitter is dependent on the PHB configuration parameters (C) such as ef_r for EF-CBQ, where as neither EF-EDGE nor EF-CORE shows such dependency. On the other hand, the jitter (J) still has an inverse relationship with the input EF traffic parameters.

4.1.6 Over-Provisioned EF-CBQ PHB with Background Traffic

The purpose of experiment 10 is to investigate the effect of the background traffic on EF traffic. The characteristics of the background traffic are listed in Table 15.

As shown in Table 11, the delay and jitter models, with respect to the input background traffic parameters, are similar to those in Section 4.1.3, where the delay (D) logarithmically depends on the background traffic volume and the jitter (J) linearly depends on the three factors, b_{pkt}, b_n, and R_{ab}. The EF-CBQ implementation also cannot protect the EF traffic against packet losses. The loss (L) depends mostly on the volume of the background traffic and the interaction with other factors, as shown in Table 11.

The polynomial regression model for delay is presented in Eq. (5) with a coefficient of determination (R^2) of 89%. Although the delay does not depend on individual factors, such as b_{pkt} and b_n as seen from the ANOVA table, it is dependent on their interactions with R_{ab} and their interactions. Therefore, we need to include b_{pkt} and b_n into the regression model. However, when comparing the coefficients values, we find that the coefficients for R_{ab} and their powers are orders-of-magnitude larger than the other factors, which is consistent with the ANOVA analysis. Given that the normalized values of b_n and b_{pkt} are in the range of (0,1), we can exclude them from the model and approximate the model as shown in Eq. (6).

$$\begin{aligned}\log(D) \quad = \quad & 1.91 - 46.65R_{ab} + 2.33b_n - 6.03b_{pkt} + 121.86R_{ab}^2 + 1.89R_{ab}b_n + 2.74R_{ab}b_{pkt} \\ & -5.25b_n^2 + 1.19b_nb_{pkt} + 8.61b_{pkt}^2 - 78.24R_{ab}^3 - 1.34R_{ab}^2b_n + 0.51R_{ab}^2b_{pkt} \\ & +0.79R_{ab}b_n^2 - 1.89R_{ab}b_nb_{pkt} - 1.96R_{ab}b_{pkt}^2 + 1.90b_n^3 + 1.99b_n^2b_{pkt} \\ & -1.88b_nb_{pkt}^2 - 3.44b_{pkt}^3 \end{aligned} \tag{5}$$

$$\log(D) \quad \approx \quad 1.91 - 46.65R_{ab} + 121.86R_{ab}^2 + 1.89R_{ab}b_n + 2.74R_{ab}b_{pkt}$$

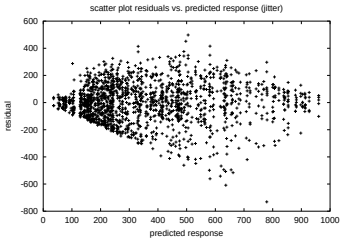


Figure 12: Residuals versus predicted response for jitter – experiment 5

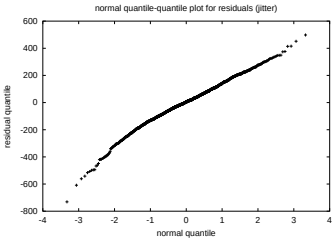


Figure 13: Normal Q-Q plot for jitter residuals – experiment 5

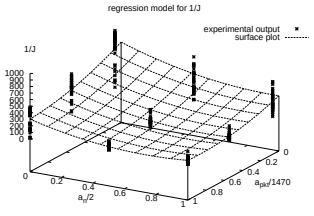


Figure 14: J Response surface for jitter – experiment 5

Response Variable	Transformation	a_r	a_{pkt}	a_n	ef_r	(a_{pkt},a_n)	(a_n,ef_r)	Error
$\sqrt{BW}$	square	96.73%	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$
$1/D$	inverse	$\approx 0\%$	94.55%	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	2.51%
$1/J$	inverse	$\approx 0\%$	14.1%	37.5%	6.8%	2.0%	2.0%	22.55%

Table 10: ANOVA results for experiment 7

$$-78.24R_{ab}^3 - 1.34R_{ab}^2 b_n + 0.51R_{ab}^2 b_{pkt} + 0.79R_{ab} b_n^2$$
$$-1.89R_{ab} b_n b_{pkt} - 1.96R_{ab} b_{pkt}^2 \quad (6)$$

4.2 Characterizing AF PHB

We present results for the AF PHB to show the generality of our approach and to provide an insight into the behavior of the AF PHB. The relevant per-hop QoS attributes, according to the IETF standard, are throughput (BW) and loss (L). However, some applications may also impose delay and jitter requirements on AF-based services, so we present results for them as well. In this experiment, we mark the designated flow with AF11 DSCP; and the background traffic (filling up the rest of the network capacity) with AF12 and AF13 DSCPs. The input factors and their levels are listed in Table 16.

Table 12 lists[5] the ANOVA results for the four QoS attributes. The results indicate that the throughput (BW) depends mostly on the average and peak sending rates, and their interactions. They also exhibit some dependency on the threshold levels of the AF11 virtual queue. In another experiment (not shown here), we find this dependency increases with the increase of the drop probability for the virtual queue. The results for loss (L), delay (D), and jitter (J) show similar dependencies, although the percentages vary with each of the QoS parameters. We notice a small, but non-trivial, interaction between the packet size of the designated and that of the background traffic contributing to the jitter of the assured traffic.

4.3 Discussion

From the above results, we can capture the functional relationships of a number of PHBs and identify the differences in their dependencies on the various input factors. To summarize: for the EF PHB cases, the results for jitter (J) indicate an inverse relationship with the factors in set I_a, and a direct relationship with the factors in set I_b. The delay (D) has a direct relationship with factors in set I_a and a logarithmic one with factors in set I_b. Also, different PHB realizations exhibit different relationships among their inputs and the outputs. For example, delay has a direct relationship with factors in set I_a in EF-EDGE and EF-CORE PHBs, while it has

[5]The error column is not shown due to space limitation.

Response Variable	Transformation	b_{pkt}	b_n	R_{ab}	(b_{pkt},b_n)	(b_{pkt},R_{ab})	(b_n,R_{ab})	(b_{pkt},b_n,R_{ab})	Error
BW	linear	10.64%	$\approx 0\%$	15.76%	7.97%	31.44%	8.53%	23.65%	0.4%
$\log(D)$	logarithmic	$\approx 0\%$	$\approx 0\%$	83.23%	2.34%	3.53%	2.04%	7%	0.1%
J	linear	11.88%	$\approx 0\%$	22.01%	5.63%	35.6%	5.68%	16.93%	0.27%
$\sqrt{L}$	sqrt	8.87%	2.38%	36.89%	5.14%	25.84%	4.8%	15.45%	0.53%

Table 11: ANOVA results for experiment 10

Res. Var.	Trans.	a_r	a_p	a_{pkt}	max_{th}	min_{th}	(a_r,a_p)	(a_r,max_{th})	(a_r,min_{th})	(a_p,max_{th})	(a_{pkt},b_{pkt})	(a_r,a_p,max_{th})
BW	linear	51.38%	12.51%	$\approx 0\%$	2.67%	2.38%	13.27%	3.27%	2%	2%	$\approx 0\%$	2.37%
D	linear	38.2%	25.42%	2.85%	$\approx 0\%$	3.1%	25.06%	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$
J	inverse	14.75%	17.35%	34.82%	$\approx 0\%$	$\approx 0\%$	18.94%	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	4.09%	$\approx 0\%$
L	linear	29.32%	17.12%	$\approx 0\%$	7.14%	4.17%	16.98%	7.4%	3.89%	3.91%	$\approx 0\%$	4.03%

Table 12: ANOVA results for AF PHB

an inverse relationship with I_a in EF-CBQ PHB. The throughput (BW) has a square root relationship with I_a for EF-CBQ only, but it has a direct relationship in the other EF PHBs.

Note that the purpose of the present study is not to cover all possible performance ranges or configurations for a particular PHB. Instead, we present a technique for analyzing the PHB performance under different situations. The levels of factors used in the experimental analysis represent only a certain range of the PHB operation. Additional parameters – such as the expected loads, the type of physical links connected to the PHB node, etc. – should be considered when applying this technique to characterize PHBs in real-world networks.

It is worth mentioning that the errors in our study can be divided into three categories as follows:

1. *Experimental errors* are due to the experimental methods and we use a reasonable number of repetitions to minimize them. These errors are captured in the ANOVA analysis.

2. *Model errors* are due to factor truncation. When we discard the insignificant factors and perform regression with the most significant ones, small errors can be potentially introduced.

3. *Statistical or fitting errors* are due to difference between the fitted regression model and the actual data. The regression usually fits an approximate model to the actual data by using the least-square fit to minimize the deviation from the actual data.

Finally, we provide an example of the potential benefit of the statistical approach we proposed. This example is beneficial for network operators as it provides a proof of using the above results in controlling the per-hop QoS of a PHB. From the results of Section 4.1.6, we observe that for b_{pkt} = 600 Bytes, b_n = 1 and R_{ab} = 2, the average delay experienced by the designated EF flow is 0.4136 msec. Now, suppose we increase the number of background flows from 1 to 3, and the average packet size of the background traffic from 600 to 1470 Bytes. Can the PHB configuration be controlled in such a way that it provides the same delay value to the assured traffic as in the first case? Basically, we have to find the right value for R_{ab} which drives the PHB to deliver the same delay. By applying the delay model in Eq. (6) to the two sets of levels, we calculate the required value for R_{ab} to be 0.494. Substituting this value back into the model, we get a delay value of 0.3652 msec which is within the accuracy ($1-R^2$ or 11%) of the model. Now, when we run the experiment again with this acquired value of R_{ab}, we get the same delay as the original controlled value. Therefore, by simply controlling the factors in the derived model, we can achieve predictable per-hop QoS. Such scenario is quite realistic. Consider the assured flow that belongs to a delay-sensitive application such as video-streaming. The adjusted R_{ab} value suggests that the streaming rate at the server needs to be lower for guaranteeing the same delay bound. Although, the focus of our experiments is per-hop QoS while guaranteeing delay bounds for applications requires an end-to-end approach, such per-hop delay bounds are essential for satisfying the end-to-end guarantees.

5 Related Work

Very few previous studies used ANOVA and nonlinear regression for experimental analyses and modeling of IP QoS networks. The authors of [13] applied a full factorial design and ANOVA to compare a number of marking schemes for TCP acknowledgments in a DiffServ network. The factors in their study were categorical in nature. Their results suggested an optimal strategy for marking the acknowledgment packets for both assured and premium flows. The performance of AF PHB was analyzed in [14] using the ANOVA method where the authors compared the performance of different techniques for bandwidth and buffer management. Compared with these previous studies, our approach is more general in dealing with different network topologies, QoS frameworks, and network applications.

The authors of [15, 16] utilized a *ring* network topology, which is similar to the one used here for experimental studies on the EF PHB. Using incremental experiments, they investigated the effect of traffic aggregation, EF traffic load, the number of EF streams, packet size as well as the scheduling algorithms used — Priority Queueing (PQ) and Weighted Fair Queueing (WFQ) — on one-way delay, jitter, packet loss, and maximum burstiness of the EF traffic. Their findings can be summarized as follows. The EF burstiness was greatly affected by the number of EF streams and packet loss. Moreover, WFQ was found to be more immune to burstiness of traffic than PQ, but had less timely delivery guarantees. Our work complements these studies and provide a more rigorous way to identify these affects.

The authors of [8] conducted an experimental analysis of the EF PHB by incorporating a part of the Internet2 QBone and using the QBone Premium Service (QPS). They used the same QoS metrics (throughput, delay, jitter and packet loss) we use. The difference, however, is that their metrics are measured as end-to-end quantities, while ours are per-hop quantities. Moreover, they did not provide any functional relationships or models for designing premium services over Internet2.

The AF PHB has been studied in [17]. The authors used modeling techniques to evaluate the performance of AF-based services with respect to Round Trip Time (RTT), number of microflows, size of target rate, and packet size. The performance metric was the throughput perceived by the AF traffic. Another study [18] on the AF PHB was mainly concerned with interaction of TCP and UDP traffic within an AF service. The authors investigated the use of different drop precedences and RED models to achieve fairness between AF-marked UDP and TCP traffic. An earlier study [19] presented similar findings. Collectively, these studies identified important effects on the performance of the AF PHB. By contrast, we have attempted to build simple mathematical models to calculate the performance and response surfaces applying statistical procedures on the experimental data.

In [20], the authors compared two different router mechanisms (threshold dropping and priority scheduling), and two packet marking mechanisms (edge-discarding and marking). They used analytical methods in estimating the loss and delay behaviors using different combinations of these mechanisms.

A rigorous theoretical study to find probabilistic bounds for EF was presented in [21]. The authors used a combination of queueing theory and network calculus to obtain delay bounds for backlogs of heterogeneous traffic as well as traffic regulated by a leaky bucket. They also derived bounds on loss ratio under statistical multiplexing of EF input flows. Our approach is purely experimental; instead of providing bounds on delay or packet loss, we seek to identify the parameters necessary to construct simple performance models of per-hop QoS mechanisms, and eventually to control their run-time behavior.

6 Conclusions and Extensions

In this paper we have presented a framework for statistically characterizing the outcome of a PHB — the per-hop QoS. We have conducted an experimental study followed by rigorous statistical analysis, including ANOVA and regression models, to find the functional relationships for the per-hop QoS. We point out operational differences

among the different PHB implementations, and explain these differences based on the internal structure of the PHBs. Our framework for measurement and experimental design is automated – the confi guration of the PHBs and the measurement of the corresponding output parameters are automatically performed. Our results are promising since they can further quantify the end-to-end QoS by concatenating per-hop QoS attributes along the path.

The approaches presented in Sections 2 and 3 can be extended to more complex systems, e.g., edge-to-edge QoS building blocks such as a Per-Domain Behavior or PDB in DiffServ, especially if the PDB is constructed by concatenation of multiple similar PHBs. Although we focused on per-hop QoS at the network layer, the statistical characterization approach can be applied to parameters across the entire protocol stack, with suitable instrumentation of the related layers. This will allow us to study end-to-end QoS parameters for any distributed application in a given network. We are also exploring statistical controls based on the most important factor(s) for controlling per-hop QoS of network nodes.

References

[1] S. Blake, D. Black, M. Carlson, E. Davis, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF, RFC 2475, Dec. 1998.

[2] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," IETF, RFC 2638, July 1999.

[3] B. Davis, A. Charny, F. Baker, J. Boudec, W. Courtney, V. Firoiu, and K. Ramakrishnam, "Expedited forwarding PHB group," IETF, draft-ietf-diffserv-rfc2598bis-02.txt, RFC 2598, Sept. 2001.

[4] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," IETF, RFC 2597, June 1999.

[5] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," IETF, RFC 1633, June 1994.

[6] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., 1991.

[7] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson, "On the impact of policing and rate guarantees in diff-serv networks: A video streaming application perspective," in *Proceedings of ACM SIGCOMM'01*, Aug. 2001.

[8] A. Mohammed *et al.*, "Diffserv experiments: Analysis of the premium service over the Alcatel-NCSU internet2 testbed," in *proceedings of the 2nd European Conference on Universal Multiservice Networks ECUMN'2002, CREF,Colmar, France*, Apr. 2002, pp. 124–130.

[9] U. Hengartner, J. Bolliger, and T. Gross, "TCP vegas revisited," in *Proceedings of IEEE INFOCOM'00*, Mar. 2000.

[10] J. Neter, W. Wasserman, and M. Kutner, *Applied Linear Statistical Models: Regression, Analysis of Variance, and Experimental Designs*. Homewood, R.D. Irwin, Inc., 1985.

[11] (2000, February) Iperf 1.1.1. [Online]. Available: http://dast.nlanr.net/Projects/Iperf/

[12] W. Almesberger, J. H. Salim, and A. Kuznetsov, "Differentiated services on Linux," Internet draft (work on progress)," draft-almesberger-wajhak-diffserv-linux-01.txt, June 1999.

[13] K. Papagiannaki *et al.*, "Preferential treatment of acknowledgment packets in a differentiated services network," in *Proceedings of the IWQoS 2001*, June 2001.

[14] M. Goyal, A. Durresi, R. Jain, and C. Liu, "Performance analysis of assured forwarding," Internet draft (work on progress)," draft-goyal-diffserv-afstdy.txt, Feb. 2000.

[15] T. Ferrari, "End-to-end performance analysis with traffi c aggregation," *Computer Network Journal, Elsevier*, vol. 34, no. 6, pp. 905–914, Dec. 2000.

[16] T. Ferrari and P. Chimento, "A measurement-based analysis of expedited forwarding PHB mechanisms," in *Proceedings of IWQoS 2000, Pittsburgh, IEEE 00EXL00*, June 2000, pp. 127–137.

[17] N. Seddigh, B. Nandy, and P. Pieda, "Bandwidth assurance issues for TCP flows in a differentiated services network," in *proceedings of GLOBECOM'99*, Mar. 1999.

[18] ——, "Study of TCP and UDP interaction for the AF PHB," Internet draft (work on progress)," draft-nsbnpp-diffserv-tcpudpaf-01.txt, Aug. 1999.

[19] J. Ibanez and K. Nichols, "Preliminary simulation evaluation of an assured service," Internet draft (work on progress)," draft-ibanez-diffserv-assured-eval-00.txt, Aug. 1998.

[20] S. Sahu, D. Towsley, and J. Kurose, "A quantitative study of differentiated services for the internet," in *Proceedings of the IEEE Globecom'99, Rio De Janiero, Brazil*, Dec. 1999.

[21] M. Vojnovic and J. L. Boudec, "Stochastic analysis of some expedited forwarding networks," in *proceedings of INFOCOM'02, New York*, June 2002.

A Appendix

The input factors (I_a and C) used in each experiment scenario are listed in Tables 13, 14, and 15 for EF-EDGE, EF-CORE, and EF-CBQ, respectively. Table 16 lists the factors used in the AF PHB experiment. The factors marked with (X) are not active in the corresponding experiment. An inactive factor is a factor with one level only, and therefore, has no effect on the ANOVA results.

	Factor Name (unit)	Exp1 levels	Exp2 levels	Exp4 levels
Set I_a	a_r (Mbps)	0.5,1,2,4	3,3.5,4,4.5	1 (X)
	a_p (Mbps)	6 (X)	8 (X)	2 (X)
	a_{pkt} (bytes)	100,400,600,900	800,900,1000,1200	1000 (X)
	a_b (bytes)	20000 (X)	40000 (X)	20000 (X)
	a_n (fbws)	1,2,3,5	1,2,3,5	1 (X)
	NI	1,2,3,5	1,2,3,5	1,2,3,4
Set C	ef_r (Mbps)	20 (X)	2,2.2,2.5,3	3 (X)
	ef_b (bytes)	40000 (X)	20000 (X)	40000 (X)
	ef_{mtu} (bytes)	1000,1200,1400,1520	1520 (X)	1520 (X)
Set I_b	R_{ab}	(X)	(X)	2,1,0.1.0.01
	b_{pkt} (bytes)	(X)	(X)	200,1000,1200,1470
	b_b (bytes)	(X)	(X)	10000,20000,30000,40000
	b_n (fbws)	(X)	(X)	1,2,3,4

Table 13: Factors and their levels for EF-EDGE

	Factor Name (unit)	Exp5 levels
Set I_a	a_r (Mbps)	0.5,1,2
	a_p (Mbps)	(X)
	a_{pkt} (bytes)	100,800,1470
	a_b (bytes)	5000,10000,60000
	a_n (fbws)	1,3
	NI	1,3
Set C	ef_{limit} (packets)	5,100
Set I_b	R_{ab}	(X)
	b_{pkt} (bytes)	(X)
	b_b (bytes)	(X)
	b_n (fbws)	(X)

Table 14: Factors and their levels for EF-CORE

	Factor Name (unit)	Exp7 levels	Exp10 levels
Set I_a	a_r (Mbps)	0.5,1,1.5,2	1 (X)
	a_p (Mbps)	6 (X)	2 (X)
	a_{pkt} (bytes)	100,400,600,900	1000 (X)
	a_b (bytes)	20000 (X)	20000 (X)
	a_n (fbws)	1,2,3,5	1 (X)
	NI	1,2,3,5	1,2,3,4
Set C	ef_r (Mbps)	10,12,14,16	4 (X)
	ef_b (bytes)	10 (X)	10 (X)
	ef_{avpkt} (bytes)	1000,1200,1470	1000 (X)
Set I_b	R_{ab}	(X)	2,1,0.1.0.01
	b_{pkt} (bytes)	(X)	200,1000,1200,1470
	b_b (bytes)	(X)	10000,20000,30000,40000
	b_n (fbws)	(X)	1,2,3,4

Table 15: Factors and their levels for EF-CBQ

	Factor Name (unit)	Exp levels
Set I_a	a_r (Mbps)	50,100
	a_p (Mbps)	60,100
	a_{pkt} (bytes)	600,1200
	a_b (bytes)	20000,40000
	a_n (fbws)	1 (X)
	NI	3 (X)
Set C	min_{th} (Kbytes)	10,20
	max_{th} (Kbytes)	40,55
	$prob$	0.02 (X)
Set I_b	R_{ab}	0.01 (X)
	b_{pkt} (bytes)	600,1200
	b_b (bytes)	20000,40000
	b_n (fbws)	2 (X)

Table 16: Factors and their levels for AF PHB