Loss Classification in Optical Burst Switching Networks using Machine Learning Techniques: Improving the Performance of TCP

A. Jayaraj, T. Venkatesh, and C. Siva Ram Murthy, *Senior Member, IEEE*

***Abstract*— Optical burst switching (OBS) is considered as a contending technology for the core of the Internet in future. However, due to lack of the buffers, losses occur due to contention among simultaneously arriving bursts at the core nodes. Contention losses do not necessarily indicate a situation of congestion in the network. Thus differentiation (classification) of losses is essential in many applications to avoid false identification of congestion. In this paper, we propose a loss classification technique for the OBS networks based on machine learning techniques. We devise a new measure to differentiate between congestion and contention losses, which is derived from the observed losses, called the number of bursts between failures (NBBF). We observe that the NBBF follows a Gaussian distribution with different parameters for contention and congestion losses. This feature is used in differentiation. We use both a supervised learning technique (hidden Markov model (HMM)) and an unsupervised learning technique (expectation maximization (EM) clustering) on the observed losses and classify them into a set of states (clusters) after which an algorithm differentiates between the congestion and contention losses. We also demonstrate the use of loss differentiation in improving the performance of Transport Control Protocol (TCP) over OBS networks. We modify congestion control mechanism of TCP suitably to arrive at two variants of TCP, HMM-TCP and EM-TCP. Their performance is compared with TCP NewReno, TCP SACK, and Burst TCP [1]. Simulation results demonstrate the effectiveness and accuracy of the loss classification technique in different network scenarios.**

***Index Terms*— Optical burst switching, machine learning, hidden Markov model, EM clustering, loss classification, TCP.**

I. Introduction

OPTICAL burst switching (OBS) is a promising next-generation optical switching paradigm in wavelength division multiplexing (WDM) backbone networks [2]. It combines the merits of both optical circuit switching and packet switching because it avoids wastage of the resources due to wavelength reservation even in absence of traffic and does not require the use of optical processing logic or buffering. Packets from the Internet routers are aggregated into data bursts at an edge node, called ingress node, and transported entirely in optical domain at the intermediate nodes, called core nodes. It is generally assumed that core nodes do not have the capability of buffering optical bursts. The optical data bursts are disassembled again into packets at the edge nodes called egress nodes. Prior to sending a data burst, a control packet (also known as burst header packet (BHP)) is sent to reserve wavelength for the duration of the data burst. This separation of the control and the data planes eliminates the need for optical buffering and processing logic at the core nodes. However, due to the lack of confirmation of resource reservation, data bursts are dropped at the core nodes when a wavelength is not available during any time of the burst duration. Such a loss of bursts that occurs when the number of bursts simultaneously arriving at a core node exceeds the number of available wavelengths is called contention loss. Since contention losses mainly occur due to the use of one-way reservation mechanism and lack of optical buffers, they do not usually indicate a situation of congestion. Research in OBS mainly revolves around the different proposals to reduce the burst loss probability (BLP) [3] due to contention. For further details on the research issues in OBS the reader may refer to [2].

Usually, in the context of OBS networks congestion has been used to indicate prolonged contention losses [4]. Either loss rate or the link load or both have been used to differentiate between congestion and contention. The threshold values used for load and loss rate in differentiation of the losses are chosen qualitatively. In this paper, we use the definition for contention and congestion based on the overlap degree given in [5]. Loss of bursts when the number of bursts arriving simultaneously at a core node exceeds the number of wavelengths for a short duration is called *contention* loss. If bursts are lost persistently due to the shortage of bandwidth it is called *congestion*. Differentiation of these two types of losses is essential in many applications. Most of the work similar to that in [6] and [7] proposed routing protocols that consider the congestion on a path to minimize the losses. The authors of [8] proposed a feedback-based OBS network architecture wherein the nodes along a path communicate the load to the ingress nodes. This is used to reduce the rate of transmission of bursts and avoid congestion. False identification of contention losses with congestion leads to a TCP sender timing out (due to loss of multiple packets in a burst). Work in [1] provided a method to identify the false timeouts and proposed Burst TCP (BTCP), a variant of TCP for OBS networks, to improve the throughput. It was observed that identifying the type of loss (congestion or contention) improves the throughput of TCP by avoiding the reduction of congestion window size by half for each

Manuscript received March 24, 2008; revised May 22, 2008. This work was supported by Microsoft Corporation and Microsoft Research India under the Microsoft Research India Ph.D. Fellowship Award.

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai 600036, India (e-mail: jayaraj@cse.iitm.ernet.in, tvenku@cs.iitm.ernet.in, murthy@iitm.ac.in).

Digital Object Identifier 10.1109/JSACOCN.2008.033508.

packet in a burst lost. However, such cross-layer techniques to improve TCP performance use statistics collected at the OBS core nodes by the TCP source. For a survey on the issues in the study of TCP over OBS networks and different solutions in the literature, the reader may refer to [9].

The problem of loss classification can be formalized with Bayesian analysis. Let $T = \{C, W\}$ be the possible type of loss with C and W denoting congestion and contention, respectively. Let $L \in T$ be a random variable signifying the type of loss and R be a random variable taking a value of $\{0, 1\}$ signifying the outcome of each burst transmitted with 0 indicating a successful transmission and 1 indicating a burst loss. The quality of differentiation in a loss l possible with a particular outcome r can be expressed as $P[L = l|R = r]$. From a Bayesian perspective,

$$P[L = l|R = r] = \frac{P[L = l].P[R = r|L = l]}{\sum_{l' \in T} P[R = r|L = l'].P[L = l']}$$

In the above equation, we see that reasonable estimates can be made for the prior $P[R = r|L = l]$ for every l.[1] The distribution in the denominator can be directly estimated from the distribution of losses observed. That leaves only the priors $P[L = l]$ unknown. From this formulation we see that when the distribution of $P[R = r|L = l]$ is sufficiently distinct, we get a good estimate of the loss pattern. In this work, we use machine learning techniques to estimate the priors.

Machine learning techniques have been used earlier to classify packets and flows in the Internet (for example, in [10], [11]). Several methods have been proposed to differentiate congestion and wireless losses which are used to improve the throughput of TCP in wireless networks [12]. Generally, the loss classification algorithms depend on the analysis of statistical behavior of some observed values such as, packet inter-arrival time and round trip time with user defined threshold values. Since there are no queues at the intermediate nodes in an OBS network, it is not possible to use parameters like the variation in delay or the inter-arrival time to observe the incipience of congestion. We classify the losses in OBS networks based on a new metric, the number of bursts between failures (NBBF) defined in this work. We justify the use of this metric by showing that the NBBF in a set of observed losses follows a normal distribution. In this paper, we use machine learning techniques to classify the losses in OBS networks into two categories *viz*, contention and congestion. Machine learning techniques are popular for the classification of traffic or losses because they are known to identify any observable pattern in a desired metric even for dynamic traffic [11]. Supervised machine learning techniques require the definition of threshold values for classification which increases the chances of misclassification of losses around the threshold values. Therefore, we also use an unsupervised learning technique for the loss classification. We train hidden Markov model (HMM) (a supervised learning method) and expectation maximization (EM) clustering (an unsupervised learning method) on the pattern of NBBF and classify all the loss events into a certain number of states or clusters. Then a state labeling algorithm is proposed to classify the states (clusters) into congestion and contention. Using the loss classifier developed, we modify the congestion control mechanism of TCP to avoid sharp reduction in congestion window for a burst loss and thus improve the throughput. The end-to-end loss classification method proposed in this work avoids any explicit notification on the nature of loss from the core nodes or any measurement of load and loss rate at each node (except in the training phase of HMM-based technique). Instead of using the outcome of each burst transmitted to know the level of congestion on a path, we use the distribution of losses over a period of time. Intuitively, congestion is associated with a higher utilization of resources compared to contention so that we observe larger number of simultaneous burst drops[2]. Further, our classification approach has the advantage that it can be tuned to favour the identification of one type of loss over the other. For example, in this work since we use it to improve TCP performance, accuracy of identification of the congestion losses is favored. We observe from the simulation results that the loss differentiation in OBS networks finds a major application in improving the performance of TCP. Apart from this, the end-to-end loss classification can be adapted for use in routing, path and wavelength selection, and similar problems to improve the performance of OBS networks. We believe that the work reported in this paper is a first step towards differentiating a situation of congestion from contention losses in OBS networks.

The rest of this paper is organized as follows. We present a basic introduction to the machine learning techniques, HMM and EM clustering in Section II. Section III proposes the framework used to observe losses and the loss classification technique. We evaluate the performance of the classifiers as well as the TCP sender that uses the loss classification in Section IV and conclude the paper in Section V.

II. Machine Learning

In this section, we give a brief description of the supervised and unsupervised machine learning techniques used in our work. Using these techniques to classify the losses in OBS networks forms a part of the next section. Machine learning is programming systems to optimize a performance criterion using example data or past experience [13]. We have a model defined on some parameters and learning is optimization of parameters of the model using the past experience. The model developed may be *predictive* to make predictions in the future, or *descriptive* to gain knowledge from data, or both. Machine learning techniques have two phases; learning (training) phase and classification phase. In the training phase, the prior estimates are captured by building a model from the training data. The model built using training data is input to a classifier which then classifies the data set. Machine learning techniques can be divided into two categories; unsupervised and supervised. In unsupervised learning no labels are used on the training data to be classified. On the other hand,

[1]We obtain this from the distribution of NBBF, a metric defined in this work.

[2]We use 70% utilization of the bottleneck link to indicate high resource utilization. The reason for this choice is explained later in Section III-C.1.

supervised learning learns the classification using a set of man-made examples. In this work, we use EM clustering as a representative of unsupervised learning and HMM for supervised learning. For more details on machine learning, the reader may refer to [13].

A. Hidden Markov Models

HMM is a powerful modeling tool for two main reasons: first, HMM is a statistical signal model which can provide the basis for a theoretical description of any signal processing system; second, it works very well when applied appropriately [14]. A signal is normally expressed as a time series visualized to come from a discrete time Markov chain. At each state change, the chain generates an observation (signal) based on a probability distribution associated with the current state. In general, the observation can be either in discrete or in continuous form. In this paper, we focus only on HMM with discrete observations. Formally, an HMM is defined by the following elements [14]:

- N, the number of states in the model. We denote the state space as $S_1, S_2, \ldots, S_N$.
- M, the number of distinct observations. We denote the observation space as $v_1, v_2, \ldots, v_M$.
- The state transition probability distribution $A = a_{ij}$ where $a_{ij} = P[s_{t+1} = S_j | s_t = S_i], 1 \leq i, j \leq N$.
- For each state S_j, the probability of observing an observation symbol o, is given by $B = b_j(o)$ where $b_j(o) = P[o_t = v_m | s_t = S_j]$.
- The initial state distribution $\pi = [\pi_1, \pi_2, \ldots, \pi_N]$ where $\pi_i = P[s_1 = S_i], 1 \leq i \leq N$.

Training the HMM requires finding appropriate values for $\{a_{ij}\}$, $\{b_j\}$ and π. This inverse problem is usually attacked using an iterative approach. The approach seeks to find the parameters that make the set of observations most likely. With increasing number of iterations, the current HMM becomes more likely than the previous one to have generated the series of observations. Good explanation on training an HMM is given in [14], [15].

Once we have a trained model, we seek to find the state sequence $\hat{s} = \{\hat{s_1}, \hat{s_2}, \ldots, \hat{s_T}\}$ that corresponds to a given sequence of the observations $\hat{o} = \{\hat{o_1}, \hat{o_2}, \ldots, \hat{o_T}\}$. For this we use the Viterbi algorithm [14], which uses dynamic programming to perform an efficient inference of state sequence. If $\hat{\lambda}$ is the trained model (*i.e.*, the model after multiple iterations of the algorithm), the Viterbi algorithm produces a state sequence $\hat{s}$ such that the *a posteriori* log-likelihood $L(\hat{s}|\hat{o}; \hat{\lambda})$ is maximized, meaning that given the observation sequence $\hat{o}$, $\hat{s}$ is the most likely sequence of states followed by the model $\hat{\lambda}$.

B. EM Clustering

EM clustering is an unsupervised machine learning approach that uses the EM algorithm to classify un-labeled training data into groups called "clusters" based on similarity. The clustering phase is unsupervised because the algorithm does not have *a priori* knowledge of the true class of the object. For example, in our problem the algorithm does not know if the loss is due to congestion or contention. A good set of clusters should exhibit high intra-cluster similarity and high inter-cluster dissimilarity [16]. The algorithm computes some predefined set of parameters until a desired convergence value is achieved. The EM algorithm estimates the statistical parameters of a mixture of normal distributions. The finite mixtures model assumes that all the attributes of the objects to be clustered are independent random variables. A mixture is a set of N probability distributions where each distribution represents a cluster (N is the number of clusters). An individual instance is assigned a probability that it would have a certain set of values to the attributes given that it was the member of a specific cluster. The probability distribution is assumed to be normal and individual instances consist of a single real-valued attribute (in this case, the NBBF). The job of the algorithm is to determine the value of the mean, standard deviation, and the sampling probability (fraction of instances that an object belongs to that cluster) for each cluster. The general procedure for EM clustering is as follows [17]:

1) Guess initial values for the parameters; mean, standard deviation and sampling probability.
2) Use the probability density function (pdf) for a normal distribution to compute the cluster probability for each instance. In the case of a single independent variable with mean μ and standard deviation σ, the pdf for normal distribution is given by $f(x) = \frac{1}{(\sqrt{2\pi}\sigma)} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$. In case of two clusters, we have two pdfs each with a different mean and a standard deviation.
3) Use the probability scores to re-estimate the parameters.
4) Terminate the algorithm if the parameters do not change by more than a desired value.
5) Else, return to the computation of the pdf in Step 2.

The algorithm terminates when the measured cluster quality no longer shows a significant improvement. To measure the cluster quality, we use the likelihood that the desired data is similar to that determined by clustering. This is done by simple multiplication of the sum of probabilities for each of the instances. Other measures for the termination of the algorithm are fixing the number of iterations and the minimum allowed standard deviation. Once an acceptable set of clusters has been found using the above procedure, a classifier can be built out of the same. For each object in the classification phase, one of the clusters is identified as the most probable cluster depending on the probability that it belongs to each cluster. Ties among two or more clusters are broken randomly.

III. Loss Differentiation Technique

The loss classification technique proposed in this work has three phases. Using the $ns-2$ simulator with necessary OBS modules [18], we collect the data used for classification which is a sequence of outcomes of bursts transmitted. Then we apply HMM and EM clustering to classify the losses into congestion and contention losses. In the next section, we use both these classifiers to modify the congestion control mechanism of TCP and show an improvement in the throughput.

A. Framework for Collection of the Loss Statistics

We collected the loss statistics using the $ns-2$ simulator with OBS modules [18]. We used the NSFNET topology in which 6 edge nodes were used as ingress-egress node pairs. Traffic was generated with about 300 sources of which 90% were TCP and the rest were UDP. We placed sources uniformly across all the ingress nodes. We chose the source-destination pairs randomly and varied the load (by varying the number of TCP sources and the rate of UDP sources) on the path under consideration as well as the other correlated paths. Such a variation in the number of sources (as well as the source-destination pairs) and the rate of each source creates congestion in the link (characterized by more than 70% utilization of the link). In the topology used, we identified a few bottleneck links that get congested when the load is varied and ensure that measurement of losses is made along the paths with these links[3]. We measured the load as a percentage of the utilization of a link. When we refer to 70% load it indicates that 70% of the link capacity is used by the traffic. Each link had 64 wavelengths (of which 6 were used for control data) with 1Gbps transmission rate per wavelength. The propagation delay of the link was kept at 5ms. This simulation was run for a fixed duration (about $2,500$ secs) at the end of which we collected $25,000$ observations. An observation is basically the result of a burst transmission on a path. The observations were represented as a string of 0s and 1s with 1 representing a burst loss. The following procedure was used to observe the outcome of a burst sent on each path. Each burst sent between a pair of nodes has a unique identifier. Whenever a burst is successfully received at the egress node it is acknowledged and the ingress node marks the outcome as 0. We assume no out-of-order transmission so that a burst loss is detected at the egress whenever a burst with larger identifier is received.

B. Metric to Classify Losses

To determine the state of a path, *i.e.*, if it is congested or not, we use the consecutive number of bursts successfully sent in between two bursts lost. We define a parameter, the *Number of Bursts Between Failures (NBBF)* which is the number of bursts successfully received at an egress between any two bursts lost. We use the NBBF as an attribute for training (clustering) in the proposed classification techniques. The outcome of the bursts sent between the same ingress-egress pair is represented as a sequence of 0s and 1s. The sequence of observations (a binary string) is converted into a sequence of the NBBF. For example, if the sequence of outcomes is 10001000011001 then it is represented by a sequence of the NBBF as $(3,4,0,2)$. In case of the HMM-based classification, the reason for the loss (characterized by the utilization of the link) is also recorded whenever a loss is observed at the core node (this is done only to measure the accuracy of classification).

We note that the utilization of a link is different in case of contention and congestion losses. As mentioned earlier, persistence of burst losses is taken as a situation of congestion.

[3]Since there is no literature on OBS networks that uses any testbed experiments, we do not have more realistic ways of creating congestion scenarios. In the future work, we would investigate alternate ways of generating congestion losses.

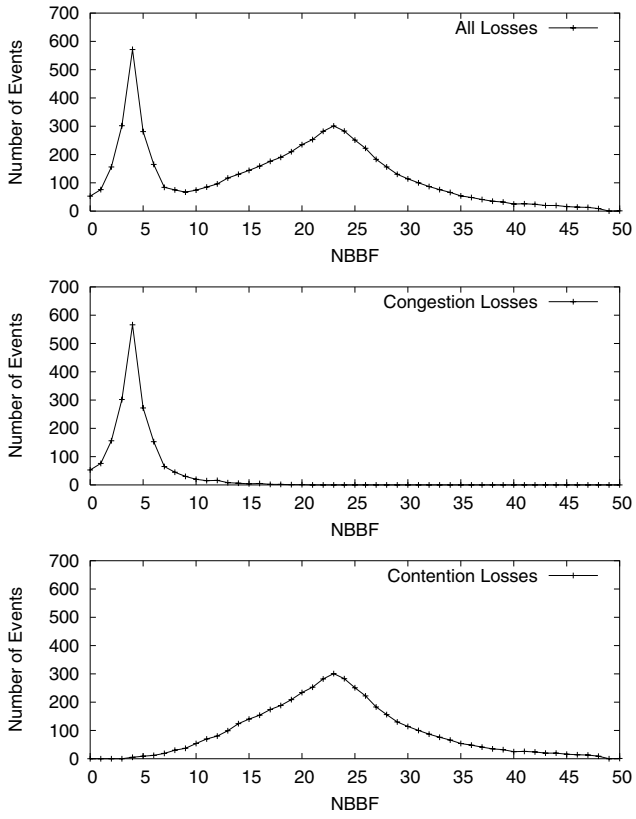

Fig. 1. Distribution of NBBF for losses by type (all, congestion and contention).

So we use the NBBF as a metric to characterize the state of a path. The distribution of NBBF obtained from the $25,000$ observations collected is shown in Fig. 1. The figure shows the distribution of NBBF for both contention losses and congestion losses. We show that the NBBF follows a different distribution for both types of losses and it captures the persistence of burst losses accurately. We observe that the NBBF values for congestion losses are distributed around a low mean with a small variance. This indicates that the path is congested and hence most of the time only a few bursts are sent successfully before a loss occurs. On the other hand, the NBBF values for contention losses have a larger mean and variance. We also observe that the NBBF follows a Gaussian distribution and hence it can be used as a metric to differentiate congestion from contention losses. In the supervised approach of classification, along with the distribution of losses we also use a threshold value for the utilization of link to label the state of a path (as given in the next section). In the unsupervised approach, we do not need to fix any such label and the loss events are clustered by the EM algorithm iteratively.

C. Design of the Loss Classifier

In this section we use the NBBF distribution in the observed losses and classify them into congestion or contention states. This is done by using either one of the two techniques independently to classify the observed set of losses into a given number of states (clusters) and then using a state labeling algorithm to identify the states of congestion and contention among them.

TABLE I
GAUSSIAN PARAMETERS FOR EACH STATE

State	Mean (μ)	Std. Dev. (σ)	CV (σ/μ)
S_1	3.03	0.48	0.16
S_2	3.57	0.57	0.16
S_3	4.17	1.21	0.29
S_4	5.56	2.32	0.42
S_5	8.34	2.31	0.28
S_6	12.50	3.40	0.27
S_7	25.00	5.87	0.24
S_8	33.35	6.24	0.19

1) HMM-based Approach: We use an HMM with 8 states, trained on $25,000$ observations[4]. For each loss at the core node, we also record its type to allow validation of our classifier. We use the link utilization to distinguish between congestion and contention. As seen in Fig. 1, the NBBF distribution is Gaussian. We show the mean, standard deviation and coefficient of variation (CV) for each of the eight states in Table I.

Note that the parameters of state 1 are similar to the NBBF distribution for the congestion losses shown in Fig. 1. So state 1 resembles the state of congestion closer than any other state. To verify this conjecture, we use the Viterbi algorithm [13] to estimate the states and determine the most likely state for each observation. Then, we use the information on utilization of the path to compute the number of times a loss event corresponds to each state which is plotted in Fig. 2. This plot along with Table I confirms that state 1 is in fact the state that corresponds to congestion closer than the other states. As mentioned earlier, we use 70% utilization of the link to indicate congestion. To arrive at this threshold, for different values of link utilization above 50%, we classified the losses into congestion and contention and then computed the error in classification. We found that this threshold value gave the lowest error in classification[5].

We see that the distribution of NBBF for congestion losses is quite different from that of contention losses. This difference is clearly due to the way in which these losses occur. Any state of the model, that corresponds to the congestion losses should have a fairly compact distribution of NBBF, with a low average value; *i.e.*, its Gaussian distribution should show a relatively smaller variance and mean. On the other hand, the NBBF for contention losses may typically be larger than that for congestion losses. These observations demonstrate that the mean, standard deviation and CV of each state help to classify the losses into congestion and contention. In Section IV-A, we measure the accuracy of this classifier using a metric *misclassification probability*.

2) EM Clustering-based Approach: We apply EM clustering on the losses observed with the number of clusters fixed to 8. In Section IV-A, we vary the number of clusters to observe its impact on the performance of the classifier. EM algorithm is terminated when there is no significant change in the log-likelihood value (10^{-4} here). After the termination, the mean, standard deviation, CV and sampling probability obtained for all the clusters are presented in Table II.

[4]It has been shown in [15] that in a number of settings, 4 states were sufficient to characterize a network communication channel using observations of loss events, and that $10,000$ observations were sufficient to train such a model.

[5]The error in classification is represented by the misclassification probability defined in Section IV-A

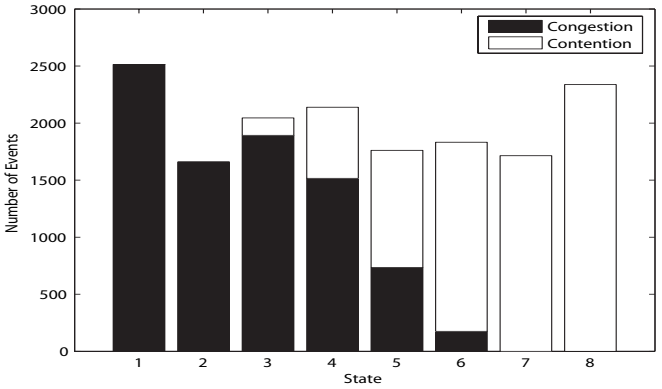

Fig. 2. Number of loss events corresponding to a state. For each state, the shaded portion denotes the events in the presence of congestion.

TABLE II
GAUSSIAN PARAMETERS FOR EACH CLUSTER

Cluster	Mean(μ)	Std.Dev.(σ)	CV(σ/μ)	Samp. Prob.
C_1	3.43	0.53	0.15	0.18
C_2	3.76	0.59	0.15	0.14
C_3	4.35	1.07	0.24	0.10
C_4	5.88	1.59	0.27	0.08
C_5	7.78	1.67	0.21	0.09
C_6	11.89	3.34	0.28	0.11
C_7	23.87	5.98	0.25	0.13
C_8	30.86	6.83	0.22	0.17

Similar to the HMM-based classification, we see that in each cluster the NBBF follows a Gaussian distribution. Note that the parameters corresponding to cluster C_1 represent congestion losses. From the parameters in the table, we see that clusters closer to C_1 can be labeled as those that correspond to congestion. It is difficult to measure the accuracy of unsupervised learning techniques because the data is not labeled manually. One good metric used to measure the quality of classification is the intra-cluster similarity, which is represented by the average CV. Further discussion on the accuracy of both the techniques is given in Section IV-A.

D. State Labeling Algorithm

Once we obtain the parameters for each state (cluster), we use a state labeling algorithm to classify the states (clusters) into two types; congestion or contention. The state labeling algorithm uses the Gaussian parameters as an input and creates two partitions based on the weights computed as given below. The same algorithm is used to classify both the set of states or clusters into two partitions to enable loss classification. In the algorithm, we use the term state to denote both the states of an HMM and the clusters in EM clustering technique. When the clustering method is used, we replace the state transition

TABLE III
WEIGHTS FOR THE 8 CLUSTERS COMPUTED BY THE STATE LABELING ALGORITHM

State	Mean(μ)	Std.Dev.(σ)	CV(σ/μ)	Weight
C_1	3.43	0.53	0.15	73
C_2	3.76	0.59	0.15	145
C_3	4.35	1.07	0.24	221
C_4	5.88	1.59	0.27	295
C_5	7.78	1.67	0.21	363
C_6	11.89	3.34	0.28	440
C_7	23.87	5.98	0.25	510
C_8	30.86	6.83	0.22	580

probabilities with the probability of an object moving between the clusters.

1) Partitioning the states into two groups:

- *Given*: The initial state distribution π, the state transition matrix A = $[a_{ij}]$, mean (μ), standard deviation (σ) and CV ($\frac{\sigma}{\mu}$) for each state.
- Partition the states into two disjoint sets with the restriction that each set should contain at least one state. There are $2^{N-1}-1$ such bipartite functions. Suppose that $\{P_1, P_2\}$ is one such partition and S_i be a state, then we define the flow between P_1 and P_2 as

$$f(P_1, P_2) = \sum_{S_i \in P_1, S_j \in P_2} (f_{ij} + f_{ji})$$

  where $f_{ij} = \pi_i.a_{ij}$ for $1 \leq i, j \leq N$ (product of a row of A with corresponding π).
- Sort the resulting $2^{N-1} - 1$ flows in order and select the two partitions associated with the median of inter-partition flow.

2) Label assignment to each partition:

- For $1 \leq i \leq N$, sort $[\mu_i]$, $[\sigma_i]$ and $[\sigma_i/\mu_i]$ in increasing order and determine their ranks. Rank(μ_i) $\geq$ Rank(μ_j) iff $\mu_i \geq \mu_j$. The ranks are computed similarly for $[\sigma_i]$ and CV.
- Assign a weight to each state S_i as follows:

$$w(S_i) = \alpha_1 Rank(\mu_i) + \alpha_2 Rank(\sigma_i) + Rank(\sigma_i/\mu_i)$$

  We use $\alpha_1 = N^2$ and $\alpha_2 = N$ after experimenting with different values since they are found to yield good results.
- The states in the partition with the smallest weight are termed as congestion states and the states in the other partition are termed as contention states. The weight of a partition is computed as the sum of the weights of all states in the partition.

Note that this method is subject to refinement for different scenarios. The weight assigned for each state is a function of the parameters of the state. Using the rank instead of the actual parameters improves the performance and adaptability of the algorithm.

Table III presents the weights computed by the state labeling algorithm for different clusters in the EM clustering method along with the parameters given in Table II. We note that the clusters C_1 to C_4 correspond to congestion states while the others represent contention losses. Note that the state labeling algorithm is same for both the learning techniques (HMM and EM clustering). Average CV for the EM clustering is 0.22 while that for the HMM technique is 0.25. This means that the EM clustering algorithm produces states with a higher degree of similarity than the HMM technique.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the accuracy and performance of the proposed loss classification techniques. We vary the load in the network, the number of wavelengths and use both dynamic and static traffic to evaluate the performance of the classifiers. Further, we demonstrate the use of loss classification to improve the performance of TCP over OBS networks. All the simulations were run on the $ns-2$ simulator with necessary OBS modules [18]. We used the same setting used to collect loss observations mentioned in Section III-A. We used the Latest Available Unused Channel with Void Filling (LAUC-VF) scheduling algorithm, Just-Enough-Time (JET) reservation protocol, and a mixed time-threshold assembly algorithm [2]. We do not assume wavelength conversion capability at the core nodes. The TCP traffic was generated using FTP connections with average duration of $1,500$ secs[6]. All the results were obtained with 95% confidence level and the intervals are shown in the plots.

A. Performance of Classification Techniques

In a supervised learning technique, the data is labeled manually *i.e.*, we define states and classify losses into them. Hence, the accuracy of the technique can be measured by *misclassification probability*. Let $P[t|t']$ be the probability of an observation being classified as belonging to the loss type t given that it is in fact due to the loss type t', for $t, t' \in T$, where T is the type of loss (contention or congestion). Specifically, let the event "being classified as a contention loss" be denoted as W and "being classified as a congestion loss" be denoted C; and let the event "the loss is actually due to congestion" be denoted $\mathcal{C}$ and "the loss is actually due to contention" be denoted $\mathcal{W}$. Among these metrics, $P[W|\mathcal{W}]$ is appropriate for evaluating the accuracy of classifying contention losses, and $P[C|\mathcal{C}]$ is appropriate for the congestion losses. It is important to note that these metrics can vary simultaneously. The *misclassification probability*, η is defined as $\eta = P[C|\mathcal{W}] + P[W|\mathcal{C}]$.

Different network parameters such as the load (measured as a percentage of the capacity of the bottleneck link), the number of wavelengths and the type of traffic (static or dynamic) may affect the accuracy of loss classification by affecting the distinctiveness of the distribution of NBBF around each type of loss. The misclassification probability under varying network load is presented in Fig. 3. It is observed that η is highest in the case of moderate load, because the distribution of NBBF contains a mixture of both congestion and contention losses and hence an HMM cannot make a clear distinction between them. Under low and high load HMM can easily associate the NBBF distribution with a type of loss and thus η is small.

[6]We considered only long-lived TCP flows in this work. Using loss classification for short-lived flows needs to be addressed separately.

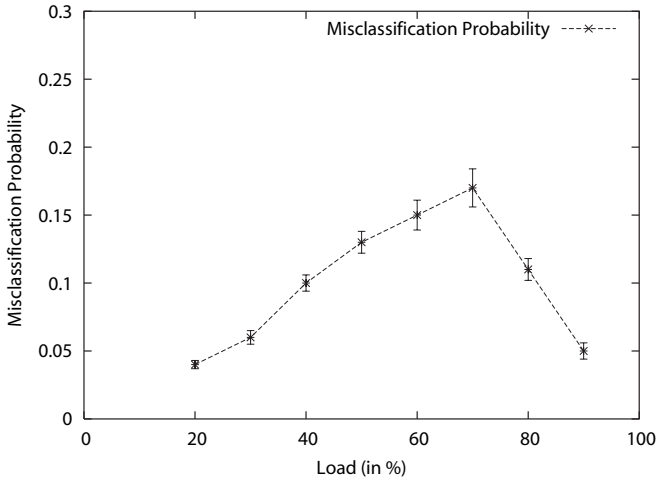

Fig. 3. Variation of misclassification probability with network load. Confidence intervals are also shown.

TABLE IV
ACCURACY OF CLASSIFICATION FOR DIFFERENT TYPES OF TRAFFIC

Type of traffic	BLP	$P[C\|\mathcal{C}]$	$P[W\|\mathcal{W}]$
Static	0.17	1.00	0.90
Dynamic	0.19	1.00	0.84

Table IV shows the accuracy of the HMM-based classification for static and dynamic traffic. In case of static traffic the load (the number of TCP sources) on all the paths is fixed for the duration of the simulation while it is varied continuously for the dynamic case. We can see that the classification technique has better accuracy for the static traffic than the dynamic case because, HMM has to react to the variation in the loss pattern under dynamic traffic. Since we favoured accurate identification of the congestion losses, $P[C|\mathcal{C}]$ is unity. We also study the effect of the number of wavelengths on a link on the classification accuracy. From Table V, we note that as the number of wavelengths increases, the misclassification probability decreases. This is because when there are more wavelengths, HMM can easily differentiate between the case of continuous lack of resources from the case of unavailability for a short duration. In the results shown in both Table IV and Table V we varied the load from 20% to 80% and computed the average values of burst loss probability (BLP) and the accuracy (error) in classification.

Unlike HMM-based classification, EM clustering works on un-labeled data so that its accuracy cannot be measured with a parameter like the misclassification probability. An important parameter that affects the performance of this technique is the number of clusters. If the number of clusters specified is large, then the clusters have higher intra-cluster similarity which is reflected in a low CV. We vary the number of clusters in the EM clustering and evaluate the Gaussian parameters for each cluster. The average CV for a given number of clusters is plotted in Fig. 4. As the number of clusters increases, the clusters have a higher degree of intra-cluster similarity and hence lower average CV. In our work, we observed that increasing the number of clusters beyond eight did not improve

TABLE V
ACCURACY OF CLASSIFICATION FOR DIFFERENT NUMBER OF WAVELENGTHS ON A LINK

No. of Wavelengths	BLP	Misclassification Probability
8	0.15	0.14
16	0.15	0.10
24	0.17	0.10
32	0.14	0.08
64	0.18	0.04

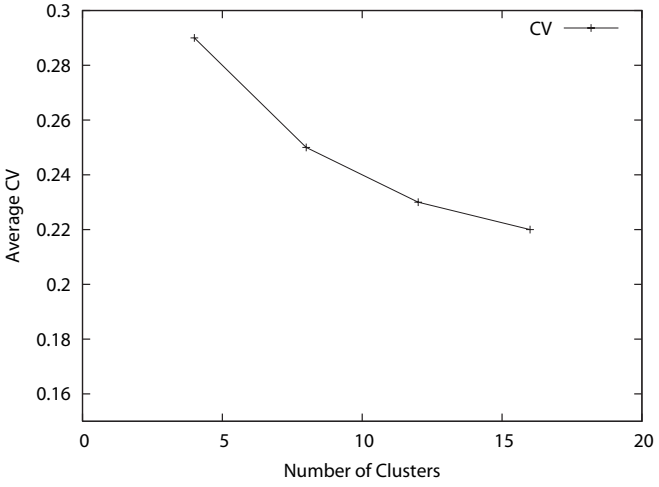

Fig. 4. Variation in the average CV with number of clusters.

the performance of the classifier because we ultimately map all the clusters to two sets.

In conclusion we discuss the pros and cons of both these techniques for loss classification before we apply them to improve the performance of TCP. One of the main disadvantages of the HMM-based loss classification in OBS networks is that it required the definition of a threshold value for link utilization to label the congestion states. As mentioned in Section III-C.1 we found that identifying congestion with losses associated with link utilization of 70% minimized the error in classification. The choice of the threshold value can vary in other scenarios. It is also difficult to maintain the utilization of all the links at the edge nodes. Finally, special care has to be taken while obtaining the training data (NBBF distribution) from the simulator (or from a real network) so that different scenarios of the network are considered. EM clustering (an unsupervised technique in general) overcomes most of these limitations. It works on the training data iteratively and classifies the loss events into clusters. Without any label given to the clusters, one can still make reasonable use of the pattern in losses to identify the state of a path. In the next section, when we use both the classification techniques to improve the performance of TCP, we see that the EM clustering performs slightly better than the HMM classification. It must be noted that the state labeling algorithm in Section III-D also affects the accuracy of classification.

Note that the machine learning techniques require a training phase to initially classify the losses. This is done offline using the observations collected from the simulator. On a Pentium

4 computer with a 3.0 GHz processor, and 512MB RAM it took about 18 secs with 26 iterations for an 8-state HMM to classify the 25,000 observations. Similarly, it took about 17 secs with 24 iterations for the EM clustering. However, we found that it takes much lesser time to classify losses online with a trained classifier (i.e., time taken by the TCP sender to associate losses with different states).

B. Application of Loss Classification to Improve TCP Performance

We demonstrate the use of machine learning-based loss classification to improve the performance of TCP over OBS networks. We modify the congestion control mechanism of TCP such that it halves the congestion window for a burst loss only if the path is observed to be congested at that time. To know if the path is congested or not, we use the classifiers developed in the previous section. We found that after training the classifiers, to identify the state of a path, it took only about 1.5 times the round trip time of a TCP connection (in the order of a few *ms*). As mentioned in [1], whenever a burst is lost in OBS network, the TCP sender perceives it as a simultaneous loss of all the packets in the burst. In the congestion avoidance phase, TCP halves the current congestion window for each packet lost. Depending on the number of packets in a burst lost, the TCP sender might set a low value of congestion window or go to a timeout state. This leads to a low throughput most of the time. The over-cautious approach of TCP is not desired in OBS networks because burst losses do not necessarily indicate congestion. To improve the throughput, several variants have been proposed in the literature [9]. Most of the work uses a method of notifying the TCP sender that a burst loss is not associated with congestion on the path and thus avoids drastic reduction of the window. Some of them use explicit feedback from the core nodes while the others use end-to-end methods to identify false congestion (contention losses).

The loss classification techniques are mainly used to know if the path is congested or not and we can avoid halving the congestion window unnecessarily (for the temporary contention losses). We use both HMM and EM clustering methods to identify congestion and detect false timeouts. This improves the throughput by reducing number of times the congestion window is halved. Whenever a burst loss occurs in the core network, the TCP window size is reduced only if the classifier identifies that the path is in a congested state. Since TCP sender automatically identifies a loss with duplicate acknowledgments, we do not need an explicit feedback to detect a burst loss. To know if the path is congested, we use the techniques proposed for loss differentiation. We call the two variants of TCP, that use loss classification at the ingress node to identify congestion as, HMM-TCP (TCP using HMM-based loss classification) and EM-TCP (TCP using the EM-clustering based loss classification). The congestion window is reduced only once irrespective of the number of packets lost due to a burst loss.

We compare the performance of these variants with BTCP proposed in [1], TCP NewReno, and TCP Selective Acknowledgment (SACK). BTCP is also capable of handling multiple

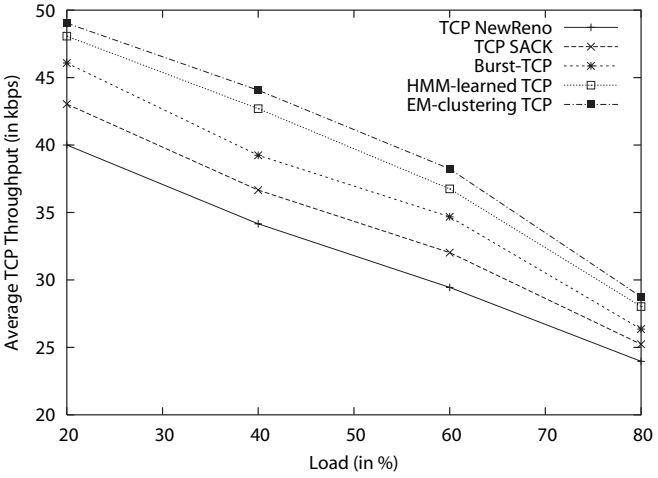

Fig. 5. Variation in average TCP throughput with load.

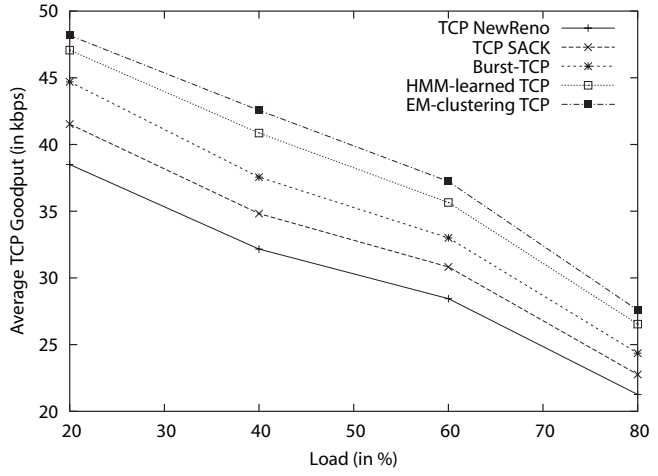

Fig. 6. Variation in average TCP goodput with load.

losses in a single burst. It reduces the window size only once for many packets lost in a single burst and so performs better than TCP NewReno. TCP SACK uses the selective acknowledgments option in the acknowledgment packet to indicate all the packets lost which are retransmitted at once. It was found by the authors of [1] that TCP SACK has the best performance among the three variants of TCP (*viz.*, Reno, NewReno and SACK) in OBS networks [1]. BTCP uses explicit feedback and does not actually differentiate between a contention loss and congestion. It must be noted that since we intend to show improvement in TCP performance only as an application of the loss differentiation technique, we do not concentrate on tuning the loss classifiers to improve TCP throughput.

We evaluate HMM-TCP, EM-TCP, BTCP, TCP NewReno, and TCP SACK by comparing the average throughput, goodput and number of timeouts for a single flow. We vary the load on the path, measured as a percentage of the maximum link capacity (64Gbps in all results), and plot these metrics. As can be seen from Fig. 5 and Fig. 6, both TCP variants that use the machine learning-based classification have a higher throughput and goodput than the other variants. The improvement is higher for low and medium load because the machine learning

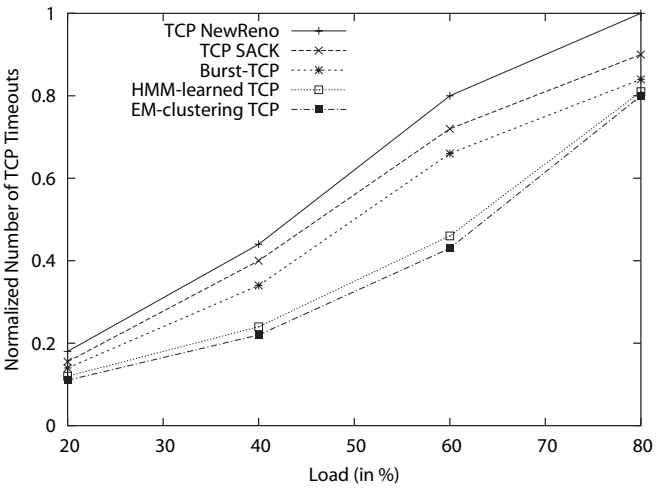

Fig. 7. Normalized number of TCP timeouts with varying load.

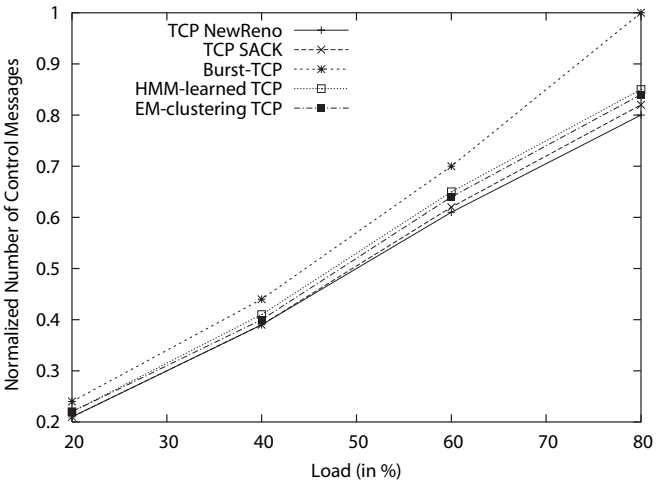

Fig. 8. Variation in the normalized number of control messages with load.

techniques have better accuracy of classification.

We can also see from Fig. 7 that the number of times a TCP sender goes to timeout state is also smaller for our variants than that for the others. We normalized the number of timeouts with that obtained at 80% load to show the relative performance. It can be seen that EM-TCP performs slightly better than HMM-TCP with respect to all the metrics as the clusters formed by the EM algorithm have a higher degree of "intra-cluster similarity"(*i.e.*, lower average CV).

We also study the control overhead incurred due to the messages used to know the state of a path (including the TCP acknowledgments) in the machine learning-based variants of TCP. Fig. 8 shows the variation in the number of control messages (normalized with that at 80%) for all the TCP variants as the load increases. The control overhead of our variants is lower than that for BTCP but higher than that for TCP NewReno and SACK (which do not use any additional messages apart from the regular acknowledgments).

V. Conclusion

In this paper, we addressed the problem of differentiating contention losses from a situation of congestion in OBS networks. We proposed a new measure called the NBBF which was found to follow a Gaussian distribution with different parameters for contention and congestion losses. We used a supervised machine learning technique *viz*, HMM and an unsupervised learning technique *viz*, EM clustering to train on the NBBF observed and classify the set of losses into states (clusters). The set of losses were classified into two types, congestion and contention using a heuristic technique. We evaluated the accuracy of the classifiers under different network conditions. We compared our TCP variants (which use loss differentiation at the source) with TCP NewReno, TCP SACK, and BTCP to demonstrate the use of end-to-end loss differentiation mechanisms in improving the performance of TCP. Apart from this application, loss classification can also be used in routing, path and wavelength selection, and similar problems to improve the performance of OBS networks. We believe that the work done in this paper is a first step taken to differentiate contention losses from congestion losses in a better fashion. We also believe that this work opens new directions in the problem of loss differentiation (classification) in OBS networks. Using machine learning techniques for loss classification has an advantage that they can be tuned for use in different applications even in dynamic network scenarios.

References

[1] X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false time out detection in OBS networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 774–784.

[2] V. M. Vokkarane and J. P. Jue, *Optical Burst Switched Networks*, 1st ed. Springer, 2004.

[3] C. M. Gauger, M. Kohn, and J. Scharf, "Comparison of contention resolution strategies in OBS network scenarios," in *Proc. IEEE International Conference on Transparent Optical Networks*, July 2004, pp. 18–21.

[4] A. Maach, G. V. Bochman, and H. Mouftah, "Congestion control and contention elimination in optical burst switching," *Telecommun. Systems*, vol. 27, no. 2-4, pp. 115–131, 2004.

[5] J. Li, C. Qiao, J. Xu, and D. Xu, "Maximizing throughput for optical burst switching networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 1853–1863.

[6] G. P. V. Thodime, V. M. Vokkarane, and J. P. Jue, "Dynamic congestion-based load balanced routing in optical burst-switched networks," in *Proc. IEEE GLOBECOM*, Dec. 2003, pp. 2628–2632.

[7] J. Li, G. Mohan, and K. C. Chua, "Dynamic load balancing in IP-over-WDM optical burst-switched networks," *Computer Networks*, vol. 47, no. 3, pp. 393–408, 2005.

[8] L. Y. Kim, S. Lee, and J. Song, "A dynamic load-aware congestion control scheme in optical burst switching networks," *Photonic Network Commun.*, vol. 13, no. 3, pp. 277–287, June 2007.

[9] B. Shihada and P.-H. Ho, "Transport control protocol (TCP) in optical burst switched networks: Issues, solutions, and challenges," to appear in *IEEE Commun. Surveys and Tutorials*, 2008.

[10] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *Proc. Passive and Active Measurements Workshop*, Apr. 2004.

[11] A. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *Proc. ACM SIGMETRICS*, June 2005, pp. 50–60.

[12] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 703–717, 2003.

[13] E. Alpaydin, *Introduction to Machine Learning*, 1st ed. MIT Press, 2004.

[14] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proc. IEEE*, June 1989, pp. 257–286.

[15] K. Salamatian and S. Vaton, "Hidden Markov modeling for network communication channels," in *Proc. ACM SIGMETRICS*, June 2001, pp. 92–101.

[16] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.

[17] A. Dempster, N. Paird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.
[18] OIRC OBS-ns Simulator. [Online]. Available: http://wine.icu.ac.kr/ obsns/index.php

A Jayaraj is currently in the final year of the Dual Degree (B.Tech and M.Tech) programme at the Department of Computer Science and Engineering, Indian Institute of Technology Madras, India. He is presently working in the area of Optical Burst Switched Networks. His research interest includes performance evaluation of systems and networks, wireless and sensor networks, real-time and distributed systems and optical networks.

T Venkatesh received his B.Sc (Hons) in Physics and M.Sc in Physics from Sri Sathya Sai Institute of Higher Learning, India in 1999 and 2001 respectively, and M.Tech degree in Electrical Engineering from the Indian Institute of Technology, Kanpur in 2003. After that he worked with the Applied Research Group of Satyam Computer Services Ltd., Bangalore for an year. He is currently pursuing his Ph.D. in the Department of Computer Science and Engineering, Indian Institute of Technology, Madras, India. He is a recipient of Microsoft Research India Fellowship and his research interests include WDM networks and systems, performance analysis of computer networks and IP over WDM networks.

C Siva Ram Murthy earned his B.Tech. degree in Electronics and Communications Engineering from Regional Engineering College (now National Institute of Technology), Warangal, India, in 1982, M.Tech. degree in Computer Engineering from the Indian Institute of Technology (IIT), Kharagpur, India, in 1984, and Ph.D. degree in Computer Science from the Indian Institute of Science, Bangalore, India, in 1988. He has been with the Department of Computer Science and Engineering at IIT Madras, since 1988, where he is currently a Professor.

He is the co-author of the textbooks *Resource Management in Real-time Systems and Networks*, (MIT Press, Cambridge, Massachusetts, USA), *WDM Optical Networks: Concepts, Design, and Algorithms*, (Prentice Hall, Upper Saddle River, New Jersey, USA), and *Ad Hoc Wireless Networks: Architectures and Protocols*, (Prentice Hall, Upper Saddle River, New Jersey, USA). His research interests include real-time systems, lightwave networks, and wireless networks. He is a Fellow of the Indian National Academy of Engineering, an Associate Editor of *IEEE Transactions on Computers*, and a Subject Area Editor of *Journal of Parallel and Distributed Computing*.