Online Flow Size Prediction for Improved Network Routing

Pascal Poupart
University of Waterloo
Waterloo, Ontario, Canada
ppoupart@uwaterloo.ca

Zhitang Chen
Huawei Technologies
Shatin, Hong Kong
chenzhitang2@huawei.com

Priyank Jaini
University of Waterloo
Waterloo, Ontario, Canada
pjaini@uwaterloo.ca

Fred Fung
Huawei Technologies
Shatin, Hong Kong
fred.fung@huawei.com

Hengky Susanto
HKUST
Clear Water Bay, Hong Kong
hsusanto@cs.uml.edu

Yanhui Geng
Huawei Technologies
Shatin, Hong Kong
geng.yanhui@huawei.com

Li Chen
HKUST
Clear Water Bay, Hong Kong
lchenad@ust.hk

Kai Chen
HKUST
Clear Water Bay, Hong Kong
kaichen@cse.ust.hk

Hao Jin
HKUST
Clear Water Bay, Hong Kong
hjinae@connect.ust.hk

ABSTRACT

We describe an emerging application of data mining in the context of computer networks. This application concerns the problem of predicting the size of a flow and detecting elephant flows (very large flows). Flow size is a very important statistic that can be used to improve routing, load balancing and scheduling in computer networks. Flow size prediction is particularly challenging since flow patterns continuously change and predictions must be done in real time (milliseconds) to avoid delays. We describe how to formulate the problem as an online machine learning task to continuously adjust to changes in flow traffic. We evaluate the predictive nature of a set of features and the accuracy of three online predictors based on neural networks, Gaussian process regression and online Bayesian Moment Matching on three datasets of real traffic. We also demonstrate how to use such online predictors to improve routing (i.e., reduced flow completion time) in a network simulation.

Keywords

Networks, online flow size prediction, routing

1. INTRODUCTION

Data centers keep growing in size and their applications are increasingly distributed. The increasing demand for bandwidth is met by multiplying the number of links and paths between nodes in commodity data centers to achieve full-bisection bandwidth. Initially, this multiplicity was not thought to be an issue since one can easily distribute flows (packet sequences) over all equal length paths (e.g., ECMP [7]) in order to take advantage of the aggregate bandwidth, however some flows consume much more bandwidth than others and in practice some paths may become congested while others are underutilized. This arises in part because the size of flows varies tremendously (small mice flows of a few kilobytes to large elephant flows of many gigabytes) and the time they occupy a path varies considerably (milliseconds to hours). Since elephant flows occupy a path much longer than mice flows, there is a risk that the number of *active* flows become imbalanced due to a concentration of elephant flows in some links. For instance, simple heuristics that ignore flow size and distribute flows evenly over all equal length paths (e.g., ECMP [7]) often lead to congestion and load balancing heuristics must be used to detect and correct imbalances. Flow size is also very important in scheduling techniques that seek to minimize the average flow completion times (FCT) [5]. For instance, shortest job first is a popular and effective scheduling technique that prioritizes flows based on their size [13].

Recent work has shown that it is possible to infer and even predict flow sizes for some applications. Mahout [4] installs a shim layer at the end hosts to detect elephant flows by monitoring socket buffers that exceed some threshold. More generally, since it may not be possible to modify the networking stack in end hosts, several approaches have been proposed to monitor flows in the network and detect elephants based on thresholds with respect to the amount of data transmitted so far or the bandwidth utilized [1, 6]. These approaches can only detect elephant flows after they have been flowing for a while and therefore they only permit re-routing and re-scheduling. This is not ideal since congestion may occur until elephant flows are detected and new routes are chosen by load balancing or the priorities of the elephant flows are decreased to allow mice flows to complete without any delays.

We propose to use data mining to estimate the size

of each flow as it starts and to detect elephant flows without modifying any application or end host. Flow size estimators based on machine learning can be easily inserted in logically centralized controllers for software defined networks. Such estimators can use features of the first few packets (e.g., packet size, header information and TCP synchronization) to predict the size of each flow as it starts. The predicted size can then be made available to routing and scheduling modules to improve traffic engineering and reduce completion times. In this work, we describe how to use three machine learning techniques (Gaussian processes, Gaussian Mixture Model with Bayesian Moment Matching and neural networks) to estimate the size of each flow based on historical data as well online streaming data.

The paper is structured as follows. Section 2 reviews previous work about elephant flow detection and scheduling techniques based on flow sizes. Section 3 describes how to use Gaussian processes, Gaussian Mixture Models and neural networks for flow size prediction in an online fashion in order to gradually tailor predictions to a network and continuously adapt to changes in traffic patterns. Section 4 evaluates the accuracy for flow size estimation on real data from three sites. Section 5 evaluates the benefits of elephant flow predictions as part of a simple routing heuritic to minimize average flow completion time in a network simulation. We conclude in Section 6.

2. BACKGROUND

2.1 Elephant Flow Detection

Flow size estimation techniques can be divided in two groups: those that modify applications and/or end hosts [4, 14, 10] and those that do not [6, 12]. While the most reliable way of estimating flow sizes is by modifying the applications to expose the size of the flows they generate, this is not always possible. Hence, many generic approaches have been proposed to detect elephant flows by monitoring some statistics in the network [1, 6, 14]. They typically define some thresholds on the amount of data transmitted or the bandwidth utilized and any flow that exceeds a threshold is categorized as elephant. Thresholding the amount of data transmitted ensures that large flows are detected, but if the rate at which the flow is generated by an application is limited, the flow may use a negligible amount of bandwidth, in which case there is no point flagging this flow since it will not cause any congestion. Instead some systems focus on thresholding the amount of bandwidth used [1, 6], but in heavily congested scenarios, many flows may compete to the point where each flow may receive only a small portion of the bandwidth without exceeding the threshold. The assumption is that congestion will gradually build up due to an increasing number of elephant flows, allowing the system to detect the first few elephant flows and to re-route them before their share of the bandwidth dips below the threshold due to heavy competition.

Monitoring techniques are *reactive* in the sense that they only detect elephant flows after they have been flowing for some time. Realizing the limitation of the monitoring techniques, we investigate in Section 3 the use of three machine learning techniques (Gaussian process, Gaussian Mixture Models and neural networks) that can estimate flow sizes (in addition to detecting elephant flows) and that can be trained in an online fashion.

3. FLOW SIZE PREDICTION

We explore the use of machine learning techniques to predict the size of each flow and detect elephant flows as they start. Formally, we are looking for a function $f : features \rightarrow size$ that takes as input some features of a flow and outputs an estimate of the size of the flow. In some cases, the function may return a distribution over the size (i.e., $f : features \rightarrow pdf(size)$). A single size estimate can be obtained by computing the mean (i.e., $mean = \int_{size} pdf(size)\ size$) or the mode (i.e., $mode = argmax_{size}\ pdf(size)$) of the distribution, while the distribution indicates the confidence of the predictor in the estimate. In other situations, we only care about detecting elephant flows that are larger than some threshold. The output of the function can then be thresholded (i.e., $elephant = true \iff size > threshold$) or a classifier that directly predicts whether a flow is an elephant or mouse can be learned (i.e., $f : features \rightarrow \{elephant, mouse\}$).

In the following subsections we describe the features extracted, how they are preprocessed, the machine learning techniques that use those features and how to continuously learn by online learning.

3.1 Features and Preprocessing

We consider seven features that are readily available for each flow: source IP, destination IP, source port, destination port, protocol, server vs client and size of the first three packets. Here, the first three packets refer to the first three *data* packets (after the protocol/synchronization packets). When the protocol of the flow is TCP, the server vs client feature indicates whether the flow is a request initiated by a *client* or a response provided by a *server*. The server vs client feature is absent in the case of UDP flows. The remaining features can be extracted from the header of the first packet. Hence flow size can be estimated immediately upon the start of a flow.

3.2 Machine Learning Techniques

A wide range of supervised regression and classifi-

cation techniques can be used for flow size prediction and/or elephant detection. We consider neural networks, Gaussian processes and the Gaussian Mixture Models as representative techniques. Each aglorithm can be trained both in an offline and online fashion.

- Neural Network (NN) [3]: We used an input layer of 106 binary nodes, two hidden layers of 60 and 40 hyperbolic tangent nodes respectively and an output layer of one node. For classification, we use the hyperbolic tangent activation function to output a value between -1 and 1. When the output of the neural network is negative, the flow is classified as mouse and when the output is positive the flow is classified as elephant. The 106 inputs consist of the features described in the previous section, where each bit is transformed into a -1/+1 value.

- Gaussian Process Regression (GPR) [11]: This is a non-parametric Bayesian model that allows us to compute a distribution over flow sizes based on the input features. More specifically, when given a new flow f, the model computes a Gaussian distribution $N(s; \mu, var)$ over the size s of the flow. The predicted size s^* of a new flow f^* is often taken to be the mean μ, which is a weighted sum of the flow sizes $\mathbf{s}$ in the network:

$$s^* = \mu = K(f^*, F)[K(F, F) + \eta I]^{-1}\mathbf{s} \quad (1)$$

  Here I is the identity matrix and η is a noise estimate (variance) of the flow sizes for any fixed set of features. Suppose we have a training set of n flows, then F denotes this set of flows and $\mathbf{s}$ is a $n \times 1$ column vector of sizes corresponding to those flows. $K(f^*, F)$ is a $1 \times n$ row vector indicating the similarity between f^* and each flow in F. Similarly, $K(F, F)$ is a $n \times n$ matrix indicating the similarity between every pair of flows in F.

- Online Bayesian Moment Matching (oBMM) [9]: The idea is to use Gaussian Mixture Models (GMMs) for elephant flow prediction. Let C_e denotes the class of elephant flows and C_m denotes the class of mice flows. We assume that feature of each class is dervied from a GMM as follows:

$$\mathbf{x}|C_e \sim \sum_{i=1}^{N_e} w_i^e \mathcal{N}(\mathbf{x}; \mu_i^e, \Sigma_i^e)$$
$$\text{and, Similarly } \mathbf{x}|C_m \sim \sum_{i=1}^{N_m} w_i^m \mathcal{N}(\mathbf{x}; \mu_i^m, \Sigma_i^m), \quad (2)$$

  where the parameters $(w_i^e, \mu_i^e, \Sigma_i^e)$ and $(w_i^m, \mu_i^m, \Sigma_i^m)$ are learnt from data using online Bayesian Moment Matching (oBMM) [8]. oBMM is an tractable online Bayesian learning algorithm for learning mixture models using the method of moments. Online Bayesian learning for mixture models is intractable because the number of terms in the posterior grow at an exponential rate with the data points observed. oBMM circumvents this problem by approximating the exact posterior with a different family of distribution. This approximate distribution is obtained by matching the sufficient moments of the distribution of the exact posterior. When we do prediction, we just assign a flow with features $\mathbf{x}$ to class C_e if $P(C_e|\mathbf{x}) > P(C_m|\mathbf{x})$, where $P(C_e|\mathbf{x})$ and $P(C_m|\mathbf{x})$ are obtained by Bayes Theorem.

4. FLOW SIZE PREDICTION EXPERIMENTS

We report several experiments to evaluate the accuracy and efficiency of flow size prediction. Those experiments are performed with three public datasets of real traffic from two universities [2] and an academic building at Dartmouth College[1]. Each dataset consists of packet traces for more than 3 million flows. The datasets from the two universities include TCP and UDP flows, while the dataset from Dartmouth College consists of the TCP flows of one building only (spring02/AcadBldg16). We vary the threshold to separate elephant flows from mice flows from 10 Kb to 1 Mb. Since practitioners may choose different thresholds depending on how the predictions will be used, in the following experiments we report the accuracy of the predictors for each threshold.

Since the percentage of elephant flows is low (12% to less than 0.1%), the classes of elephant and mice flows are imbalanced. This can impact negatively the training of a classifier since the classifier will tend to classify all flows as mice unless there is clear evidence that a flow is elephant. To mitigate the impact of class imbalances, we re-weight the data instances to increase the importance of misclassifying elephant flows while decreasing the importance of misclassifying mice flows. More specifically, we multiply the errors associated with elephant misclassification by a weight that is inversely proportional to their percentage during training.

4.1 Prediction Accuracy

We evaluate the accuracy of elephant flow detection for various flow size thresholds. We report the true positive rate (TPR) and true negative rate (TNR) which correspond respectively the percentage of elephant flows that are correctly identified and the percentage of mice

[1]http://crawdad.org/dartmouth/campus/

flows that are correctly identified.

$$TPR = \frac{\#\text{ of correctly identified elephant flows}}{\#\text{ of elephant flows}} \quad (3)$$

$$TNR = \frac{\#\text{ of correctly identified mice flows}}{\#\text{ of mice flows}} \quad (4)$$

Figures 1 report the TPR and TNR of each algorithm as we vary the classification threshold. There is a natural tradeoff between TPR and TNR, which is why the algorithm that performs best on one measure is not usually the one that perform best on the other measure. The neural network and the online Bayesian Moment Matching tend to be affected by class imbalances more than the Gaussian process which explains why their accuracy often suffers as the classification threshold increases. The Gaussian process is more robust to class imbalances since it effectively maintains a distribution over all possible hypotheses.

5. ROUTING EXPERIMENTS

In this section, we devise a simple routing policy that uses flow size predictions to identify elephant flows and route them via a least congested path. We compare this approach to Equal Cost Multiple Path (ECMP) [7], which is a simple and popular routing policy that is often used in real networks. Assuming a network with a fixed structure, all least cost paths (in terms of the number of hops) are pre-computed for every pair of source and destination IP addresses. When a new flow enters the network, ECMP chooses a path from this set at random or according to a hash function on the flow id. In both cases, flows tend to be distributed evenly over the set of least cost paths. However, even if the flows are distributed perfectly evenly, congestion often occurs since elephant flows occupy their paths much longer than mice flows, creating load imbalances in some parts of the network. To counter this, we devise a simple policy that inspects each new flow to predict whether it will be an elephant or a mouse flow. If a flow is predicted to be a mouse, it is routed by ECMP, but if it is predicted to a be an elephant, it is routed via a least congested path. An approximately least congested path is determined by monitoring the number of active elephant flows in each link and choosing the path that will yield the most bandwidth assuming fair sharing of the bandwidth in each link among the elephant flows going through each link. While it is tempting to bypass the detection of elephant flows and route all flows (including mice flows) via a least congested path, this does not work well in practice. Mice flows might all be assigned the same path, creating very short collisions that are not detectable by most load monitors. Nevertheless, these collisions will increase significantly the completion time of those mice flows. Hence distributing mice flows by ECMP works much better in practice. From now on, we will denote by least congested (LC) path the policy that detects elephant flows (greater than 100 Kb) based on a flow size predictor and routes them via a least congested path, while routing mice flows by ECMP.

In order to compare LC to ECMP, we utilize a network simulator that works at the flow level (instead of the packet level). Given a network topology, incoming traffic and a routing policy, the simulator measures the completion time of each flow. We used the network topology shown in Fig. 2, which is a simple fat tree topology common in data centers. There are 16 servers groups in 4 clusters that are connected to 2 switches. Each link has a bandwidth of 1 Gb. Instead of simulating artificial traffic, we used the real traffic from the University 1 dataset and re-mapped the source and destination IP addresses into 16 equal-size buckets corresponding to the 16 servers.

We compare the normalized flow completion time of LC to ECMP. Here, normalized completion time is the completion time of a flow divided by its size where the completion time includes the time to predict flow size, the time to route by LC or ECMP and the travel time to reach the destination. Figures 3 and 4 compare the normalized completion time of elephant and mice flows respectively. The curves show the cumulative distribution of the flows based on their normalized completion time for different combinations of routing technique, flow size predictor and features. The curve for ECMP serves as a baseline. Since this is a simulation and therefore we know the size of each flow, the curve for LC-TFS reports what would be obtained by least cost (LC) routing if we had access to the true flow size (TFS). In this experiment, we use GPR and oBMM with different features to predict the flow size. Comparing the other curves labeled LC-FSP that are based on flow size predictions (FSP) help us understand the impact of inaccurate predictions by Gaussian Process Regression (GPR) when using features in the *header* of the first packet only, the size of the first *3 Packets* only or all the features (*Header&3Packets*). Since GPR yields results close to TFS, the accuracy of GPR is good enough to have a negligible impact on routing. Figure 4 shows that there is not much improvement possible for LC in comparison to ECMP for mice flows since the curve for ECMP is very close to that of LC with TFS. This make sense since mice flows are so short that one would not expect much improvement in the completion time. Note that among the curves for GPR the one based on the size of 3 Packets only performs worse, which suggests that the header features are very important to ensure good results.

Table 1 compares the improvement in the 99th percentile normalized flow completion time achieved by LCP over ECMP for elephant and mice flows respectively. The 99th-percentile of normalized completion

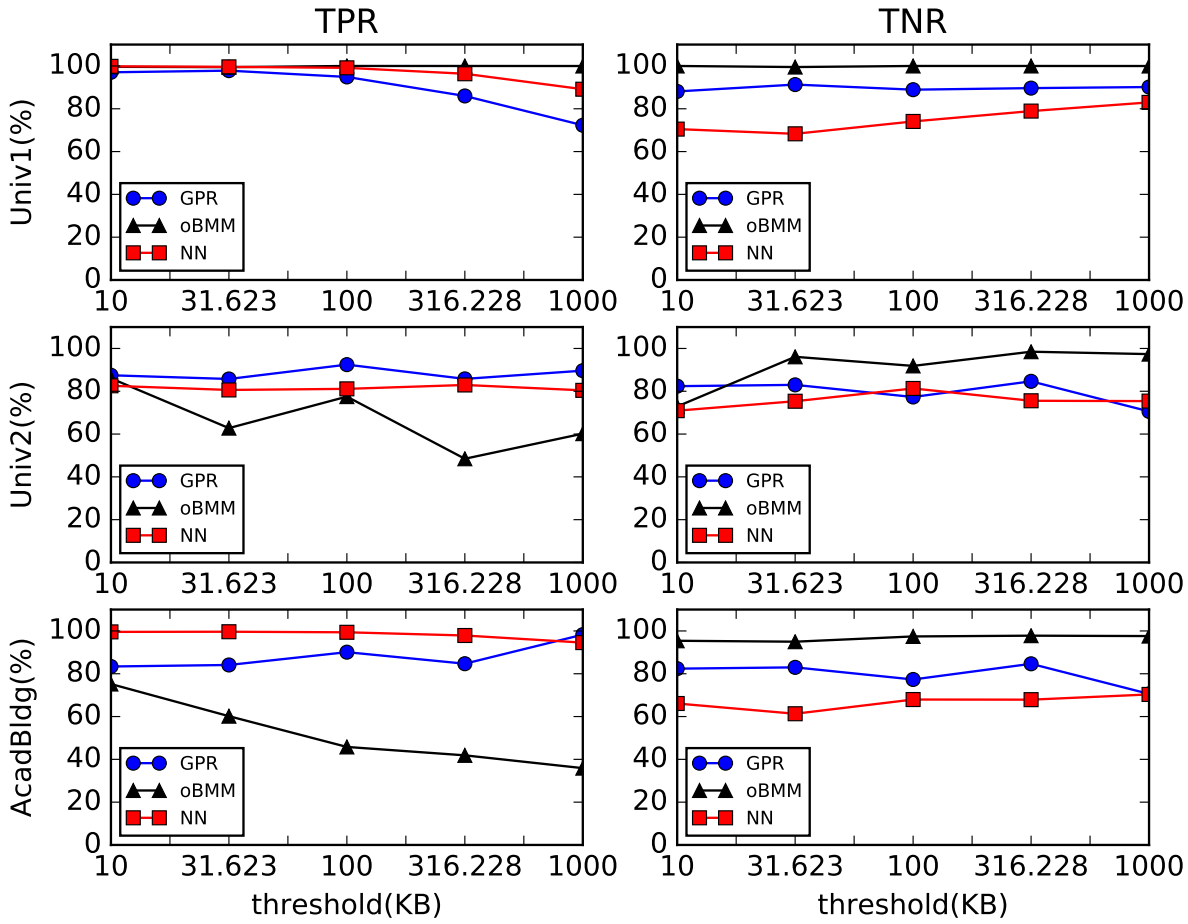

Figure 1: True Positive Rate (TPR) and True Negative Rate (TNR) for online learning

Table 1: 99th Percentile of Normalized Elephant Flow Completion Time Improvement over ECMP

Features	Algo	Elephant	Mice
Header	GPR	6.6%	-1.93%
3Packets	GPR	13.66%	-5.25%
Header&3Packets	GPR	14.17%	0.59%
3Packets	oBMM	13.5%	-1.93%

time indicates the time by which 99% of the flows have completed, which can be thought as an upper bound while treating the remaining 1% as outliers. The results show that GPR with all the features (Header & 3Packets) achieve the best improvements for elephant flows while slightly impacting mice flows.

6. CONCLUSION

In this work, we described how to use several machine learning techniques to estimate flow sizes and detect elephant flows based on information that can be extracted from the first few packets of each flow. This approach does not require any modification to the applications or end hosts and it allows predictions to be made immediately upon the start of each flow. This is particularly useful in scheduling and routing since the ability to predict the size of a flow as it starts allows us to route it in a least congested path,therefore avoiding congestion and mitigating the need for load balancing. We show the use of 6 additional features that can be extracted from the header of the first packet and TCP synchronization, provide useful information to predict the flow size. We also showed the benefits of elephant detection in routing by assigning a least congested path to elephant flows. The resulting routing policy reduced the average flow completion time of elephant flows while keeping the average flow completion time of mice flows roughly the same. Based on those positive results, we plan to implement the flow size predictors and least cost path routing policy in a real network.

7. REFERENCES

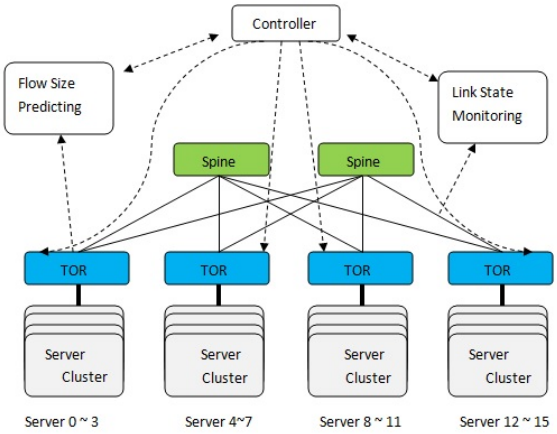

Figure 2: Network topology

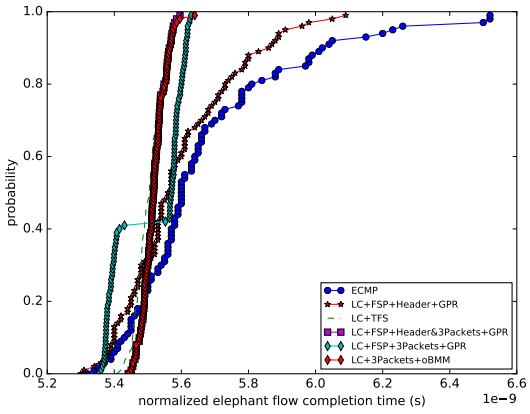

Figure 3: Performance of Elephant flows

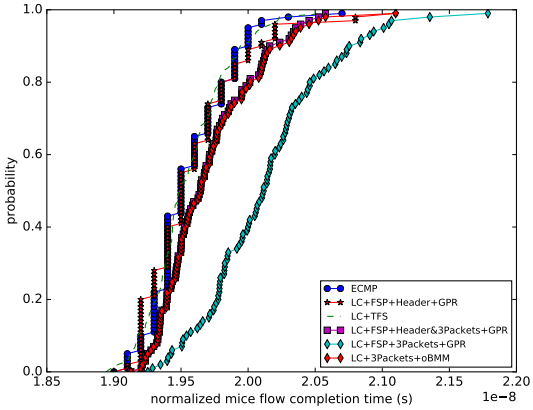

Figure 4: Performance of Mice flows

[1] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI*, volume 10, pages 19–19, 2010.

[2] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280. ACM, 2010.

[3] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[4] A. R. Curtis, W. Kim, and P. Yalagandula. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In *INFOCOM, 2011 Proceedings IEEE*, pages 1629–1637. IEEE, 2011.

[5] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *ACM SIGCOMM Computer Communication Review*, 36(1):59–62, 2006.

[6] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 41(4):339–350, 2011.

[7] C. E. Hopps. Analysis of an equal-cost multi-path algorithm. 2000.

[8] P. Jaini, A. Rashwan, H. Zhao, Y. Liu, E. Banijamali, Z. Chen, and P. Poupart. Online algorithms for sum-product networks with continuous variables. In *The International Conference on Probabilistic Graphical Models*, 2016.

[9] F. Omar. Online bayesian learning in probabilistic graphical models using moment matching with applications. 2016.

[10] Y. Peng, K. Chen, G. Wang, W. Bai, Z. Ma, and L. Gu. Hadoopwatch: A first step towards comprehensive traffic forecasting in cloud computing. In *INFOCOM, 2014 Proceedings IEEE*, pages 19–27. IEEE, 2014.

[11] C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[12] Y. Shao, B. Yang, J. Jiang, Y. Xue, and J. Li. Emilie: Enhance the power of traffic identification. In *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pages 31–35. IEEE, 2014.

[13] A. Silberschatz, P. B. Galvin, G. Gagne, and A. Silberschatz. *Operating system concepts*. 1998.

[14] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan. c-through: Part-time optics in data centers. *ACM SIGCOMM Computer Communication Review*, 41(4):327–338, 2011.