A Machine Learning-based Approach for Estimating Available Bandwidth

Ling-Jyh Chen
Institute of Information Science, Academia Sinica
Cheng-Fu Chou and Bo-Chun Wang
Department of Computer Science and Information Engineering, National Taiwan University

***Abstract*— In this paper, we propose a machine learning-based approach for estimating available bandwidth. We evaluate the approach via simulations using two probing models: a *packet train probing model* and a *pathChirp-like probing model*. The simulation results show that the former cannot yield accurate estimates in our system; however, using the *pathChirp-like probing model*, the proposed approach can estimate the available bandwidth with moderate traffic overhead more accurately than two widely used tools, pathChirp and Spruce. Moreover, we propose a normalization method that improves our approach's ability to estimate available bandwidth, even if there are no samples with similar properties to the measured path in the training dataset. The effectiveness and simplicity of this novel approach make it a promising scheme that goes a long way toward achieving accurate estimation of available bandwidth on Internet paths.**

I. Introduction

Estimating the available bandwidth of an Internet path is a fundamental research problem in computer networking; hence, a number of available bandwidth estimation techniques have been proposed in recent years [2] [4] [5] [6] [9] [10] [11] [14]. Such schemes can be divided into two main categories, *statistical cross-traffic models* and *self-induced congestion models*, according to their basic design principles.

More specifically, the statistical cross-traffic model measures the time interval between the arrival of any two successive probing packets and uses dispersion measurements to estimate the available bandwidth. In contrast, the self-induced congestion model relies on the simple intuition that if the probing rate exceeds the available bandwidth of the path, the receiver will measure a dispersion larger than the initial inter-packet gap. However, the above approaches have two drawbacks: they are either inaccurate when network scenarios are complex (e.g., multi-hopped, wired/wireless mixed, or with multiple bottleneck links) or they behave intrusively (i.e., they input a lot of probes into the network to obtain one estimate). A feasible solution that can accurately estimate the available bandwidth of Internet paths is thus highly desirable.

In this paper, we propose a machine learning-based approach for available bandwidth estimation. Using Support Vector Machine (SVM) [3] as the machine learning tool, we evaluate two probing models (a packet train model and a pathChirp-like model) via simulations. The results show that the pathChirp-like probing model yields more accurate estimation results than the packet train probing model, while the proposed approach outperforms two widely used tools, pathChirp and Spruce, in all test cases. Moreover, we propose a normalization method that enables the proposed approach to estimate the available bandwidth of a path, even if there are no samples with similar properties to the measured path in the training dataset.

The remainder of the paper is organized as follows. In Section II, we discuss related work. In Section III, we present the proposed approach and discuss the two probing models employed. In Section IV, we evaluate the models and compare the performance of the proposed approach with that of other tools in terms of accuracy and overhead. We then present our conclusions in Section V.

II. Related Work

Cprobe [2], the first method to estimate the available bandwidth of a network path in an end-to-end fashion, sends a train of ICMP packets and estimates the available bandwidth by measuring the dispersions at the receiver. The approach has been extended by another tool called Pipechar [6].

Subsequent estimation schemes can be divided into two categories, namely, statistical cross-traffic models and self-induced congestion models. Statistical cross-traffic models measure the time interval between the arrival of any two successive probing packets at the receiver and use the dispersion measurements to estimate the available bandwidth. Several popular tools, such as Delphi [10], IGI [4], and Spruce [14] are based on this concept.

More specifically, Delphi [10] sends a train of probes with constant spacing, and uses the dispersion measurements at the receiver to estimate the amount of cross traffic, and thereby determine the available bandwidth. IGI [4] is similar to Delphi, except that it iteratively probes the network by using different transmission rates to alleviate the self-induced congestion problem. In contrast, Spruce [14] uses a Poisson process to probe the network with a number of packet pairs, and prevents self-induced congestion by enlarging all inter-pair gaps. All these tools make a strong assumption that there is only one bottleneck link along the path, and the tightest link is exactly the same as the bottleneck link. As a result, such methods are likely to fail when networks become as diverse as todays Internet.

This material is based upon work supported by the National Science Council under grant number NSC 94-2218-E-001-002.

Tools based on the self-induced congestion model rely on the simple intuition that if the probing rate is lower than the available bandwidth, the probe packets will not experience additional queueing delay during transmission. On the other hand, if the probing rate exceeds the available bandwidth, the probing packets must be queued at some router along the path such that the overall delivery latency will involve an additional queueing delay. A number of tools fall into this category, such as, TOPP [9], Pathload [5], and pathChirp [11].

TOPP [9] uses packet pairs transmitted at irregular intervals to probe the network, and estimates the available bandwidth based on the measured dispersion at the receiver. Pathload [5], which uses a long CBR packet train to estimate available bandwidth, varies the sending rate to approximate the available bandwidth. Although this approach has proved accurate in almost all test scenarios, its overhead is considered too high in terms of time complexity and the amount of traffic input to the network. Meanwhile, pathChirp [11], which uses exponentially spaced probing chirps, is widely considered to be the most efficient approach, and several refined variants, such as PTR [4], have been proposed.

III. System Setting

A. Network Scenarios

A successful machine learning-based system must be able to collect a sufficiently large amount of training data that is representative of realistic Internet scenarios. However, it is widely recognized that, due to the diversity and dynamics of the Internet, collecting and verifying the correctness of data about specific properties of such a large-scale network is very difficult. Therefore, in this study, we do not collect the training data directly from the Internet; instead, we use the NS-2 simulator to create a representative network using realistic network traces with well-established network traffic models, and we monitor each link of the simulated network while transmitting probing packets between selected node pairs.

More precisely, our network scenario is based on the Tiscali topology of Rocketfuel's trace [13]. This topology has 750 links and 506 nodes, of which 221 are end-users. The propagation delay of each intermediate link is chosen randomly and uniformly in a 10ms to 20ms range, and the buffer size of each link is fixed at 20. Moreover, we assume that fifty percent of the end-users use ADSL, while the remainder use an academic network. The capacity of the links between the core routers is 1,000Mbps. For the academic network links, the capacity is 100Mbps, and for the ADSL links, it is 3Mbps and 1Mbps for the download and upload links respectively.

In addition to the network topology, we generate network traffic based on the measurement results reported in [7][12]. Specifically, [7] monitored the weekday traffic of a campus-wide network, and classified the traffic types (i.e., web, P2P, FTP, streaming, and other applications) by reading the headers of captured packets. In [12], the session duration and the packet size of each type of application were analyzed statistically using multiple network traces. Using these previous findings, we let each node in our system decide whether or not to generate a new flow every second based on a given network utilization factor. When generating a new flow, the node first determines the traffic type (i.e., web, P2P, FTP, or streaming) in accordance with the distribution reported in [7], and then calculates the session duration and the packet size based on the measurement results in [12]. Thus, the generation of network traffic in our simulation is expected to be representative of realistic Internet scenarios.

B. Probing Models

We employ two probing models in our evaluation: the packet train model and the pathChirp-like model. In the former, the sender transmits eleven packets in one burst (i.e., back-to-back) in each round; whereas in the latter, the sender transmits a chirp of fifteen packets with a spread factor of $\gamma = 1.2$ such that the lowest sending rate is five percent of the bottleneck capacity. For each model, we randomly select a pair of nodes (i.e., the sender and the receiver) and start transmitting probing packets, each of which is 1,500 bytes. After a round finishes, we record the available bandwidth (of the tightest link), the path capacity (of the bottleneck link), the hop count along the path, and the dispersion of every pair of contiguous packets in each probe. Hence, there are ten dispersions in the packet train model and fourteen in the pathChirp-like model. If a packet is lost during transmission, we deem the corresponding dispersion to be missing data.

C. Machine Learning Tool

Machine learning techniques can be classified into two types: unsupervised learning and supervised learning. Unsupervised learning techniques, such as the *EM* algorithm and *K-means clustering*, treat input objects as a set of random variables, and perform learning without receiving any feedback from the environment. In contrast, the goal of supervised learning techniques is to find a mapping function based on the training data that can predict the system's output for any input data. Examples of this type of technique are the *K-Nearest Neighbor (k-NN)* algorithm and *Support Vector Machine (SVM)*.

We use SVM, one of the most popular supervised techniques as our machine-learning tool, and run it on the *R* statistical computing platform [1]. The advantages of SVM are that it can handle missing data caused by packet loss and also deal with the regression problem. More precisely, SVM can interpolate/extrapolate the system output based on the properties of the test input, even when the system output does not physically exist in the training dataset. Furthermore, the computation overhead of SVM is affordable, and it has been employed in numerous applications because of its speed and accuracy.

IV. Performance Evaluation

A. Evaluation of the Packet Train Model

In the first set of simulations, we employ SVM as the machine learning tool, and probe the network with the packet train model. Using the network topology and traffic model mentioned earlier, we collect 16,000 samples as the training

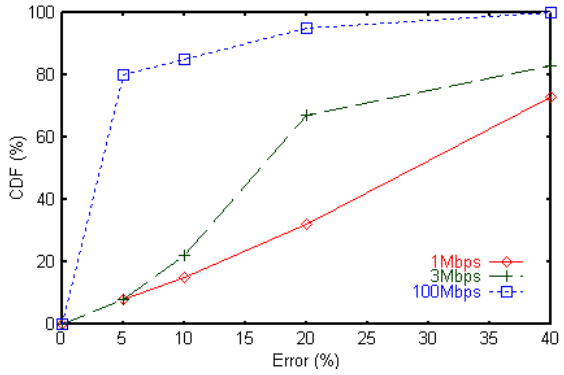

Fig. 1. The accuracy of the proposed machine learning-based approach using the packet train model with 1Mbps, 3Mbps, and 100Mbps bottleneck link capacities.

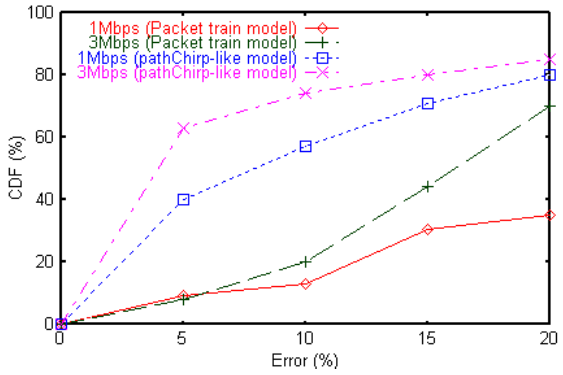

Fig. 2. Comparison of the proposed machine learning-based bandwidth estimation method using the pathChirp-like model and the packet train model (with two bottleneck link capacities: 1Mbps and 3Mbps).

dataset and 1,500 samples as the test data. Each sample is the probing result of a randomly selected node pair (i.e., a sender and a receiver), and each probe consists of eleven back-to-back packets. Specifically, for the training data, each sample is comprised of the ten dispersions, the hop count, the bottleneck capacity, and the tightest link's available bandwidth. For the test data, each sample is comprised of all above information, except the available bandwidth. We divide the results into three groups based on their bottleneck link capacity, and depict the accuracy of the proposed approach in Cumulative Distribution Function (CDF) curves, as shown in Fig. 1.

From Fig. 1, it is evident that the accuracy of the estimation results is only acceptable when the bottleneck link capacity is 100Mbps, and the performance degrades substantially as the bottleneck link capacity becomes narrow. However, from the distribution of the training data of the 100Mbps bottleneck link capacity, we observe that most samples are clustered in a region that has higher values of the real available bandwidth and the estimated available bandwidth. In contrast, the distribution of samples is more scattered when the bottleneck link capacity is 1Mbps and 3Mbps. This observation explains the results that the *packet train model* can only yield accurate available bandwidth estimates when the training data and the test data are clustered in the same region; if the samples are scattered, the accuracy of this model is unacceptable. Unfortunately, due to the dynamics and diversity of the Internet, the samples are deemed to be scattered in reality. Thus, the *packet train probing model* is clearly inappropriate for machine learning-based available bandwidth estimation.

B. Evaluation of the pathChirp-like Model

Here, we evaluate the machine learning-based solution using the pathChirp-like probing model, which probes the network using *chirps* in the same way as pathChirp [11]. In our evaluation, each chirp consists of 15 packets with a spread factor γ equal to 1.2. Similar to the previous experiment, we collect 16,000 samples as the training dataset and 1,500 samples as the test data; each training and test sample is the probing result of a randomly selected source and receiver pair.

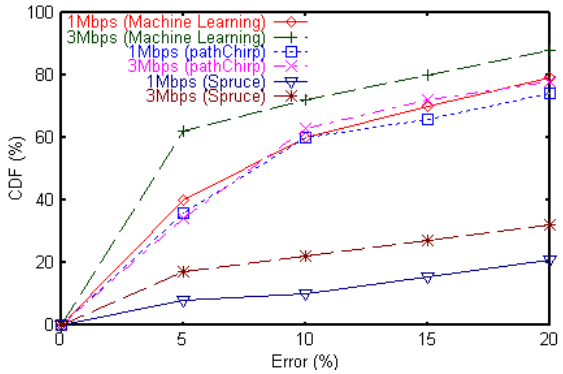

Fig. 3. Comparison of the proposed machine learning-based bandwidth estimation approach (using the pathChirp-like probing model), Spruce, and pathChirp with different bottleneck link capacities (1Mbps and 3Mbps).

We compare the accuracy of the pathChirp-like probing model with the packet train probing model on two bottleneck capacity settings (i.e., 1Mbps and 3Mbps), as shown in Fig. 2.

The results in Fig. 2 clearly show that the pathChirp-like model outperforms the packet train model in both test cases. Moreover, when the bottleneck link is 3Mbps, 75% of the pathChirp-like model's estimation results fall within a $\pm 10\%$ error range, compared to only 25% of the packet train model's estimates. The reason for this phenomenon is that the pathChirp-like model employs different inter-packet gaps in a chirp to represent different sending rates. As a result, the 14 measured dispersions can indicate whether the available bandwidth is larger or smaller than each corresponding sending rate; whereas the 10 measured dispersions of the packet train model do not have this capability. In addition, the packet train model tends to cause self-congestion, which affects the accuracy of the estimation. As the pathChirp-like model yields more accurate available bandwidth estimates, we use it in the proposed machine learning-based approach.

C. Comparison with Other Tools

We compare the accuracy of the proposed machine learning-based approach using the pathChirp-like probing model with two widely used tools, namely pathChirp [11] and Spruce [14].

We run 1,500 tests for both pathChirp and Spruce in the same network scenario as the previous experiments, and divide the results into two groups based on their bottleneck link capacity.

The comparison results, shown in Fig. 3, indicate that both pathChirp and the proposed machine learning-based approach substantially outperform Spruce in all test cases, which confirms the results in [8] that Spruce-like techniques can not provide accurate bandwidth estimation on multihop paths. Moreover, the results show that the proposed method consistently outperforms pathChirp in all cases. The reason is that pathChirp only uses one round of probing to estimate the available bandwidth, while the proposed approach estimates the bandwidth using a database that includes a large number of historical records.

D. Scale-Free Approach

There is a feasibility issue with regard to the proposed approach in that it only collects training data from a very limited network scenario (i.e., the number of nodes in the network is limited, and the bottleneck link capacity of a connection path is either 1Mbps, 3Mbps, or 100Mbps). In contrast, the Internet is not only vast, but also dynamic in terms of its diversity and complexity. The cost of building a new database that covers all types of Internet scenarios would be prohibitively expensive if it were possible. Therefore, we propose a *scale-free* approach to normalize all properties of our system so that we can use our database to estimate the available bandwidth in other network scenarios.

The *normalization* process is implemented as follows. First, for each training data sample, we divide the 14 dispersion measurements by the sample's corresponding initial inter-packet gap, and replace the observation of the real available bandwidth with the utilization of the bottleneck link. Second, for each test data sample, we also divide the dispersion measurements by the sample's corresponding initial inter-packet gap and input the results into the machine learning system. As a result, the output of the SVM model represents the utilization of the bottleneck link, and the available bandwidth estimate is obtained by taking the product of the utilization and the bottleneck link capacity.

Using the pathChirp-like probing model and the same training database, we run two test cases (with 1,500 samples in each case) on the same topology and traffic model with two bottleneck capacities, 6 Mbps, and 50 Mbps, not included in the training database. The evaluation results, shown in Fig. 4, demonstrate that the proposed approach can accurately estimate the available bandwidth. Specifically, over 50% of the estimates are within a 10% error range, which is comparable to the results of pathChirp and the proposed approach in homogeneous network configurations (i.e., Fig. 3).

V. Conclusion and Future Work

We propose a machine learning-based approach for estimating the available bandwidth of a network path. Using SVM as the machine learning tool, we employ two probing models in the proposed system and evaluate their accuracy in estimating available bandwidth via simulations. The results show that the pathChirp-like model outperforms the packet train model in all test cases. Furthermore, by using the pathChirp-like model in conjunction with SVM, the proposed method yields more accurate estimates than two widely used tools, pathChirp and Spruce. By normalizing all attributes in the system, we show that this novel approach is capable of accurately estimating available bandwidth, even if there are no samples with similar properties to the measured path in the training dataset. Work on evaluating the proposed approach on the Internet is ongoing. We will report the results in the near future.

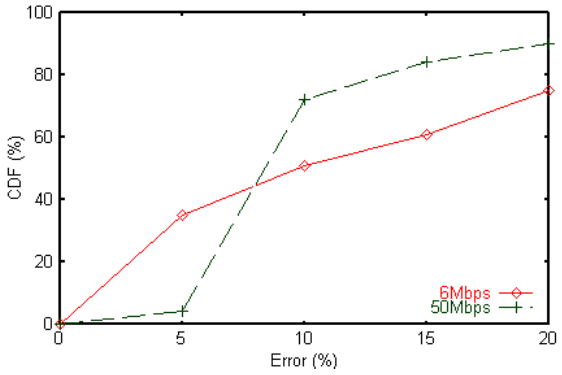

Fig. 4. Simulation results of the proposed approach after normalization. The test samples were collected from network scenarios with 6Mbps and 50 Mbps bottleneck capacities, while the training data was collected from network scenarios of 1, 3, and 100 Mbps bottleneck link capacities.

References

[1] R project. http://www.R-project.org.

[2] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27(8):297–318, Oct. 1996.

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20, 1995.

[4] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. In *IEEE JSAC*, August 2003.

[5] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. In *ACM SIGCOMM*, 2002.

[6] G. Jin, G. Yang, B. Crowley, and D. Agarwal. Network characterization service (ncs). In *IEEE HPDC*, 2001.

[7] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. In *ACM SIGCOMM*, 2005.

[8] L. Lao, C. Dovrolis, and M. Y. Sanadidi. The probe gap model can underestimate the available bandwidth of multihop paths. *ACM SIGCOMM Computer Communication Review*, 36:29–34, October 2006.

[9] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *IEEE Globecom*, 2000.

[10] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, and R. Baraniuk. Multifractal cross-traffic estimation. In *ITC Seminar on IP Traffic Measurement*, 2000.

[11] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cot. pathchirp: Efficient available bandwidth estimation for network paths. In *PAM Workshop*, 2003.

[12] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *ACM IMC*, 2004.

[13] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *ACM SIGCOMM*, 2002.

[14] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *ACM IMC*, 2003.