A Machine Learning Approach to TCP Throughput Prediction

Mariyam Mirza, Joel Sommers, Paul Barford, Xiaojin Zhu

***Abstract*—*TCP throughput prediction* is an important capability for networks where multiple paths exist between data senders and receivers. In this paper we describe a new, lightweight method for TCP throughput prediction. Our predictor uses Support Vector Regression; prediction is based on both prior file transfer history and measurements of simple path properties. We evaluate our predictor in a laboratory setting where ground truth can be measured with perfect accuracy. We report the performance of our predictor for *oracular* and *practical* measurements of path properties over a wide range of traffic conditions and transfer sizes. For bulk transfers in heavy traffic using *oracular* measurements, TCP throughput is predicted within 10% of the actual value 87% of the time, representing nearly a 3-fold improvement in accuracy over prior history-based methods. For *practical* measurements of path properties, predictions can be made within 10% of the actual value nearly 50% of the time, approximately a 60% improvement over history-based methods, and with much lower measurement traffic overhead. We implement our predictor in a tool called *PathPerf*, test it in the wide area, and show that *PathPerf* predicts TCP throughput accurately over diverse wide area paths.**

***Index Terms*—TCP Throughput Prediction, Active Measurements, Machine Learning, Support Vector Regression.**

I. Introduction

The availability of multiple paths between sources and receivers enabled by content distribution, multi-homing, and overlay or virtual networks suggests the need for the ability to select the "best" path for a particular data transfer. A common starting point for this problem is to define "best" in terms of the throughput that can be achieved over a particular path between two end hosts for a given sized TCP transfer. In this case, the fundamental challenge is to develop a technique that provides an accurate TCP throughput forecast for arbitrary and possibly highly dynamic end-to-end paths.

Prior work on the problem of TCP throughput prediction has largely fallen into two categories: those that investigate *formula-based* approaches and those that investigate *history-based* approaches. Formula-based methods, as the name suggests, predict throughput using mathematical expressions that relate a TCP sender's behavior to path and end host properties such as RTT, packet loss rate, and receive window size. In this case, different measurement tools can be used to gather the input data that is then plugged into the formula to generate a prediction. However, well-known network dynamics and limited instrumentation access complicate the basic task of gathering timely and accurate path information, and the ever evolving set of TCP implementations means that a corresponding set of formula-based models must be maintained.

History-based TCP throughput prediction methods are conceptually straightforward. They typically use some kind of standard time series forecasting based on throughput measurements derived from prior file transfers. In recent work, He *et al.* show convincingly that history-based methods are generally more accurate than formula-based methods. However, the authors carefully outline the conditions under which history-based prediction can be effective [11]. Also, history-based approaches described to date remain relatively inaccurate and potentially heavy weight processes focused on bulk transfer throughput prediction.

Our goal is to develop an accurate, lightweight tool for predicting end-to-end TCP throughput for arbitrary file sizes. We investigate the hypothesis that the accuracy of history-based predictors can be improved and their impact on a path reduced by augmenting the predictor with periodic measurements of simple path properties. The questions addressed in this paper include: 1) Which path properties or combination of path properties increase the accuracy of TCP throughput prediction the most? and 2) What is a minimum set of file sizes required to generate history-based throughput predictors for arbitrary file sizes? Additional goals for our TCP throughput prediction tool are: 1) to make it robust to "level shifts" (*i.e.*, when path properties change significantly) which He *et al.* show to be a challenge in history-based predictors, and 2) to include a confidence value with predictions—a metric with little treatment in prior history-based throughput predictors.

We use Support Vector Regression (SVR), a powerful machine learning technique that has shown good empirical performance in many domains, for throughput prediction. SVR has several properties that make it well suited for our study: 1) It can accept multiple inputs (*i.e.*, multivariate features) and will use all of these to generate the throughput prediction. 2) SVR does not commit to any particular parametric form, unlike formula-based approaches. Instead, SVR models are flexible based on their use of so-called non-linear kernels. This expressive power is the reason why SVR has potential to be more accurate than prior methods. 3) SVR is computationally efficient, which makes it attractive for inclusion in a tool that can be deployed and used in the wide area. For our application, we extend the basic SVR predictor with a confidence interval estimator based on the assumption that prediction errors

An earlier version of this paper appeared in SIGMETRICS 2007.

Mariyam Mirza (mirza@cs.wisc.edu), Paul Barford (pb@cs.wisc.edu) and Xiaojin Zhu (jerryzhu@cs.wisc.edu) are with the Department of Computer Sciences, University of Wisconsin-Madison. Joel Sommers (jsommers@colgate.edu) is with the Department of Computer Science, Colgate University. This work was supported in part by NSF grants numbers CNS-0347252, CNS-0646256, CNS-0627102, and CCR-0325653. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

are normally distributed, an assumption that we test in our laboratory experiments. Estimation of confidence intervals is critical for on-line prediction, since retraining can be triggered if measured throughput falls outside a confidence interval computed through previous measurements.

We begin by using laboratory-based experiments to investigate the relationship between TCP throughput and measurements of path properties including available bandwidth (*AB*), queuing delays (*Q*), and packet loss (*L*). The lab environment enables us to gather highly accurate passive measurements of throughput and all path properties, and develop and test our SVR-based predictor over a range of realistic traffic conditions. Our initial experiments focus on bulk transfers and compare actual throughput of TCP flows with predictions generated by our SVR-based tool. Our results show that throughput predictions can be improved by as much as a factor of 3 when including path properties in the SVR-based tool versus a history-based predictor. For example, our results show that the SVR-based predictions are within 10% of actual 87% of the time for bulk transfers under heavy traffic conditions (90% average utilization on the bottleneck link). Interestingly, we find that the path properties that provide the most improvement to the SVR-based predictor are *Q* and *L* respectively, and that including *AB* provides almost no improvement to the predictor.

Next, we expand the core SVR-based tool in three ways. First, the initial tests were based entirely on passive traffic measurements, which are unavailable in the wide area, so we tested our SVR-based approach using measurements of *Q* and *L* provided by the BADABING tool [25]. The reduction in accuracy of active versus passive measurements resulted in a corresponding reduction in accuracy of SVR-based throughput predictions for bulk transfers under heavy traffic conditions on the order of about 35%—still a significant improvement on history-based estimates. Also, throughput prediction based on training plus lightweight active measurements results in a dramatically lower network probe load than prior history-based methods using long-lived TCP transfers and heavyweight probe-based estimates of available bandwidth such as described in [11]. We quantify this difference in Section VII. Second, unlike prior work which focuses only on bulk transfers, we enabled SVR to predict throughput accurately for a wide range of file sizes. We found that a training set of only three file sizes results in accurate throughput predictions for a wide range of file sizes. Third, He *et al.* showed that "level shifts" in path conditions pose difficulties for throughput prediction [11], suggesting the need for adaptivity. To accomplish this, we augmented the basic SVR predictor with a confidence interval estimator as a mechanism for triggering a retraining process. We show in Section VI-C1 that our technique is able to adapt to level shifts quickly and to maintain high accuracy on paths where level shifts occur. We implement these capabilities in a tool called *PathPerf* [1] that we tested in the wide area. We present results from 18 diverse wide area paths in the RON testbed [3], and show that *PathPerf* predicts throughput accurately under a broad range of conditions in real networks.

[1]PathPerf will be openly available for download at http://wail.cs.wisc.edu/waildownload.py

II. RELATED WORK

Since seminal work by Jacobson and Karels established the basic mechanisms for modern TCP implementations [13], it has been well known that many factors affect TCP throughput, such as the TCP implementation, the underlying network structure, and the dynamics of the traffic sharing the links on the path between two hosts. A number of studies have taken steps toward understanding TCP behavior, including [1], [2] which developed stochastic models for TCP based on packet loss characteristics. A series of studies develop increasingly detailed mathematical expressions for TCP throughput based on modeling the details of the TCP congestion control algorithm and measurements of path properties [4], [8], [10], [17], [18]. While our predictor also relies on measurement of path properties, the SVR-based approach is completely distinguished from prior formula-based models.

A large number of empirical studies of TCP file transfer and throughput behavior have provided valuable insight into TCP performance. Paxson conducted one of the most comprehensive studies of TCP behavior [19], [20]. While that work exposed a plethora of issues, it provided some of the first empirical data on the characteristics of packet delay, queuing, and loss within TCP file transfers. Barford and Crovella's application of critical path analysis to TCP file transfers provides an even more detailed perspective on how delay, queuing, and loss relate to TCP performance [6]. In [5], Balakrishnan *et al.* studied throughput from the perspective of a large web server and showed how it varied depending on end-host and time of day characteristics. Finally, detailed studies of throughput variability over time and correlations between throughput and flow size can be found in [31], [32], respectively. These studies inform our work in terms of the basic characteristics of throughput that must be considered when building our predictor.

Past studies of history-based methods for TCP throughput prediction are based on standard time series forecasting methods. Vazhkudia *et al.* compare several different simple forecasting methods to estimate TCP throughput for transfers of large files and find similar performance across predictors [30]. A well-known system for throughput prediction is the Network Weather Service [28]. That system makes bulk transfer forecasts by attempting to correlate measurements of prior large TCP file transfers with periodic small (64KB) TCP file transfers (referred to as "bandwidth probes"). The DualPats system for TCP throughput prediction is described in [16]. That system makes throughput estimates based on an exponentially weighted moving average of larger size bandwidth probes (1.2MB total). Similar to our work, Lu *et al.* found that prediction errors generally followed a normal distribution. As mentioned earlier, He *et al.* extensively studied history-based predictors using three different time series forecasts [11]. Our SVR-based method includes information from prior transfers for training, but otherwise only requires measurements from lightweight probes and is generalized for all files sizes, not just bulk transfers.

Many techniques have been developed to measure path properties (see CAIDA's excellent summary page for exam-

ples [9]). Prior work on path property measurement directs our selection of lightweight probe tools to collect data for our predictor. Recent studies have focused on measuring available bandwidth on a path. AB is defined informally as the minimum unused capacity on an end-to-end path, which is a conceptually appealing property with respect to throughput prediction. A number of studies have described techniques for measuring AB including [14], [26], [27]. We investigate the ability of AB measurement as well as other path properties to enhance TCP throughput predictions.

Finally, machine learning techniques have not been widely applied to network measurement. One notable exception is in network intrusion detection (*e.g.,* [12]). The only other application of Support Vector Regression that we know of is to the problem of using IP address structure to predict round trip time latency [7].

III. A Multivariate Machine Learning Tool

The main hypothesis of this work is that history-based TCP throughput prediction can be improved by incorporating measurements of end-to-end path properties. The task of throughput prediction can be formulated as a regression problem, *i.e.,* predicting a real-valued number based on multiple real-valued input features. Each file transfer is represented by a feature vector $\mathbf{x} \in \mathbb{R}^d$ of dimension d. Each dimension is an observed feature, *e.g.,* the file size, proximal measurements of path properties such as queuing delay, loss, available bandwidth, etc. Given $\mathbf{x}$, we want to predict the throughput $y \in \mathbb{R}$. This is achieved by training a regression function $f : \mathbb{R}^d \mapsto \mathbb{R}$, and applying f to $\mathbf{x}$. The function f is trained using training data, *i.e.,* historical file transfers with known features and the corresponding measured throughput.

The analytical framework that we apply to this problem is *Support Vector Regression (SVR)*, a state-of-the-art machine learning tool for multivariate regression. SVR is the regression version of the popular Support Vector Machines [29]. It has a solid theoretical foundation, and is favored in practice for its good empirical performance. We briefly describe SVR below, and refer readers to [21], [23] for details, and to [15] as an example of an SVR software package.

To understand SVR we start from a linear regression function $f(\mathbf{x}) = \beta^\top \mathbf{x} + \beta_0$. Assume we have a training set of n file transfers $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. Training involves estimating the d-dimensional weight vector β and offset β_0 so that $f(\mathbf{x}_i)$ is close to the truth y_i for all training examples $i = 1 \ldots n$. There are many ways to measure "closeness". The traditional measure used in SVR is the ε-insensitive loss, defined as

$$L(f(\mathbf{x}), y) = \begin{cases} 0 & \text{if } |f(\mathbf{x}) - y| \le \varepsilon \\ |f(\mathbf{x}) - y| - \varepsilon & \text{otherwise.} \end{cases} \quad (1)$$

This loss function measures the absolute error between prediction and truth, but with a tolerance of ε. The value ε is application-dependent in general, and in our experiments we set it to zero. Other loss functions (*e.g.,* the squared loss) are possible too, and often give similar performance. They are not explored in this paper.

It might seem that the appropriate way to estimate the parameters β, β_0 is to minimize the overall loss on the training set $\sum_{i=1}^n L(f(\mathbf{x}_i), y_i)$. However if d is large compared to the number of training examples n, one can often fit the training data perfectly. This is dangerous, because the truth y in training data actually contain random fluctuations, and f is partly fitting the noise. Such f will generalize poorly, *i.e.* causing bad predictions on future test data. This phenomenon is known as *overfitting*. To prevent overfitting, one can reduce the degree of freedom in f by selecting a subset of features, thus reducing d. An implicit but more convenient alternative is to require f to be *smooth*[2], defined as having a small parameter norm $\|\beta\|^2$. Combining loss and smoothness, we estimate the parameters β, β_0 by solving the following optimization problem

$$\min_{\beta, \beta_0} C \sum_{i=1}^n L(f(\mathbf{x}_i), y_i) + \|\beta\|^2, \quad (2)$$

where C is a weight parameter to balance the two terms. The value of C is usually selected by a procedure called cross-validation, where the training set is randomly split into two parts, then regression functions with different C are trained on one part and their performance measured on the other part, and finally one selects the C value with the best performance. In our experiments we used $C = 3.162$ using cross-validation. The optimization problem can be solved using a quadratic program.

Nonetheless, a linear function $f(\mathbf{x})$ is fairly restrictive and may not be able to describe the true function y. A standard mathematical trick is to augment the feature vector $\mathbf{x}$ with non-linear bases derived from $\mathbf{x}$. For example, if $\mathbf{x} = (x_1, x_2)^\top$, one can augment it with $\phi(\mathbf{x}) = (x_1, x_1^2, x_1 x_2, x_2, x_2^2)$. The *linear* regressor in the augmented feature space $f(\mathbf{x}) = \beta^\top \phi(\mathbf{x}) + \beta_0$ then produces a *non-linear* fit in the original feature space. Note β has more dimensions than before. The more dimensions $\phi(\mathbf{x})$ has, the more expressive f becomes.

In the extreme (and often beneficial) case $\phi(\mathbf{x})$ can even have infinite dimensions. It seems computationally impossible to estimate the corresponding infinite-dimensional parameter β. However, if we convert the *primal* optimization problem 2 into its *dual* form, one can show that the number of dual parameters is actually n instead of the dimension of $\phi(\mathbf{x})$. Furthermore, the dual problem never uses the augmented feature $\phi(\mathbf{x})$ explicitly. It only uses the inner product between pairs of augmented features $\phi(\mathbf{x})^\top \phi(\mathbf{x}') \equiv K(\mathbf{x}, \mathbf{x}')$. The function K is known as the *kernel*, and can be computed from the original feature vectors $\mathbf{x}, \mathbf{x}'$. For instance, the Radial Basis Function (RBF) kernel $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$ implicitly corresponds to an infinite dimensional feature space. In our experiments we used a RBF kernel with $\gamma = 0.3162$, again selected by cross-validation. The dual problem can still be efficiently solved using a quadratic program.

SVR therefore works as follows: For training, one collects a training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, and specifies a kernel K. SVR solves the dual optimization problem, which equivalently finds the potentially very high-dimensional parameter β and

[2] If β are the coefficients of a polynomial function, the function will tend to be smooth (changes slowly) if $\|\beta\|^2$ is small, or noisy if $\|\beta\|^2$ is large.

β_0 in the augmented feature space defined by K. This produces a potentially highly non-linear prediction function $f(\mathbf{x})$. The function $f(\mathbf{x})$ can then be applied to arbitrary test cases $\mathbf{x}$, and produces a prediction. In our case, test cases are the file size for which a prediction is to be made and current path properties based on active measurements.

IV. Experimental Environment and Methodology

This section describes the laboratory environment and experiments that we used to evaluate our throughput predictor.

A. Experimental Environment

The laboratory testbed used in our experiments is shown in Figure 1. It consisted of commodity end hosts connected to a dumbbell-like topology of Cisco GSR 12000 routers. Both measurement and background traffic was generated and received by the end hosts. Traffic flowed from the sending hosts on separate paths via Gigabit Ethernet to separate Cisco GSRs (hop B in the figure) where it was forwarded on OC12 (622 Mb/s) links. This configuration was created in order to accommodate a precision passive measurement system, as we describe below. Traffic from the OC12 links was then multiplexed onto a single OC3 (155 Mb/s) link (hop C in the figure) which formed the bottleneck where congestion took place. We used an AdTech SX-14 hardware-based propagation delay emulator on the OC3 link to add 25 milliseconds delay in each direction for all experiments, and configured the bottleneck queue to hold approximately 50 milliseconds of packets. Packets exited the OC3 link via another Cisco GSR 12000 (hop D in the figure) and passed to receiving hosts via Gigabit Ethernet.

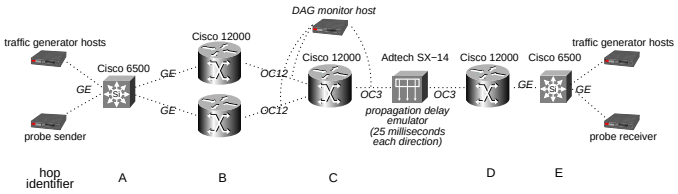


Fig. 1. Laboratory testbed. Cross traffic flowed across one of two routers at hop B, while probe traffic flowed through the other. Optical splitters connected Endace DAG 3.5 and 3.8 passive packet capture cards to the testbed between hops B and C, and hops C and D. Measurement traffic (file transfers, loss probes, and available bandwidth probes) flowed from left to right. Congestion in the testbed occurred at hop C.

The measurement hosts and traffic generation hosts were identically configured workstations running FreeBSD 5.4. The workstations had 2 GHz Intel Pentium 4 processors with 2 GB of RAM and Intel Pro/1000 network cards. They were also dual-homed, so that all management traffic was on a separate network than depicted in Figure 1. We disabled the TCP throughput history caching feature in FreeBSD 5.4, controlled by the variable net.inet.tcp.inflight.enable, to allow TCP throughput to be determined by current path properties rather than throughput history.

A key aspect of our testbed was the measurement system used to establish the true path properties for our evaluation. Optical splitters were attached to both the ingress and egress links at hop C and Endace DAG 3.5 and 3.8 passive monitoring cards were used to capture traces of *all* packets entering and leaving the bottleneck node. By comparing packet headers, we were able to identify which packets were lost at the congested output queue during experiments, and accurately measure available bandwidth on the congested link. Furthermore, the fact that the measurements of packets entering and leaving hop C were synchronized at a very fine granularity (*i.e.*, a single microsecond) enabled us to precisely measure queuing delays through the congested router.

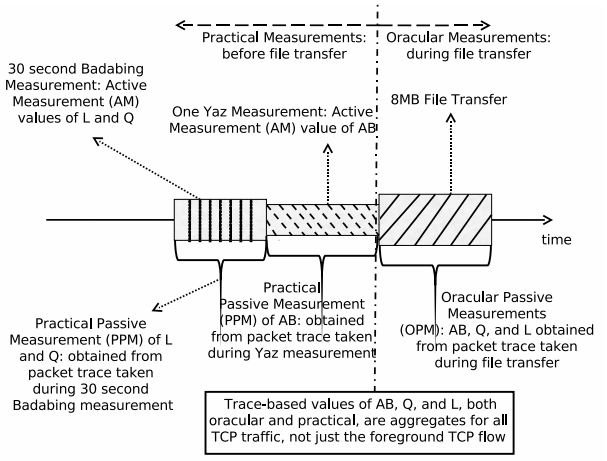


Fig. 2. Measurement traffic protocol, and oracular and practical path measurements used for SVR-based throughput prediction.

B. Experimental Protocol

We generated background traffic by running the Harpoon IP traffic generator [24] between up to four pairs of traffic generation hosts as illustrated in Figure 1. Harpoon produced open-loop self-similar traffic using a heavy-tailed file size distribution, mimicking a mix of application traffic such as web and peer-to-peer applications common in today's Internet. Harpoon was configured to produce average offered loads ranging from approximately 60% to 105% on the bottleneck link (the OC3 between hops C and D).

As illustrated in Figure 2, measurement traffic in the testbed consisted of file transfers and active measurements of AB, L, and Q. We also measured all three metrics passively using packet traces. All measurements are aggregates for all flows on the path. Measurements were separated by 30 second intervals. For the measurement traffic hosts, we increased the TCP receive window size to 128 KB (with the window scaling option turned on). In receive window limited transfers, file transfer throughput was approximately 21 Mb/s. That is, if the available bandwidth on the bottleneck link was 21 Mb/s or more, the flow was receive window (*rwnd*) limited, otherwise it was congestion window (*cwnd*) limited. We increased the receive window size because we wanted file transfers to be *cwnd* limited so we could evaluate our predictor thoroughly; *rwnd*-limited transfers have constant throughput for a given *rwnd* size, so any reasonable predictor performs well on *rwnd*-limited transfers. For this reason, we do not report any results for *rwnd*-limited flows.

Series of measurements were collected and then split up into mutually exclusive training and test sets for SVR. Notions of separate training and test sets are not required for history-based methods; rather, predictions are made over the continuous notion of distant and recent history. In our evaluation of history-based methods, we use the final prediction of the training set as the starting point for the test set.

Figure 2 shows that we gather three types of measurements for each experiment, *Oracular Passive Measurements (OPM)*, *Practical Passive Measurements (PPM)*, and *Active Measurements (AM)*. Oracular measurements, taken during the file transfer, give us perfect information about network conditions, so we can establish the best possible accuracy of our prediction mechanism. In practice, this *oracular* information is not available for making predictions. *AM*, taken before the file transfer, are available in practice and can thus be used for prediction. *PPM*, passive measurements taken at the same time *AM* are taken, show the best possible accuracy of our prediction mechanism with practical measurements, or show how much better SVR would perform if the active measurements had perfect accuracy.

AB is defined as the spare capacity on the end-to-end path. YAZ estimates available bandwidth using a relatively low-overhead, iterative method similar to PATHLOAD [14]. The time required to produce a single AB estimate can vary from a few seconds to tens of seconds. The passive measurement value of AB is computed by subtracting the observed traffic rate on the link from the realizable OC3 link bandwidth.

We use BADABING for active measurements of L and Q. BADABING requires the sender and receiver to be time-synchronized. To accommodate our wide area experiments, the BADABING receiver was modified to reflect probes back to the sender, where they were timestamped and logged as on the original receiver. Thus, the sender clock was used for all probe timestamps. BADABING reports loss in terms of *loss episode frequency* and *loss episode duration*, defined in [25]. Although we refer to *loss frequency* and *loss duration* together as L or loss for expositional ease, these two metrics are two different features for SVR. Q is defined as the difference between the RTT of the current BADABING probe and the minimum probe RTT seen on the path. We set the BADABING probe probability parameter p to 0.3. Other parameters were set according to [25]. Passive values of Q and L are computed using the same definitions as above, except that they are applied to all TCP traffic on the bottleneck link instead of BADABING probes.

For experiments in the wide area, we created a tool, *Path-Perf*. This tool, designed to run between a pair of end hosts, initiates TCP file transfers and path property measurements (using our modified version of BADABING), and produces throughput estimates using our SVR-based method. It can be configured to generate arbitrary file size transfers for both training and testing and initiates retraining when level shifts are identified as described in Section VII.

C. Evaluating Prediction Accuracy

We denote the actual throughput by R and the predicted throughput by $\hat{R}$. We use the metric *relative prediction error* E introduced in [11] to evaluate the accuracy of an individual throughput prediction. *Relative prediction error* is defined as $E = \frac{\hat{R}-R}{min(\hat{R},R)}$ In what follows, we use the distribution of the absolute value of E to compare different prediction methods.

V. BUILDING A ROBUST PREDICTOR

This section describes how we developed, calibrated, and evaluated our prediction mechanism through an extensive set of tests conducted in our lab test-bed.

A. Calibration and Evaluation in the High Traffic Scenario

The first step in developing our SVR-based throughput predictor is to find the combination of training features which lead to the most accurate predictions over a wide variety of path conditions. We trained the predictor using a feature vector for each test that contained different combination of our set of target path measurements (AB, Q, L) and the measured throughput. The trained SVR model is then used to predict throughput for a feature vector containing the corresponding sets of network measurements, and we compare the prediction accuracy for the different combinations.

We also compare the accuracy of SVR to the exponentially weighted moving average (EWMA) History-Based Predictor (HB) described in [11], $\hat{R}_{i+1} = \alpha R_i + (1-\alpha)\hat{R}_i$. We use an α value of 0.3 because it is used in [11].

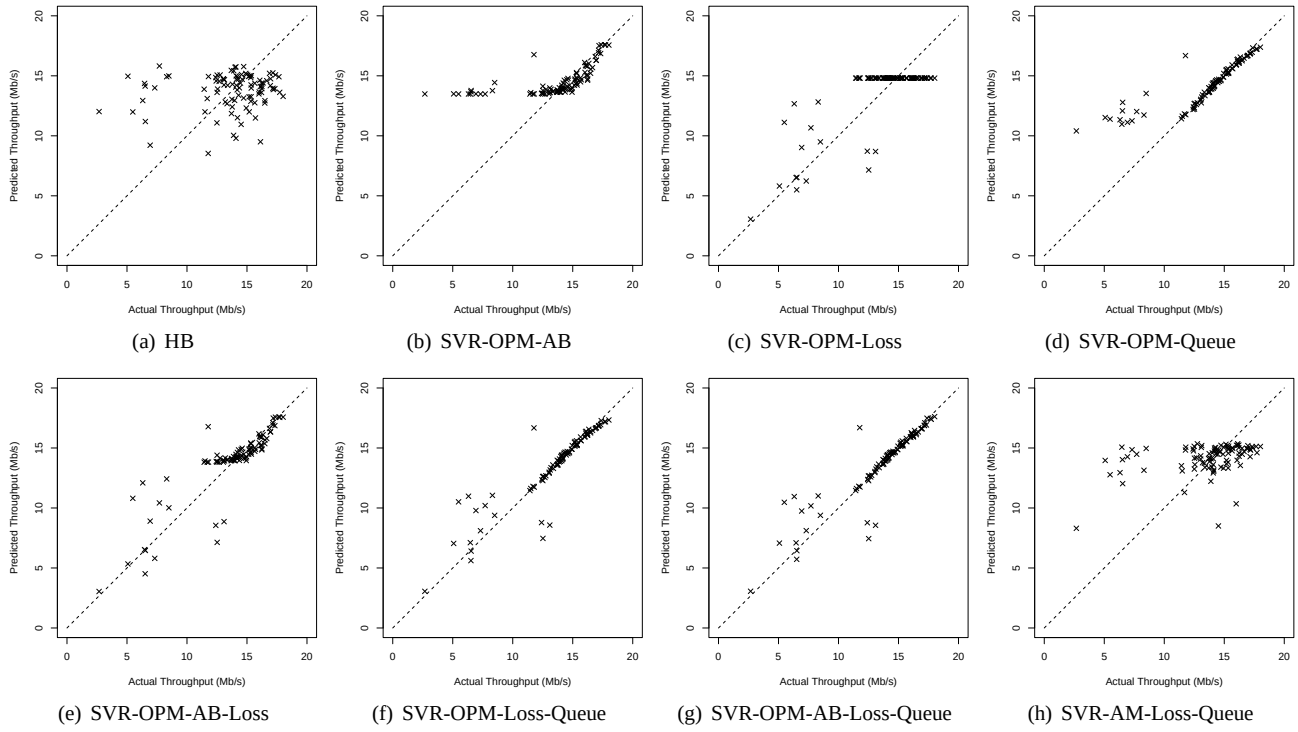
For most of the results reported, we generated an average of 140 Mb/s of background traffic to create high utilization on our OC3 (155 Mb/s) bottleneck link. We used one set of 100 experiments for training and another set of 100 experiments for testing. An 8 MB file was transferred in each experiment.

Figures 3(a) to 3(h) show scatter plots comparing the actual and predicted throughput using different prediction methods as discussed below. A point on the diagonal represents perfect prediction accuracy; the farther a point is from the diagonal, the greater the prediction error.

1) Using Path Measurements from an Oracle: Figure 3(a) shows the prediction accuracy scatter plot for the HB method. Figures 3(b) to 3(g) show the prediction error with SVR using *Oracular Passive Measurements (OPM)* for different combinations of path measurements in the feature vector. For example, *SVR-OPM-Queue* means that only queuing delay measurements were used to train and test, while *SVR-OPM-Loss-Queue* means that both loss and queuing delay measurements were used to train and test.

Table I shows relative prediction errors for HB forecasting and for *SVM-OPM*-based predictions. Values in the table indicate the fraction of predictions for a given method within a given accuracy level. For example, the first two columns of the first row in Table I mean that 32% of HB predictions have relative prediction errors of 10% or smaller while 79% of *SVR-OPM-AB* predictions have relative prediction errors of 10% or smaller. We present scatter plots in addition to tabular data to provide insight into how different path properties contribute to throughput prediction in the SVR method.

From Figure 3(a), we can see that the predictions for the HB predictor are rather diffusely scattered around the diagonal, and that predictions in low-throughput conditions tend to have



(a) HB (b) SVR-OPM-AB (c) SVR-OPM-Loss (d) SVR-OPM-Queue

(e) SVR-OPM-AB-Loss (f) SVR-OPM-Loss-Queue (g) SVR-OPM-AB-Loss-Queue (h) SVR-AM-Loss-Queue

Fig. 3. Comparison of Prediction Accuracy of HB, SVR-OPM, and SVR-AM in High Background Traffic Conditions.

large relative error. Figures 3(b), 3(c), and 3(d) show the behavior of the SVR predictor using a single measure of path properties in the feature vector—AB, L, and Q respectively. The AB and Q graphs have a similar overall trend: predictions are accurate (*i.e.*, points are close to the diagonal) for high actual throughput values, but far from the diagonal, almost in a horizontal line, for lower values of actual throughput. The L graph has the opposite trend: points are close to the diagonal for lower values of actual throughput, and form a horizontal line for higher values of actual throughput.

The explanation for these trends lies in the fact that file transfers with low actual throughput experience loss, while file transfers with high actual throughput do not experience any loss. When loss occurs, the values of AB and Q for the path are nearly constant. AB is almost zero, and Q is the maximum possible value (which depends on the amount of buffering available at the bottleneck link in the path). In this case, throughput depends on the value of L. Hence, L appears to be a good predictor when there is loss on the path, and AB and Q, being constants in this case, have no predictive power, resulting in horizontal lines, *i.e.*, a single value of predicted throughput. On the other hand, when there is no loss, L is a constant with value zero, so L has no predictive power, while AB and Q are able to predict throughput quite accurately.

Figure 3(e) and 3(f) show improvements in prediction accuracy obtained by using more than one path property in the SVR feature vector. We can see that when L is combined with AB or Q, the horizontal lines on the graphs are replaced by points much closer to the diagonal. Combining AB or Q with L allows SVR to predict accurately in both lossy and lossless conditions. Since both AB and Q help predict throughput in lossless conditions, do we really need both AB and Q, or can we use just one of the two and still achieve the same prediction accuracy? To answer this question, we compared AB-Loss and Loss-Queue predictions with each other and with AB-Loss-Queue predictions (*i.e.*, Figures 3(e), 3(f), and 3(g)). The general trend in all three cases is the same: the horizontal line of points is reduced or eliminated, suggesting that prediction from non-constant-value measurements is occurring for both lossy and lossless network conditions. If we compare the AB-Loss and Loss-Queue graphs more closely, we observe two things. First, in the lossless prediction case, the points are closer to the diagonal in the Loss-Queue case than in the AB-Loss case. Second, in the Loss-Queue case, the transition in the prediction from the lossless to the lossy case is smooth, *i.e.*, there is no horizontal line of points, while in the AB-Loss case there is still a horizontal line of points in the actual throughput range of 11–14 Mb/s. This suggests that Q is a more accurate predictor than AB in the lossless case. The relative prediction error data of Table I supports this: SVR with a feature vector containing Loss-Queue information predicts throughput within 10% of actual for 87% of transfers, while a feature vector containing AB-Loss measurements predicts with the same accuracy level for 78% of transfers. Finally, there is no difference in accuracy (either qualitatively or quantitatively) between Loss-Queue and AB-Loss-Queue.

The above discussion suggests that AB measurements are not required for highly accurate throughput prediction, and that a combination of L and Q is sufficient. This observation is not only surprising, but rather good news. Prior work has shown that accurate measurements of AB require at least moderate amounts of probe traffic [22], [26], and some formula-based

TABLE I
RELATIVE ACCURACY OF HISTORY-BASED (*HB*) THROUGHPUT PREDICTION AND SVR-BASED PREDICTORS USING DIFFERENT TYPES OF ORACULAR PASSIVE PATH MEASUREMENTS (*SVR-OPM*) IN THE FEATURE VECTOR. TABLE VALUES INDICATE THE FRACTION OF PREDICTIONS WITHIN A GIVEN ACCURACY LEVEL.

Relative Error	*HB*	*AB*	*L*	*Q*	*AB-L*	*AB-Q*	*L-Q*	*AB-L-Q*
10%	0.32	0.79	0.54	0.87	0.78	0.87	0.86	0.86
20%	0.67	0.87	0.86	0.87	0.87	0.87	0.90	0.90
30%	0.80	0.87	0.92	0.87	0.91	0.87	0.90	0.90
40%	0.87	0.87	0.94	0.87	0.92	0.87	0.93	0.93
50%	0.88	0.88	0.95	0.89	0.97	0.89	0.96	0.96
60%	0.88	0.88	0.97	0.92	0.97	0.92	0.97	0.97
70%	0.89	0.89	0.97	0.94	0.97	0.94	0.98	0.98
80%	0.92	0.91	0.98	0.95	0.98	0.95	0.99	0.99
90%	0.92	0.92	0.98	0.96	0.98	0.96	0.99	0.99

TABLE II
RELATIVE ACCURACY OF HISTORY-BASED (*HB*) THROUGHPUT PREDICTION AND SVR-BASED PREDICTORS USING TRACE-BASED PASSIVE PATH MEASUREMENTS (*PPM*) OR ACTIVE PATH MEASUREMENTS (*AM*). TABLE VALUES INDICATE THE FRACTION OF PREDICTIONS WITHIN A GIVEN ACCURACY LEVEL.

Relative Error	*HB*	*PPM*		*AM*	
		AB-L-Q	*L-Q*	*AB-L-Q*	*L-Q*
10%	0.32	0.49	0.53	0.49	0.51
20%	0.67	0.77	0.81	0.78	0.76
30%	0.80	0.86	0.86	0.86	0.86
40%	0.87	0.86	0.89	0.86	0.86
50%	0.88	0.88	0.89	0.86	0.87
60%	0.88	0.90	0.89	0.88	0.87
70%	0.89	0.90	0.91	0.88	0.88
80%	0.92	0.91	0.94	0.90	0.90
90%	0.92	0.92	0.95	0.92	0.92

TCP throughput estimation schemes take as a given that AB measurements are necessary for accurate throughput prediction [11]. In contrast, measurements of L and Q can be very lightweight probe processes [25]. We discuss measurement overhead further in Section VII.

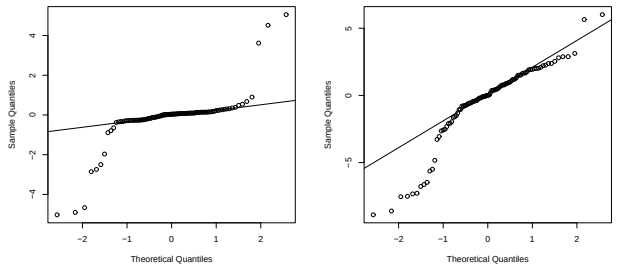
2) Using Practical Passive and Active Path Measurements: So far, we have considered the prediction accuracy of SVR based on only *Oracular Passive Measurements (OPM)*. This gives us the best-case accuracy achievable with SVR, and also provides insight into how SVR uses different path properties for prediction. Table I shows that HB predicts 32% of transfers within 10% of actual while *SVR-OPM-Loss-Queue* predicts 87%, an almost 3-fold improvement. In practice, however, perfect measurements of path properties are not available, so in what follows we assess SVR using measurements that can be obtained in practice in the wide area.

Table II presents relative prediction error data for HB, *SVR-PPM* and *SVR-AM*. Due to space limitations, we present only *Loss-Queue* and *AB-Loss-Queue* results for SVR. We choose these because we expect *AB-Loss-Queue* to have the highest accuracy as it has the most information about path properties, and *Loss-Queue* because it is very lightweight and has accuracy equal to *AB-Loss-Queue* for *SVR-OPM*. We wish to examine three issues: first, whether our finding that *Loss-Queue* has the same prediction accuracy as *AB-Loss-Queue* from the *SVR-OPM* case holds for the *SVR-PPM* and *SVR-AM* case; second, whether *SVR-PPM* and *SVR-AM* have the same accuracy; and third, how *SVR-AM* accuracy compares with HB prediction accuracy.

All prediction accuracy results in the *SVR-PPM* and *SVR-AM* columns in Table II are very similar. *AB-Loss-Queue* has approximately the same accuracy as *Loss-Queue* for both *SVR-PPM* and *SVR-AM*. This is encouraging because it is consistent with the observation from *SVR-OPM*, *i.e.*, that we can achieve good prediction accuracy without having to measure AB. *SVR-AM* has accuracy similar to *SVR-PPM*, *i.e.*, using active measurement tools to estimate path properties yields predictions almost as accurate as having ground-truth passive measurements. This is important because in real wide-area paths instrumentation is generally not available for collecting accurate passive measurements.

Finally, we compare HB with *SVR-AM*. Although Figures 3(a) and 3(h) are qualitatively similar, *SVR-AM* has a tighter cluster of points around the diagonal for high actual throughput than HB. Thus, *SVR-AM* appears to have higher accuracy than HB. As Table II shows, *SVR-AM-Loss-Queue* predicts throughput within 10% of actual accuracy 49% of the time, while HB does so only 32% of the time. Hence, for high traffic scenarios, *SVR-AM-Loss-Queue*, the practically deployable lightweight version of the SVR-based prediction mechanism, significantly outperforms HB prediction.

The above observations raise the question of why SVR performs so much better than HB at high accuracy levels. HB is a weighted average that works on the assumption that transfers close to each other in time will have similar throughputs. HB loses accuracy when there are rapid and large changes in throughput, because in that case there is a large difference between the weighted average and individual throughputs. SVR predicts throughput by constructing a function mapping path properties to throughput values. This function is able to capture complex non-linear relationships between the path properties and throughput. As long as the mapping between path properties and resulting throughput is consistent, SVR can predict throughput accurately, regardless of the variation between the throughput of neighboring transfers. SVR accuracy breaks down only when the mapping between path properties and throughput becomes inconsistent, *e.g.*,, when path conditions are so volatile that they change between the active path measurement and the file transfer, or when an aggregate path measure does not hold for the tranfer, such as when there is loss on the bottleneck link but the file transfer experiences more or less loss than the aggregate traffic. Thus, SVR outperforms HB because *(a)* it has access to additional information, in the form of path properties such as AB, L, and Q, while HB has access only to past throughputs, and *(b)* it constructs a much more sophisticated functional mapping of path properties to throughput compared to the simple time-series analysis that HB uses. The fact that SVR's supremacy over HB is limited to high accuracy levels, and HB catches up with SVR for lower accuracy levels (predicted throughput within 40%-90% of actual, Table II), is a feature of the distribution of throughputs in our test set. We can see from the graphs in Figure 3(a)- 3(h) that the average throughput of the test set is between 10 and 15 Mb/s. We expect the HB prediction to be around the average throughput. Figure 3(a) shows that this is indeed the case: all HB predictions are



(a) Q-Q Plot SVR-OPM-Loss-Queue. (b) Q-Q Plot for SVR-AM-Loss-Queue.

Fig. 4. Normal Q-Q Plots for Prediction Errors with (a) oracular measurements and (b) active measurements of *Loss-Queue*.

between 10 and 15 Mb/s. All but one transfers have actual throughput between 5 Mb/s and 18 Mb/s, so the relative error is 100% or less ($\frac{10-5}{5} * 100 = 100$) for low throughputs (all predictions within a factor of two of actual) and 80% or less ($\frac{18-10}{10} * 100 = 80$) for high throughputs. If the range of actual throughputs had been wider but the average had been the same, HB would have been less able to match SVR accuracy for relative error values of 40%-90%.

A remaining question is why Q is a better predictor of throughput than AB. When we compare AB and Q prediction behavior in Figures 3(b) and 3(d), we see that as throughput decreases, *i.e.*, as the network moves from lossless to lossy conditions, AB loses its predictive power sooner than Q does, around 15 Mb/s instead of 11 Mb/s for Q. As soon as the link experiences loss, AB is zero. While Q eventually reaches a maximum value under sufficiently high congestion, under light loss it retains its resolution because Q will be different depending on the percentage of packets that experience the maximum possible queuing delay and those which experience less than the maximum. Hence Q is a better predictor of throughput than AB most likely because it can be measured with better resolution for a wider range of network conditions.

3) The Nature of Prediction Error: Lu *et al.* [16] observed in their study that throughput prediction errors were approximately normal in distribution. As the authors noted, normality would justify standard computations of confidence intervals. We examined the distribution of errors in our experiments and also found evidence suggestive of normality.

Figure 4 shows two normal quantile-quantile (Q-Q) plots for *SVR-OPM-Loss-Queue* (Figure 4(a)) and *SVR-AM-Loss-Queue* (Figure 4(b)). Samples that are consistent with a normal distribution form approximately a straight line in the graph, particularly toward the center. In Figures 4(a) and 4(b), we see that the throughput prediction samples in each case form approximately straight lines. These observations are consistent with normality in the distribution of prediction errors. Error distributions from other experiments were also consistent with the normal distribution, but are not shown due to space limitations. We further discuss the issues of retraining and of detecting estimation problems in Section VII.

TABLE III
RELATIVE ACCURACY OF *SVR*-BASED PREDICTOR USING ORACULAR PASSIVE MEASUREMENTS (*OPM*) AND TRAINING SETS CONSISTING OF 1, 2, 3, 6, OR 8 DISTINCT FILE SIZES.

Relative Error	No. of distinct file sizes in training				
	1	2	3	6	8
10%	0.06	0.24	0.49	0.35	0.34
20%	0.16	0.40	0.57	0.48	0.51
30%	0.18	0.52	0.64	0.54	0.54
40%	0.19	0.61	0.66	0.59	0.61
50%	0.22	0.64	0.67	0.65	0.66
60%	0.24	0.67	0.68	0.66	0.67
70%	0.24	0.69	0.68	0.67	0.67
80%	0.29	0.71	0.69	0.68	0.68
90%	0.30	0.72	0.70	0.68	0.68

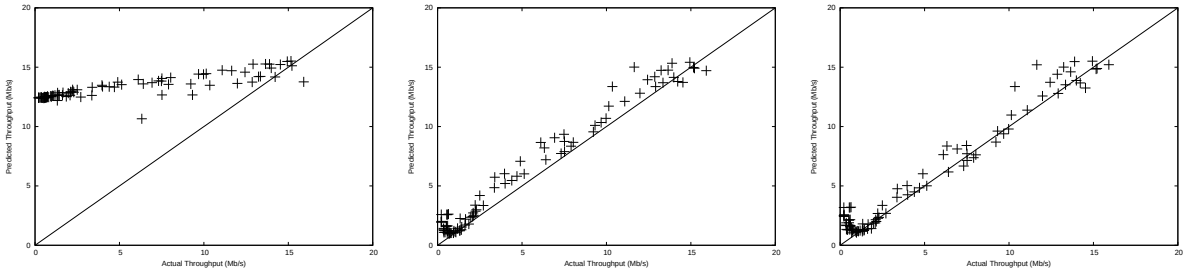
B. Evaluation of Prediction Accuracy for Different File Sizes

Thus far, we have predicted throughput only for bulk transfers of 8MB. However, our goal is accurate prediction for a range of file sizes, not just bulk transfers, so in this section we evaluate prediction accuracy for a broad range of file sizes, something not considered in prior HB prediction studies.

We conducted experiments using background traffic at an average offered load of 135 Mb/s. We used 9 different training sets, each of size 100, consisting of between 1 and 9 unique file sizes. The file sizes for the training sets were between 32 KB (2^{15}bytes) and 8 MB (2^{23}bytes). The first training set consisted of 100 8 MB files, the second of 50 32 KB files and 50 8 MB files, the third of 33 each of 32 KB (2^{15}bytes), 512 KB (2^{19}bytes) and 8 MB (2^{23}bytes) files, the fourth of 25 each of 2^{15}, $2^{17.66}$, $2^{20.33}$, and 2^{23} byte files, and so on, dividing the range of the exponent, 15 to 23, into more numerous but smaller intervals for each subsequent training set.

Test sets for our experiments consist of 100 file sizes between 2 KB (2^{11}bytes) and 8 MB (2^{23}bytes). To generate file sizes for the test set, uniformly distributed random numbers r between 11 and 23 were generated. The file size was set to 2^r bytes, rounded to the closest integer; note that $2^{11} = 2KB$, and $2^{23} = 8MB$. This log-based distribution of file sizes contains a much greater number of smaller files than a uniform linear distribution of file sizes from 2KB to 8MB would. The bias towards smaller file sizes is important for proper evaluation of predictor performance for small files because it generates a more even distribution between files that spend all their transfer time in slow-start, those which spend some time in slow-start and some in congestion avoidance, and those which spend a negligible amount of time in slow-start. If the uniform linear distribution had been used, most of the file sizes generated would have spent a negligible amount of time in slow-start. We also use a wider range of small files in the test set compared to the training set – the test set starts with 2KB files while the training set starts with 32KB – because we want to see how well our predictor extrapolates for unseen ranges of small file sizes.

Tables III and IV present prediction accuracy for training sets consisting of 1, 2, 3, 6, or 8 distinct file sizes for *SVR-OPM* and *SVR-AM*. Graphs in Figure 5 present *SVR-AM* results for one, two, and three file sizes in the training set. We do not include graphs for remaining training sets due to space limitations: they are similar to those for two and three



(a) *SVR-AM* with file size of 8 MB in training set.

(b) *SVR-AM* with file sizes of 32 KB and 8 MB in training set.

(c) *SVR-AM* with file sizes of 32 KB, 512 KB, and 8 MB in training set.

Fig. 5. Scatter plots for the *SVR*-based predictor using 1, 2, or 3 distinct file sizes in the training set. All results shown use active measurements to train the predictor. Testing is done using a range of 100 file sizes from 2 KB to 8 MB.

TABLE IV
RELATIVE ACCURACY OF *SVR*-BASED PREDICTOR USING ACTIVE MEASUREMENTS (*AM*) AND TRAINING SETS CONSISTING OF 1, 2, 3, 6, OR 8 DISTINCT FILE SIZES.

Relative Error	No. of distinct file sizes in training				
	1	2	3	6	8
10%	0.10	0.29	0.40	0.29	0.29
20%	0.15	0.41	0.51	0.47	0.47
30%	0.16	0.53	0.59	0.52	0.55
40%	0.19	0.58	0.64	0.57	0.61
50%	0.23	0.64	0.65	0.62	0.64
60%	0.23	0.66	0.66	0.64	0.65
70%	0.26	0.70	0.67	0.64	0.66
80%	0.28	0.70	0.68	0.64	0.67
90%	0.31	0.71	0.69	0.65	0.68

file sizes in the training set. The first observation is that for the single file size in training, the prediction error is very high. This inaccuracy is expected because the predictor has been given no information about the relationship between size and throughput for small files. The second observation is that for more than one file size, prediction becomes dramatically more accurate, *i.e.*, the predictor is able to successfully extrapolate from a handful of sizes in training to a large number of sizes in testing. The third observation is that relative error is low for large file sizes (corresponding to high actual throughput) while it is higher for small files (low actual throughput), or, in other words, predicting throughput accurately for small files is more difficult than for large files. For small files, throughput increases rapidly with file size during TCP's slow-start phase due to the doubling of the window every RTT. Also, packet loss during the transfer of a small file has a larger relative impact on throughput. Due to the greater variability of throughput for small files, predicting throughput accurately is more difficult. The fourth observation is that for small file sizes (*i.e.*, small actual throughput), the error is always that of over-prediction. The smallest file in the training set is 32KB while the smallest file in the test set is 2KB. This difference is the cause of over-prediction errors: given the high variability of throughput for small file sizes due to slow-start effects and losses having a greater relative impact on throughput, without a broader training set, the SVR mechanism is unable to extrapolate accurately for unseen ranges of small file sizes.

A final observation is that prediction accuracy reaches a maximum at three file sizes in the training set, and there is no clear trend for four to nine file sizes in the training set. The number of transfers in our training set is always constant at 100, so for the single training size, there are 100 8 MB transfers, for the two training sizes, there are 50 32 KB transfers and 50 8 MB transfers, and so on. We believe that accuracy is maximum at three training sizes in our experiments because there is a trade-off between capturing a diversity of file sizes and the number of samples for a single file size. We would not see maximum accuracy occurring at three file sizes and would instead see an increase in accuracy with increasing number of file sizes in the training set if we kept the number of samples of a single file size constant at 100 in the training set and allowed the size of the training set to increase from 100 to 200, 300, etc., as we increase the number of file sizes in the training set.

VI. WIDE AREA EXPERIMENTS

To further evaluate our SVR-based TCP throughput prediction method we created a prototype tool called *PathPerf* that can generate measurements and make forecasts on wide area paths. We used *PathPerf* to conduct experiments over a set of paths in the RON testbed [3]. This section presents the results of our wide area experiments.

A. The RON Experimental Environment

The RON wide area experiments were conducted in January 2007 over 18 paths between 7 different node locations. Two nodes were in Europe (in Amsterdam and London), and the remainder were located at universities in the continental United States (Cornell, Maryland, New Mexico, NYU, and Utah). Of the 18 paths, two are trans-European, 9 are trans-Atlantic, and 7 are trans-continental-US. The RON testbed has a significantly larger number of available nodes and paths, but two considerations limited the number of nodes and paths that we could use. The first consideration was that the nodes should have little or no other CPU or network load while our experiments were running: this is required for BADABING to measure loss accurately. The second issue was that we could not use any nodes running FreeBSD 5.x because the TCP throughput history caching feature in FreeBSD 5.x, controlled by the variable net.inet.tcp.inflight.enable, is on by default, and interfered with our experiments. Consequently, we were

TABLE V
MINIMUM RTTS FOR THE 18 RON PATHS USED FOR WIDE-AREA EXPERIMENTS

Paths (Abbreviations)	Minimum RTT (ms)
Amsterdam-London (A-L, L-A)	8
Cornell-Amsterdam (C-A)	92
Cornell-London (C-L)	83
Cornell-NYU (C-NY)	8
Cornell-Utah (C-U, U-C)	71
London-Maryland (L-M)	81
London-NYU (L-NY, NY-L)	72
London-Utah (L-U, U-L)	145
New Mexico-Maryland (NM-M)	106
New Mexico-Cornell (NM-C)	87
NYU-Amsterdam (NY-A)	80
NYU-London (NY-L)	62
NYU-Utah (NY-U, U-NY)	67

restricted to using nodes running FreeBSD 4.7, which does not have the throughput history caching feature.

We use the following convention for path names: A-B means that A is the TCP sender and B is the receiver, *i.e.*, the TCP data flow direction is from A to B, and TCP ack flow is in the reverse direction. A-B and B-A are considered two different paths, because routing and background traffic level asymmetry between the forward and reverse directions can lead to major differences in throughput.

The wide area measurement protocol was the following: *(1)* run BADABING for 30 seconds, *(2)* transfer a 2MB file, *(3)* sleep 30 seconds, and *(4)*repeat the above 200 times.

In Section V, we showed that available bandwidth measurements are not required for accurate TCP throughput prediction, so we omit running YAZ in the wide area. Thus, the wide area prediction mechanism is the *SVR-AM-Loss-Queue* protocol from Section V. We reduced the size of the file transfers from 8MB in the lab experiments to 2MB in the wide area experiments because the typical throughput in the wide area was an order of magnitude lower compared to the typical throughput in the lab (1-2 Mbps versus 10-15 Mbps).

The measurements were carried out at different times of the day for different paths depending upon when the nodes were available, *i.e.*, when the nodes had little or no other CPU and network activity in progress. Depending again on node availability, we ran a single set of experiments on some paths, and multiple sets on others. We can only conduct active measurements in the wide area because the infrastructure required for passive measurements is not present.

Table V lists the minimum round trip times observed on the wide area paths over all BADABING measurements taken on each path. The shortest paths have a minimum RTT of 8ms and the longest a minimum RTT of 145 ms. In Table V we ignore path directions, *e.g.*, we list Amsterdam-London and London-Amsterdam as a single entry, because minimum RTT is the same in both directions.

B. Wide Area Results

Figure 6 compares the accuracy of the SVR and HB throughput predictions for the 18 wide area paths. The results in Figure 6 are obtained by dividing the 200 measurements gathered for each path into two consecutive sets of 100, and using the first 100 as the training set and the second 100 as the test set. Some paths feature more than once because node availability allowed us to repeat experiments on those paths.

We observe two major trends from Figure 6. First, for the majority of experiments, prediction accuracy is very high for both HB and SVR: most paths have greater than 85% of predictions within 10% of actual throughput. Second, for 5 out of the 26 experiments – Cornell-Amsterdam, London-Maryland-1, London-Utah-2, London-Utah-5 and NYU-Amsterdam – the SVR prediction accuracy is quite low, as low as 25% for London-Utah-2. The reasons for this poor accuracy will be analyzed in detail in Section VI-C.

Next, we take a more detailed approach to assessing prediction accuracy. We divide the 200 samples into sets of *k* and *200-k* for values of *k* from 1 to 199. The first *k* samples are used for training, and the remaining *200-k* samples are used for testing. This allows us to understand the trade-off between training set size and prediction accuracy, which is important for a practical online prediction system where it is desirable to start generating predictions as soon as possible. This analysis also allows us to identify the points in a trace where an event that has an impact on prediction accuracy, such as a level shift, occurs, and whether retraining helps maintain prediction accuracy in the face of changing network conditions. We present this data for SVR only; for HB, there is no division of data into training and test sets because retraining occurs after every measurement.

Figure 7 presents the SVR prediction accuracy for *k*=1, 5, and 10 for those experiments in Figure 6 that had high prediction accuracy for *k=100*. For all but three of the experiments in Figure 7, there is little difference between prediction accuracy for training set sizes of 1, 5, 10, and 100. This is because there is little variation in the throughput observed during the experiments. A path with little or no variation in observed throughput over the course of an experimental run is the easy case for both SVR and HB throughput predictors, so these experiments will not be discussed any further in this paper. For three experiments in Figure 7, Amsterdam-London, London-Utah-1, and Utah-Cornell, the prediction accuracy for *k* values of 1, 5, and 10 is significantly lower compared to that for *k=100*. The reason for this poor accuracy will be discussed in detail in Section VI-C.

C. Detailed Analysis of Wide Area Results

In this section we analyze the reasons for poor SVR prediction accuracy for 5 paths from Figure 6 (Cornell-Amsterdam, London-Maryland-1, London-Utah-2, London-Utah-5, and NYU-Amsterdam) and 3 paths from Figure 7 (Amsterdam-London, London-Utah-1, and Utah-Cornell). We find that there are two dominant reasons for poor SVR prediction accuracy: background traffic level shifts and changes in background traffic on small timescales. We find that retraining can improve prediction accuracy for level shifts but not for changes in network conditions on small timescales.

In the analysis that follows, we use a pair of graphs per experiment to illustrate the details of each experiment. Consider Figures 8(a) and 8(b) for London-Utah-5. The first graph, the *throughput profile*, is a time series representation

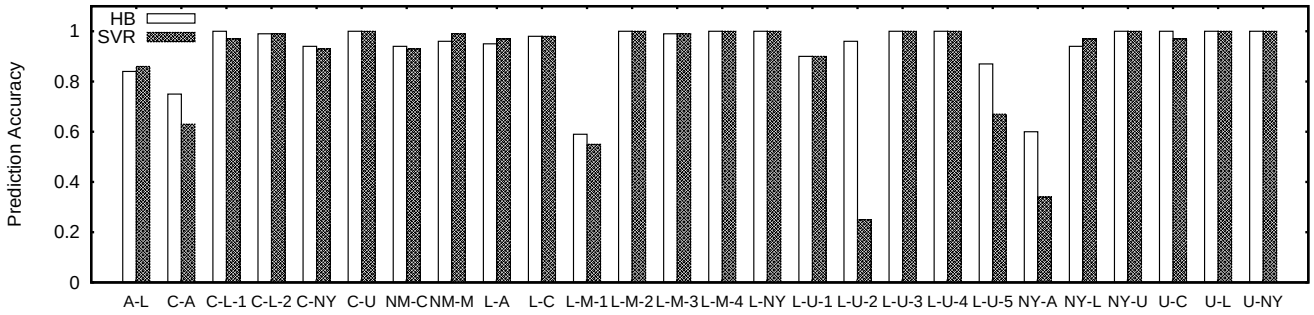


Fig. 6. Wide Area Results for HB and SVR predictions: the HB and SVR columns present the fraction of predictions with relative error of 10% or less. Training and test sets consisted of 100 samples each. 2MB of data was transferred. Labels use node initials; see Section V for complete node names.

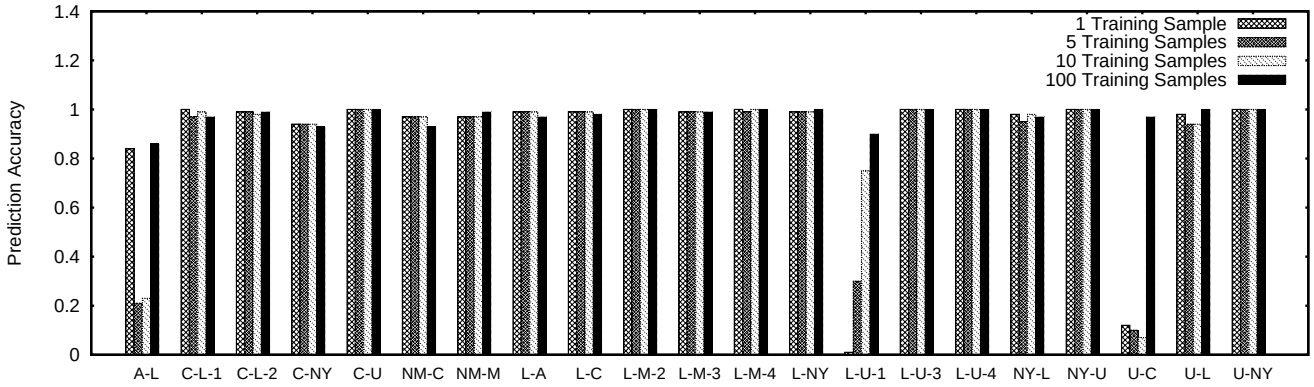
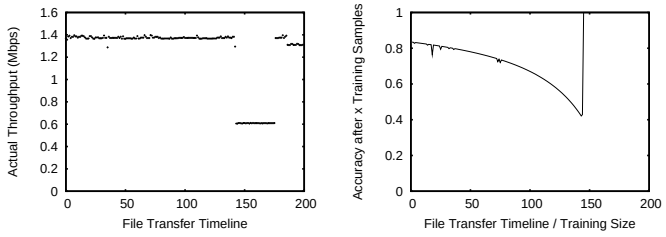
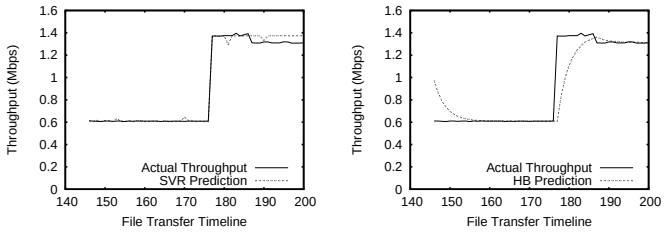


Fig. 7. Effect of training set size on SVR prediction accuracy in the wide area. For paths with high prediction accuracy in Figure 6, this table shows how large a training set size was needed to achieve high accuracy. Labels use node initials; see Section V for complete node names.



(a) Throughput Profile (b) Prediction Accuracy

Fig. 8. London-Utah-5: An Example of a Wide Area Path with Level Shifts. Time on the x-axis is represented in terms of file transfer number



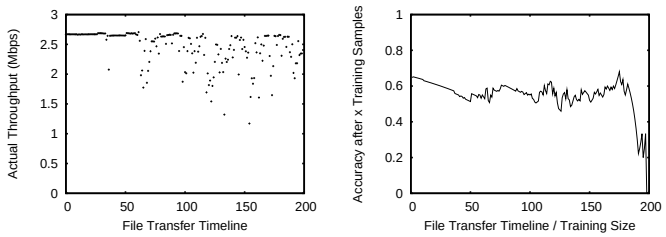
(a) SVR Prediction Accuracy (b) HB Prediction Accuracy

Fig. 9. London-Utah-5: Comparison of SVR and HB prediction accuracy for background traffic with level shifts. The x axis is the file transfer timeline starting at 145. The y axis is throughput.

of the actual throughput observed during the experiment. The second graph, *prediction accuracy*, is a more detailed version of the bar graphs of Figure 7, incorporating all possible values of *k*. At an x value of *k*, the first *k* of the 200 samples are used as the training set, and the remaining *200-k* samples as the test set. The y value is the fraction of samples in the test set of *200-k* whose predicted throughput is within 10% of the actual throughput. As the value of *k* increases, the training set becomes larger compared to the test set, because the total number of samples is fixed at 200. For values of *k* close to 200, the test set is very small, and thus the prediction accuracy values, whether they are high or low, are not as meaningful.

1) Level Shifts: Level shifts were responsible for poor prediction accuracy in four experiments: Utah-Cornell, London-Utah, London-Utah-2, and London-Utah-5. We discuss London-Utah-5 in detail as a representative example.

Figures 8(a) and 8(b) are the *throughput profile* and *prediction accuracy* graphs for London-Utah-5. Figure 8(a) shows that a level shift from a throughput of 1.4 Mbps to 0.6 Mbps occurs after 144 file transfers, and a level shift from 0.6 Mbps back to 1.4 Mbps occurs after 176 transfers. Figure 8(b) shows that prediction accuracy decreases from 0.84 at 1 file transfer to a global minimum of 0.40 at 144 file transfers, and then increases very rapidly to 1.00 at 145 file transfers. This result can be explained by the fact that the level shift is not included in the training data. Thus, the prediction function will generate samples at the first throughput level accurately, and those at



(a) Throughput Profile (b) Prediction Accuracy

Fig. 10. London-Maryland-1: An example of a wide area path with throughput changes on small timescales

the second throughput level inaccurately. Prediction accuracy decreases from 1 to 144 because there are relatively fewer samples at the first level and more at the second level in the test set as the value of x increases, so there are relatively more inaccurate predictions, leading to a decreasing trend in prediction accuracy. If the division into training and test sets is at the level shift boundary, in this case the first 144 file transfers, the training set consists only of measurement samples before the level shift and the test set of samples after the level shift. All predictions will be inaccurate (assuming the level shift changes the throughput by more than 10%, our threshold) because the prediction function has never encountered the new throughput level. Hence, we observe minimum prediction accuracy at the level shift boundary, *i.e.*, 144 transfers. If the division into training and test set includes a level shift, some samples with the new network conditions and resultant new throughput are included in the training set. The prediction function is now aware of the new network conditions, and is able to make better predictions in the new conditions. In our example, including only *one* sample from after the level shift in the training set, the 145th sample, is sufficient to allow all throughputs at the lower levels to be predicted accurately. That is, the SVR predictor needs the minimum possible training set size (one single sample) for the new network conditions before it can generate accurate predictions.

Figures 9(a) and 9(b) compare the behavior of SVR and HB predictors for a level shift. The training set for SVR consisted of the first 145 samples, *i.e.*, 144 samples at the first throughput level and 1 sample at the second throughput level. The test set consisted of the remaining 55 samples. For HB, recall that there is no separation into training and test sets, and retraining occurs after every measurement sample. Comparing Figures 9(a) (SVR) and 9(b) (HB), we see that the SVR predicted throughput follows the actual throughput very closely, while the HB predicted throughput takes some time to catch up with actual throughput after a level shift. If the SVR predictor has knowledge of the level shift, its prediction accuracy is much better than that of HB. After the second level shift (176 samples) no further training of the SVR predictor is required to predict the remaining 23 correctly. The predicted throughput in Figure 9(a) follows the actual throughput closely at the level shift after 176 transfers even though the training set consists of only the first 146 samples.

The above example shows that the SVR predictor has two advantages over the HB predictor. First, it can adapt instantaneously, *i.e.*, after a single training sample, to a level shift, while the HB predictor takes longer. Second, it shows that unlike the HB predictor, the SVR predictor needs to be trained only once for a given set of conditions. The results for the other three wide area experiments that contained level shifts are similar to those of the London-Utah-5 experiment (omitted due to space considerations).

2) Changes Over Small Timescales: Network condition changes over small timescales reduced SVR prediction accuracy for Amsterdam-London, Cornell-Amsterdam, London-Maryland-1, and NYU-Amsterdam. We consider London-Maryland-1 as a representative example.

Figures 10(a) and 10(b) present the throughput profile and prediction accuracy of the London-Maryland-1 experiment. Figure 10(a) shows that until about the 60th file transfer, throughput is fairly steady around 2.7 Mbps, after which it starts to vary widely in the range between approximately 1.2 Mbps and 2.7 Mbps. Figure 10(b) shows that prediction accuracy is at a maximum of 65% at one file transfer, decreases in accuracy between 1 and 60 transfers, and varies between 50% and 60% between 60 and 180 transfers.

Unlike for level shifts, after the throughput profile changes and measurement samples of the new network conditions are included in the training set, prediction accuracy does not improve. Recall that we measure network conditions using BADABING for 30 seconds, and then transfer a file. In this experiment, after the 60th transfer, the network conditions vary so rapidly that they change between the BADABING measurement and the file transfer. Consequently, there is no consistent mapping between the measured network conditions and the transfer throughput, so a correct SVR predictor cannot be constructed, leading to poor prediction accuracy.

The HB predictor also performs poorly in these conditions. As shown in Figure 6, for 100 training samples for London-Maryland-1, HB has a prediction accuracy of 59% while SVR has an accuracy of 55%. When network conditions vary as rapidly as in the above example, it is not possible to predict throughput accurately using either SVR or HB because of the absence of a consistent relationship in network conditions just before and during a file transfer.

One difference we observed in experiments with level shifts versus experiments with throughput changes on small timescales was that level shifts occurred under lossless network conditions while throughput changes on small timescales occurred under conditions of sustained loss. Thus if only the average queuing delay on a network path changes, we observe a level shift; if a network goes from a no-loss state to a lossy state, we observe throughput changes on small timescales. A possible explanation is that if the network is in a lossy state, a particular TCP flow may or may not experience loss. Since TCP backs off aggressively after a loss, flows that do experience loss will have significantly lower throughput compared to those that do not experience loss, leading to large differences in throughput of flows. However, if only average queuing delay on a path changes, every flow on the path (unless it is extremely short) experiences the change in queuing delay, leading to a change in throughput of all flows, *i.e.*, a level shift, on the path.

VII. Discussion

This section addresses two key issues related to running *PathPerf* in operational settings.

Network Load Introduced by *PathPerf*. Traffic introduced by active measurement tools is a concern because it can skew the network property being measured. Furthermore, network operators and engineers generally wish to minimize any impact measurement traffic may have on customer traffic.

For history-based TCP throughput estimation methods, the amount of traffic introduced depends on the measurement protocol. For example, two approaches may be taken. The first method is to periodically transfer fixed-size files; the second is to allow a TCP connection to transfer data for a fixed time period. The latter approach was taken in the history-based evaluation of He *et al.* [11]. To estimate the overhead of a history-based predictor using fixed-duration transfers, assume that (1) the TCP connection is not *rwnd*-limited, (2) that the fixed duration of data transfer is 50 seconds (as in [11]), (3) throughput measurements are initiated every 5 minutes, and (4) throughput closely matches available bandwidth. For this example, assume that the average available bandwidth for the duration of the experiment is approximately 50 Mb/s. Over a 30 minute period, nearly 2 GB in measurement traffic is produced, resulting in an average bandwidth of about 8.3 Mb/s.

In the case of *PathPerf*, measurement overhead in training consists of file transfers and queuing/loss probe measurements. In testing, overhead consists solely of loss measurements. Assume that we have a 15 minute training period followed by a 15 minute testing period. Assume that file sizes of 32 KB, 512 KB, and 8 MB are transferred during the training period, using 10 samples of each file, and that each file transfer is preceeded by a 30 second BADABING measurement. With a probe probability of 0.3, BADABING traffic for each measurement is about 1.5 MB. For testing, only BADABING measurements must be ongoing. Assume that a 30 second BADABING measurement is initiated every three minutes. Thus, over the 15 minute training period about 130 MB of measurement traffic is produced, resulting in an average bandwidth of about 1.2 Mb/s for the first 15 minutes. For the testing period, a total of 7.5 MB of measurement traffic is produced, resulting in a rate of about 66 Kb/s. Overall, *PathPerf* produces 633 Kb/s on average over the 30 minute measurement period, dramatically different from a standard history-based measurement approach. Even if more conservative assumptions are made on the history-based approach, the differences in overhead are significant. Again, the reason for the dramatic savings is that once the SVR predictor has been trained, only lightweight measurements are required for accurate predictions.

Detecting Problems in Estimation. An important capability for throughput estimation in live deployments is to detect when there are significant estimation errors. Such errors could be indicative of a change in network routing, causing an abrupt change in delay, loss, and throughput. It could also signal a pathological network condition, such as an ongoing denial-of-service attack leading to endemic network loss along a path. On the other hand, it may simply be a measurement outlier with no network-based cause.

As discussed in Section V-A3, normality allows us to use standard statistical machinery to compute confidence intervals (*i.e.*, using measured variance of prediction error). We show that prediction errors are consistent with a normal distribution and further propose using confidence intervals as a mechanism for triggering retraining of the SVR as follows. Assume that we have trained the SVR predictor over n measurement periods (*i.e.*, we have n throughput samples and n samples of L and Q). Assume that we then collect k additional throughput samples, making predictions for each sample and recording the error. We therefore have k error samples between what was predicted and what was subsequently measured. Given a confidence level, *e.g.*, 95%, we can calculate confidence intervals on the sample error distribution. We can then, with low frequency, collect additional throughput samples to test whether the prediction error exceeds the interval bounds. (Note that these additional samples may be application traffic for which predictions are used.) If so, we may decide that retraining the SVR predictor is appropriate. A danger in triggering an immediate retraining is that such a policy may be too sensitive to outliers regardless of the confidence interval chosen. More generally, we can consider a threshold m of consecutive prediction errors that exceed the computed confidence interval bounds as a trigger for retraining.

Predictor Portability In this paper, since we have only considered predictors trained and tested on the same path, a number of features, such as the identity of a path and MTU, are implicit. This makes SVR simpler to use for path-specific prediction, because a lot of path details that would be required by other approaches, such as formula-based prediction, can safely be hidden from the user.

If two paths are similar in all features, implicit and explicit, then a predictor trained on one path using only Q and L (and possibly AB) can be used directly for the other path. If paths differ in implicit features, then those features will have to be added explicitly to the predictor to allow the predictor to be portable. Fortunately, SVR extends naturally to include new features. If a complete set of features effecting TCP throughput can be compiled, and there is a wide enough set of consistent training data available, then a *universal* SVR predictor that works on just about any wireline Internet path might be a possibility. A potential challenge here is identifying and measuring all factors – path properties, TCP flavors, operating system parameters – that might affect throughput. However, this challenge can be surmounted to quite an extent by active path measurements, using TCP header fields, and parsing operating systems configuration files to include operating parameters as SVR features.

Acknowledgments

We would like to thank David Andersen for providing us access to the RON testbed, and Ana Bizzaro for help with the WAIL lab testbed. We also thank the anonymous reviewers for their helpful suggestions.

References

[1] A. Abouzeid, S. Roy, and M. Azizoglu. Stochastic Modeling of TCP Over Lossy Links. In *Proceedings of IEEE INFOCOM '00*, Tel-Aviv, Israel, March 2000.

[2] E. Altman, K. Avachenkov, and C. Barakat. A Stochastic Model of TCP/IP with Stationary Random Loss. In *Proceedings of ACM SIGCOMM '00*, August 2000.

[3] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *Proceedings of 18th ACM Symposium on Operating Systems Principles (SOSP)*, Banff, Canada, October 2001.

[4] M. Arlitt, B. Krishnamurthy, and J. Mogul. Predicting Short-Transfer Latency from TCP Arcana: A Trace-based Validation. In *Proceedings of the ACM Internet Measurement Conference*, Berkeley, CA, October 2005.

[5] H. Balakrishnan, S. Seshan, M. Stemm, and R. Katz. Analyzing Stability in Wide-Area Network Performance. In *Proceedings of ACM SIGMETRICS '97*, Seattle, WA, June 1997.

[6] P. Barford and M. Crovella. Critical Path Analysis of TCP Transactions. In *Proceedings of ACM SIGCOMM '00*, Stockholm, Sweeden, August 2000.

[7] R. Beverly, K. Sollins, and A. Berger. SVM Learning of IP Address Structure for Latency Prediction. In *Proceedings of the SIGCOMM Workshop on Mining Network Data*, Pisa, Italy, September 2006.

[8] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *Proceedings of IEEE INFOCOM '00*, Tel-Aviv, Israel, March 2000.

[9] Cooperative Association for Internet Data Analysis. http://www.caida.org/tools, 2006.

[10] M. Goyal, R. Buerin, and R. Rajan. Predicting TCP Throughput From Non-invasive Network Sampling. In *Proceedings of IEEE INFOCOM '02*, New York, NY, June 2002.

[11] Q. He, C. Dovrolis, and M. Ammar. On the Predictability of Large Transfer TCP Throughput. In *Proceedings of ACM SIGCOMM '05*, Philadelphia, PA, August 2005.

[12] K. Ilgun, R. Kemmerer, and P. Porras. State Transition Analysis: A Rule-Based Intrusion Detection Approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[13] V. Jacobson and M. Karels. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM '88*, Palo Alto, CA, August 1988.

[14] M. Jain and C. Dovrolis. End-to-end Available Bandwidth Measurement Methodology, Dynamics and Relation with TCP Throughput. *ACM/IEEE Transactions on Networking*, 11(4):537 – 549, August 2003.

[15] Thorsten Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[16] D. Lu, Y. Qiao, P. Dinda, and F. Bustamante. Characterizing and Predicting TCP Throughput on the Wide Area Network. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, Columbus, OH, June 2005.

[17] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), July 1997.

[18] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM '98*, Vancouver, Canada, Setpember 1998.

[19] V. Paxson. End-to-End Internet Packet Dynamics. In *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997.

[20] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California Berkeley, 1997.

[21] B. Schölkopf and A. J. Smola. *Learning With Kernels*. MIT Press, Cambridge, MA, 2002.

[22] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and kc claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *Proceedings of Passive and Active Measurement Workshop '05*, pages 306–320, 2005.

[23] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[24] J. Sommers and P. Barford. Self-Configuring Network Traffic Generation. In *Proceedings of the ACM Internet Measurement Conference*, Taormina, Italy, October 2004.

[25] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving Accuracy in End-to-end Packet Loss Measurements. In *Proceedings of ACM SIGCOMM '05*, Philadelphia, PA, August 2005.

[26] J. Sommers, P. Barford, and W. Willinger. A Proposed Framework for Calibration of Available Bandwidth Estimation Tools. In *Proceedings of IEEE Symposium on Computers and Communication (ISCC '06)*, June 2006.

[27] J. Strauss, D. Katabi, and F. Kaashoek. A Measurement Study of Available Bandwidth Tools. In *Proceedings of ACM Internet Measurement Conference '03*, Miami, FL, November 2003.

[28] M. Swany and R. Wolski. Multivariate Resource Performance Forecasting in the Network Weather Service. In *Proceedings of Supercomputing*, pages 1–10, Baltimore, MD, November 2002.

[29] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, N.Y., second edition, 1995.

[30] S. Vazhkudai, J. Wu, and I. Foster. Predicting the Performance of Wide Area Data Transfers. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS '02)*, Fort Lauderdale, FL, April 2002.

[31] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, August 2002.

[32] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *Proceedings of the Internet Measurement Workshop*, San Francisco, CA, October 2001.

Mariyam Mirza is a graduate student in Computer Science at the University of Wisconsin-Madison. She received a BSE in Computer Science from Princeton University in 2002. Her research interests include network measurement and performance evaluation.

Joel Sommers received a Ph.D. in Computer Science from the University of Wisconsin-Madison in 2007. He is an assistant professor of Computer Science at Colgate University, and his current research focus is the measurement and analysis of networked systems and network traffic.

Paul Barford received his BS in Electrical Engineering from the University of Illinois at Champaign-Urbana in 1985, and his Ph.D. in Computer Science from Boston University in December, 2000. He is an associate professor of Computer Science at the University of Wisconsin-Madison. He is the founder and director of the Wisconsin Advanced Internet Laboratory, and his research interests are in the measurement, analysis and security of wide area networked systems and network protocols.

Xiaojin Zhu is an Assistant Professor of Computer Science at University of Wisconsin-Madison. His research interests are statistical machine learning, and its applications to natural language processing. He received his Ph.D. in Language Technologies from Carnegie Mellon University in 2005, and B.Sc. in Computer Science from Shanghai Jiao Tong University, China in 1993. From 1996-1998 he was a research member at IBM China Research Laboratory in Beijing, China.