# A Machine Learning Approach to TCP Throughput Prediction

Mariyam Mirza, Joel Sommers, Paul Barford, Xiaojin Zhu
Computer Science Department, University of Wisconsin-Madison
{mirza, jsommers, pb, jerryzhu}@cs.wisc.edu

## ABSTRACT

*TCP throughput prediction* is an important capability in wide area overlay and multi-homed networks where multiple paths may exist between data sources and receivers. In this paper we describe a new, lightweight method for TCP throughput prediction that can generate accurate forecasts for a broad range of file sizes and path conditions. Our method is based on Support Vector Regression modeling that uses a combination of prior file transfers and measurements of simple path properties. We calibrate and evaluate the capabilities of our throughput predictor in an extensive set of lab-based experiments where ground truth can be established for path properties using highly accurate passive measurements. We report the performance for our method in the ideal case of using our passive path property measurements over a range of test configurations. Our results show that for bulk transfers in heavy traffic, 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. In the same lab environment, we assess our method using less accurate active probe measurements of path properties, and show that predictions can be made within 10% of the actual value nearly 50% of the time over a range of file sizes and traffic conditions. This result represents approximately a 60% improvement over history-based methods with a much lower impact on end-to-end paths. Finally, we implement our predictor in a tool called *PathPerf* and test it in experiments conducted on wide area paths. The results demonstrate that *PathPerf* predicts TCP throughput accurately over a variety of paths.

**Categories and Subject Descriptors:** C.4 **[Performance of Systems]**: Measurement Techniques

**General Terms:** Measurement

**Keywords:** TCP Throughput Prediction, Active Measurements, Machine Learning, Support Vector Regression

## 1. 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.

There are several difficulties in generating accurate TCP throughput predictions. Prior work on the problem 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, gathered either passively (*e.g.,* by tapping a link) or actively (*e.g.,* by periodically sending a file). 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.

The analytical framework for the study that we report in this paper is based on the use of Support Vector Regression (SVR), a powerful machine learning technique that has shown good empirical performance in many domains. SVR has several attractive 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, which is a requirement for our approach. 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 an important reason for the potential for more accurate predictions than formula-based 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 an assumption that prediction errors 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 ground truth measurements of throughput of target TCP flows (*i.e.*, actual throughput) with predicted TCP throughput values generated by multiple instances of our SVR-based tool trained with different combinations of path properties. We compare the actual and predicted throughput using the Relative Prediction Error (E) metric described in [11]. 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.

Toward our goal of developing a robust tool that can be used in the wide area, we expand the core SVR-based tool in three ways. First, the initial tests were based entirely on passive traffic measurements, which are unlikely to be widely available in the Internet. To address this, 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 of *Q* and *L* 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-base estimates. It is also important to note that 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 heavy-weight probe-based estimates of available bandwidth such as described in [11]. We quantify this difference in Section 7. Second, we experimented with training data in order to enable predictions over a range of file sizes instead of only bulk transfers which is the focus of prior work. We found that a training set of only three file sizes results in accurate throughput predictions for a wide range of file sizes, which highlights another strength of our SVR-based approach. 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 5.2 that our technique is able to adapt to level shifts quickly and to maintain high accuracy on paths where level shifts occur.

This combination of capabilities was sufficient for us to develop an active probe tool for TCP throughput prediction we call *PathPerf* [1] that we deployed and tested in the wide area. Through a series of experiments over six end-to-end paths in the RON testbed [3] paths, we found that in the best case *PathPerf* provides TCP throughput estimates within 10% of actual value 100% of the time, and in the worst case provides estimates within 10% of actual 35% of the time. We believe that improvements in tools for gathering path property information will lead to corresponding improvements in throughput prediction accuracy. We plan to investigate these possibilities in future work.

## 2. RELATED WORK

Since the 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. In general, these include the TCP implementation, the underlying network structure, and the dynamics of the traffic sharing the links on the path between two hosts. Steps toward understanding TCP behavior have been taken in a number of studies 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 the use of 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 attempt-

[1]PathPerf is openly available at http://wail.cs.wisc.edu/waildownload.py

ing 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 examples [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].

## 3. 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 $\beta$ and offset $\beta_0$ 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 $\varepsilon$-insensitive loss, defined as

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

This loss function measures the absolute error between prediction and truth, but with a tolerance of $\varepsilon$. The value $\varepsilon$ 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 $\beta, \beta_0$ 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 $\beta, \beta_0$ by solving the following optimization problem

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

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 $\beta$ 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 $\beta$. 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 $\beta$ and $\beta_0$ in the augmented feature space defined by $K$. This produces a potentially

[2]If $\beta$ 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.

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.

## 4. EXPERIMENTAL ENVIRONMENT AND METHODOLOGY

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

### 4.1 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.

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.

### 4.2 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).

Measurement traffic in the testbed consisted of file transfers, and active measurements of queuing delay, packet loss, and available bandwidth. For the measurement traffic hosts, we set the TCP receive window size to 128 KB. 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 experimented with both *rwnd*- and *cwnd*-limited scenarios.

For active measurements of available bandwidth and queuing/loss, we used the YAZ [26] and BADABING [25] tools, respectively. YAZ estimates end-to-end available bandwidth using a relatively low-overhead, iterative method similar to PATHLOAD [14]. Since the probe process is iterative, the time taken to produce an estimate can vary from a few seconds to tens of seconds.

BADABING reports two characteristics of *loss episodes*, namely the *frequency* of loss episodes, and *mean duration* of loss episodes, using a lightweight probe process. We used a probe probability parameter $p$ of 0.3. Other parameters were set according to [25]. In the rest of the paper, we refer to both loss characteristics combined as loss or $L$. 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.

We used BADABING to measure loss characteristics because it is the most accurate loss characteristics measurement tool currently available; if accurate *loss rate* measurement tools become available in the future, loss rate may replace frequency and duration as the loss characteristic we use for our prediction mechanism.

The measurement collection protocol consisted of the following:

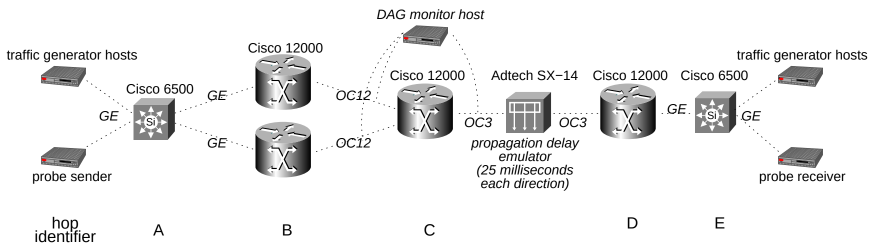
1. Run BADABING for 30 seconds.

2. Run YAZ to obtain an estimate of the available bandwidth.

3. Transfer a file.

In the remainder of the paper, we refer to the above series of steps as a single *experiment*, and to a number of consecutive experiments as a *series*. Experiments in the wide area omit the available bandwidth measurement. Individual experiments in a series are separated by a 30 second period. Series of experiments differ from each other in that either the background traffic is different between series, or the distribution of file sizes transferred is different, or the experiments are conducted over different physical paths. Each series of experiments is divided into two mutually exclusive training and test sets for the SVR. The SVR mechanism does not require that the sets of experiments that form the training and test set be consecutive or contiguous in time. In contrast, history-based prediction methods generally require consecutive historical information since they rely on standard timeseries-based forecasting models. In our evaluation we use contiguous portions for training and test sets, *i.e.* the beginning part of a series becomes the training set and the rest the test set. The number of experiments in training and test sets may be the same or different. Notions of separate training and test data 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.

From each series of experiments, we gather three different sets of measurements. The first set, *Oracular Passive Measurements (OPM)*, are AB, Q, and L measurements during a file transfer that we obtain from packet traces. We refer to these measurements as *oracular* because they give us essentially perfect information about



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.

network conditions. In practice, this information would not be available when making a prediction for an arbitrary path. We use this information to establish the best possible accuracy of our prediction mechanism. The second set, *Active Measurements (AM)*, are the measurements from our active measurement tools. Note that, unlike the *OPM*, the *AM* provide AB, Q, and L values before the actual transfer. The third set, *Practical Passive Measurements (PPM)*, are trace-based measurements of AB, Q and L taken at the same time as *AM* are taken. Their purpose is to show the best possible accuracy of our prediction mechanism with measurements that can be obtained in practice, or to show how much better the accuracy would be if the active measurements had perfect accuracy. All measurements are aggregates for conditions on the path: they are not specific to any single TCP flow on the path.

For experiments in the wide area, we created a tool, *PathPerf*. 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 7.

### 4.3 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.

## 5. 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.

### 5.1 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. Although we refer to both *loss frequency* and *loss duration* together as L or loss for expositional ease, these measures are two different features in the feature vector. 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], $\hat{R}_{i+1} = \alpha R_i + (1-\alpha)\hat{R}_i$, with an $\alpha$ value of 0.3.

We do not report detailed results from tests with low utilization on the bottleneck link, *i.e.*, receive window bound flows, because in the low utilization scenarios there is very little variance in throughput of flows. Our SVR-based predictor generated 100% accurate forecasts for these tests. However, any reasonable prediction method can forecast throughput quite accurately in this scenario. For example, the formula-based predictor used in [11], which is based on [18], performs poorly in high utilization scenarios, but is accurate in low utilization scenarios.

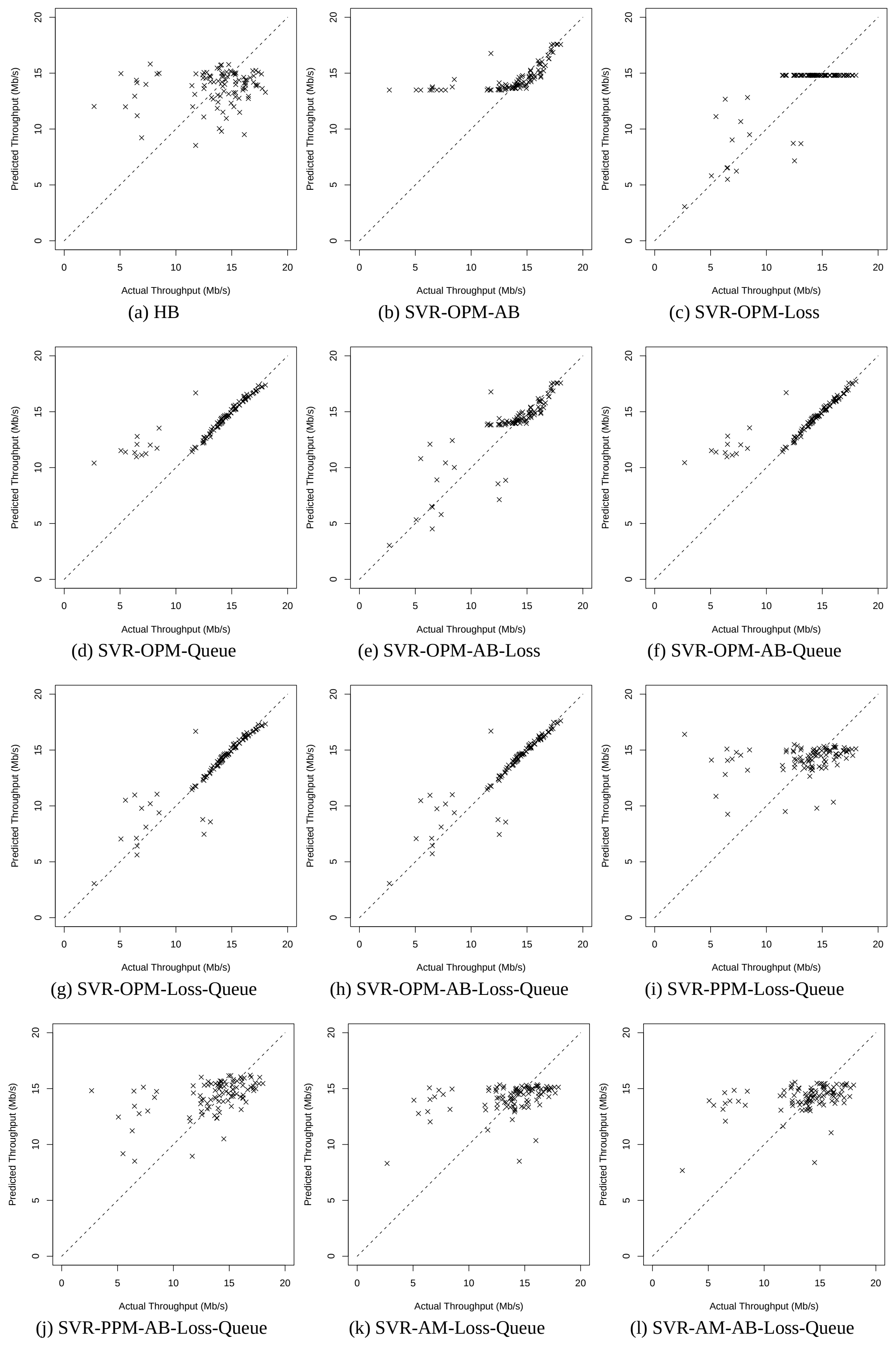
For most of the results reported, we generated an average of 140 Mb/s of background traffic to create high utilization on our OC3 (155 Mb/s) bottleneck link. We used one set of 100 experiments for training and another set of 100 experiments for testing. An 8 MB file was transferred in each experiment. In our initial experiments we use an 8 MB file because it is large enough to make slow-start an insignificant factor in throughput for our receive window size of 128 KB; in later sections we consider smaller file sizes where slow-start has a bigger impact on overall throughput.

Figures 2a to 2l 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.

#### 5.1.1 *Using Path Measurements from an Oracle*

Figure 2a shows the prediction accuracy scatter plot for the HB method. Figures 2b to 2h 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-AB-Queue* means that both available bandwidth and queuing delay measurements were used to train and test.

Table 1 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 1 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



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

(d) SVR-OPM-Queue (e) SVR-OPM-AB-Loss (f) SVR-OPM-AB-Queue

(g) SVR-OPM-Loss-Queue (h) SVR-OPM-AB-Loss-Queue (i) SVR-PPM-Loss-Queue

(j) SVR-PPM-AB-Loss-Queue (k) SVR-AM-Loss-Queue (l) SVR-AM-AB-Loss-Queue

2: Comparison of Prediction Accuracy of HB, SVR-OPM, SVR-PPM, and SVR-AM in High Background Traffic Conditions.

in addition to tabular data to provide insight into how different path properties contribute to throughput prediction in the SVR method.

From Figure 2a, 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 large relative error. Figures 2b, 2c, and 2d 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.

Figures 2e to 2h show improvements on the prediction accuracy obtained by using more than one path property in the SVR feature vector. We can see that when L is combined with either AB or Q, the horizontal lines on the graphs are replaced by points much closer to the diagonal, so combining L with AB or Q allows the SVR method to predict accurately in both lossy and lossless network conditions.

Measurements of AB and Q appear to serve the same function, namely, helping to predict throughput in lossless conditions. This observation begs the question: 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 2e, 2g, and 2h). The general trend in all three cases, as seen from the scatter plots, 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 1 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 (given our framework). 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 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 7.

1: 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 |

2: 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 | *HB* | *PPM* | | *AM* | |
|---|---|---|---|---|---|
| Error | | *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 |

### 5.1.2 *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 baseline for the best-case accuracy with SVR, and also provides insight into how SVR uses different path properties for prediction. Table 1 and the graphs in Figure 5.1 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 are more like those available in the wide area.

We compare HB with 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*.

Figures 2i and 2j show predicted and actual throughput for *SVR-PPM* and Figures 2k and 2l show predicted and actual throughput for *SVR-AM*. Table 2 presents relative prediction error data for HB, *SVR-PPM* and *SVR-AM*. 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.

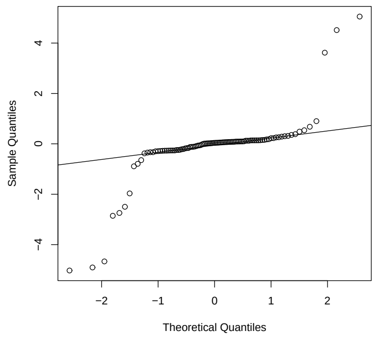
All the scatter plots from Figures 2i to 2l are qualitatively very similar; this is encouraging because it suggests two things. First, *Loss-Queue* has similar prediction accuracy as *AB-Loss-Queue, i.e.*, the observation from *SVR-OPM* still holds, so we can achieve good prediction accuracy without having to measure AB. The relative prediction error data in Table 2 supports this graphical observation: *AB-Loss-Queue* has only slightly better accuracy than *Loss-Queue* for both *SVR-PPM* and *SVR-AM*. Second, *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 measurements. The data in Table 2 further substantiates this observation. Similar accuracy between *SVR-PPM* predictions and *SVR-AM* predictions is important because in real wide-area Internet paths instrumentation is generally not available for providing accurate passive measurements.

Finally, we compare HB with *SVR-AM*. Although Figures 2a, 2k and 2l 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 2 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.

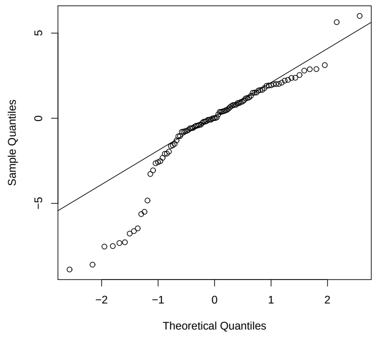
### 5.1.3 The Nature of Prediction Error

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 3 shows two normal quantile-quantile (Q-Q) plots for *SVR-OPM-Loss-Queue* (Figure 3a) and *SVR-AM-Loss-Queue* (Figure 3b). Samples that are consistent with a normal distribution form approximately a straight line in the graph, particularly toward the center. In Figures 3a and 3b, 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 7.



(a) Q-Q Plot SVR-OPM-Loss-Queue.



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

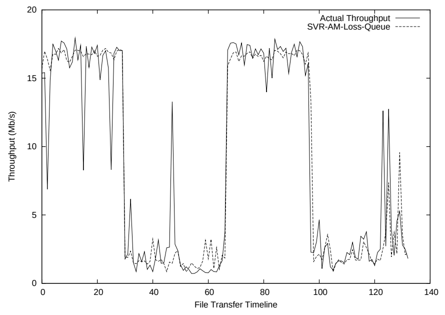
3: Normal Q-Q Plots for Prediction Errors with (a) oracular measurements and (b) active measurements of *Loss-Queue*.

## 5.2 Evaluation of Prediction Accuracy with Level-shift Background Traffic

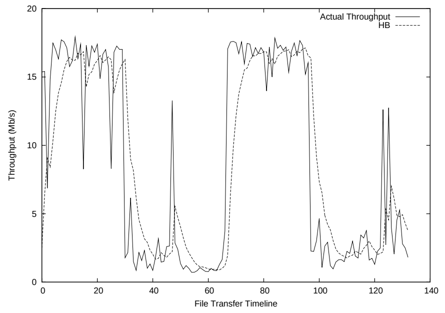
In the previous section, we considered SVR and HB prediction accuracy under variable but stationary background traffic. In this section, we consider the case where there is a shift in average load of background traffic.

We configure our traffic generation process to cycle between 120 Mb/s and 160 Mb/s average offered loads. With 120 Mb/s traffic, there is virtually no loss along the path and the throughput is bound by the receive window size. With 160 Mb/s traffic, the OC3 bottleneck is saturated and endemic loss ensues. The procedure is to run 120 Mb/s background traffic for 75 minutes, followed by 160 Mb/s of traffic for 2 hours 15 min, repeating this cycle several times. We follow the measurement protocol in Section 4.2 for collecting AB, L, Q, and throughput measurements. We first train our SVR predictor in the 120 Mb/s environment, and test it in the 160 Mb/s environment. The column labeled *OPM-L-Q TrainLowTestHigh* in Table 3 shows the results. Clearly, the prediction accuracy is very poor: all predictions are off by a factor of two or more. Next, we train the predictor on one whole cycle of 120 Mb/s and 160 Mb/s, and test it on a series of cycles. The last two columns of Table 3, *OPM-Loss-Queue* and *AM-Loss-Queue*, and Figure 4a show the results from these experiments. The x-axis in Figure 4a is time represented by the number of file transfers that have completed, and the y-axis is throughput. In this case, both *SVR-OPM* and *SVR-AM* predict throughput with high accuracy. *SVR-OPM* predicts throughput within 10% of actual 62% of the time, and *SVR-AM* 52% of the time. For comparison, the first column of Table 3 and Figure 4b present results for the history-based predictor. HB accuracy is significantly lower than SVR accuracy, only about half as much as *SVR-OPM*. Figure 4b explains why. After every level-shift in the background traffic, the HB predictor takes time to re-adapt. In contrast, no adaptation time is required for the SVR predictor if it has already been trained on the range of expected traffic conditions.

One issue regarding shifts in average traffic volume is how many samples from a new, previously unseen traffic level are needed to adequately train the predictor. To examine this issue, we train our predictor on two new kinds of average traffic loads: a step constituting 75 minutes of 120 Mb/s traffic followed by 10 minutes of 160 Mb/s traffic resulting in 5 experiments at the higher traffic level, and a step constituting of 75 minutes of 120 Mb/s traffic followed by 20 minutes of 160 Mb/s traffic resulting in 10 experiments



(a) Level Shift: SVR-AM-Loss-Queue



(b) Level Shift: HB

4: Prediction Accuracy in Fluctuating Background Traffic

3: Relative accuracy of history-based (*HB*) and SVR-based throughput predictors using oracular path measurements (*OPM*) and active measurements (*AM*) when predictors are subjected to shifts in average background traffic volume.

| **Relative Error** | ***HB*** | ***OPM-L-Q TrainLowTestHigh*** | ***OPM-Loss-Queue*** | ***AM-Loss-Queue*** |
|---|---|---|---|---|
| 10% | 0.29 | 0.00 | 0.62 | 0.52 |
| 20% | 0.40 | 0.00 | 0.72 | 0.61 |
| 30% | 0.52 | 0.00 | 0.81 | 0.69 |
| 40% | 0.55 | 0.00 | 0.84 | 0.73 |
| 50% | 0.62 | 0.00 | 0.88 | 0.77 |
| 60% | 0.64 | 0.00 | 0.90 | 0.78 |
| 70% | 0.66 | 0.00 | 0.91 | 0.80 |
| 80% | 0.71 | 0.00 | 0.92 | 0.83 |
| 90% | 0.74 | 0.00 | 0.92 | 0.84 |

4: Comparison of relative accuracy of SVR-based throughput prediction using active measurements (*AM*) and different numbers of training samples. 40 training samples are used in *AM-Loss-Queue*, 5 samples for *AM-L-Q-5*, and 10 samples for *AM-L-Q-10*. Background traffic consists of shifts in average traffic volume between 120 Mb/s and 160 Mb/s.

| **Relative Error** | ***AM-Loss-Queue*** | ***AM-L-Q-5*** | ***AM-L-Q-10*** |
|---|---|---|---|
| 10% | 0.52 | 0.44 | 0.45 |
| 20% | 0.61 | 0.49 | 0.53 |
| 30% | 0.69 | 0.51 | 0.55 |
| 40% | 0.73 | 0.55 | 0.59 |
| 50% | 0.77 | 0.58 | 0.62 |
| 60% | 0.78 | 0.60 | 0.65 |
| 70% | 0.80 | 0.62 | 0.67 |
| 80% | 0.83 | 0.65 | 0.73 |
| 90% | 0.84 | 0.67 | 0.73 |

at the higher traffic level. We call these conditions *SVR-AM-L-Q-5* and *SVR-AM-L-Q-10* respectively. The prediction accuracy of these schemes is presented in Table 4. As can be seen in the table, these training schemes yield most of the accuracy of *SVR-AM-L-Q*, which trains at the 160 Mb/s level for approximately 40 experiments. These results demonstrate that fairly small training sets from new traffic levels are needed for accurate prediction. A comparison of time series plots from these experiments (not shown due to space constraints) provides further insight into how prediction accuracy improves with larger training sets. Both *SVR-AM-L-Q-5* and *SVR-AM-L-Q-10* have periods where the predicted throughput is virtually constant even though the actual throughput is not. Such periods are shorter for *SVR-AM-L-Q-10* compared to *SVR-AM-L-Q-5*. *SVR-AM-L-Q* (Figure 4a) has no such periods. The reason for this effect is that predictors incorporating fewer samples may not include a broad enough range of possible network conditions. However, the SVR prediction mechanism can still yield high accuracy using a small set of training samples if the samples are representative of the range of network conditions along a path.

## 5.3 Evaluation of Prediction Accuracy for Different File Sizes

We have so far considered only 8 MB file transfers in step 3 of the experimental protocol of Section 4.2. For a TCP receive window of 128 KB, the majority of the lifetime of an 8 MB transfer is spent in TCP's congestion avoidance phase. However, our goal is an accurate predictor for a range of file sizes, not just bulk transfers. Predicting throughput for small files is complicated by TCP's slow start phase in which throughput changes rapidly with increasing file size due to the doubling of the window every round-trip time, and because packet loss during the transfer of a small file can have a large relative impact on throughput. We hypothesize that using different file sizes to train the SVR predictor will lead to accurate forecasts for a broad range of file sizes - something not treated in prior HB prediction studies.

We conducted experiments using background traffic at an average offered load of 135 Mb/s and using a series of 9 training sets consisting of between 1 and 9 unique file sizes. The file sizes for the training sets are between 32 KB and 8 MB. The first training set consists of a single file size of 8 MB, the second training set consists of two file sizes of 32 KB and 8 MB, and the third training set adds a file size of 512 KB. Subsequent training sets sample the range between 32 KB and 8 MB such that the fraction of a transfer lifetime spent in slow start covers a wider range.

Test sets for our experiments consist of 100 file sizes between 2 KB and 8 MB – a much more diverse set than the training set. The test file sizes are drawn from a biased random number generator in such a way that the resulting file transfers exhibit a wide range of behavior, *i.e.*, files that are fully transferred during slow start, and transfers consisting of a varying proportion of time spent in slow start versus congestion avoidance. We use a wider range of small files in the test set to allow us to see how our predictor performs for unseen and difficult to predict file sizes. We do not use a wider range for large file sizes because we expect the throughput to be almost constant (*i.e.*, window or congestion limited) once slow start becomes an insignificant fraction of the transfer time.

Tables 5 and 6 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

5: 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 |

6: 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 |

7: Details of RON Paths used for wide-area experiments

| Path Number | Node Locations | Path RTT (ms) |
|---|---|---|
| Path 1 | Amsterdam-Utah | 144 |
| Path 2 | Utah-Maryland | 54 |
| Path 3 | Maryland-Utah | 54 |
| Path 4 | Maryland-New York | 10 |
| Path 5 | New Mexico-Ithaca | 86 |
| Path 6 | New Mexico-New York | 83 |

8: Wide Area Results for 512KB Transfers

| Relative Error | Path 1 | Path 2 | Path 3 | Path 4 | Path 5 | Path 6 |
|---|---|---|---|---|---|---|
| 10% | 0.43 | 0.96 | 0.35 | 0.55 | 1.00 | 0.92 |
| 20% | 0.76 | 1.00 | 0.73 | 0.90 | 1.00 | 0.96 |
| 30% | 0.86 | 1.00 | 0.98 | 0.98 | 1.00 | 0.98 |
| 40% | 0.94 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 50% | 0.94 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 60% | 0.94 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 70% | 0.96 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 80% | 0.96 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 90% | 0.96 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

9: Wide Area Results for 2MB Transfers

| Relative Error | Path1 | Path2 | Path3 | Path4 | Path5 | Path6 |
|---|---|---|---|---|---|---|
| 10% | 0.48 | 0.98 | 0.92 | 0.52 | 0.96 | 0.98 |
| 20% | 0.77 | 1.00 | 0.98 | 0.75 | 0.98 | 1.00 |
| 30% | 0.92 | 1.00 | 0.98 | 0.94 | 0.98 | 1.00 |
| 40% | 0.98 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 50% | 0.98 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 60% | 0.98 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 70% | 0.98 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 80% | 0.98 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 90% | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

those for two and three 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). This is consistent with our expectation that it would be more difficult to accurately predict throughput for small files. 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: the relationship between file size and throughput is complicated for small files, and without a broader training set, the SVR mechanism is unable to provide accurate prediction.

An important 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. A feature of our training set is that the number of transfers is always constant at one hundred, so for the single training size, there are one hundred 8 MB transfers, and for the two training sizes, there are fifty 32 KB transfers and fifty 8 MB transfers. 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—this was alluded to in our discussion of the number of samples needed for good prediction accuracy in Section 5.2. In other words, we believe that 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 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. A thorough characterization of the trade-off between diversity of file sizes and number of samples of each file size is future work.

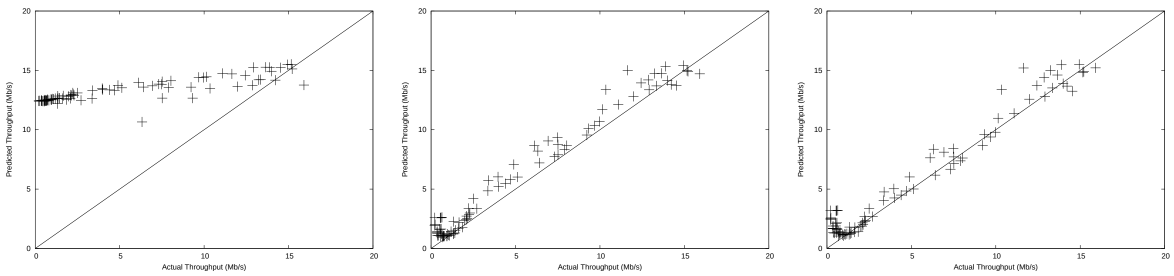
## 6. WIDE AREA TEST RESULTS

To further evaluate our SVR 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 small set of paths in the RON testbed [3].

The RON nodes used in our experiments have a range of CPU and memory configurations, but all ran FreeBSD 4.7 and had limited additional experimental load during the time of our tests. We conducted our tests over the set of paths described in Table 7. These include five trans-continental paths and one trans-Atlantic path. While this path set is modest in size, it has relative diversity in path characteristics and round-trip times that range between 10ms and 144ms.

Since we have shown that AB measurement does not improve throughput prediction accuracy, we eliminate it in the experimental protocol described in Section 4.2. We conduct experiments for two target file sizes: 512 KB and 2MB. The training and test sets consist of 50 transfers of a particular file size. The experiments were conducted between October 20, 2006 and October 31, 2006.

Tables 8 and 9 show the results of the tests. The prediction accuracy is high on all paths for both file sizes: in many cases SVR



(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.

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.

prediction accuracy is within 10% of actual greater than 90% of the time. In most cases, the prediction accuracy for 512KB and 2MB transfers is similar. The one exception is Path 3, where accuracy is within 10% of original only 35% of the time for 512KB transfers, but is much higher at 92% of the time for 2MB transfers. We found that on this particular path the coefficient of variation for the actual throughput of 512KB files is twice that of 2MB files, making prediction for 512KB files more difficult and suggesting that path conditions were more dynamic during this test.

Our measurement data showed that many wide area paths were lightly loaded during our tests and exhibited very little variation in throughput between different file transfers. We are in the process of extending our wide area experiment set considerably in the hope that we will encounter more variability on these paths, which will enable us to further refine *PathPerf's* capability. Finally, we also evaluated the HB predictor's performance in these experiments and found it's forecasts to be approximately the same as SVR. We expect that, as we find paths with more dynamic conditions, our SVR predictor will distinguish itself as our lab experiments demonstrate.

## 7. 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 of concern because excessive traffic 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 specific 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. Thus, 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 the training period consists of file transfers and queuing/loss probe measurements. In the testing phase, 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 5.1.3, 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 in the following way. 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.

## 8. SUMMARY AND FUTURE WORK

In this paper we address the problem of how to generate accurate TCP throughput predictions for arbitrary paths in the Internet. Our approach uses a powerful machine learning tool - Support Vector Regression - which provides an efficient mechanism for generating a predictor using multiple inputs. We investigate measurements of path properties including queuing delay, packet loss, and available bandwidth that can be used along with prior throughput measurements as the feature set for our predictor. Through an extensive series of lab-based experiments we find that our SVR predictor makes highly accurate forecasts using measurements of queuing and loss, and that available bandwidth measurement does not improve predictions. In heavy traffic conditions, the SVR forecasts are nearly 3 times more accurate than prior HB predictors. We make a series of extensions to the SVR predictor to make it operationally viable. Further lab experiments show that these extensions enable the predictor to work well for a wide range of file sizes, to be robust to measurements from active probe tools, and to adapt to changing path conditions. We created a tool called *PathPerf* which enables us to test our SVR predictor in the Internet. In initial tests in the RON testbed, we show that *PathPerf* generates highly accurate throughput predictions. *PathPerf* also generates far less probe traffic compared to a HB predictor configured as suggested in prior work.

In future work, we intend to continue to refine *PathPerf* by investigating how to better tune the training set, and possibly experiment with other machine learning tools. We are continuing to expand the tests run in the wide area so that we can evaluate *PathPerf's* capability under a broader range of path conditions.

## 9. ACKNOWLEDGEMENTS

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. 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.

## 10. 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*, 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*, 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.