On Exploiting Long Range Dependence of Network Traffic in Measuring Cross Traffic on an End-to-end Basis

Guanghui He and Jennifer C. Hou
Dept. of Computer Science
Univ. of Illinois at Urbana Champaign
Urbana, Illinois 61801
Email: {ghe,jhou}@cs.uiuc.edu

***Abstract*— In this paper we present three theoretically grounded methods: prediction, reconstruction and interpolation, for measuring cross traffic on the bottleneck link of an end-to-end path. The objective is to infer cross traffic as accurately as possible, while not injecting a significant amount of probe packets into the network. In the prediction-based method, we take advantage of the *LRD* characteristic of the cross traffic to predict the future traffic based on the recent information obtained by probe packets. In the reconstruction method, we rebuild the entire cross traffic process with the information obtained by probe packets. In the interpolation method, we periodically send closely-spaced probe packet pairs to sample cross traffic of the bottleneck link, and infer cross traffic between two sampling points using interpolation. The simulation study indicates that (i) the prediction-based and reconstruction methods can give good mean measurement of cross traffic, while the interpolation method usually captures the instantaneous value of cross traffic better; and (ii) all three methods are adaptive to the dynamic change of cross traffic and are quite robust in the presence of multiple bottleneck links on an end-to-end path.**

I. Introduction

To facilitate design and development of better resource management protocols, it will be greatly helpful to better understand the dynamic properties and behavior of end-to-end paths in the Internet. Moreover, not to overload routers with traffic measurement and report tasks, it is more desirable for end hosts to infer these properties on an end-to-end basis. To this end, several end-host-based or edge-based measurement infrastructure projects (such as IPMA [12], NIMI [17], Felix [8], and Surveyor [26]) and academic research projects (that use the one-packet techniques [13], [25], the packet-pair techniques [1], [3], [19], a combination thereof [15], or the multicast-based inference technique [21], [5], [6]) have been proposed to collect and analyze end-to-end measurements between a number of hosts.

The common feature of the above mentioned techniques is to inject one or more unicast/multicast *measurement* packets and measure/record the round-trip time (as in the one-packet techniques), the difference in the arrival times of two consecutive packets (as in the packet-pair techniques), or the pattern of packets received in a multicast group (as in the multicast-based inference technique). The measured information is then used to infer the available bandwidth or the packet loss probability, over links of interest. The measurement results can then be utilized for better resource control. For example, the work in [10] [18] enables rate based congestion control based on the estimate of the attainable throughput inferred at the network edge. Cetinkaya *et al.* [4] and Rubenstein *et al.* [23] develop algorithms to perform admission control and detect flows that are subject to the same congestion points. Most of the above approaches, perhaps except [2], [5] (which is grounded on a maximum likelihood estimation approach) and [15] (which is grounded on rigorous algebraic derivation), are devised based on some simple heuristics and observation.

A number of recent empirical studies of traffic measurements from a variety of working packet networks have convincingly demonstrated that network traffic is self-similar or long-range dependent (*LRD*) in nature [16], [7], [27].This implies the existence of concentrated periods of high activity and low activity (i.e., burstiness) at a wide range of time scales. In the context of end-to-end measurement, this implies, at first glimpse, a large amount of packets have to be sent in order to make accurate measurements at both small and large time scales. A closer investigation actually reveals that the existence of *LRD* implies the existence of nontrivial correlation structures at multiple time scales which can actually be exploited to better infer traffic properties. To this end, a multi-fractal-model based cross traffic estimation algorithm, called *Delphi* algorithm, was proposed in [22]. Based on the multi-fractal model, special temporally-spaced probe packets (called "chirp packet trains") were sent and the cross traffic was inferred based on the information thus obtained. The major advantage of the *Delphi* algorithm is that it requires only a small number of probe packets for end-to-end measurement. However, two measurement errors were induced: first, as it is impossible to fit real traffic perfectly into the multi-fractal model without introducing error; and second, the algorithm proposed in [22] to infer the amount of cross traffic is heuristic-based (although with good theoretical reasoning) and also introduces error. The simulation results reported in [22] shown that the performance of the *Delphi* algorithm depends heavily on the bottleneck link utilization.

In this paper we proposed three theoretically grounded methods that are either prediction-based or interpolation-based to measure cross traffic of the bottleneck link. In the first method, the future traffic is predicted based on recent traffic measurements. Only a fixed number of probe packets are sent at the beginning of every T period and LRD-based prediction is made to infer the cross traffic for the remaining time of the period. In the second method, we attempt to reconstruct the entire cross traffic process based on the information obtained by probe packets. Specifically, we use the information to estimate the power spectral density (PSD) of cross traffic, and use inverse Fourier transform to obtain the estimate of the entire process under the assumption that the cross traffic is statistically stationary. In the third method, we periodically send closely-spaced probe packet pairs to "sample" cross traffic of the bottleneck link. Then we interpolate the process between every two sample points. According to the Nyquist criterion, the entire process can be "reconstructed" as long as the sampling rate is at least 2 times larger than the bandwidth of the signal. By virtue of the existence of *LRD*, the sampling rate can be very low. A finite impulse response (FIR) filter is used to implement the interpolation process under the minimum mean square error criterion.

To assess the three methods with respect to accuracy, packet overhead, robustness, and capability to deal with traffic changes, we conduct *ns-2* simulation on both the dumbbell topology and arbitrary topologies. The simulation study indicates that the prediction-based and reconstruction methods can give good mean measurement of cross traffic, while the interpolation method can, with proper design of the *FIR* filter, capture both the mean and instantaneous values of cross-traffic. All three methods perform well under different bottleneck link utilizations and are adaptive to the dynamic change of cross traffic and are quite robust in the presence of multiple bottleneck links on an end-to-end path.

The rest of the paper is organized as follows. In Section II, we succinctly summarize the notion of LRD, state the assumptions made and describe the pattern of probe packets used in this paper. Then we delve into the detailed description of the three proposed methods in Sections III–IV. Following that, we present simulation results in Section VI and conclude the paper with future work in Section VII.

II. Preliminary

A. Assumptions and probing packets pattern

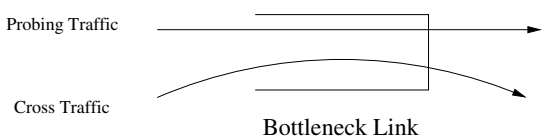


Fig. 1. Single bottle neck link model.

We consider an end-to-end path with a bottleneck link. Closely-spaced probe packets are sent along the path, and the time interval between arrivals of two consecutive packets at the destination is measured and used to infer the cross traffic (Fig. 1). The assumptions made in the paper are: (**A1**) there exists only one bottleneck link on the end-to-end path; (**A2**) packets will not be queued before or after the bottleneck link; and (**A3**) the capacity of the bottleneck link is known. Under these assumptions, the volume of cross traffic entering the queue between two probe packets can be exactly computed. This is because under (**A2**) the time interval between these two probe packets does not change after they leave the bottleneck link until they arrive at the destination. Without (**A2**) (i.e., if the two probe packets experience queuing delay at some queues other than the queue at the bottleneck link), the time interval between arrivals of the two packets at the destination may be stretched or squeezed, leading to underestimate or overestimate of the cross traffic. Although (**A3**) is reasonable[1], the first two assumptions may not hold true in real networks. In Section VI, we will study through *ns-2* simulation the robustness of our methods if these two assumptions do not hold.

Fig. 2. Two back-to-back probe packets.

The temporal pattern of probe packets is shown in Fig. 2. At the sender, two closely-spaced probe packets of length L (called a *probe packet pair*) are sent. Suppose the interval between their sending times is t. If there exists no cross traffic at the bottleneck link, and $t < \frac{L}{C}$, where C is the bottle-neck link capacity, then the dispersion of the arrival times of the two packets $t^{'}$ at the destination satisfies:

$$t^{'} = \frac{L}{C} \tag{1}$$

or

$$C = \frac{L}{t^{'}} \tag{2}$$

Eqs. (1)–(2) are accurate under the condition: $t < \frac{L}{C}$. Since with capacity C, at most L traffic can be served in $\frac{L}{C}$ seconds, before the first packet leaves the queue, the second packet has been queued, the time interval between the arrivals of the two probing packets is exactly $\frac{L}{C}$.

Let the amount of cross traffic that arrive during the time interval $[t_0, t_0 + t]$ be denoted as $f_t(t_0)$, where t_0 is the time instant the first packet of the probe packet pair traverses the bottleneck link. Then the dispersion of the arrival times of the probe packet pair at the destination is:

$$\tau = \frac{L + f_t(t_0)}{C}, \tag{3}$$

or

$$f_t(t_0) = C\tau - L. \tag{4}$$

[1]One can use traceroute to determine the path on which the packets traverse and find out the type each link (T-1, T-3, or OC-n) on the path; alternatively, one may use the end-to-end measurement technique in [15] to infer the bottleneck bandwidth.

Since the time interval τ can be measured exactly at the receiver, with the knowledge of C and L, we can infer the cross traffic that arrive in interval $[t_0, t_0 + t]$.

A naive method to obtain the volume of cross traffic at any time is to send constantly closely-spaced probe packets and measure the dispersion of arrival times of consecutive probe packets. However, since C is usually large and L is small as compared to C, the value of t has to be very small. For example, if $C = 2$Mbps, $L = 1000$ bytes, then $t < 4ms$. That is, a large amount of probe packets have to be sent and the bandwidth of the bottleneck link will be consumed mainly by probe packets. In this paper, we will devise three methods to accurately infer the amount of cross traffic without sending a large amount of probe packets.

B. Long Range Dependency

Let $f(t), t \in Z+$, be a time series representing the instantaneous cross traffic rate (in units of Mbps) measured at some fixed time granularity. By the above definition, $f_t(t_0)$ is the amount of cross traffic that arrives in time interval $[t_0, t_0 + t]$, and $\frac{f_t(t_0)}{t}$ in the rate of cross traffic at time t_0. Suppose we take m samples of $f(t)$ in a time interval of t, then $\frac{f_t(t_0)}{t}$ can be represented as

$$\frac{f_t(t_0)}{t} = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} f(i), \tag{5}$$

where k is used to index the measurement intervals. We call $\frac{f_t(t_0)}{t}$ the aggregated cross traffic at time scale t and denote it as $f_a(t)$. Notice that the average (aggregation) in Eq. (5) can be taken at any time scale, and hence we can further aggregate $f_a(t)$ to obtain aggregation at some larger time scale.

Let $R(k)$ and $R^m(k)$ denote the autocorrelation functions of $f(t)$ and $f_a(t)$, respectively. The time series $f(t)$ is (asymptotically second-order) self-similar, if the following conditions hold:

$$R^m(k) \sim R(k), \tag{6}$$

$$R(k) \sim \text{const} \cdot k^{-\beta}, \tag{7}$$

$$\tag{8}$$

for large values of k and m, where $0 < \beta < 1$. That is, $f(t)$ is self-similar in the sense that the correlation structure is preserved with respect to time aggregation (Eq. (6)) and $R(k)$ behaves hyperbolically with $\sum_{k=0}^{\infty} R(k) = \infty$ (Eq. (7)). The former property (Eq. (6)) is also referred to as long range dependency (LRD). The proposed methods take advantage of this LRD characteristic to infer cross traffic without incurring excessive probing overhead. Succinctly, if we can obtain the estimate of the autocorrelation structure of cross traffic at some time scale, we can obtain the autocorrelation structure (and infer cross traffic) at any time scale.

III. The Prediction-Based Method

In the prediction-based method, the sender sends $n + 1$ closely-spaced probe packets (with the temporal distance between the sending times of two consecutive packets being t). The destination measures the inter-arrival times of the $n + 1$ probe packets, and obtains n samples of τ. The time series $f_t(t)$ with length n can then be constructed using Eq. (4), and furthermore the aggregated time series $f_a(k), k = 1, 2, ...n$ can be obtained by dividing $f_t(t)$ by t.

Based on these aggregate series samples, we predict the future cross traffic using the Linear Minimum Mean Square Error (*LMMSE*) estimator. Specifically, given the series $f_a(k), k = 1, \ldots, n$, the aggregate cross traffic series in the next time interval t, $f_a(n+1)$, can be expressed as a weighted linear combination of the past n samples, where the weights are determined to minimize the mean square error. That is, the estimate (written as $\hat{f}_a(n+1)$) of $f_a(n+1)$ is expressed as

$$\hat{f}_a(n+1) = \begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} \begin{bmatrix} f_a(1) \\ f_a(2) \\ ... \\ f_a(n) \end{bmatrix}, \tag{9}$$

where $a_1, a_2, ..., a_n$ are the *LMMSE* coefficients and can be expressed as

$$\begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} = \begin{bmatrix} R(n) & R(n-1) & ... & R(1) \end{bmatrix} \times \begin{bmatrix} R(0) & R(1) & ... & R(n-1) \\ R(1) & R(0) & ... & R(n-2) \\ ... & ... & ... & ... \\ R(n-1) & R(n-2) & ... & R(0) \end{bmatrix}^{-1} \tag{10}$$

where $R(n)$ is the covariance function of the time series, and can be estimated in practice as

$$R(i) \cong R^{(m)}(i) = \frac{1}{n} \sum_{t=i+1}^{n} f_a(t) f_a(t-i), \tag{11}$$

Sang *et al.* [24] has shown that the above estimator can make very good prediction of $\hat{f}_a(n+1)$. After $\hat{f}_a(n+1)$ is predicted, $\hat{f}_a(n+2)$ can be predicted by using $f_a(k), k = 1, 2, ..., n$ and $\hat{f}_a(n+1)$, $\hat{f}_a(n+3)$ can be predicted by using $f_a(k), k = 1, 2, ..., n$, $\hat{f}_a(n+1)$ and $\hat{f}_a(n+2)$, and so on. The process continues until N predictions are made.

Determination of tunable parameters:: There are two tunable parameters that we need to determine: one is the number, n, of closely-spaced packets that are sent at the beginning, and the other is the number, N, of predictions that can be made before prediction accuracy is impaired as a result of accumulated prediction errors in each step.

To determine the value of n, we have conducted *ns-2* simulation in which traffic traces are generated on a dumbbell topology and on arbitrary network topologies and the above LMMSE-based approach is used to predict the future traffic. Table I gives the relative prediction error for different values of n under the dumbbell topology given in Fig. 8, where

n	5	10	15	20	30	40	50
$err(\%)$	14.2	11.5	9.3	7.8	7.7	7.7	7.6

TABLE I

Relative prediction error for different values of n.

N	n	$2n$	$3n$	$4n$	$5n$	$6n$	$8n$
$err(\%)$	7.8	8.5	9.1	10.2	11.8	15.4	19.8

TABLE II

RELATIVE PREDICTION ERROR FOR DIFFERENT VALUES OF N.

the number of TCP connections varies from 10 to 100, and the relative error is defined as $\frac{|\hat{f}(t)-f(t)|}{f(t)}$. (The result under arbitrary topologies are similar and hence omitted.) As shown in Table I, the larger the value of n, the smaller the relative error (i.e., the more accurate the prediction result). However, the performance improvement levels off as n exceeds 20. This is due to the fact that $R(\tau)$ decreases quite dramatically, and hence adding more history information can not further improve the prediction accuracy. The interested reader is referred to [11] for a detailed account of experiment setups and discussion on the findings. In the simulation study we set $n = 20$.

To determine the value of N, we also conduct the same set of experiments (with n set to 20) to study the effect of varying the value of N on the prediction accuracy. The experiment results are shown in Table II. When N grows beyond $5n$ the relative prediction error is more than 15%. Hence, in the simulation study, we set $N = 5n$, i.e., the sender sends $n+1$ closely-spaced probe packets at the beginning and infers the amount of cross-traffic in the next $5n \times t$ period. After that, the sender sends another $n+1$ probe packets, and the entire process repeats.

IV. THE RECONSTRUCTION METHOD

Similar to the traffic prediction method, the sender sends $n+1$ closely-spaced probe packets, and obtains the time series $f_a(k), k = 1, 2, ..., n$. The time series is then used to reconstruct the *entire* process under the assumption that the amount of cross traffic is statistically stationary. Conceptually, we obtain the autocorrelation function $R^m(k)$ of $f_a(k)$ using Eq. (11), and estimate the power spectral density, $p(s)$, of $f_a(k)$, i.e., the *Fourier* transform of $R^m(k)$. Since the power spectral density is the square of the *Fourier* transform of the original time series, $f_a(k)$ can be obtained in principle using the inverse *Fourier* transform of the square root of $p(s)$.

To practically implement this method, we consider two issues: (i) how to get the estimate of the autocorrelation function $R(k)$; and (ii) how to reconstruct the process $f_a(k)$. We leverage the method by Davis *et al.* [14]. Succinctly, given a Gaussian, zero-mean time series of length n with autocorrelation function $\gamma(i), i = 0, 1, ..., n-1$,[2], we perform the following operations:

1) Define the finite *Fourier* transform g_k of the sequence $\gamma(0), \gamma(1), ..., \gamma(n-2), \gamma(n-1), \gamma(n-2), ..., \gamma(1)$ as follows. Let
$$\lambda_k \triangleq \frac{2\pi(k-1)}{(2n-2)}, \tag{12}$$
for $k = 1, 2, ..., 2n-2$, and
$$R(i) \triangleq \begin{cases} \gamma(i), & i = 0, 1, ..., n-1, \\ \gamma(2n-2-i), & i = n, n+1, ..., 2n-3. \end{cases} \tag{13}$$
Note that for any real WSS (wide sense stationary) random process, its autocorrelation function is even, and we use the one sided autocorrelation function $\gamma(i)$ to generate $R(i)$. Then, the finite *Fourier* transform g_k is defined as
$$g_k = \sum_{j=0}^{2n-3} R(j) \cdot e^{ij\lambda_k}, k = 1, 2, ..., 2n-2. \tag{14}$$

2) Generate two independent series of zero mean normal random variables, $U_1, U_2, ..., U_n$ and $V_2, ..., V_{n-1}$, such that $var(U_1) = var(U_n) = 2$ and for $k \neq 1, n$, $var(U_k) = var(V_k) = 1$. Let $V_1 = V_n = 0$ and define complex random variables Z_k as:
$$Z_k = \begin{cases} U_k + iV_k, & k = 1, 2, ..., n \\ U_{2n-k} - iV_{2n-k}, & k = n+1, ..., 2n-2. \end{cases} \tag{15}$$

3) For $t = 1, 2, ..., n$, define
$$X_t = \frac{1}{2\sqrt{n-1}} \sum_{k=1}^{2n-2} \sqrt{g_k} e^{i(t-1)\lambda_k} Z_k. \tag{16}$$

X_t corresponds to the time series, $f_a(k)$, we would like to reconstruct. The rationale behind constructing a complex random variables Z_k is as follows. For any real WSS random process, its autocorrelation function is a deterministic function of time lag, τ, and the corresponding Fourier transform function g_k is also deterministic (due to the fact that $R(\tau)$ is even). The process is not reversible, i.e., after taking the square root of g_k, we obtain a deterministic signal, whose inverse Fourier transform is also deterministic (i.e., the recovered signal is no longer random). To reconstruct a random variable we have to rely on another random variable with the uniform distribution. Here, Z_k plays this role. Multiplying $\sqrt{g_k}$ with Z_k with is equivalent to convoluting $r(\tau)$ with a constant (due to the fact that Z_k is white). In this way, we introduce randomness to the reconstructed process while keeping the autocorrelation structure unchanged.

Note that in step 3 a n-point inverse *Fourier* transform is performed to reconstruct the original process. Since the original process exhibits self similarity, its autocorrelation structure is asymptotically the same at different time scales (Eqs. (7)–(6)) and we can envision the reconstructed process as the amount of aggregated cross traffic at any time scale. For example, if we envision the reconstructed process as an aggregated process at the time scale of $2t$, we can get the estimate in the period of $2nt$. In theory as long as the process is self similar, the sender needs only to send $n+1$ probe packets and can obtain estimates in the entire time domain. However,

[2]Under the stationary assumption, one can transform an arbitrary process into a process with zero mean by subtracting the mean from the process, and adding the mean back at the end of this algorithm.

N	n	$2n$	$3n$	$4n$	$5n$	$6n$	$8n$
$err(\%)$	8.7	9.2	9.7	10.6	11.3	13.4	15.1

TABLE III

RELATIVE RECONSTRUCTION ERROR FOR DIFFERENT VALUES OF N.

in reality since the process is not strictly self-similar, and the autocorrelation structure Estimated based on the $n+1$ probe packets may introduce estimate error, the amount of cross-traffic can not be estimated at arbitrarily large time scales. In other words, $n+1$ probe packets have to be sent every Nt period, and similar to the traffic prediction method, the values of both n and N have to be determined.

Determination of tunable parameters:: We have conducted the same set of *ns-2* simulation runs as in Section III to study the effect of varying the value of n on the reconstruction error. The result is similar to that obtained in the traffic prediction method, i.e., as n goes beyond 20, the reduction in the reconstruction error levels off. This is not a coincidence, as both methods depend heavily on the autocorrelation structure of the time series. Henceforth, in the simulation study we set $n = 20$.

To determine the value of N, we have to consider two issues: accuracy and computation complexity. Again we conduct the same set of experiments as in Section III. As shown in Table III, the larger the value of N, the more pronounced the reconstruction error but the lower the computation complexity ($n+1$ probe packets per Nt time units). In the simulation study we set $N = 4n$, partially due to the facts that the reconstruction error is $\leq 10\%$ and that the factor $4 = 2^2$ facilitates application of fast Fourier transform algorithm.

V. THE INTERPOLATION-BASED METHOD

Both the traffic prediction and reconstruction methods require periodic sending of $n+1$ closely-spaced probe packets. The computational complexity in the traffic reconstruction method is also non-negligible, although fast Fourier transform algorithm can be used. In addition, both methods can only capture the mean value of the time series of interest. In this section, we propose an interpolation-based method that requires a smaller number of probe packets and yet gives better estimates. In what follows, we first give an overview of this method, and then delve into the discussion of implementation details.

A. Overview

According to the *Nyquist* criterion, a signal can be reconstructed as long as the rate at which the signal is sampled is at least twice as large as the bandwidth of the signal. As the cross traffic exhibits long-range dependency, we have $\sum_{k=0}^{\infty} R(k) = \infty$, and hence the power spectral density $p(s) \to \infty$ as $s \to 0$. In other words, the power spectrum of the cross-traffic has the $\frac{1}{f}$ property, i.e., the cross traffic has a much narrower bandwidth (and hence requires a much smaller sampling rate) as compared to traditional Gaussian white noise.

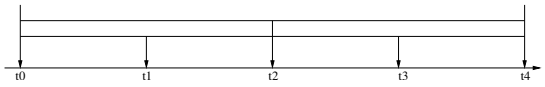


Fig. 3. Use of the cross traffic information obtained at t_0 and t_4 to interpolate the cross traffic at the three middle points t_1, t_2, and t_3.

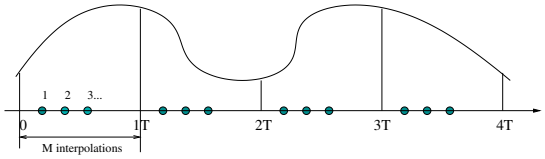


Fig. 4. The input signal to the FIR filter with m interpolated points.

By Eq. (7), the power spectral density satisfies

$$p(s) \sim \text{const} \cdot s^{\beta-1}, \tag{17}$$

where $0 < \beta < 1$ is some constant and is related to the Hurst parameter through $H = 1 - \frac{\beta}{2}$. As the typical value of the Hurst parameter for Internet traffic is $H = 0.8$, we have $\beta = 0.4$. Since $p(s)$ goes to infinity at $s = 0$, we can not calculate the $3dB$ bandwidth as usual. Instead, we calculate the bandwidth when $p(s)$ drops to $\frac{1}{2} \cdot const$ (denoted as B), where $const$ is the constant defined in Eq. (17). Note that B is much larger than $3dB$ bandwidth. By Eq. (17), we have $B = 3.2$, i.e., if we sample the original signal at the rate $r \geq 6.4$ or the sampling interval $T \leq 156ms$, we can reconstruct the original signal without incurring error. (We will demonstrate later that T can be even larger, due to the self-similarity of the signal.)

In the end-to-end measurement problem considered, the signal (the amount of cross traffic on the bottleneck link) is sampled by having the sender send a pair of closely-spaced probe packets every T seconds. Each pair of such packets gives one sample of aggregated cross traffic $f_a(t)$ at time t. Then we employ the interpolation-based method to rebuild the time series. Consider, for example, Fig. 3: the sender sends a pair of probe packets at time instants t_0 and t_4 ($T = t_4 - t_0$). Then the information obtained at t_0 and t_4 is used to interpolate the cross traffic at the middle point between t_0 and t_4. Similarly, after the information at t_2 is obtained, it is used (along with information obtained at t_0 and t_4) to interpolate the cross traffic at time instants t_1 and t_3. This process repeats recursively until a desirable time granularity is reached.

There are two issues that must be considered in order to implement the interpolation-based method: (i) how to determine the value of the sampling cycle T; and (ii) how to interpolate the amount of cross traffic with the cross traffic information available at the two endpoints. We will elaborate on the second issue in the next subsection, and defer the discussion of the first issue to Section VI.

B. Implementation of the Interpolation-based Method

Fig. 3 demonstrates that one can interpolate, with the use of the cross traffic information obtained at the two endpoints, the

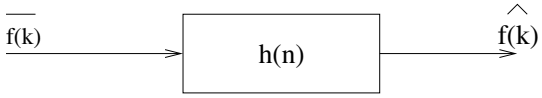


Fig. 5. The FIR filter.

amount of cross traffic at the middle point of an interval T. As a matter of fact, by designing an appropriate finite impulse response (*FIR*) filter one can interpolate the amount of cross traffic at M points that are evenly distributed in an interval T. Specifically, let $f_a(k)$ be the cross traffic in the discrete time domain as defined before. With the sampled values of $f_a(k)$ at time T, $2T$, ..., we would like to interpolate the amount of cross traffic at the M evenly-spaced points between any two sampled values.

The interpolation procedure with the use of *FIR* filters is as follows. As shown in Fig. 4, we first set the values at the M points in each interval T to be 0, and pass the sequence, $\overline{f}_a(k)$, with M zero values inserted, to the *FIR* filter. The output of the *FIR* filter, $\hat{f}_a(k)$, is the interpolated signal (Fig. 5). The impulse response of the filter $h(n)$ is designed to have the following properties:

1) $h(n)$ is a time series of length $2M+1$.
2) $h(-i) = h(i) = a_i, 1 \le i \le M$, and $h(0) = 1$.

Since $\hat{f}_a(k) = \overline{f}_a(k) \star h(k)$, where $\star$ is the convolution operation, it is straightforward to see that $\hat{f}_a(iT) = \overline{f}_a(iT), i = 0, 1, 2, ...$ This is desirable, because the output should be the same as the input at sampling points. The symmetric design of $h(n)$, on the other hand, guarantees that the FIR filter has the linear phase and hence a constant time delay can be guaranteed.

Next we need to determine the values of a_i's in order to achieve the minimum mean square error. By the definition of convolution, the i-th interpolated value can be represented as:

$$\hat{f}_a(k+i) = a_{M-i+1}\overline{f}(k) + a_i\overline{f}(M+k+1), i = 1, 2, ..., M. \quad (18)$$

To fulfill the minimum mean square error criterion, we have

$$E((f_a(k+i) - \hat{f}_a(k+i)) \cdot \hat{f}_a(k+i)) = 0. \quad (19)$$

After some algebraic operations, Eq. (19) gives

$$a^2_{M-i+1}R(0) + a_i^2R(0) + 2a_ia_{M-i+1}R(M+1) - a_{M-i+1}R(i) - a_iR(M+1-i) = 0, i = 1, 2, ..., M \quad (20)$$

where $R(k)$ is the autocorrelation function of $f_a(k)$. To determine the values of a_i's, we have to estimate the autocorrelation structure $R(k)$ of the cross traffic $f_a(k)$. To this end, in the interpolation-based method we enable the sender to send $n+1$ closely-spaced probe packets to get n samples initially, so that we can estimate $R(k)$ using Eq. (11). Since only $R(k)$, $0 \le k \le M$, are needed, and M is usually small (≤ 5), we can get accurate estimates of $R(k)$, even if n is not large. In contrast to the prediction-based and reconstruction methods, the interpolation-based method does not require that the sender sends periodically $n+1$ probe packets. After the sender sends $n+1$ probe packets initially, it needs only to send one pair of probe packets every T seconds.

Note that Eq. (20) contains M equations with M unknowns. However, because the coefficients are symmetric, the M equations are not independent, and the M coefficients cannot be uniquely determined. In order to determine the values of the M coefficients, we have to introduce some other relation among the coefficients. (We will discuss this further below.) Next we consider two special cases: $M = 1$ and $M = 2$.

$M = 1$:: The impulse response of the filter $h(n)$ has the following property: (i) $h(n)$ is a time series of length 3; and (ii) $h(-1) = h(1) = \alpha$, and $h(0) = 1$. What is left to determine is $\alpha = h(1)$. By Eq. (18), we have

$$\begin{aligned} \hat{f}_a(k) &= \alpha\overline{f}_a(k-1) + \alpha\overline{f}_a(k+1) \\ &= \alpha f_a(k-1) + \alpha f_a(k+1). \end{aligned} \quad (21)$$

Also, to achieve minimum mean square error, $f_a(k) - \hat{f}_a(k)$ should be perpendicular to $\hat{f}_a(k)$, i.e.,

$$E((f_a(k) - \hat{f}_a(k)) \cdot \hat{f}_a(k)) = 0, \quad (22)$$

or,

$$\alpha^2R(0) + \alpha^2R(2) - \alpha R(1) = 0. \quad (23)$$

Hence,

$$\alpha = \frac{R(1)}{R(0) + R(2)}. \quad (24)$$

$M = 2$:: The impulse response of the filter $h(n)$ is a time series of length 5 and with two unknown coefficients. Let $a = h(2) = h(-2)$ and $b = h(1) = h(-1)$ denote the two coefficients of the *FIR* filter yet to be determined. Using the similar technique as above we obtain

$$a^2R(0) + b^2R(0) + 2abR(3) - aR(2) - bR(1) = 0. \quad (25)$$

Due to the fact that coefficients are symmetric, the other equation is the same as Eq. (25). Hence, we have to add another condition. To enforce a linear decrease in the coefficients, we set $b = \frac{1+a}{2}$. Other relations between a and b are also possible. For example, we may enforce the coefficients to decrease exponentially from 1 to a. We have conducted simulations to investigate the impact of the additional condition on the performance, and found that in all the simulation runs the performance is rather insensitive to the condition as long as b is larger than a. Thus, we choose $b = \frac{1+a}{2}$ so as to obtain closed form results:

$$a = \frac{-c_2 + \sqrt{c_2^2 - 4c_1c_3}}{2c_1}, \quad (26)$$

Where $c_1 = 5R(0) + 4R(3)$, $c_2 = 2R(0) + 4R(3) - 4R(2) - 2R(1)$, and $c_3 = R(0) - 2R(1)$.

Discussion:: Note that the choice of M represents a trade-off between the ability to interpolate the signal at more points between the two given samples, the accuracy of the interpolation results, and the complexity of the resulting *FIR* filter. As shown in Fig. 6, if the interpolation is performed at one point ($M = 1$), 4 pairs of probe packets are needed; on the other hand, if the amount of cross traffic is interpolated at two points between the two given samples ($M = 2$), only 3

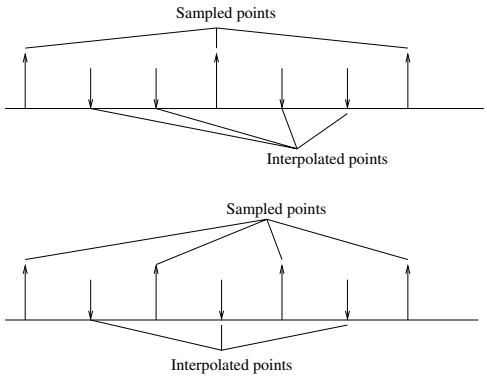


Fig. 6. An example that shows the difference between interpolating the signal at 1 or 2 points between the two given samples.

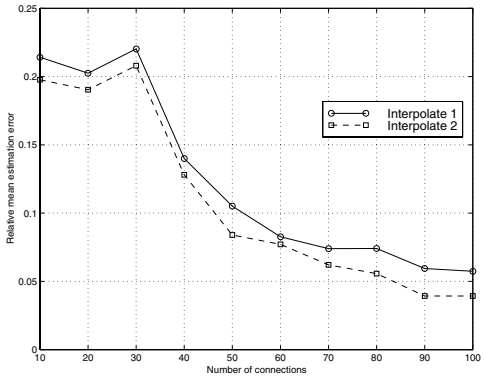


Fig. 7. The mean interpolation error in the cases of $M = 1$ and $M = 2$.

pairs of probe packets need to be sent. However, the saving in the number of probe packets does not come without a cost. The *FIR* filter for $M = 2$ is more complex.

We have conducted the same set of *ns-2* simulation runs as in Section III to study the effect of varying the value of M on the interpolation error. We have also implemented the FIR filter in *Matlab* for both $M = 1$ and $M = 2$. Fig. 7 gives the relative mean interpolation error. Contrary to our intuition, the interpolation error is smaller under the case of $M = 2$. This can be explained as follows. In the case of $M = 2$, the first interpolated value(and so is the second interpolated value), I, is $\frac{2T}{3}$ time units away from the right sample A, and $\frac{T}{3}$ time units away from the left sample B, and I is calculated as $I = a \cdot A + b \cdot B$, where $a < b$, a and b are the parameters of the FIR filter in Eq. (25). As B is temporally closer to I, the estimate of I is more accurate by giving more weight to B. In contrast, in the case of $M = 1$, equal weights are assigned to both the two samples, A and B, that are $\frac{T}{2}$ time units away. One should, however, not generalize on this result, i.e., it may not be true that the larger the value of M, the more accurate the interpolation results will be. This is because as M increases, the design of the FIR filter becomes more difficult, and the interpolation results will be very sensitive to the coefficients. In the simulation study, we implement FIR filters with $M = 1$ and $M = 2$.

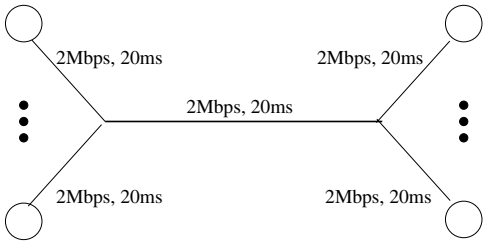


Fig. 8. The single-bottleneck dumbbell topology used in the simulation.

VI. Simulation Results

We have implemented the prediction-based, reconstruction, and interpolation-based methods in *ns-2* and conducted a simulation study to validate the proposed design and compare the performance. The performance metrics of interest are (i) relative mean error $err = \frac{\hat{f}(t)-f(t)}{f(t)}$, (ii) standard deviation of the error std, (iii) ability of adapting to the changes of traffic load on the bottleneck link, and (iv) robustness in the case that some of the assumptions (**A2**) are relaxed.

We examine the behavior of these methods under a variety of network topologies and traffic sources. In particular, we have considered the network topologies with a single bottleneck link, with multiple bottleneck links, as well as arbitrary topologies. The maximum buffer size of each router is set to 100 packets (each of size 1000 bytes). The interval between two consecutive, back-to-back probe packets is set to be $0.005s$. We have used an assortment of traffic sources (e.g., TCP sources that generate packets according to the on-off model or real traffic traces down-loaded from the Internet, and constant bit rate UDP sources) as the sources of cross traffic. Each data point is the result averaged over 10 simulation runs, and each simulation run lasts for 50 seconds. Due to the space limitation, we only report on a small set of the simulations which we believe is the most representative. In spite of numerous system parameters involved, the results are found to be quite robust in the sense that the conclusion drawn from the performance curves for a representative set of parameter values (reported below) is valid over a wide range of parameter values.

A. Experiment 1: Performance under Different Link Utilizations

In the first set of experiments, we evaluate the performance of the three methods under different link utilizations. We also study the effect of varying the value of T (the interval between two samples) on the performance of the interpolation-based method.

The network topology used (along with all the relevant network parameters) is shown in Fig. 8. The cross traffic on the bottleneck link is made up of 30-100 UDP/TCP connections, with the left-hand-side (right-hand-side) hosts being the sources (destinations), and with $Pareto$ on-off models (with the shape parameter $\alpha = 1.5$) being the traffic generation

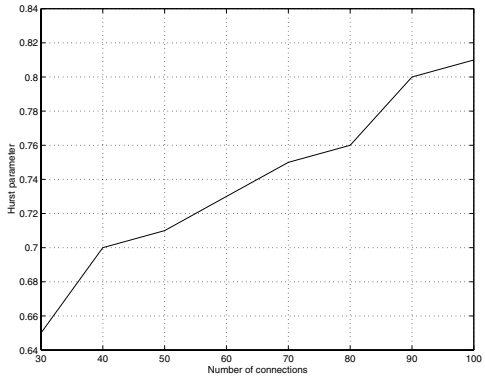


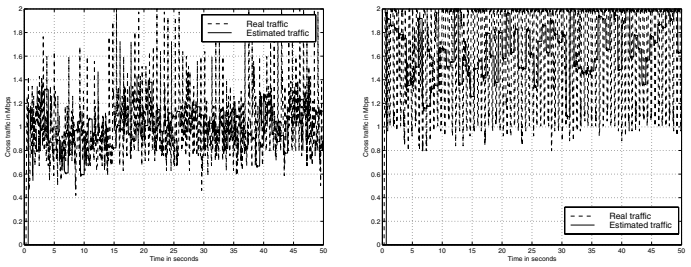
Fig. 9. The Hurst parameter of the cross traffic in Experiment 1.

models. The end-to-end measurement is performed by another source-destination pair that send probe packets in compliance with the method under consideration. As shown in Fig. 9, the *Hurst* parameter H of the cross traffic ranges from 0.65 to 0.81, as the number of connections increases from 30 to 100, indicating that the cross traffic does exhibit the LRD characteristics. If there were no cross traffic, the interval t should satisfy: $t \leq \frac{L}{C} = \frac{8000}{2000000} = 0.004s$. In the presence of cross traffic, t can be set to be a little larger. We keep track of the cross traffic and its estimate.

In the prediction-based method, 21 back-to-back probe packets are sent at the beginning of each interval of length 0.6 second (120 times t). Since it takes $20 \times 0.005 = 0.1s$ to send 21 back-to-back probe packets, the prediction-based method estimates the amount of cross traffic for the remaining 0.5 second. As mentioned in Section III, this amount of time is equal $5nt = 100 \times 0.005$. In the reconstruction method, again 21 back-to-back probe packets are sent at the beginning of each interval of length 0.5 second. An 80-point inverse *Fourier* transform is performed to reconstruct the original cross traffic process in an time interval of of length $4nT = 0.4s$. In the interpolation-based method, 21 back-to-back probe packets are sent initially. Following that, in every T seconds 2 back-to-back probe packets are sent. The amount of cross traffic in the T interval is estimated by interpolation. Note that the choice of T will have an impact on the performance of the interpolation-based method, as there exists a trade-off: the smaller the value of T, the smaller the relative mean error; however, as more probe packets have to be sent, the larger the standard deviation of the error. We have experimented with different values of T, and will show below the results in the cases of $T = 0.05$ s and $T = 0.5$ s.

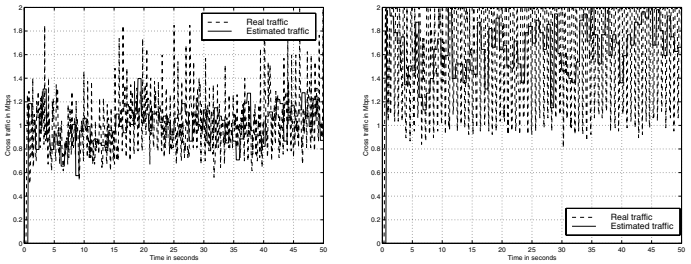
Figs. 10–12 give the simulation results under the prediction-based, reconstruction, and interpolation-based methods, respectively, in the existence of 30 and 50 sources of cross traffic. Tables IV–VI give err and std under the three methods. Several observations are in order:

- As shown in Figs. 10–11, both the prediction-based and reconstruction methods capture the mean value of the cross traffic very well. On the other hand, the estimates



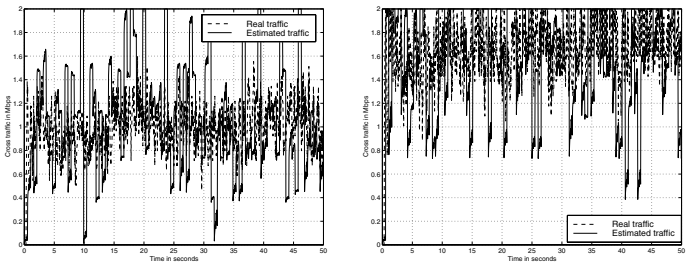
(a) 30 sources (b) 50 sources of cross traffic

Fig. 10. Cross traffic estimated using the prediction-based method versus actual traffic in the existence of 30 and 50 sources of cross traffic.



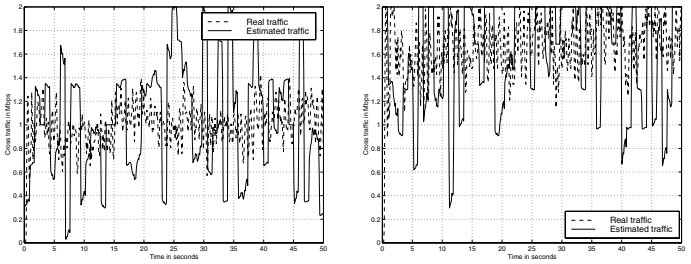
(a) 30 sources (b) 50 sources

Fig. 11. Cross traffic estimated using the reconstruction method versus actual traffic in the existence of 30 and 50 sources of cross traffic.



(a) 30 sources (b) 50 sources

Fig. 12. Cross traffic estimated using the interpolation-based method (with $T = 0.05$) versus actual traffic in the existence of 30 and 50 sources.



(a) 30 sources (b) 50 sources

Fig. 13. Cross traffic estimated using the Interpolation-based method (with $T = 0.5$) versus actual traffic in the cases that 30 and 50 sources of cross traffic are present.

# sources	30	40	50	60	70	80	90
Link util.	0.39	0.56	0.78	0.87	0.89	0.90	0.91
err	0.06	0.02	0.04	0.08	0.06	0.05	0.05
std	0.17	0.21	0.24	0.22	0.21	0.19	0.18

TABLE IV

RELATIVE MEAN ERROR AND THE STANDARD DEVIATION OF ERRORS UNDER THE PREDICTION-BASED METHOD.

# sources	30	40	50	60	70	80	90
Link util.	0.39	0.56	0.78	0.87	0.89	0.90	0.91
err	0.02	0.04	0.03	0.06	0.09	0.08	0.07
std	0.13	0.19	0.22	0.20	0.19	0.18	0.17

TABLE V

RELATIVE MEAN ERROR AND THE STANDARD DEVIATION OF ERRORS UNDER THE RECONSTRUCTION METHOD.

obtained using the interpolation method oscillate around the actual values. This is because the interpolation-based method depends heavily on the instantaneous sampled values of cross traffic, as the interpolated values are linear combinations of sampled values. As a result, in addition to keeping track of the mean value of the actual amount of cross traffic, the interpolation-based method also attempts to capture the instantaneous value of the actual traffic. This is especially true for small values of T.

- As shown in Figs. 12–13, the interpolation result oscillates much less significantly when T is large (e.g.,$T = 0.5$) than when T is small ($T = 0.05$). This is because when T is large, the *FIR* filter filters out detailed information within the interval of length T and gives much smoother results.
- All three methods perform well under different utilizations in terms of relative mean error.

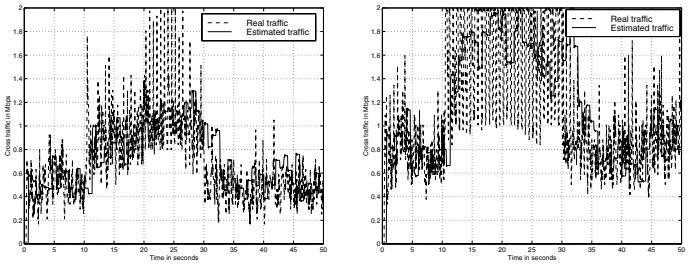
B. Experiment 2: Adaptability to Traffic Load Changes on the Bottleneck Link

Recall that In the first set of experiments all the connections that comprise the cross traffic last for 50 seconds, i.e., the amount of cross traffic does not change throughout the simulation. To evaluate the three methods in terms of their adaptability to the changes in the amount of cross traffic, we repeat the same experiments but vary the number of *effective* connections that comprise the cross traffic as follows. At time 0, $\frac{N}{2}$ cross-traffic connections commence, at time 10 s, another $\frac{N}{2}$ connections commence, and at time 30 s, $\frac{N}{2}$ connections terminate, where N varies from 30 to 100. All simulation runs last for 50 s. Figs. 14–16 give the corresponding simulation results. As shown in Figs. 14–16, all the three methods perform well and capture the changes in the amount of cross traffic almost immediately.

# sources	30	40	50	60	70	80	90
Link util.	0.39	0.56	0.78	0.87	0.89	0.90	0.91
$err_{T=0.05}$	0.07	0.05	0.01	0.05	0.04	0.01	0.02
$std_{T=0.05}$	0.24	0.27	0.25	0.19	0.16	0.12	0.11
$err_{T=0.5}$	0.09	0.08	0.04	0.07	0.09	0.08	0.07
$std_{T=0.5}$	0.18	0.18	0.17	0.14	0.12	0.09	0.08

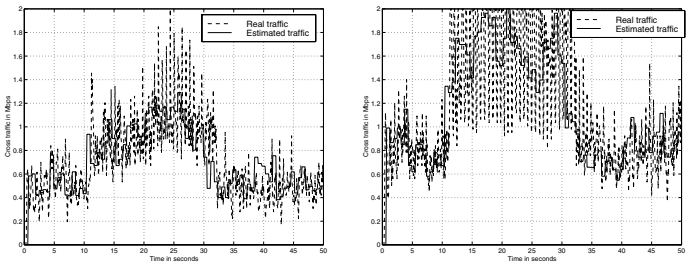
TABLE VI

RELATIVE MEAN ERROR AND THE STANDARD DEVIATION OF ERRORS UNDER THE INTERPOLATION-BASED METHOD WITH $T = 0.05$ AND $T = 0.5$.



(a) 30 sources (b) 50 sources

Fig. 14. Cross traffic estimated using the prediction-based method versus actual traffic in the case that the amount of cross traffic dynamically changes.

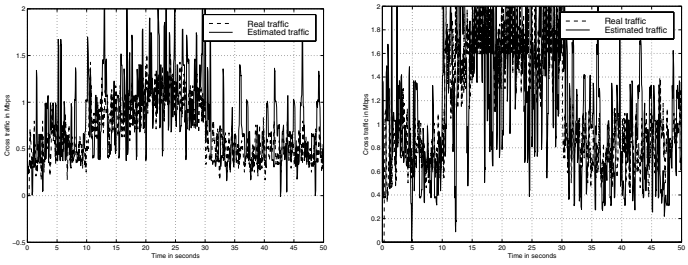


(a) 30 sources (b) 50 sources

Fig. 15. Cross traffic estimated using the reconstruction method versus actual traffic in the case that the amount of cross traffic dynamically changes.

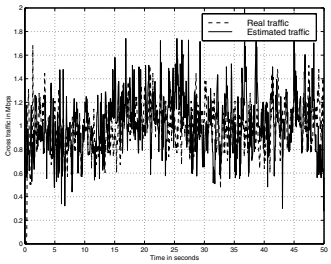
C. Experiment 3: Effect of Varying the Number of Interpolated Values on Performance

In this set of experiments, we study the effect of varying the number, M, of interpolated values between two samples in the

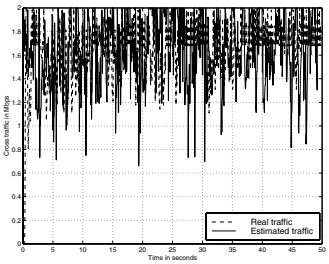


(a) 30 sources (b) 50 sources

Fig. 16. Cross traffic estimated using the interpolation-based method (with $T = 0.05$ s) versus actual traffic in the case that the amount of cross traffic dynamically changes.

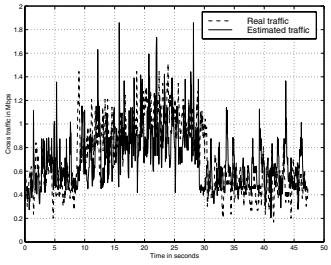


(a) 30 sources

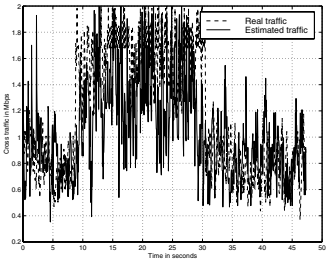


(b) 50 sources

Fig. 17. Cross traffic estimated using the interpolation-based method (with $T = 0.05$ and $M = 2$ versus actual traffic in the existence of 30 and 50 traffic sources.

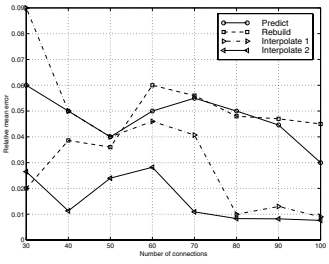


(a) 30 sources

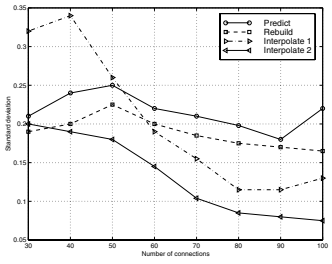


(b) 50 sources

Fig. 18. Cross traffic estimated using the interpolation-based method (with $T = 0.05$ s and $M = 2$) versus actual traffic in the case that the amount of cross traffic dynamically changes.



(a) Relative mean error



(b) Std dev of error

Fig. 19. The relative mean error and standard deviation of the error under the prediction-based method, the reconstruction method, and the interpolation-based methods with $M = 1$ and $M = 2$.

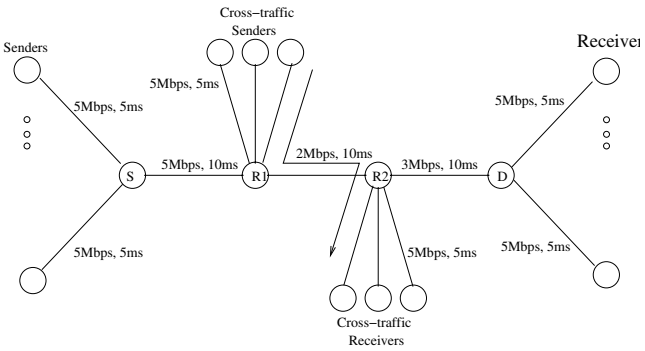


Fig. 20. The topology with multiple bottleneck links.

interpolation-based method. We repeat the same experiments as in Sections VI-A–VI-B but use the interpolation-based method with $M = 2$ for inferring the amount of cross traffic (set $T = 0.05s$). Figs. 17–18 give the corresponding simulation results. It turns out that the Interpolation-based method with $M = 2$ can capture not only the real mean value, but also the instantaneous value very well. Also, it incurs smaller values of err and std. To better illustrate this, we depict err and std under the prediction-based method, the reconstruction method, and the interpolation-based methods with $M = 1$ and $M = 2$ in Fig. 19. From the figure we can draw a conclusion: if we want to keep track of both the mean and the instantaneous value of the cross traffic while achieve smaller err and std, interpolation with $M = 2$,$T = 0.05s$ is the choice.

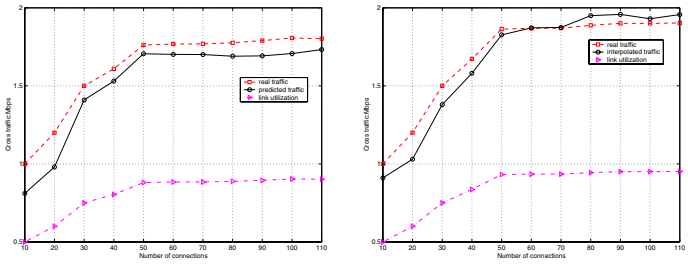
D. Experiment 4: Performance with Respect to Robustness

As mentioned in Section II-A, we assume that probe packets only experience queuing at the bottleneck link. In real networks, this may not be true. In this set of experiments, we study the robustness of the three methods by relaxing the first two assumptions ((**A1**) and (**A2**)) in Section II-A. The simulation scenario is depicted in Fig. 20: link $R1 - R2$ is the bottleneck link, but probe packets may also experience queuing at links $S - R1$ and $R2 - D$. The bandwidths of the links are, respectively, $B_{R1-R2} = 2Mbps$, $B_{S-R1} = 5Mbps$, $B_{R2-D} = 3Mbps$, and $5Mbps$ for all the other links. Cross traffic is generated by the left-hand-side hosts (labeled as senders), and traverses links $S - R1$, $R1 - R2$, and $R2 - D$. In addition, another additional 2-10 cross traffic connections are generated by hosts attached to $R1$ (labeled as cross-traffic senders) and traverse link $R1 - R2$.

As probe packets may get queued before or after link $R1 - R2$, the inter-arrival time τ between two back-to-back probe packets at the destination will be stretched/squeezed so that Eq. (3) does not hold. Specifically, if the second probe packet is queued before it arrives at link $R1 - R2$, when it arrives at link $R1 - R2$ the first probe packet may have already left. As a result, τ may be stretched, and the three methods will underestimate the amount of cross traffic that traverse link $R1 - R2$. Conversely, if the first probe packet is queued after it leaves link $R1 - R2$, τ may be squeezed, and the three methods will overestimate the amount of cross traffic.

Table VII gives the mean error err and std under the three methods in the case that probe packets may be queued before/after the bottleneck link. As compared to Talbes IV–VI, the relative mean error in this scenario is only slightly larger (about 5%) than that in the idealistic case, while std is almost the same under both cases. This suggests that the three methods are pretty robust in the case that (**A2**) does not hold. In the figure to be presented below, we will see that the increase in err results from the effect of underestimating or overestimating the amount of cross traffic on the bottleneck link.

Fig. 21 gives the estimated and real mean values of cross



(a) Prediction & Reconst. (b) Interpolation-based

Fig. 21. Estimated and real mean values of cross traffic under the prediction-based, reconstruction, and interpolation-based methods.

# sources	30	40	50	60	70	80	90
link util.	0.41	0.62	0.82	0.88	0.92	0.93	0.93
Pred.:err	0.12	0.08	0.08	0.07	0.09	0.08	0.09
Pred.:std	0.19	0.23	0.20	0.21	0.21	0.20	0.18
Recon.:err	0.08	0.10	0.09	0.08	0.07	0.09	0.10
Recon.:std	0.14	0.18	0.22	0.18	0.19	0.17	0.18
Inter.:err	0.15	0.10	0.09	0.12	0.08	0.12	0.13
Inter.:std	0.21	0.23	0.20	0.17	0.15	0.14	0.11

TABLE VII

ESTIMATION MEAN ERROR err AND std FOR DIFFERENT METHODS

traffic under different link utilizations. As shown in Fig. 21, the estimated mean value of cross traffic is smaller than the real traffic under the prediction-based and reconstruction methods. In the interpolation-based method, when the link utilization is very high (≥ 0.95), the estimated mean value of cross traffic is higher than that of real traffic.

VII. CONCLUSION

In this paper, we demonstrate that the self similarity property of cross-traffic can be exploited to infer on an end-to-end basis the amount of cross traffic on the bottleneck link, and investigate three such theoretically-grounded methods: prediction, reconstruction, and interpolation. The simulation study indicates that the prediction-based and reconstruction methods can give good mean measurement of cross traffic, while the interpolation method can, with proper design of the *FIR* filter, capture both the mean and instantaneous values of cross-traffic. All three methods are adaptive to the dynamic change of cross traffic and are quite robust in the presence of multiple bottleneck links on an end-to-end path.

While the work lays a theoretical foundation for exploiting LRD characteristics in measuring cross traffic, there remain several practical issues that should be considered in deploying the proposed methods in real networks. For example, if the bandwidth of the bottleneck is large, the dispersion, t, of the sending times of two back-to-back packets has to be extremely small (Eqs. (1)–(2)). Sending extremely closely-spaced packets is subject to the limitation of the OS at end hosts and may not be feasible. We will study how to resolve this issue. We will also further improve the robustness of the proposed methods in the cases that probe packets may be queued before/after the bottleneck link (violation of assumption (**A2**)) and that the bottleneck itself may change as a result of dynamic network traffic changes.

REFERENCES

[1] J. C. Bolot. End-to-end packet delay and loss behavior in the Internet, in *Proc. ACM SIGCOMM'93*, pp. 289-298, 1993.

[2] R. Cáceres, N. G. Duffield, J. Horowitz, and D. F. Towsley. Multicast-based inference of network-internal loss characteristics, *IEEE Journal on Selected Areas in Communications*, Vol. 45, No. 7, November 1999.

[3] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. In *Proc. of ACM PERFORMANCE'96*, October 1996.

[4] C. Cetinkaya and E. Knightly. Egress admission control, in *Proc. IEEE INFOCOM 2000*, March 2000.

[5] N. G. Duffield, J. Horowitz, D. Towsley and T. Bu. Multicast-based inference of network-internal characteristics: accuracy of packet loss estimation, in *Proc. IEEE INFOCOM'99*, April 1999.

[6] N.G. Duffield and F. Lo Presti. Multicast inference of packet delay variance at interior network Links. In *Proc. of IEEE INFOCOM 2000*, April 2000.

[7] A. Erramilli, O. Narayan, and W. Willinger. Experimental queuing analysis with long-range dependent traffic, *IEEE/ACM Transactions on Networking*, April 1996.

[8] Felix. Independent monitoring for network survivability. For more information, refer to http://www.bellcore.com/pub/mwg/felix/index.html.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking*, Vol.1, No.4, August 1993.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation based congestion control for unicast applications, in *Proc. ACM SIGCOMM 2000*, August 2000.

[11] G. He and J. C. Hou. On exploiting LRD to stabilize queues in AQM. *Tech. report, Dept. of Computer Science, UIUC,* May 2002.

[12] IPMA: Internet performance measurement and analysis. For more information, refer to http://www.merit.edu/ipma.

[13] V. Jacobson. Pathchar — a tool to infer characteristics of Internet paths. ftp://ftp.ee.lbl.gov/pathchar/, 1997.

[14] J. Beran. Statistics for Long-Range Process, *Ch.4. CHAMPAN & HALL*, 1994.

[15] K. Lai, M. Baker. Measuring link bandwidths using a deterministic model of packet delay, in *Proc. ACM SIGCOMM 2000*, August 2000.

[16] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson. On the self-similar nature of Ethernet traffic (extended version), *IEEE/ACM Transactions on Networking*, February 1994.

[17] J. Mahdavi, V. Paxson, A. Adams, and M. Mathis. Creating a scalable architecture for Internet measurement. In *Proc. INET'98,* 1998.

[18] M. Mathis, J. Semkeand, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm, *Computer Communication Review*, Vol.27, pp.67-82, July 1997.

[19] V. Paxson. End-to-end Internet packet dynamics, *IEEE/ACM Trans. Networking*, Vol.7, pp.277-292, June 1999.

[20] P. Stoica and R. Moses. Introduction to Spectral Analysis, *Ch.2, Prentice-Hall, Upper Saddle River, NJ*, 1997.

[21] S. Ratnasamy and S. Mccanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements, in *Proc. IEEE INFOCOM'99*, March 1999.

[22] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks and R. Baraniuk, Multifractal cross-traffic estimation, in *Proc. of the ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Monterey, California*, September 2000.

[23] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement, in *Proc. ACM SIGMETRICS*, September 2000.

[24] A. Sang, and S.-q. Li. A predictability analysis of network traffic, *Proc. of IEEE INFOCOM'2000*, March 2000.

[25] S. Savage. Sting: A TCP-based network measurement tool. In *Proc. of the USENIX Symp. on Internet Technologies and Systems*, 1999.

[26] Surveyor. For more information, refer to http://io.advanced.org/surveyor.

[27] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level, In *Proc. ACM SIGCOMM'97*, pp. 149–157, 1997.