Proceedings ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management,

September 2000, Monterey, CA

Multifractal Cross-Traffic Estimation

Vinay Ribeiro, Mark Coates, Rudolf Riedi, Shriram Sarvotham,
Brent Hendricks and Richard Baraniuk *

Department of Electrical and Computer Engineering, Rice University
6100 South Main Street, Houston, TX 77005, USA
Email:{vinay, mcoates, riedi, shri, brentmh, richb}@rice.edu. URL: www.dsp.rice.edu.

Abstract

In this paper we develop a novel model-based technique, the Delphi *algorithm, for inferring the instantaneous volume of competing cross-traffic across an end-to-end path. By using only end-to-end measurements, Delphi avoids the need for data collection within the Internet. Unique to the algorithm is an efficient exponentially spaced probing packet train and a parsimonious multifractal parametric model for the cross-traffic that captures its multiscale statistical properties (including long-range dependence) and queuing behavior. The algorithm is adaptive; it requires no a priori traffic statistics and effectively tracks changes in network conditions. ns (network simulator) experiments reveal that Delphi gives accurate cross-traffic estimates for higher link utilization levels while at lower utilizations it over-estimates the cross-traffic. Also, when Delphi's single bottleneck assumption does not hold it over-estimates the cross-traffic.*

1 Introduction

1.1 Edge-based estimation

A better understanding of the dynamic properties and behavior of end-to-end paths would greatly benefit the design and development of future network control algorithms and protocols. It is unrealistic to expect internal routers to determine and report these properties, because this would require maintaining overwhelming amounts of per-flow state information. It becomes necessary to infer the properties from edge-based measurements, which are relatively easy and inexpensive to make. In this light, several authors have proposed edge based techniques for congestion control [1–3], estimating the bottleneck bandwidth [4–6], inferring multicast routing trees [6], performing admission control [7] and detecting flows with the same congestion points [8].

In addition to their practicality, edge-based estimation algorithms also provide a convenient abstraction of network dynamics. Exactly modeling connections that traverse multiple hops is hopelessly complex, both in terms of overhead and analysis. If inference techniques based on simplifying assumptions, for example a single bottleneck assumption, are successful then it will be possible to develop reduced complexity models that accurately reflect network behavior.

Edge-based analysis and simplified end-to-end path modeling are closely interwoven. Accurate estimates of the volume of cross-traffic competing for the available bandwidth of a path have the potential to impact a wide range of applications. Potential applications of such estimates include (1) new end-to-end based congestion control protocols, (2) workload balancing on web servers ("rate based clocking"), (3) dynamic adjustment of transmission rate to maximize quality of voice and video-conferencing connections, and (4) pricing on a connection basis according to the stress a transfer puts on the network in its current state.

1.2 The Delphi algorithm

In this paper we propose and analyze the *Delphi* algorithm, an inference procedure that uses the queuing delay experienced by probe packets to estimate, over a range of time scales, the load induced by cross-traffic on the bottleneck link of an end-to-end path. The algorithm is sender based requiring no collaboration from the network and only little feedback from the receiver as to when packets reach it.

Inherent in any probing scheme is an uncertainty principle or "accuracy-sparsity" tradeoff. The volume of cross-traffic entering a queue between two probes can be computed exactly (assuming infinite clock precision) from their delay spread at the receiver *provided* the queue does not empty in between. Unfortunately, this

*This work was supported by the NSF, grant no. ANI-9979465, by ONR, grant no. N00014-99-10813, by DARPA, grant no. R 36400, and by Texas Instruments.

situation is guaranteed only if the probes are very closely spaced. However, sending long trains of narrowly spaced probes will overwhelm the very cross-traffic we are trying to measure. If probes are spaced far apart the queue can empty in between which results in uncertainty in the cross-traffic volume.

Delphi differs from earlier techniques for estimating available bandwidth [9,10] in that it is model based. By employing short bursts of packets these techniques are naturally restricted to estimating cross-traffic only over short time periods [9]. Other methods capable of dealign with larger time scales use only an indirect measure of the competing traffic load [3]. Delphi instead uses probes spaced farther apart and improves the accuracy in cross-traffic estimates by leveraging statistical knowledge of network dynamics provided by a versatile traffic model, the *multifractal wavelet model* (MWM) [11]. Aggregated traffic on a link has been shown to be multiscale in nature (in a first approximation self-similar, or fractal [12]) and more precisely multifractal [11,13–15]. The MWM captures the multifractal properties of traffic that give it its bursty character.

Unique to Delphi is an efficient exponentially spaced probing packet train that matches the binary tree structure underlying the MWM. "Chirp packet trains" balance the accuracy-sparsity tradeoff by being highly accurate initially and highly sparse at their end. Probing the path with a series of chirp trains, we use the inter-packet delays at the receiver to estimate (using Bayesian inference techniques) the cross-traffic load at a range of scales. The efficiency of the MWM model and Delphi allows them to be applied on-line, generating estimates in real-time.

A significant advantage of Delphi is that it does not require traffic statistics in advance. Starting with arbitrary model parameters, Delphi estimates the cross-traffic from which it updates its parameters.

To study the Delphi's performance under different levels of utilization and its parameter tracking capability, we perform ns simulation experiments. Results indicate that at higher utilization levels Delphi gives accurate estimates of cross-traffic. Also, Delphi's model parameters converge to that of an MWM trained on the entire cross-traffic trace from initial arbitrary values. At very low utilization levels, however, Delphi gave less accurate estimates and the model parameters did not converge to those of the cross-traffic trace.

A fundamental assumption of Delphi is that most of the queuing delay that probe packets face is at the bottleneck queue. In a situation where the probe packets are delayed at two queues, Delphi over-estimates the cross-traffic and hence for congestion control purposes is conservative.

1.3 Overview

We review the models underpinning the Delphi algorithm (the MWM and simplified path model) in Section 2. Section 3 introduces Delphi by proposing methods for dynamically estimating the critical parameters of the MWM during the operation of the algorithm. We conduct a series of ns-2 simulations to study the Delphi's performance and demonstrate its practical applicability in Section 4. Section 4 also explores the validity of the simplified path model which lies at the core of the Delphi algorithm. Section 5 closes with a discussion and conclusions.

2 Modeling Framework

In this section, we introduce Delphi's intelligent probing methodology. Its overall goal is the real-time inference of end-to-end path network properties from measurements made at the edge of the network. In particular, Delphi attempts to estimate the cross-traffic intensity at the bottleneck queue in the path, a concept we clarify later in this section. Delphi is founded on the amalgamation of a simplified model of the end-to-end path and a statistical model for the cross-traffic stream. In this section, we first detail the simplified path model and what can be deduced about traffic behavior on the basis of its adoption. We then explain and motivate the traffic model.

2.1 Path modeling

After Bolot [4], we employ a simple model for the path that captures its essential features (see Figure 1). Basically, we reduce the path to a single bottleneck link/router connected to source and receiver by infinitely fast links. In addition, we assume that the propagation and processing delay of the path is constant and use the constant D to represent this fixed component of the overall end-to-end delay; a single-server queue with finite buffer and FIFO service discipline models the variable component of the delay. Entering this queue is both the traffic between source and receiver and a *cross-traffic stream*, which is the superposition of all the other connections that share network resources utilized by the path. The service rate C is related to the slowest link speed or queue service rate along the path; this is the bottleneck queue.

We now define some terminology. The intensity B_T at time-scale T denotes the number of bytes of cross-traffic that arrive in time-interval T. We call the average amount of cross-traffic per second that could be inserted by the source of the path over a time-interval T the *dynamic available bandwidth* at time-scale T, and denote it D_T. More specifically, if we use Q to denote

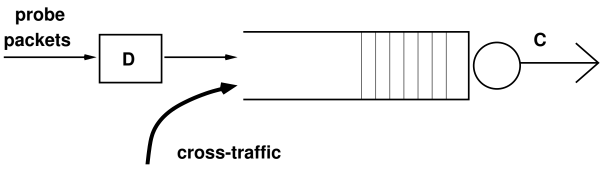

Figure 1: **Simplified cross-traffic model for an end-to-end path.**

the index set of the queues in the path, and C_i and $B_T(i)$ to denote, respectively, the bandwidth and cross-traffic intensity of the i-th queue ($i \in Q$), then:

$$D_T = \min_{i \in Q} \left[\frac{C_i T - B_T(i)}{T} \right]. \qquad (1)$$

Generally, we expect that this minimum arises at the bottleneck queue.

We use the term *virtual cross-traffic* of an end-to-end path to denote the aggregated cross-traffic that must be inserted in the single queue model so that the dynamic available bandwidth measured using the simplified model is the same as that along the entire multi-hop path obtained from (1). Figure 2 depicts the relationship between dynamic available bandwidth and virtual cross-traffic.

The specific aim of the Delphi algorithm is the dynamic inference of the intensity B_T of the cross-traffic present at the bottleneck queue over a range of different time scales T. In performing this inference, the algorithm makes a fundamental assumption, namely that the delay in excess to the constant propagation delay observed on the path is due to the cross-traffic on the bottleneck link alone. In many cases, this corresponds to estimating the virtual cross-traffic of the path and thus allows the development of a piecewise constant estimate of the path's dynamic available bandwidth. Section 4 explores the behavior of the algorithm when the delay assumption is not valid. In such circumstances, knowledge of the cross-traffic at the bottleneck queue no longer leads to the dynamic available bandwidth; the virtual cross-traffic of the path is then the quantity that must be estimated.

If we transmit two probe packets of size P bytes at times t_0 and $t_1 = t_0 + T$, then provided the queue does not empty between the queue-entry times of the first and second probes, the time difference between the arrival times a_0 and a_1 at the receiver is ideally identical with the inter-departure time at the queue. This time is proportional to the number of bytes in the queue over the time-period T:

$$B_T = C(a_1 - a_0) - P. \qquad (2)$$

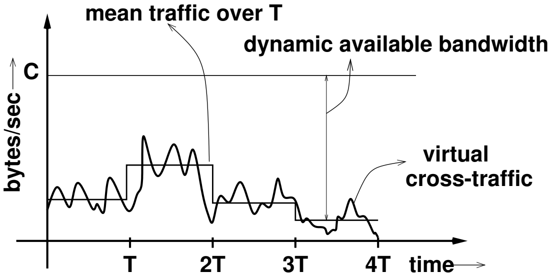

Figure 2: **Dynamic available bandwidth at time-scale T, together with a plot of virtual cross-traffic of the path.**

Unfortunately, if the queue does empty between probe-packet arrivals, then the inter-arrival spread can substantially overestimate the cross-traffic arriving in the period T.

For probe packets of size P, we use $T_{NEQ} = P/C$ to denote the coarsest time-scale that ensures that the queue cannot empty between the probe arrivals. If we could send probe-trains that were this finely spaced, then there would be no risk of underestimation. Unfortunately, such fine scale probing is impossible because it overwhelms the network after a short period and disrupts the measured traffic. Rather, we propose the use of a *packet chirp* as a probing device. The packet chirp consists of probes sent in an exponential flight pattern, with the first three probes spaced by a time $T_n \leq T_{NEQ}$ and then the spacing between subsequent probes increasing by a factor of two each time (see Figure 3). The first three probes of the packet chirp thus provide initial fine scale probing which can provide exact knowledge of $B_{T_{NEQ}}$.

The packet chirp probing strategy balances the trade-off between (1) generating reliable and accurate estimates and (2) overburdening and disturbing the network. The fine-scale probing anchors cross-traffic estimates made at coarser levels, but the rapid increase in probe-spacing makes the probing efficient.

2.2 Multifractal wavelet model

Accurate estimation of the cross-traffic from a relatively coarsely spaced train of probe packets is not possible without some form of statistical model for the cross-traffic. The cross-traffic stream is the superposition of many data flows that share common Internet resources with the probe connection. Such superpositions have been shown to exhibit self-similarity [12], burstiness, long-range dependence (LRD), and even *multifractal* behavior [11,13–15], all of which can have a major impact on network performance. These characteristics can be captured with an appropriate statistical traffic model.

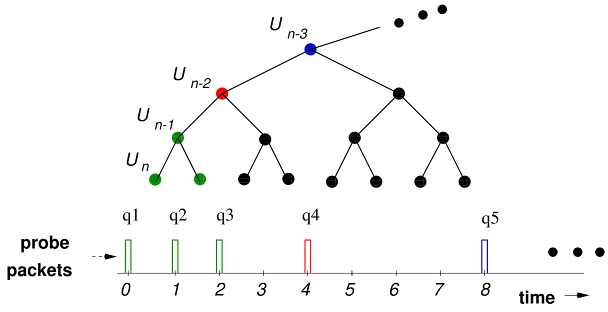

Figure 3: **Packet chirp probe train has an exponential flight pattern to balance the trade-off between accuracy and sparsity.**

Our model of choice for the cross-traffic is the *multifractal wavelet model* (MWM) [11, 14]. The MWM is based on a computationally efficient tree structure which represents the cross-traffic load at multiple aggregation scales on a binary tree (see Figure 4(a)). The MWM's tree is closely related to the Haar wavelet system and hence the term "wavelet" in its name [11].

We use T_0 to denote the time-interval between the first and last probes in the packet train. Within this interval, the tree coefficients $U_{j,k}$, $j \geq 0$, $k = 0, 1, \ldots, 2^j - 1$, correspond to the total sum of cross-traffic bytes arriving at the model queue in the interval $[2^{-j}kT_0, 2^{-j}(k+1)T_0]$. Here j denotes the *scale* of interest (see Figure 3). Note that each parent coefficient is the sum of its two children

$$U_{j,k} = U_{j+1,2k} + U_{j+1,2k+1}. \quad (3)$$

Note that we take a different normalization than in [11, 14]. Thus, we can move *up* the tree from some finest scale to obtain all $U_{j,k}$ at coarser scales. To move *down* the tree while ensuring that (3) is preserved, we model the parent bytes $U_{j,k}$ as split between its children by a random factor:

$$U_{j+1,2k} = B_{j,k}\, U_{j,k}\,, \quad U_{j+1,2k+1} = (1 - B_{j,k})\, U_{j,k} \quad (4)$$

with $B_{j,k}$ a random variable distributed between 0 and 1 (see Figure 4(b)). The use of symmetric beta random variables for the multipliers $B_{j,k}$ is proposed in [11]. So the MWM is a *parametric* model for bursty non-Gaussian traffic. Its parameters are (1) a global mean-rate parameter (the aggregate at the coarsest scale) and (2) beta multiplier parameters (one for each scale).

To train the MWM to a target traffic data set, we simply flow up the tree to form the $U_{j,k}$, collect their statistics, and estimate the beta parameters (see [11,14] for the details). Delphi does not have access to the $U_{j,k}$ (indeed, it is a subset of these that we are attempting to infer); it becomes necessary to jointly infer the appropriate beta parameters and the traffic. We outline a sequential adaptive algorithm in Section 3 that performs this estimation. The MWM can match the exact multiscale second-order statistics of a traffic trace and provides a much closer match of the higher-order statistics (non-Gaussianity) that lead to burstiness than other possible traffic models [12, 16].

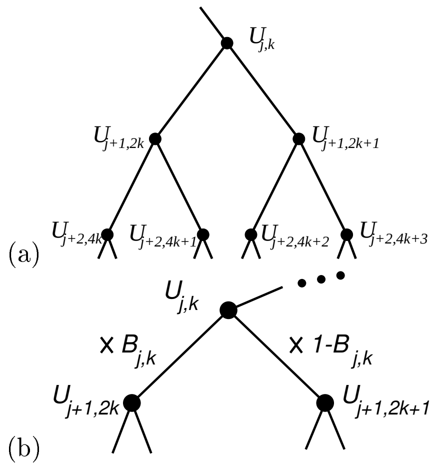

Figure 4: **Multifractal wavelet model (MWM). (a) Binary tree structure of aggregated traffic. (b) Beta multipliers split parent aggregate into two child aggregates at the next finer scale.**

Most importantly, the MWM provides a natural and efficient means to estimate, given the beta parameters, the queuing behavior of a synthetic trace [17]. The *multiscale queuing formula* (MSQ) approximates the tail queue probabilities for *arbitrary* buffer sizes (not just asymptotic). We use the derivative of the MSQ to obtain a probability density function for queue sizes. The computational efficiency, and queuing formula approximation of the MWM make it a natural modeling framework for our cross-traffic estimation algorithm.

3 MWM-based Inference Algorithm

In this section, we present the *Delphi* algorithm itself which estimates the amount of cross-traffic that arrives in the time interval between the arrival at the queue in our model (recall Figure 1) of the first and last probe packets of a packet chirp. We model the cross-traffic as an MWM process with unknown beta parameters. In a simpler form, the Delphi algorithm uses fixed beta parameters that have been estimated from previous measurements. In the latter part of this section we detail a method for adaptively estimating the beta parameter estimates on the fly as traffic estimates are generated.

Delphi attempts to form reliable estimates whilst transmitting as few packets as possible. It does this by sending a packet "chirp" occupying the time-interval T_0; the interval is partitioned according to the exponential spacing of the probes. Figure 3 depicts this exponential flight pattern and its natural relationship with the MWM tree. If we send $n+2$ probes in the packet chirp, then T_0 labels the interval between the first and last probes, T_1 the interval between the first and second last, and so on, such that T_n is the interval between the first and second probes. The probe structure guarantees that $T_n \leq T_{NEQ}$, the maximum separation between probes that ensures that the queue does not empty between probe arrivals. For dynamic measurements, Delphi issues a new chirp every T_0 seconds to obtain a piecewise constant estimate of the cross-traffic load.

Estimation of the cross-traffic arriving in the interval T_0 requires an estimate for the cross-traffic that appeared in the interval T_1, which in turn needs an estimate of the cross-traffic that appeared in the interval T_2 and so on. This recursive requirement flows down the tree until it hits T_{n-1}, at which point the distance between probes is such that the queue is guaranteed not to empty (recall (1)), and an exact measurement of cross-traffic is available. Once this point has been reached, the algorithm flows back up the tree, calling an inference procedure to estimate U_i for $i = (n-2), (n-1), \ldots, 0$. See Figure 5 for pseudo-code detailing the nature of the Delphi algorithm.

The inference procedure (*infer* in Figure 5) develops an approximate maximum likelihood estimate. When we assume that the bottleneck bandwidth is known, the delay each probe experiences ideally provides a instantaneous measure of the bottleneck queue size, which we denote q_i for the i-th probe (see Figure 3). This measurement is usually noisy due to the granularity of clocks, the drift between sender and receiver clocks and additional queuing delays at other queues along the path. At time scale $n-2$, we wish to maximize the likelihood of jointly observing both the measured traffic over T_{n-1} and the measured queue size q_4:

$$p(u_{n-1}, q_4 | u_{n-2}) \quad = \quad p(q_4 | u_{n-1}, u_{n-2})\, p(u_{n-1} | u_{n-2}) \tag{5}$$

We approximate the probability of observing q_4 given the traffic arriving in T_{n-1} and T_{n-2} as the probability of observing q_4 given the traffic arriving only in the latter half of T_{n-2}. We are left with the expression

$$p(u_{n-1}, q_4 | u_{n-2}) \approx p(q_4 | u_{n-2} - u_{n-1})\, p(u_{n-1} | u_{n-2}).$$

The first term on the right hand side can be approximately evaluated using the MSQ formula [17]; the second term is equal to the appropriate beta distribution in the MWM model. We maximize this likelihood over the range of possible u_{n-2} values (which is constrained by the queue sizes q_3 and q_4). Now we have an estimate $\widehat{u_{n-2}}$ that can be used to generate an estimate of u_{n-3} by maximizing the approximate likelihood function $p(\widehat{u_{n-2}}, q_5 | u_{n-3})$. The process continues until an estimate has been formed for u_0.

Delphi uses the bandwidth C to translate delays into traffic volumes. Without C Delphi can still classify the cross-traffic volumes as high or low but cannot scale them to obtain the traffic in bytes. Also the value T_{NEQ} depends on C. However, Delphi does not require the initial probe packets to be spaced closer than T_{NEQ} and can estimate cross-traffic at the finest time scale in the same fashion as at coarser scales.

In the form proposed above the Delphi algorithm relies on prior estimation of the beta multiplier parameters of the MWM that best fits the traffic. We now propose a method to adapt the beta parameters on-line while the cross-traffic estimation is conducted. The method is similar to parameter adaptation techniques that underpin sequential estimation procedures in problems as diverse as speech processing and target tracking. At the commencement of the Delphi algorithm, we choose an initial estimate of the beta multipliers, which may be based on previous measurements or may be completely arbitrary. We then begin to estimate cross-traffic using these initial parameters. After we have made K such estimates, we use the sample second-moments of the cross-traffic estimates to obtain an *instantaneous estimate* of multiplier variances using

$$\mathbb{E}[B_j^2] = \mathbb{E}[U_{j+1}^2] / \mathbb{E}[U_j^2]. \tag{6}$$

We then update the MWM parameters using

$$\begin{aligned} \text{new variance} \quad &= \quad \alpha \times \text{instantaneous estimate} \\ &\quad\quad +(1-\alpha) \times \text{current variance} \end{aligned} \tag{7}$$

where $\alpha \in [0, 1]$ is a constant. Smaller values of α give rise to smoother changes in parameters. We could also choose different values of alpha to update parameters at different time scales depending on the smoothness required. In this paper we set $\alpha = 0.2$. Figure 8 in Section 4 demonstrates that this method generates multiplier estimates that converge to the same values derived from model matching based on the entire traffic trace.

4 Simulation Experiments

In order to test Delphi we conduct several simulation experiments using the LBNL Network Simulator (ns

Delphi Algorithm

procedure main (q, T_0, C, n) {

    u_0 = determine_traffic(q, T_0, C, n)
    return u_0

}

procedure determine_traffic(q,T,C,k) {

    if $(T < 2T_{NEQ})$
        $u = q_k - q_{k-1} + TC$
        return u
    else
        m = determine_traffic(q,T/2,C,k-1)
        u = infer(q,T,C,m,k)
        return u
    end

}

procedure infer(q,T,C,m,k)

$$\text{u\_min} = \begin{cases} m, & q_k = 0 \\ m + q_k + C - & \\ \max(q_{k-1} + C - (T-1)C, 0), & q_k > 0 \end{cases}$$

    u_max = $m + q_k - q_{k-1} - C + TC$
    return $\hat{u} \in$ [u_min,u_max] that maximizes $p(q_k, m \mid u)$

}

Figure 5: **The Delphi algorithm: q denotes the vector of queue measurements, T_0 the time interval of interest, C the service rate of the model queue, and n the number of probe-packets sent. The range of possible values of traffic, that is u_{max} and u_{min} are computed assuming a discrete-time FIFO queue taking into account the effect of the previous probe on the queue size.**

version 2) [18]. By varying the utilization, we observe that at higher utilization levels Delphi gives accurate estimates with little bias while at low utilizations it over-estimates the cross-traffic. Over-estimating the cross-traffic corresponds to under-estimating the available bandwidth which is conservative for congestion control purposes. In addition, at higher utilization levels the Delphi's model parameters converged to those of an MWM trained on the cross-traffic while at low utilization the parameters did not converge. Experiments with different buffer sizes showed little change in Delphi's performance so we do not report their details here.

To gain insight into Delphi's performance when the underlying assumption of a single bottleneck is invalid, we perform trace driven simulations. Results indicate that in situations where the probe packets experience significant delay at links other than the one with the smallest link speed on the path, Delphi gives conservative estimates for cross traffic and available bandwidth.

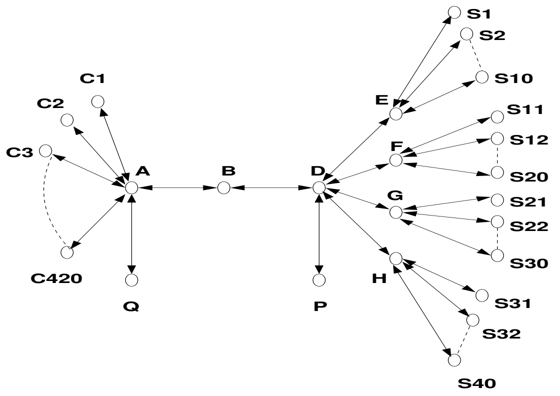

Figure 6: **Network configuration for ns-2 simulations with bottleneck link (A, B).**

4.1 ns Experiments

We simulated a bottleneck network environment where several concurrent connections are multiplexed over a shared bottleneck link. Figure 6 shows the network topology, comprising 420 web clients and 40 web servers. Table 1 gives the link characteristics of all the links in the network.

The clients engage in large-scale web traffic across the bottleneck link to the servers. The simulation is carried out for over 1500 seconds (simulation time) and number of sessions is varied to obtain different link utilizations. Each web session has 350 pages whose sizes are chosen from a heavy tailed distribution. Node P sends chirps of probe packets of size 900 bytes to node Q using the UDP protocol.

The minimum spacing between probe packets is set to $T_{NEQ} = 2.4$ ms which corresponds to a bottleneck link capacity of 3 Mbps and probe packet size of 900 bytes. This ensures that the queue does not empty between the first three probes in every chirp. We discarded the first 1000 seconds of the simulation to eliminate transients. The Delphi algorithm assumes that the bottleneck bandwidth C is known, so we supplied it with the configuration value; alternatively it could have been estimated beforehand using techniques outlined in [5].

Delphi computes the queuing delay of a probe packet as

$$\begin{aligned} \text{queuing delay} &= \text{receive time} - \text{transmit time} \\ &\quad -\text{constant delay}. \end{aligned} \tag{8}$$

The constant delay equals the sum of propagation delay and service time. This is set equal to the minimum of all delays experienced by probe packets while traversing across the network. Note that in a simulation environment synchronized clocks and error free values of these

Table 1: **Link and source parameters for the network configuration depicted in Figure 6 as used in the ns-2 simulation experiments.** $U[a,b]$ **denotes a uniform random variable over the range** $[a,b]$**.**

Link	Band Width (Mbps)	Latency (ms)
AB	3	20
BD	10	20
DE	1.5	20
DF	1.5	30
DG	1.5	40
DH	1.5	60
DP	1.5	60
ES1 to GS10	10	U[10,100]
FS11 to FS20	10	U[10,100]
GS21 to GS30	10	U[10,100]
HS31 to HS40	10	U[10,100]
AC1 to AC420	U[22,32]	U[10,100]
AQ	U[22,32]	U[10,100]

quantities are available unlike in a real world situation. However, synchronized clocks at the sender and receiver are not necessary in practice because the difference in clocks at the sender and receiver can be incorporated into the constant delay term. Other problems like clock drift and resetting can however lead to errors in (8).

In all experiments we estimate the cross-traffic arriving over time-slots of 307.2ms (the $7th$ aggregation level in the tree of Figure 4), sending a chirp of 8 probe packets in each time-slot. The probe traffic was thus equal to 6.25% of the bandwidth. We set the parameter α of (7) for updating the model parameters to 0.2. The buffer size was fixed at 50 packets.

4.2 Utilization effects

We changed the number of web sessions to obtain different utilization levels. Variance-time plots [12] show that the cross-traffic is LRD with Hurst parameters of 0.74, 0.73 and 0.61 for the experiments with 39%, 65% and 96% utilization respectively.

Delphi performed better with increasing utilization as we observe from Figure 7. Notice from Figure 7 that the traffic is more bursty at the finer time scale than at the coarse scale. Since in the process of estimating the traffic at coarser time scales Delphi gathers statistics at finer time scales too, it can serve target applications that require information about the burstiness of traffic at multiple time scales.

As a measure of accuracy we use the sample mean and standard deviation of the the error which we normalize by dividing by the sample mean of the entire cross-traffic. See Table 2 for details. At high utilizations Delphi gives accurate estimates with little bias while at low utilizations it performs less accurately. Intuitively at lower utilization levels the probe packets encounter smaller queues more often and the smaller the queue size the longer the queue can stay empty between the arrival of two probe packets. This implies a greater uncertainty in the cross-traffic volume.

From Table 2, we observe that at low utilization Delphi gives estimates with a positive bias, that is it overestimates the cross-traffic. This corresponds to underestimating the available bandwidth which is conservative for congestion control purposes. Also if competing cross-traffic volumes are low, an error of the order of magnitude of the cross-traffic will not significantly affect the estimated available bandwidth. This result at low utilization thus does not reflect a major short-coming of Delphi.

4.3 Tracking capability

To test Delphi's adaptive algorithm we initially set the variance of the beta multipliers at all time scales equal to 0.083. This corresponds to a beta parameter of 1 for the multipliers [11]. Notice from Figure 8(a) that the variances of the multipliers track the correct values after few iterations. Also observe from Figure 8(b) that the initial estimates of traffic have larger errors because of incorrect model parameters and that the error reduces with improved model parameters. This demonstrates that the Delphi does not require prior knowledge of cross-traffic statistics. The utilization for this experiment was 65%.

The improvement in inference with correct parameter values indicates that the MWM parameters indeed contain valuable statistical information about the cross-traffic. It also suggests that incorrect statistical knowledge or oversimplified traffic models can give erroneous estimates.

Delphi uses cross-traffic estimates to update model parameters. As a result, in the experiment with 39% utilization the model parameters did not converge because of less accurate cross-traffic estimates.

4.4 Validity of the model

We used trace-driven experiments to explore the validity of some of the assumptions underpinning our model. The use of the simplified path model depicted in Figure 1 hinges on the bottleneck queue providing the major queuing delay in the path.

This assumption can be invalid if the cross-traffic at

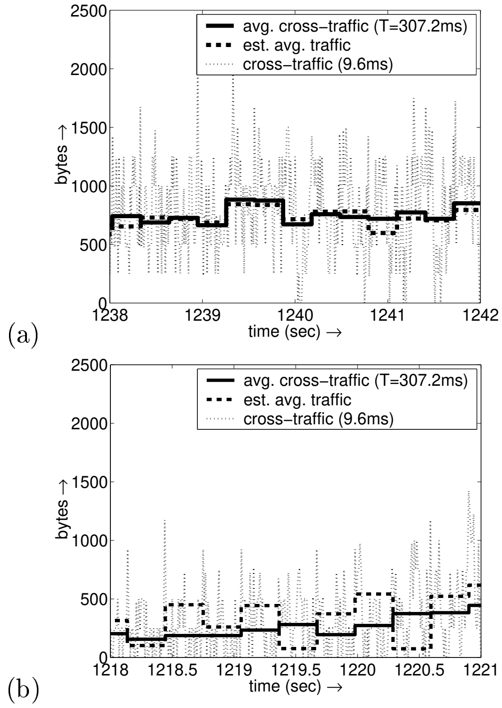

Figure 7: **(a) Cross-traffic estimates for utilization of 65%. Observe that the estimates are accurate. (b) Cross-traffic estimates for 39% utilization. Observe that the estimates give greater errors but are generally conservative.**

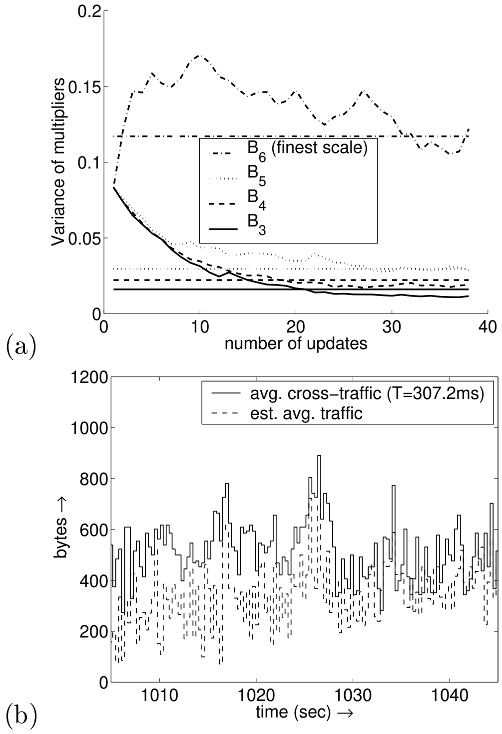

Figure 8: **(a) Adaptive estimates of variance of multipliers over time. The horizontal lines correspond to the sample variances of the multipliers estimated by matching the multifractal model to the entire traffic trace (i.e., the best model parameters we can achieve). Observe that from an arbitrary initial value of 0.083, the model parameters track the sample variances (b) Cross-traffic estimates over time. Notice that the error in estimates decrease over time as the model parameters track those of the actual cross-traffic.**

another queue in the path is substantially heavier than that at the bottleneck queue. Therefore we experimented with the two queue system depicted in Figure 9. By applying independent cross-traffic streams (X_1 at the bottleneck queue and X_2 at the second queue) to the two queues, we were able to examine the effects of heavy cross-traffic at the second queue.

A somewhat simple-minded view of the system provides valuable intuition. Say we send two probes and neither queue empties between the arrivals of the probes. If we denote the cross-traffic appearing between the two probes at the first queue x_1 and that at the second queue x_2, then the total delay induced by the system (ignoring propagation and service delays) is

$$d = x_1/C_1 + x_2/C_2. \tag{9}$$

The Delphi model assumes that this delay is entirely due to the bottleneck queue; its estimate of cross-traffic is $x = x_1 + C_1 x_2/C_2$. The second term is an error term, but because we assume that $C_2 > C_1$, it only has a substantial effect on the estimate if x_2 is much larger than x_1.

We examined the effect of the error term by conducting an experiment in which we set the bandwidth at the two congestion points to $C_1 = 1$ and $C_2 = 5$ bytes/time unit. We used second-order self-similar MWM traces with Hurst parameter $H = 0.8$ as the cross-traffic at both queues [12]. The mean and standard deviation of cross-traffic at the finest time-scale were chosen to equal half the bandwidth at their corresponding queues.

Using this experimental setup, we applied the Delphi algorithm to generate estimates of the cross-traffic at the bottleneck queue. Figure 10(a) plots the estimates (normalized by the true cross-traffic at the bottleneck) against the ratio of the true cross-traffics. The lines show the asymptotic values we anticipate from (9); for low ratios, the first term dominates and for high ratios, the second term dominates.

In Section 2, we explained that the dynamic available

Table 2: **The error of inference decreases with increasing utilization. A positive mean error implies generally conservative estimates at that level of utilization.**

Utilization (%)	Mean Error (normalized)	Std. of Error (normalized)
39.09	0.27	0.42
65.34	0.1	0.23
96.04	-0.023	0.044

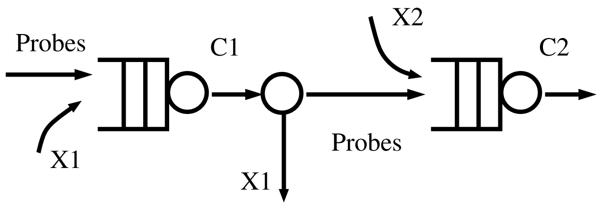

Figure 9: **A two queue system used to validate model assumptions. The first queue is the bottleneck with bandwidth C_1; the second has bandwidth C_2. The queues experience independent cross-traffic streams X_1 and X_2. Probe traffic flows through both queues.**

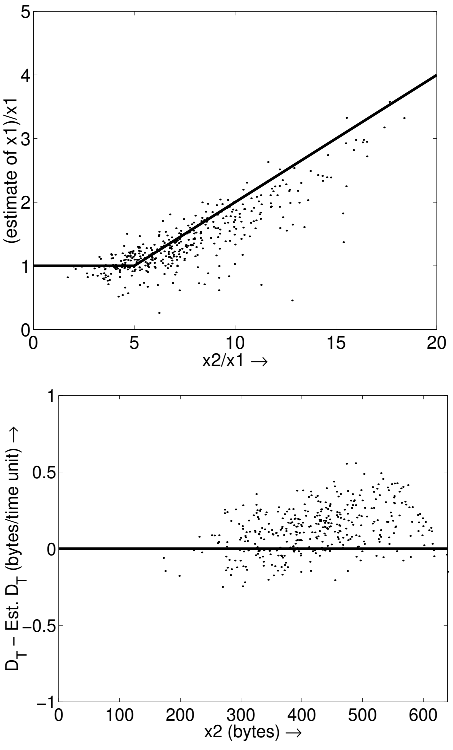

Figure 10: **(a) The ratio between the Delphi estimate and the true value of the cross-traffic x_1 at the bottleneck queue of the system in Figure 9 as a function of the ratio between x_1 and the cross-traffic at the second queue x_2. The lines depict anticipate asymptotics from (9). (b) The error between Delphi-estimated and true dynamic available bandwidth D_T (see (1)) as a function of x_2 with bottleneck bandwidth of 1 byte/time unit.**

bandwidth is really related to virtual cross-traffic rather than the cross-traffic at the bottleneck link. Virtual cross-traffic and bottleneck cross-traffic are only equivalent in our experiment when the ratio x_2/x_1 is small. Figure 10(b) compares the dynamic available bandwidth estimated using the Delphi algorithm to the true dynamic available bandwidth. Clearly even as x_2 becomes substantial (640 bytes is the maximum traffic that can be sent on the link over the time-scale studied), the Delphi estimate remains reasonable. Moreover, it becomes increasingly conservative, which is desirable behavior in times of heavy load.

5 Discussion

The ability to estimate cross-traffic loads is key to the development of a better understanding of Internet dynamics, and can potentially be used in the design of bandwidth efficient transport protocols and rate-based clocking methodologies.

We have introduced an algorithm (Delphi) that uses a novel probing strategy to dynamically estimate the cross-traffic load confronted along an end-to-end network path. Delphi can adopt an efficient probing pattern (the packet chirp) because our estimation procedure is model-based. This is critical when we wish to incorporate as much prior information about network behavior as possible so that we can derive accurate estimates from few measurements. It is fully adaptive and does not require a priori traffic statistics, is mainly sender-based requiring minor timing information from the receiver, and does not require synchronized clocks at the sender and receiver.

Simulations showed that Delphi gave smaller errors at higher utilization levels. This is not surprising, since probe packets are likely to encounter larger queues at high utilization levels. Larger queues imply a smaller chance of the queue emptying between the probes and hence a smaller uncertainty in cross-traffic volume. At low utilizations the errors in traffic estimates prevented Delphi from tracking the cross-traffic statistics. However at low utilization, errors as large as the traffic itself may not cause a large error in the estimated available bandwidth. Moreover, at low utilizations the Delphi generally over-estimates the cross-traffic which implies that for control purposes it is conservative.

A number of issues must still be addressed, and several algorithmic improvements suggest themselves. There are numerous practical issues we have not discussed in detail here, such as time-stamping issues and receiver/source-side algorithmic structure. Delphi needs to be modified to take dropped probes into account. At present chirps with dropped packets are discarded. A number of potential inaccuracies could arise, including the approximate nature of the multi-scale queuing formula. Our algorithm assumes a knowledge of bottleneck capacity; ideally capacity estimation should be performed in conjunction with the cross-traffic estimation. Knowledge about bottleneck capacity is however not essential for using the algorithm. Delphi can provide estimates of relative cross-traffic, that is not in absolute measure of bytes but as a relative comparison between cross-traffic volumes over different time intervals. There is also the question as to whether it is more natural to characterize path behavior by quantities such as dynamic end-to-end cross-traffic delays rather than explicit traffic estimates. Different characterizations of path behavior require new models portraying alternative spectra of system effects. Moreover, other packet probe configurations should be examined, including random and protocol-determined (completely passive) flight patterns.

References

[1] J. P. J. W. S. Floyd, M. Handley, "Equation based congestion control for unicast applications," *Proc. of SIGCOMM*, 2000.

[2] 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.

[3] T. Tuan and K. Park, "Multiple time scale congestion control for self-similar network traffic," in *Performance evaluation*, vol. 36, pp. 359–386, 1999.

[4] J. C. Bolot, "End-to-end packet delay and loss behavior in the internet," *Proc. SigComm '93*, pp. 289–298, 1993.

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

[6] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *Proc. INFOCOM '99*, (New York), Mar. 1999.

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

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

[9] R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in packet-switched networks," *Performance Evaluation (Proceedings of Performance'96)*, vol. 27&28, pp. 297–318, 1996.

[10] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proc. ACM SIGCOMM*, 1999.

[11] R. Riedi, M. S. Crouse, V. Ribeiro, and R. G. Baraniuk, "A multifractal wavelet model with application to TCP network traffic," *IEEE Trans. Info. Theory, Special issue on multiscale statistical signal analysis and its applications*, vol. 45, pp. 992–1018, April 1999.

[12] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Networking*, pp. 1–15, 1994.

[13] J. Lévy Véhel and R. Riedi, "Fractional Brownian motion and data traffic modeling: The other end of the spectrum," *Fractals in Engineering*, pp. 185–202, Springer 1997.

[14] V. Ribeiro, R. Riedi, M. S. Crouse, and R. G. Baraniuk, "Simulation of non-Gaussian long-range-dependent traffic using wavelets," *Proc. SigMetrics*, pp. 1–12, May 1999.

[15] A. Feldmann, A. C. Gilbert, and W. Willinger, "Data networks as cascades: Investigating the multifractal nature of Internet WAN traffic," *Proc. ACM/Sigcomm 98*, vol. 28, pp. 42–55, 1998.

[16] S. Ma and C. Ji, "Modeling video traffic in the wavelet domain," in *Proc. of 17th Annual IEEE Conf. on Comp. Comm., INFOCOM*, pp. 201–208, Mar. 1998.

[17] V. J. Ribeiro, R. H. Riedi, M. S. Crouse, and R. G. Baraniuk, "Multiscale queuing analysis of long-range-dependent network traffic," *Proc. IEEE INFOCOM*, March 26-30 2000.

[18] "ns – network simulator." For more information, see http://www-mash.cs.berkeley.edu/ns/ns.html.