Spectral Probing, Crosstalk and Frequency Multiplexing in Internet Paths

Partha Kanuparthy, Constantine Dovrolis, Mostafa Ammar
School of Computer Science
Georgia Institute of Technology
{partha,dovrolis,ammar@cc.gatech.edu} *

ABSTRACT

We present an end-to-end active probing methodology that creates frequency-domain signals in IP network paths. The signals are generated by periodic packet trains that cause short-lived queueing delay spikes. Different probers can be multiplexed in the frequency-domain on the same path. Further, a signal that is introduced by a "prober" in one path can cause a crosstalk effect, inducing a signal of the same frequency into another path (the "sampler") as long as the two paths share one or more bottleneck queues. Applications of the proposed methodology include the detection of shared store-and-forward devices among two or more paths, the creation of covert channels, and the modulation of voice or video periodic packet streams in less noisy frequencies. In this paper we focus on the first application. Our goal is to detect shared bottleneck(s) between a "sampler" and one or more "prober" paths. We present a spectral probing methodology as well as the corresponding signal processing/detection process. The accuracy of the method has been evaluated with controlled and repeatable simulation experiments, and it has also been tested on some Internet paths.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Monitoring

General Terms

Measurement, Experimentation, Performance

Keywords

Fourier Transform, Frequency Multiplexing, Signal Processing, Active Probing, Distributed Agents, Network Management

*This work was supported by a research award from Telchemy Inc and by the National Science Foundation (award ANIR-0347374).

IMC'08, October 20–22, 2008, Vouliagmeni, Greece.

1. INTRODUCTION

In analog communication systems, signals and channels, spectral (Fourier) analysis, multiplexing, frequency modulation, and channel interference (or crosstalk) are fundamental concepts. Even though these concepts are also very important in the physical-layer of packet networks, they have been traditionally viewed as foreign to the network, transport and application layers of the protocol stack. This paper proposes a manifestation of the previous concepts in IP-based packet networks. Specifically, we introduce a *Spectral Probing (SP)* framework that combines techniques for frequency-based signal generation in packet networks, frequency multiplexing of diverse signal sources, (intentional) crosstalk between network paths, and frequency modulation. Before we present applications of the SP framework, we start by introducing its key underlying ideas.

First, let us consider a network path from host A to host B as a *channel* P(A,B), assuming for now that this path is unique and constant for the time-window of interest. The obvious information-carrying signal is what is contained in the data packets sent on this path. The signal we are interested in, however, is more subtle: it is the *end-to-end queueing delay variations* on P(A,B). Specifically, suppose that we sample the one-way delay variation in P(A,B) using periodic or Poisson measurements. In the absence of any signal-carrying traffic from A to B, these measurements can be viewed as *noise*. It is caused by cross traffic at the queues along the path P(A,B). The measured queueing delay variations can be analyzed in the frequency-domain using standard Fourier analysis techniques to characterize the spectral properties of the corresponding timeseries. This has been done before, as discussed in the related work section (§ 7).

A new idea, however, is that we can generate information bearing signals on P(A,B) by sending specially crafted traffic that generates desired patterns in the end-to-end delay timeseries. As in analog communication systems, a frequency-based signal is more robust to additive noise than an amplitude-based signal [5]. So, the basic idea is that *we can cause periodic queueing delay increases in P(A,B) by sending periodically, every $T_p = 1/f_p$ seconds, a packet train with sufficiently high rate and size.* As explained in detail in the next section, if the transmission rate of the packet train is higher than the available capacity in P(A,B), and the frequency f_p is below a certain threshold, then the end-to-end delay timeseries will acquire a periodic component, a signal in other words, at frequency f_p.

A second idea is that we can *multiplex signals* of different frequencies in the same bottleneck queue, if that queue can

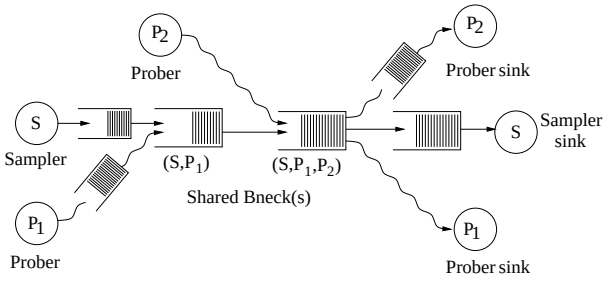


Figure 1: The sampler attempts to detect whether it shares a bottleneck queue with each of the two probing paths.

be modeled as a linear time-invariant channel. Each signal can be generated by a different source at A, or it may be that the same source generates multiple frequencies to synthesize a more complex signal. What is important here is that even though these individual signals can overlap in the time domain, they should not overlap in the frequency domain. Further, as in all bandwidth-limited communication channels, there is a limit on how many signals we can multiplex on the same queue.

A third idea is related to interference, or *crosstalk between different network paths*. In analog communications, crosstalk refers to the (most often, undesirable) leakage of signal power from one channel to a neighboring channel through capacitive/inductive coupling. In the context of IP network paths, coupling can take place when two paths P(A,B) and P(C,D) have a *shared bottleneck*.[1] In that case, queueing due to traffic bursts sent by A in P(A,B) can affect the end-to-end delays in P(C,D). In the SP framework, we exploit this coupling to convey frequency-modulated information from one network path to another.

We have thought of several applications of the SP framework and we anticipate that follow-up research will invent even more. In this paper we focus on *detecting sharing* application, illustrated next. We briefly touch upon two more applications, *frequency covert channels* and *continuous media modulation*, in Section 8.

The SP framework can be used to *detect shared store-and-forward devices* (links, routers, middleboxes, etc) between two network paths, at layer-2 or above, as long as one of the paths (the "probing" path) can induce queueing delays at a packet queue that it shares with the other path (the "sampling" path). Even though the "detection of shared congestion" problem has received significant attention in the past [10, 15, 16, 17, 24], the proposed approach is significantly different because it does not require that the shared link(s) is congested. Instead, the proposed method can detect sharing even when the shared link(s) is lightly utilized as long as the sender at the probing path can cause short-term queueing at the shared link(s).

The proposed method can be generalized, using frequency multiplexing, to detect whether each of N probing paths has *a shared bottleneck*, in the previous sense, with the sampling path. Figure 1 illustrates this application in the case of one sampler and two probers.[2] The probing path of P_1 has two shared bottlenecks with the sampler. The second bottleneck is also shared by the probing path of P_2.

In general, the actual topology of a network is rarely known even to network operators, especially when an end-to-end path crosses several administrative domains and when it includes devices at different layers of the protocol stack. It is often important, for network troubleshooting, failure-risk analysis, QoS provisioning, overlay routing, etc, to know whether two or more network paths share any infrastructure. Note that *traceroute* is not sufficient to detect sharing between two paths for several reasons: it only detects layer-3 devices, it is often disabled behind firewalls and NATs, and it can fail to detect that two IP segments may be going through the same router, sharing the same queue.[3] The proposed method in this paper *cannot* detect any type of sharing. For instance, we obviously cannot detect sharing at the physical layer. We detect sharing only when one path (the prober) can cause queueing (as in short-lived delay increases; not long-term congestion) in a packet queue of a network device that is shared by another path (the sampler). Note that the proposed technique does not require or assume that the shared queue is congested. This is an important difference with the previous work in shared congestion detection, discussed in Section 7, which aims to detect whether two or more paths (or TCP flows) share the same *congested queue.* Also, the proposed technique does not belong in the bandwidth estimation literature, because the latter focuses on the measurement of capacity or available bandwidth in a single path.

In Section 2, we describe the SP model and give basic constraints on the parameters of this probing method. In Section 3, we describe the signal processing algorithm that the sampler conducts to detect whether it has a shared bottleneck with one or more probers. In Section 4, we rely on simulations to evaluate the accuracy of the proposed shared bottleneck detection method, and to examine the impact of certain key parameters. In Section 5, we demonstrate that the method works "in the wild" with experimental results from several Internet paths, Related work is discussed in Section 7. We conclude in Section 8.

2. SPECTRAL PROBING

This section describes the basic model behind the SP framework. We first explain under which conditions a network queue can be viewed as a linear time-invariant channel. Next, we present basic constraints on the probing frequency, probing intensity, and selection of simultaneous probing frequencies. We are then ready to describe more precisely the shared bottleneck detection problem that this paper focuses on. Finally, we describe the end-to-end sampling process and give a constraint on the minimum sampling frequency.

2.1 Linearity approximation

The frequency multiplexing of different signals on the same channel requires that the channel can be modeled as *linear* (i.e., a linear combination of two or more input signals produces an output that is the corresponding linear combination of the individual outputs) and *time-invariant* (i.e., the input/output relation does not vary with time).

A packet queue, however, is *not* a linear system. To see why, consider a single FIFO queue with service rate C

[1]This term is defined more precisely in the next section.

[2]We use the terms "prober" and "probing path" interchangeably. Similarly for "sampler" and "sampling path".

[3]We show one such case in section 5.

bits/sec and buffer size B bits (denoted as {C,B}). The queue size (or backlog) $q(t_k)$ at time t_k can be determined by the following discrete-time equation for given initial conditions,

$$q(t_{k+1}) = \min\{\max\{q(t_k) + a(t_k) - C\Delta t_k, 0\}, B\} \quad (1)$$

where $a(t_k)$ is the amount of arriving traffic at t_k, and $\Delta t_k = t_{k+1} - t_k$. Obviously, the output variable $q(t_{k+1})$ is *not* a linear function of the input $a(t_k)$.

For a different selection of the output variable, however, and under certain conditions and approximations, a packet queue can be modeled as a linear system. Specifically, consider the following function,

$$q_\delta(t_k) = \{q(t_{k+1}) - q(t_k)\}^+ \quad (2)$$

where $\{x\}^+ = x$ if $x > 0$, and zero otherwise. We refer to $q_\delta(t_k)$ as the *backlog increase* function. In the following, we will assume that the arriving traffic is such that the buffer size B rarely causes losses. Note that $q_\delta(t_k)$ is positive only when the amount of arrivals at t_k exceeds what can be serviced in Δt_k,

$$q_\delta(t_k) = \{a(t_k) - C\Delta t_k\}^+ \quad (3)$$

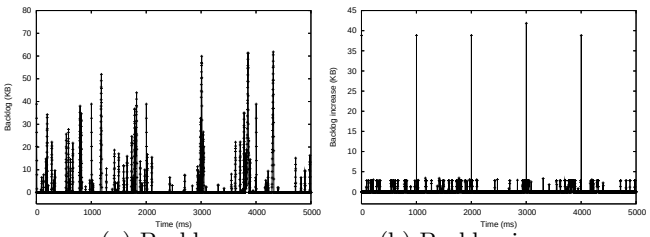
If we assume that the time interval Δt_k is very small and that the arrivals are bursty (packets arrive instantaneously), we have that $a(t_k)$ is either zero, or it is $a(t_k) \gg C\Delta t_k$. So, if one or more packets arrive at t_k the backlog increase function becomes approximately equal to the arrived traffic, otherwise it is zero,

$$q_\delta(t_k) = \{a(t_k)\}^+ \quad (4)$$

So, the backlog increase function for the sum of two or more signals $a_1(t_k) + a_2(t_k) + \ldots$ is the sum of the corresponding backlog increase functions $q_{\delta,1}(t_k) + q_{\delta,2}(t_k) + \ldots$. The queue can be further modeled as time-invariant, as long as its capacity remains constant.

There are three practical implications from the previous analysis. First, the output variable that we should be monitoring is the backlog increase function, not the backlog function. Second, the probing signals should be as bursty as possible, i.e., the probing traffic should arrive at the bottleneck queue with the highest possible rate. In the SP framework, we create a backlog increase signal by sending a sequence of L back-to-back packets (a "packet train"). If the rate with which the packet train arrives at the {C,B} queue is R_{in}, we can guarantee a backlog increase as long as $R_{in} > C$; the higher R_{in}/C, the better we approximate the previous linearity condition. So, it is important in practice that the probers have high-capacity network interfaces and they are connected to the Internet through high-capacity paths. Third, our signal transmitters should react to packet losses by decreasing their train length, so that the frequency of buffer overflows remains very low.

Figure 2 illustrates the functions $q(t_k)$ and $q_\delta(t_k)$, simulating a single queue (C=50Mbps) with random cross traffic (ON-OFF with exponential burst/idle durations) and four packet trains (L=30 packets, 1500 bytes each) at times 1000, 2000, 3000 and 4000. Note that the backlog function is noisy and it is hard to distinguish the probing signal from the cross traffic spikes. The backlog increase function, on the other hand, acts as a nonlinear filter, setting to zero the decreasing parts of the timeseries and reducing the magnitude of any gradual backlog increases.



(a) Backlog (b) Backlog increase

Figure 2: Example of a backlog function with the corresponding backlog increase function.

2.2 Probing frequency

Here we give constraints on the maximum and minimum probing frequency. Suppose that we want to create a periodic backlog increase signal of frequency f_p at a queue {C,B}. As previously mentioned, we can periodically send, every $T_p = 1/f_p$ seconds, a packet train that consists of L packets. Say that each packet has a size of s_p bits, and so the total size of the packet train is $S = s_pL$.

A first condition is that, even if there is no other traffic at that queue, it will take $\Delta_p = S/C$ to transmit a packet train. In the presence of cross traffic, Δ_p can be larger that S/C. So, the probing frequency should be much lower than $1/\Delta_p$, or otherwise successive packet trains will overflow the queue,

$$f_p \ll \frac{C}{S} \quad (5)$$

In other words, we can think of a packet queue as a *band-limited channel with a cutoff frequency that is at most C/S.*

Another reason to impose an upper bound on the probing frequency is to limit the average traffic rate of the probing signal, which is $f_p\,S$. Suppose that $R_{max} (\ll C)$ is the maximum average probing rate that we are allowed to generate. Then, the probing frequency should be upper bounded as follows,

$$f_p < \frac{R_{max}}{S} \quad (6)$$

It is important that the probing frequency is also not too low for two reasons. First, the spectral density of Internet traffic increases as $1/f$ as the frequency f tends to zero [9]. Thus, we should expect significant cross traffic noise in low probing frequencies. In practice, we have observed in many Internet paths that spectral noise increases significantly below the frequency of 1Hz. Figure 3 shows the spectrum of a timeseries of one-way delay variations at an Internet path. Note how the spectrum increases in lower frequencies, especially below approximately 1Hz.

Second, suppose that the duration of our signal is Δ_e. For practical reasons, related to measurement latency or variations in the underlying network conditions, we would like Δ_e to be as short as possible. For a given Δ_e, the number of signal iterations (number of packet trains) for a probing frequency f_p is $f_p\Delta_e$. Thus, if f_p is too low, we will not have enough signal iterations to accurately detect the signal in a noisy spectrum. In practice, we often set Δ_e=60sec and require at least 50-60 signal iterations. Thus, a probing frequency of around 1Hz seems to be a good lower bound in practice.

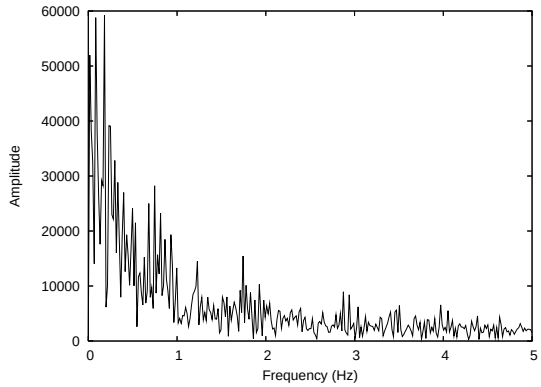


Figure 3: The spectrum of the one-way delay variations at an Internet path from procyon.cc.gatech.edu to www.dpls.lib.or.us.

2.3 Probing intensity

The transmitted signal power is determined by the size of the packet trains. There are three factors we need to consider here. First, the signal should be stronger than the noise intensity so that the receiver can detect the signal. Suppose that a packet train {S,R_{in}} arrives at the queue {C,B} with rate $R_{in} > C$. The minimum (i.e., in the absence of any cross traffic) backlog increase that the train will induce in the queue is $S(1 - C/R_{in})$. So, the packet train will create a queueing delay increase that is at least $\hat{d}_p = \alpha S/C$, where $\alpha = 1 - C/R_{in}$. Suppose that the cross traffic induces queueing delays that are typically less than $\hat{d}_n$ (we define $\hat{d}_n$ more formally in section 4). The *Signal-to-Noise Ratio* (SNR) is defined as $\hat{d}_p/\hat{d}_n$. We require that the SNR should be larger than a threshold $\gamma \approx 1$; in other words, we require that the minimum signal-induced delays should be at least as high as the larger delays induced by cross traffic. Then, a lower bound for the train size is

$$S > \frac{C\,\hat{d}_n}{\alpha} \tag{7}$$

A second consideration is that the packet trains should not be causing packet losses. In our prototype, the probers decrease L when there are packet losses at the receiver (detected using sequence numbers).

A third constraint is that the signal $\hat{d}_p$ should not be much higher than the queueing delays induced by cross traffic. In other words, the SNR should not be much higher than γ. This is important both for not being intrusive to cross traffic, but also for "stealth operation" when the SP framework is used to create covert channels of communication.

2.4 Multiple probing frequencies

Suppose that we want to multiplex several probing signals on the same queue. The basic idea is that each prober i will be sending trains at a frequency $f_p^{(i)}$, creating periodic delay increases of that frequency on the queue. As long as the queue can be modeled as a linear channel, the multiplexing of distinct frequencies will be a reversible process. Here, we examine the constraints for the selection of simultaneous probing frequencies. First, as previously discussed, there is a lower bound (about 1Hz) and an upper bound (R_{max}/S) for any probing frequency. Further, any two probing frequencies should differ by at least a *guard band* g so that the receiver can accurately distinguish them. The magnitude of the guard band depends on the exact detection process, and it is discussed in more detail in the next section.

The signal detection process that we propose in the next section does not only rely on the *fundamental frequency* $f_p^{(i)}$ of each probing signal, but it also leverages the first H harmonics of that signal.[4] So, an additional requirement is that the H+1 frequencies of each probing signal (the fundamental frequency and the first H harmonics) should differ by at least one guard band g from the H+1 corresponding frequencies of any other probing signal. Formally, we require that

$$|m\, f_p^{(i)} - n\, f_p^{(j)}| \geq g \tag{8}$$

for any probing frequencies $i \neq j$ and all $m, n = 1, \ldots H+1$. The appropriate value of g depends on the exact detection process used. In our implementation, a value of g around 0.1-0.2Hz is sufficiently large.

We determine numerically the maximum number of probing frequencies that can be multiplexed on a queue based on the previous constraint. For instance, for a minimum probing frequency of 1Hz, a maximum probing frequency of 4.76Hz, g=0.167Hz and H=4, we can multiplex five simultaneous probers.

2.5 The shared bottleneck detection problem

Now that we have described the main ideas behind spectral probing, we can define more precisely the shared bottleneck detection problem. Suppose that the "sampling" path is P_s and the "probing" paths are $P_p(1), \ldots P_p(N)$ with $N \geq 1$. Each probing path may share one or more queues with P_s. Note that two or more probing paths may be identical. Also, it may be that one or more probing paths overlap with the sampling path.

We say that a probing path $P_p(i)$ has a *shared bottleneck* (queue) with the sampling path if P_s and $P_p(i)$ share a packet queue and the packet trains transmitted by $P_p(i)$ can cause a backlog increase in that queue. As previously discussed, if the packet trains transmitted by $P_p(i)$ reach a shared queue {C,B} with rate $R_{in} > C$, then that queue is a shared bottleneck. If there is cross traffic in the shared queue, then it is possible that that queue will be a shared bottleneck even if $R_{in} < C$, as long as R_{in} is higher than the queue's available capacity.

In the shared bottleneck detection problem, the sampler aims to infer, relying exclusively on end-to-end information, whether each of the N probing paths has a shared bottleneck with P_s. Note that it is possible that there is sharing between a prober and the sampler, but that joint queue(s) is not a shared bottleneck. It is also possible that a probing path has more than one shared bottleneck with the sampling path. Also, the shared bottleneck is not necessarily the link with the minimum available bandwidth in that path.

2.6 Sampling process

The sampler cannot measure the instantaneous backlog at each queue of its path. It can measure however the end-to-end delay variations in its path. From those delay variations it aims to detect the presence of periodic increases at specific frequencies, and thus to infer which of the N probing paths have a shared bottleneck with P_s.

[4]Recall that the harmonics of a periodic signal of frequency f appear at the integer multiples of that frequency.

Specifically, suppose that the sampler sends a periodic packet stream through P_s. Let f_s be the frequency with which the sampler sends packets, and s_s be the size of those packets.[5] Say that $d(j)$ is the time difference between the receiver and sender timestamps for the j'th sampling packet. Note that clock synchronization between the sampling sender and receiver is not required; the measurements $d(j)$ will include the clock offset between the two hosts. It is important, however, to compensate for any clock skew (variable clock offset) (as described in [19] for instance). If we only have access at the sender of the sampling path, then we can rely on Round-Trip Time (RTT) measurements using a utility such as *ping*. In that case, however, the reverse channel may affect the accuracy of the method if it introduces significant delay variability.

The receiver of the sampling packets analyzes the following *delay increase* function,

$$d_\delta(j) = \{d(j) - d(j-1)\}^+ \tag{9}$$

Note how the previous function resembles the backlog increase function $q_\delta(k)$ at a single queue. With the delay increase function $\{d_\delta(j)\}$ the sampler aims to detect significant backlog increases at any queue along its path. Instead of searching for a signal at the end-to-end delay variations, which include both positive and negative differences, we aim to only detect *increases* of the measured delays. Delay decreases, on the other hand, are set to zero so that they do not contribute any power in the received spectrum. If we were analyzing, instead, the delay variations $\{d(j) - d(j-1)\}$, there would be significantly higher spectral noise due to all the negative delay differences.

2.7 Sampling frequency

According to the Nyquist sampling theorem, one would expect that f_s should be higher than twice the largest probing frequency f_p. In practice, however, we are working with time-limited signals (recall the earlier discussion about the signal duration Δ_e), while the Nyquist rate assumes infinitely-long signals. Instead, we require that the sampling period is much lower than the transmission latency of a packet train {S,R_{in}} at the bottleneck queue {C,B} so that the sampling packets can detect the increased queueing delays due to every probing train. The transmission latency of a packet train at {C,B} is at least Δ_p. So, we require that the sampling frequency is much higher, say at least five times, than $1/\Delta_p$,

$$f_s > \frac{5}{\Delta_p} = 5\frac{C}{S} \tag{10}$$

Note that the factor five is empirically chosen, based on our Internet experiments and simulations. A higher sampling frequency makes the detection process more accurate but also more intrusive. The packet size of the sampling process can be as small as possible to minimize intrusiveness, and it is typically set to 40B (UDP).

2.8 Putting it all together

To visually illustrate the significance of the previous parameters, Figure 4 shows the spectrum of four delay increase timeseries (from a simple simulation of a single queue with exponential ON-OFF random traffic). In Figure 4-a we have chosen a probing frequency, train length and sampling frequency based on the previous constrains: f_p=1Hz, L=30, and f_s=1KHz. Notice the clearly visible spectral spikes at the fundamental frequency 1Hz as well as at its first four harmonics. The next three plots show what happens when one of these important parameters is not sufficiently large. The spectral spikes are not visible, and it would also be hard to detect them with any signal processing algorithm.

[5]In all our simulations and experiments we set s_p=1500B using UDP packets.

3. DETECTION METHODOLOGY

In this section, we describe the signal processing algorithm that the sampler follows to detect whether it has a shared bottleneck with one or more probers. The sampler knows the set of $N \geq 1$ probers and the frequency $f_p^{(i)}$ that prober i uses. The goal, then, is to detect which of these N frequencies are present in the sampled signal. The detection scheme should exhibit both high sensitivity (low false negative probability) and high specificity (low false positive probability) for each prober.

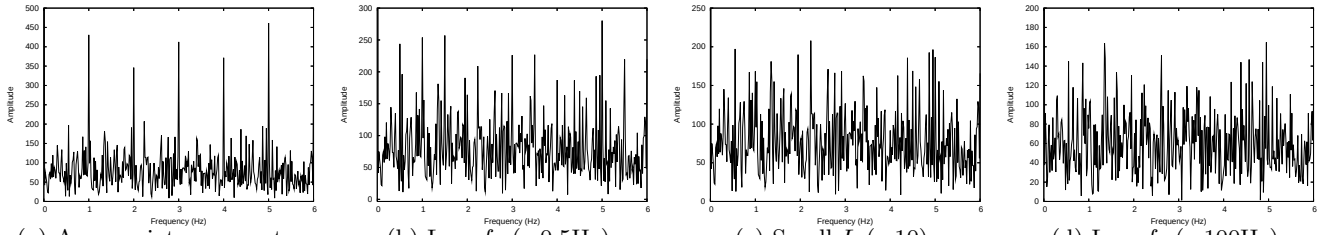
3.1 Outline

1) The entire experiment is checked in terms of traffic stationarity, path uniqueness and path constancy.
2) The measured timeseries at the sampler is "conditioned" to deal with lost samples and outliers. Then, we calculate the delay increase timeseries $d_\delta(j)$.
3) We estimate the spectrum of the delay increase timeseries using the Fast-Fourier Transform (FFT) algorithm. The spectrum is then appropriately filtered to reduce the effects of spectral leakage.
4) We infer whether the fundamental frequency, as well as the first H harmonics, of each prober are present in the estimated spectrum. To do so with a given false-positive probability, we also estimate the spectrum of the cross traffic in the sampler path *in the absence of any probing activity*.
5) We combine the H+1 frequency detection outcomes for each prober to infer whether that prober is present. If it is present, we also estimate the false positive probability.

3.2 Measured timeseries and basic checks

The proposed method involves two timeseries: *preprobing* and *probing*. Both timeseries have the same duration and sampler frequency (and thus the same number of points). The difference is that the former is collected while all probers are idle, while the latter is collected while all probers are active. In the shared bottleneck detection application the probers can communicate with the sampler to coordinate the start of the probing phase. In our simulations and Internet experiments, we set the duration of both timeseries to Δ_e=60sec. The probing timeseries is collected shortly after the preprobing phase has been completed.

Each prober, as well as the sampler, use the *Paris-traceroute* utility to examine whether their end-to-end path is unique and stays constant during the measurements [1]. If the underlying routing is not stable, or if it exhibits multipath characteristics even for packets with the same address/protocol/port IP header fields, the shared bottleneck detection problem is not well-defined. We further check whether the received timeseries at the sampler shows clear non-stationarity symptoms, such as major level-shifts. In those cases we abort the measurements. Our Internet experiments have shown that



(a) Appropriate parameters (b) Low f_p (=0.5Hz) (c) Small L (=10) (d) Low f_s (=100Hz)

Figure 4: The effect of various key parameters on the probing spectrum.

most paths satisfy the previous constraints, but there are few paths in which these constraints are often violated.

3.3 Timeseries conditioning and outliers

Because of packet losses, the two timeseries may be missing some samples. The sampler can detect those if the packets carry sequence numbers. Frequently lost samples can affect the spectrum of the received timeseries. In practice, the loss rate in our simulations and Internet experiments is much lower than 1%, and so the effect of lost samples is insignificant. We replace lost samples with the previous successfully received sample.

An important conditioning step is to detect, and somehow remove, outliers from both preprobing and probing timeseries. Outliers can be caused by measurement errors (e.g., large delays introduced by the operating system at the sending or receiving hosts) and/or by sudden and major forwarding delays (not necessarily due to queueing) in forwarding network elements. Outliers can significantly distort the spectrum of the received timeseries, hiding any periodicities that may exist due to probing. For instance, see how the presence of 3-4 outliers in Figure 5 greatly affects the spectrum of the delay timeseries.

Note that we do not detect outliers based on standard statistical techniques (see [23]). The reason is that those approaches can easily misclassify as outliers the periodic probing delay spikes, especially if the latter are larger than the delays induced by cross traffic. Instead, we develop an outlier detection method that considers the expected number of probing delay spikes.

We start by detecting outliers in the probing timeseries. Suppose initially that a single prober of frequency f_p is present in the probing delay timeseries. During Δ_e that prober will generate $f_p\Delta_e$ packet trains, each of them causing a delay increase. The basic idea of the method is to estimate an *outlier threshold* χ so that any delay spike (i.e., a continuous sequence of samples) that exceeds χ is considered an outlier, *subject to the constraint that the number of detected outliers should be much less than* $f_p\Delta_e$. In other words, χ is chosen so that it is higher than the magnitude of most probing delay spikes. If χ was lower than the magnitude of the probing delay spikes, the number of detected outliers would be more than $f_p\Delta_e$ and we would misclassify all spikes of our probing signal as outliers. In practice, we determine χ iteratively, starting from the largest delay measurement and reducing χ in each iteration until the number of detected outliers is more than $h\, f_p\Delta_e$, where $0 < h < 1$. Note that the previous method does not assume that the given prober actually has a shared bottleneck with the sampler; the important point is that, *in case there is such sharing and the corresponding signal is present in the probing timeseries*, we do not misclassify all probing delay spikes as outliers. Second, when multiple probers are active we need to consider the lowest probing frequency because that gives the minimum number of delay spikes we are allowed to detect.

We have found empirically, through several Internet experiments, than an appropriate value for h is around 40%. Such a high value of h means that, in the absence of outliers, we may detect a significant fraction of probing spikes as outliers; this is acceptable because, as discussed in the next paragraph, we truncate the amplitude of any detected outliers instead of completely removing them.

After we classify a delay spike as an outlier, we need to modify it somehow so that it does not affect the estimated spectrum. Because it is likely, however, that a fraction of the detected outliers are probing spikes, we do not want to completely remove the corresponding samples from the timeseries. Instead, we "truncate" the spike by replacing each sample of that outlier with the previous sample that is less than the threshold χ. Then, when we construct the delay increase function $d_\delta(j)$, all the successive samples that are equal to χ give zero difference. So, only the first value of the spike, which is at most χ, will remain in the delay increase timeseries.

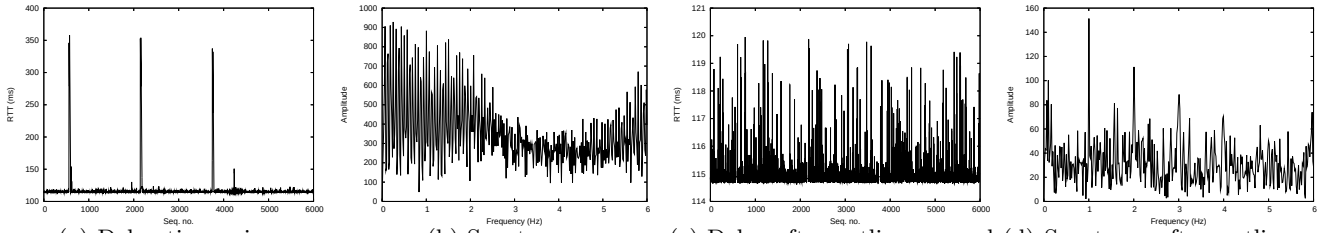
After we have detected the threshold χ from the probing timeseries, we apply the same outlier detection and truncation process on the preprobing timeseries. Finally, we calculate the delay increase function for both timeseries based on Equation 9.

3.4 Spectrum estimation and leakage reduction

Next, we estimate the amplitude $D_\delta(f)$ of the Discrete Fourier Transform (DFT) of the delay increase timeseries. The basic idea is that if the probing timeseries includes a periodic component due to prober i, we will see a significant spectral amplitude at frequency $f_p^{(i)}$ relative to the surrounding frequencies.

To calculate $D_\delta(f)$ we rely on the FFT algorithm, as implemented in Matlab. Recall that if a timeseries consists of M measurements collected with a sampling frequency f_s, then the FFT of the timeseries also consists of M frequency points covering the range from 0 Hz (DC component) to f_s with constant spacing f_s/M. The spectrum is symmetric around $f_s/2$, and so we only focus on the range $[0, f_s/2)$.

A well-known issue in the analysis of time-limited signals is *spectral leakage*. Because of this effect, even a sinusoid with frequency f_p (say an integer multiple of f_s/M) can have some power in frequencies other than f_p. A standard



(a) Delay timeseries (b) Spectrum (c) Delay after outlier removal (d) Spectrum after outlier removal

Figure 5: The significance of outlier removal in spectral analysis. The delay timeseries resulted from ping measurements on the Internet path procyon.cc.gatech.edu to mail-gw.izm.fraunhofer.de.

approach to reduce the effects of spectral leakage is to multiply the timeseries with a Hamming window before computing the FFT coefficients [3]. The Hamming window smooths the timeseries at its two boundaries, reducing the leakage of spectral power to adjacent frequencies. This is important in the proposed method, where the objective is to detect the presence of a periodic signal of a given frequency comparing the power of that frequency with the power of surrounding frequencies[6]. Figure 6 shows the effects of leakage reduction when we apply a Hamming window with the same duration as the timeseries.

To avoid leakage from high-frequency content that we are not interested in, we also perform low-pass filtering on the received timeseries. The cutoff frequency of the filter is 0.5Hz higher than the H-th harmonic of the maximum probing frequency. The filter that we use is a windowed linear-phase FIR filter of order 200. We also remove the DC component of the spectrum, by subtracting the average from the timeseries.

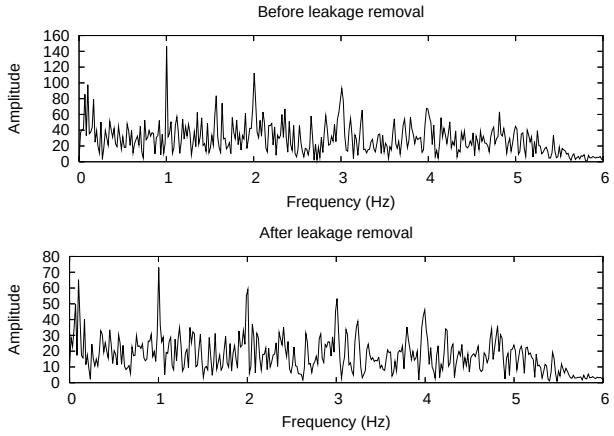


Figure 6: Effect of leakage reduction on the spectrum of a probing timeseries from the Internet path procyon.cc.gatech.edu to mail-gw.izm.fraunhofer.de.

[6]It should be mentioned that there are more sophisticated methods to estimate the spectrum of a timeseries [20]. Those methods are typically classified as non-parametric (e.g., Welch or multitaper) or parametric (e.g., Yule-Walker or Burg). We found that the DFT-based method is simpler and faster, while other methods are sometimes sensitive to the selection of their parameters.

3.5 Signal detection at a single frequency

The next step is to examine the estimated spectrum $D_\delta(f)$ and infer whether a significant periodic signal exists at the fundamental frequency $f_p^{(i)}$ or at the first H harmonics $(h+1)f_p^{(i)}$ $(h = 1, \ldots H)$, for any prober i.

Let us first consider a single frequency f_p, assuming it is the fundamental frequency of a certain prober. To examine whether a periodic signal exists at that frequency with a controlled false positive probability, we need to estimate the a priori distribution of $D_\delta(f)$ around f_p. We do so using the preprobing timeseries. In that timeseries we know that none of the probers is active, and so any power around f_p will be due to cross traffic noise.

Specifically, let $\tilde{D}_\delta(f)$ be *the spectrum of the preprobing delay increase timeseries.* A naive idea would be to detect the signal if $D_\delta(f_p)$ is much larger than $\tilde{D}_\delta(f_p)$. To make this inference with a given false positive probability, however, we need to consider $\tilde{D}_\delta(f_p)$ as a random variable in an ensemble of preprobing spectra. Let $\Omega(f_p)$ be that random variable. Our aim is to estimate a percentile of $\Omega(f_p)$ such that the false positive probability is, for instance, at most 20%,

$$\text{Prob}[\Omega(f_p) > \omega_{.2}] = 0.2 \tag{11}$$

where $\omega_{.2}$ is the 20% upper percentile. Then, we can infer that a probing signal exists at f_p, with a 20% false positive probability, when the probing spectral power at f_p is higher than $\omega_{.2}$. Later in this section we show how to further reduce the false positive probability by considering several harmonics.

To estimate the percentile $\omega_{.2}$, we consider a frequency band of width 1Hz centered around f_p in the preprobing spectrum. We then estimate the empirical CDF of $\tilde{D}_\delta(f)$ in that frequency band, and use that as an estimate of the CDF of $\Omega(f_p)$. The threshold $\omega_{.2}$ is then calculated as the 20-th upper percentile of that empirical CDF.

To have a more accurate estimate of the signal power $S(f_p)$ at f_p we consider a narrow frequency band of width 0.033Hz centered around f_p, instead of considering a single frequency point at f_p. The reason is that there may be a mismatch between the exact probing frequency at the sending host and f_p. Further, due to operating system jitter, it is possible that the prober does not transmit its packet trains at a constant frequency f_p. In practice, we observed that a frequency band of 0.033Hz is sufficient to capture the total power of the probing signal.

Finally, we calculate the SNR at frequency f_p as

$$\text{SNR}(f_p) = \frac{S(f_p)}{\omega_{.2}} \quad (12)$$

If $\text{SNR}(f_p) > 1$, we infer that a signal exists in that frequency with false positive probability 20%. The next step is to combine the inference outcome for the fundamental frequency f_p with the inference for the first H harmonics of that frequency.

3.6 Signal detection at multiple frequencies

It is possible that a probing signal is hidden in noise at its fundamental frequency f_p. Fortunately we can also rely on the harmonics of that frequency, which appear at the integer multiples of f_p. Specifically, we consider the first H harmonics of f_p. In general, the signal power decreases quickly in higher harmonics and after a certain harmonic the signal power is practically zero. We examine the effect of H on the accuracy of the method in the next section.

Given a certain H, we apply the previous frequency detection method at the fundamental frequency as well as at the first H harmonics. If H is even, the total number of detection outcomes is H+1 and there is no possibility for ties. Finally, *a signal is detected if the majority of the H+1 detection outcomes were positive.*

Even though the false positive probability for a single frequency is high (20%), the *global false positive probability* is much lower. Suppose that in reality there is no sharing and so there is no probing signal. Then, if we treat each of the H+1 detection outcomes as independent Bernoulli trials with detection probability 20%, a false positive will occur if we get at least H/2+1 individual false positives. For H={4, 6, 8} the global false positive probability is {0.027, 0.017, 0.01}, respectively.

4. EVALUATION

In this section we rely on simulations to evaluate the accuracy of the proposed shared bottleneck detection method, and to examine the impact of certain key parameters such as the train length L and the number of harmonics H. Simulations allow us to evaluate the method in a controllable and repeatable manner, which would not be possible with Internet experiments. The simulation setup is described in the Appendix. Unless if specified otherwise, the sampling frequency is 1KHz and the minimum probing frequency is 1Hz. To measure the False-Negative (FN) and False-Positive (FP) detection rates, we run each simulation 96-110 times. This gives us a margin of error that is less than 10% with 95% confidence.

4.1 Static train length

We first examine the FN rate in the case of a single prober that has a shared bottleneck with the sampler, when we use probing trains of length L. Figure 7 shows the FN rate for three values of L. As expected, as the utilization increases the delay variations induced by cross-traffic increase. So, for a fixed probing train length, the higher the utilization is, the lower the SNR becomes. When the SNR is much less than one, the detection method fails to identify the probing "spikes" in the noisy spectrum and we see frequent false negatives. With L=35 packets, this happens when the utilization becomes larger than 40-50%. As we increase L, we effectively increase the signal power, and thus the SNR. With

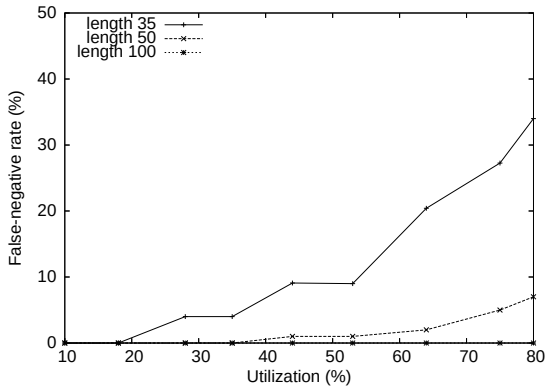


Figure 7: FN rate as utilization increases (static train length).

L=50, the FN rate is below 10% up to a utilization of 80%. With L=100, the FN rate is consistently zero in the entire utilization range.

On the other hand, as we increase the probing train length we also increase the intrusiveness of the tool, in terms of larger delay spikes and higher average probing rate. The previous observations raise the following question: *is it possible to automatically adjust L so that the method is both accurate and non-intrusive?* We examine this question in the next paragraph.

4.2 Adaptive train length

The basic idea in this variation of the method is to select a packet train length that is large enough to cause queueing delays of about the same magnitude as the 95-th percentile of the queueing delays induced by cross-traffic. In doing so, we expect that the method will be both accurate (because the probing delays will be higher than almost all cross-traffic delays) and non-intrusive (because we do not cause queueing delays that are larger than the larger delay spikes caused by cross-traffic).

To do so, we measure the distribution of delay variations (end-to-end measured delay minus the base-level minimum delay in the path) during the preprobing phase. Then, we use the 95-th percentile $d_{n,.95}$ of that distribution to estimate the desired train length L, as $L = d_{n,.95}C$, where C is a rough estimate of the capacity in the shared bottleneck. A lower bound estimate of C is the minimum of the end-to-end capacity of the sampler and prober paths. These capacities can be measured using one of the existing packet-pair methods, such as bprobe, pathrate or capbrobe. We expect that in practice the user (who may be a network manager) will often know the capacity of the potential shared bottleneck between the sampler and a prober (e.g., the access link of the user's campus network).

To not end up with values of L that are too low (when the cross-traffic delay variations are very small) or too high (in heavily loaded or low-capacity paths), we bound L so that it is not less than 35 packets and not higher than 100 packets.[7] In summary, the adaptive train length variation is described

[7]These minimum and maximum train lengths were chosen empirically, based on our simulations and Internet experiments.

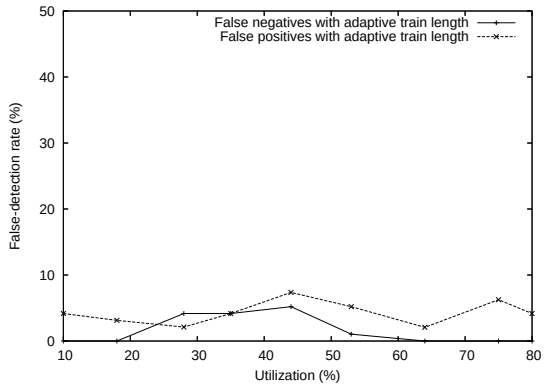


Figure 8: FN and FP rate as utilization increases (adaptive train length).

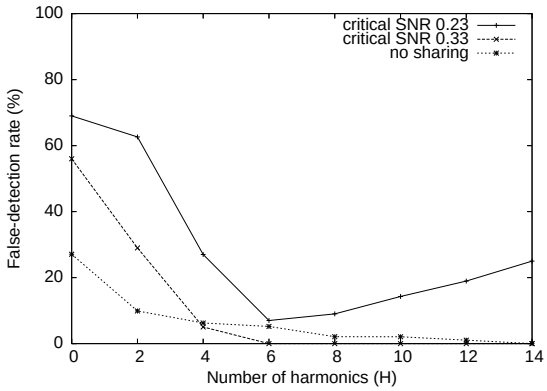


Figure 9: Effect of the number of harmonics H.

by the following equation

$$L = \max\{35, \min\{100, d_{n,.95}C\}\} \quad (13)$$

Figure 8 shows the FN rate as well as the FP rate ("no sharing" curve) when we use the previous adaptive train length variation. Here, we assume that the user knows the actual capacity of the shared bottleneck. Note that *both false detection rates are persistently below 10%.*

4.3 Number of harmonics

We next examine the effect of the number of harmonics H in the signal detection process. Recall that the method detects a signal when the majority of the H+1 frequencies (H harmonics and the fundamental frequency) show the existence of a signal in the corresponding spectral band. In the following experiment, we have a single prober that either shares a bottleneck with the sampler (to measure FN rate) or that does not share a bottleneck (to measure FP rate). In the former, we set the train length L to a low value (35 packets) so that there is a clear dependency of the FN rate to the number of harmonics.

Figure 9 shows the FN rate when there is sharing, and the FP rate when there is no sharing. Note that the FP rate drops as the number of harmonics increases. This is expected because as H increases it becomes less likely that the majority of the H+1 frequencies will randomly, due to cross-traffic noise, happen to have a spectral spike at the corresponding spectral band.

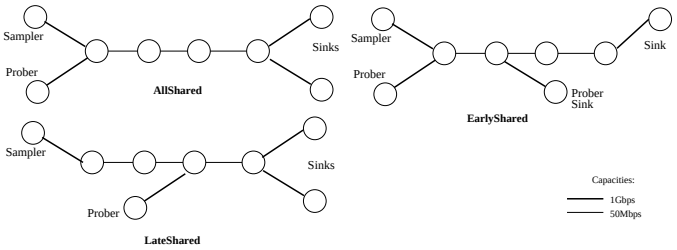


Figure 10: Topological variations with multiple bottlenecks.

The FN rate, on the other hand, initially decreases, until H becomes 6, and then it starts increasing again. The reason for the initial decrease is that as we use more frequencies, it becomes less likely that our signal will be masked by noise at the majority of those frequencies. As H increases, however, the signal power (as well as the noise power) decrease. After a certain point there is practically "no signal left" in those higher harmonics, and the resulting detection outcomes become random, causing a slow increase of the FN rate. Through many simulation experiments with various train lengths, we observed that *the optimal value of H actually varies between 4 to 8.*

4.4 Multiple shared bottlenecks

So far we assumed that the sampler and prober share a single bottleneck. Here we consider several variations of our base topology (see Appendix) to examine what happens when the paths share multiple bottlenecks ("All-Sharing"), or when the shared bottleneck appears before ("Early-Sharing") or after ("Late-Sharing") other non-shared bottlenecks. The three topological variations are shown in Figure 10. Each of the three bottlenecks (shared on not shared) are equally loaded with cross-traffic and of the same capacity (50Mbps). In these simulations we use the adaptive train length formula and H=4. The results are shown in Figure 11. Note that *the false detection rates are consistently below 10%, independent of the location of the shared bottleneck or of the number of shared bottlenecks.* It should be mentioned that the FP rate can be further decreased by decreasing the FP threshold for each harmonic (now set to 20%); that may increase the FN rate, however. The FN rate can be further decreased by increasing the train length, making the tool more intrusive though.

4.5 Multiplexing probing frequencies

In this last subsection, we examine the accuracy of the method when a number of probers are active at the same time. Some of them share the same (single) bottleneck with the sampler, while the rest do not share a bottleneck with the sampler. We measure the FN rate for the former and the FP rate for the latter. Again, we use the adaptive train length formula.

In the first experiment, there are N=11 probers, each of them using H=4 harmonics. Six probers share the sampler's bottleneck, and five do not. Table 1 shows the false detection rates. Note that despite the high multiplexing degree and the large number of allocated frequencies (6 times (4+1)=30 frequencies in the probing spectrum), the method is remarkably accurate in terms of both false positives and negatives.

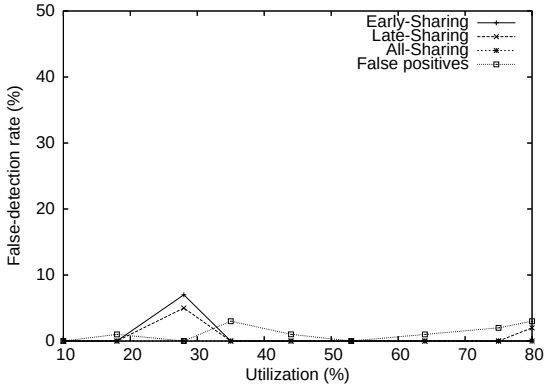


Figure 11: FN and FP rate as utilization increases in the three topological variations of Figure 10.

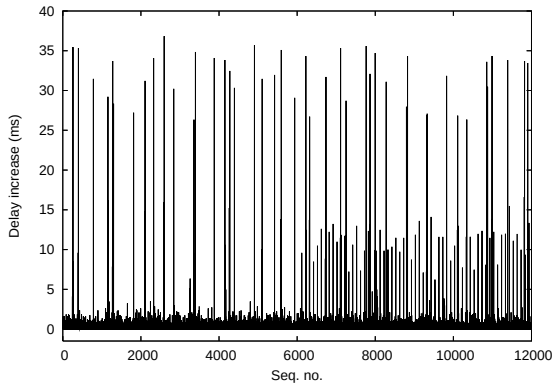


Figure 12: Traceroute does not detect sharing but the second half of the timeseries shows clear probing signal: GT to ODC paths.

In the second experiment, we have N=8 probers, each of them using only H=2 harmonics. Four probers share the sampler's bottleneck and four do not. Table 2 shows the false detection rates. Note that the false detection rates are higher, because we are using a very small number of harmonics. On the other hand, the fact that we use a wider guard band (166.7mHz vs 85.5mHz) does not amortize the negative effect of the low H value. In general, it is more important to have enough harmonics than to further isolate a small number of harmonics with a larger guard band.

Sharing		No sharing	
Funder-Freq	FN rate	Fundam-Freq	FP rate
1.0Hz	0%	2.18Hz	1.82%
2.7Hz	0%	3.1Hz	3.64%
3.43Hz	0%	3.55Hz	0%
3.7Hz	0%	3.85Hz	0%
4.1Hz	0.9%	4.55Hz	0%
4.7Hz	0%		

Table 1: 11 probers, 4 harmonics, 85.5mHz guard band.

Sharing		No sharing	
Fund-Freq	FN rate	Fund-Freq	FP rate
1.0Hz	1%	3.18Hz	1%
3.4Hz	0%	3.6Hz	10.1%
3.78Hz	0%	4.18Hz	0%
4.4Hz	1.03%	4.6Hz	1.01%

Table 2: 8 probers, 2 harmonics, 166.7mHz guard band.

5. INTERNET EXPERIMENTS

In this section we present some representative results from Internet experiments. A major issue with such experiments is that we cannot always know the ground truth, and so the objective of this section is not to conduct a large-scale measurement study. Nevertheless, in some cases we are certain that there is sharing because we see a clearly visible probing signal in the sampler measurements. In other cases we are confident that there is no sharing because, according to traceroute at least, the two paths go through different ISPs and reach different destinations.

Instead of relying on one-way delay measurements, which would require access at both ends of a path, the following experiments use a ping-like utility that we wrote for Linux. The tool, called *SharedBneck*, can run as prober or sampler, sending ICMP ECHO packets in both cases. Because it measures RTTs, it is subject to noise at the reverse-path. The source code for SharedBneck will be made publicly available. Note that the tool does not require any special timers or other non-standard operating system features. The kernel timestamps are collected using the *SIOCGSTAMP* ioctl call on Linux and they have a resolution of 1 microsecond. The transmission of periodic packet trains is scheduled using the *select* timeout, while the clock interrupt period of all hosts we experimented with is 1ms (allowing a sampling frequency of 1KHz).

Unless stated otherwise, the following experiments use the adaptive train length variation with a sampling frequency of 1KHz and a probing frequency of 1Hz. The capacity of the shared bottleneck is estimated as the minimum of the sampler/prober end-to-end capacities. The prober/sampler sources are located at Georgia Tech's Atlanta campus. We use `procyon.cc.gatech.edu` for the sampler. Unless otherwise stated, we use `foofoo.rnoc.gatech.edu` for the prober, located in a different L2 network from the sampler. The destinations are in various countries and the end-to-end paths cross multiple commercial and academic networks such as Abilene, GEANT2, Qwest and Cogent.

Detection of sharing when traceroute fails: In this pair of paths P_1 from `GT` to Oracle's data center in Texas (ODC) (destinations: sampler `bigip-wbw-adc.oracle.com`; prober `bigip40-roi-v1.oracle.com`), the prober/sampler traceroute outputs do not show sharing. Actually the traceroutes do not complete, probably because of a firewall or NAT at the destination network, which is the same for both the prober and the sampler. The output of traceroute shows the sampler AS path going through Qwest and Level3, while the prober AS path traverses Cogent and Global Exchange. Figure 12 shows the delay (RTT) increase function for both the preprobing (up to sequence-number 6000) and probing phases. Notice the periodic spikes of the probing signal (delay increase of about 10-15msec) at the right half of the plot. This means that there is certainly sharing between the two paths. Indeed, SharedBneck detects it with high SNR values.

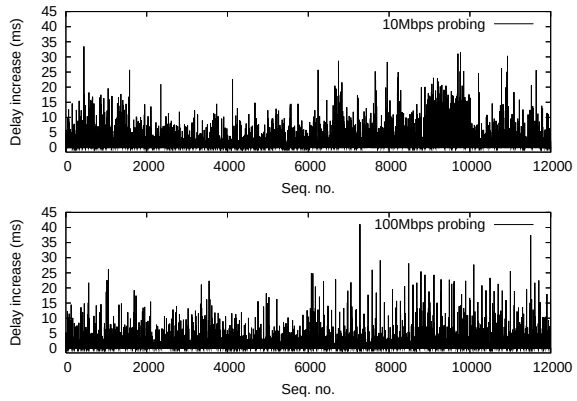


Figure 13: A low probing rate may fail to detect sharing: GT to Insead paths.

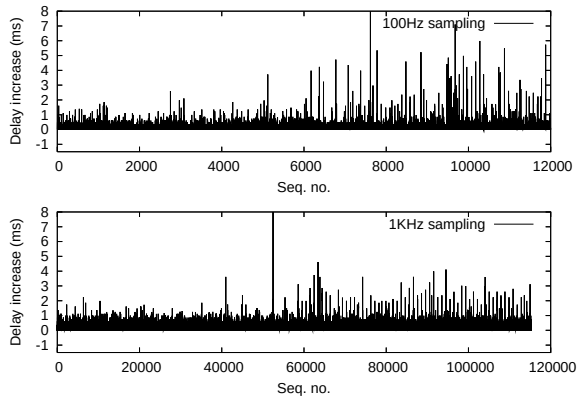


Figure 14: A low sampling frequency may fail to detect sharing: GT to UCY paths.

Effect of probing rate: This experiment illustrates the importance of a sufficiently high probing rate. In this pair of paths P_2 from `GT` to `Insead`'s campus in France (destinations: prober `knowledge.insead.edu`; sampler `faculty.insead.edu`), we do not see a signal when the prober sends trains at a rate of 10Mbps, and SharedBneck reports "no sharing". When we increase the transmission rate at the prober to 100Mbps, there is a clear signal and SharedBneck reports "sharing" with high SNR values. The AS paths traversed Qwest and ALTER.NET ASes. Figure 13 shows the delay increase function timeseries for both the preprobing (first half of the plots) and probing phases with both probing rates.

Effect of sampling frequency: This experiment illustrates the importance of a sufficiently high sampling frequency. In this pair of paths P_3 from `GT` to University of Cyprus's (`UCY`) CS department (destinations: prober `www2n.cs.ucy.ac.cy`; sampler `pac2.cs.ucy.ac.cy`), we do not see a very clear signal when the sampling frequency is 100Hz, and SharedBneck often gives false positives. When we increase f_s to 1KHz there is a clear signal at the second half of the timeseries (probing phase) and SharedBneck reports "sharing" with high SNR values. The sampler and prober paths traverse the high-capacity Abilene and GEANT2 backbones before reaching `UCY`. Figure 14 shows the two delay increase functions.

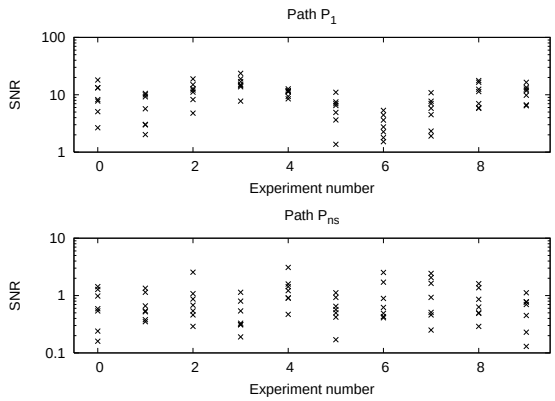


Figure 15: SNRs across harmonics and experiments for GT to ODC paths.

SNR variability across harmonics and experiments: We now present results from four cases of sharing (the previous path pairs P_1, P_2, P_3 and one more path P_4 in the `fraunhofer.de` network[8]), as well as a case of no-sharing P_{ns}. In the latter, we sample without probing on the noisy pair of paths P_1 (from `GT` to `ODC`), to make sure that there is no probing signal. We repeated each experiment 10 times, at different times on the same weekday, on multi-processor and single-processor sampler environments, for a total of 50 experiments. The important parameter values for all experiments were: L=35 packets, f_p=1Hz, f_s=1KHz, H=6 harmonics.

Out of the 50 experiments, we detected only two false detections; one false negative on P_3 and one false positive. Figure 15 shows the seven (1+H) SNRs for each experiment in two of the path-pairs, P_1 and P_{ns}. In P_1 there is sharing, and this is why most SNR values are much larger than one. Note however the significant variability of the SNR values across experiments, and also across harmonics in the same experiment.

Experiment with multiple probers: Here, we perform a three-prober experiment. In addition to the aforementioned sampler and prober (labeled A), we choose the other two probers as `vega.cc.gt.atl.ga.us` (B) and `sirius.cc.gatech.edu` (C). The sinks are chosen in the `fraunhofer.de` network (including paths P_4). The key parameter values are: L=35, f_s=100Hz, g= 133.3mHz, H=4, while the maximum allowed probing rate is R_{max} =1Mbps. The fundamental frequencies of the three probers are 1Hz, 1.72Hz, and 1.87Hz. We refer to the three probers as A, B, and C, respectively. The experiment consists of seven phases, each lasting Δ_e=60 seconds. In each phase a different combination of probers is active (see Figure 16). For instance, in the first phase only A is active.

We did not observe any false negatives or false positives for any prober during the experiment.

Figure 16 shows the resulting five (1+H) SNRs for each prober (in the order A, B, C), in each phase of the experiment. Notice that when a prober is active, its SNR values are mostly (but not always) larger than one. The plot also shows the large variability of SNRs across different harmonics.

[8]destinations: prober `pps.izm.fraunhofer.de`; sampler `www.izm.fraunhofer.de`

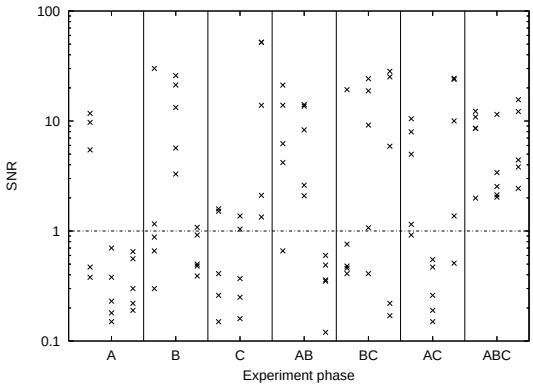


Figure 16: Three-prober experiment with seven combinations of active probers: GT to fraunhofer.de paths.

6. DISCUSSION

The previous two sections have examined the accuracy of the SharedBneck method, focusing on the significance of the sampling and probing frequencies and of the probing train length. In this section, we further discuss various other factors and conditions that can affect the accuracy of SharedBneck.

First, the definition of a shared bottleneck in Section 2.5 requires that the prober is able to cause queueing in the link that it shares with the sampler. If there is a shared link, but the prober path is such that the probing trains arrive at the shared queue with a rate that is less than that queue's available capacity, SharedBneck will not be able to detect sharing. The practical implication is that probers should be sending their trains at the maximum possible rate and that the hosts that serve as probers should have high-capacity interfaces.

It is also important that the measured paths do not change in terms of the underlying routing and traffic conditions. In our simulations and experiments, we found that simple tests for traffic stationarity, such as major level-shifts, were sufficient, and did not require more elaborate tests for stationarity in the strong or weak mathematical sense. It is important however that, if there is a shared bottleneck, that link should remain in that state during the measurements despite any available capacity fluctuations in the probing and sampling paths. For this reason, we require "operational constancy", as this notion was described by Zhang et al. in [25], and we examine for violations of this property by detecting level-shifts in the measured timeseries.

The assumption of lossless operation, stated in Section 2.1, is necessary for that model but it is not that important in practice. We have observed that SharedBneck is accurate even when there is noticeable loss rate (say 1%).

The adaptive train length method needs a capacity estimate for the potential shared bottleneck. A lower estimate for that capacity can be obtained with capacity measurements in the sampler and prober paths. In some cases, the SharedBneck user may know the capacity of the potential shared bottleneck when the measured paths are in the network that he/she manages.

Even though we pay attention to the intrusiveness issue, by limiting the probing train length, the method may still be viewed as intrusive. We have not explored whether SharedBneck can cause any performance degradation to other flows, either TCP or jitter-sensitive VoIP flows. We doubt that this would be the case, given the relatively small probing frequency (typically once per second) and the short duration of the probing delay spikes.

The preprobing phase may show that there is significant power in the same probing frequencies that SharedBneck uses. This may happen, for instance, if two different users run SharedBneck at the same time. One option is to abort the measurements after the preprobing phase. Another approach is to add some randomization in the probing frequencies, so that different users rarely overlap in the frequency domain. A third approach is to choose the probing frequencies so that they do not overlap with the observed frequencies in the preprobing spectrum.

Finally, as any other measurement tool that is more than an idealized model, SharedBneck involves several parameters, such as the SNR threshold γ, the guardband g, the duration of the measurements Δ_e, the outlier-detection parameter h, the low-pass filtering parameters, the minimum and maximum probing train length bounds (35 and 100), the sampling packet size, the false-positive probability for a single harmonic (20%), or the accuracy of the capacity estimate C. In this paper, we have investigated the effect of the probing and sampling frequencies (f_p and f_s), of the probing train length L and of the number of harmonics H. Due to space constraints we cannot present results with different values for the rest of the parameters. An extensive robustness study for the accuracy of SharedBneck in this multi-dimensional parameter space would be difficult. We cannot claim that the parameter values we give in this paper are "optimal" in any sense. They have been working well, however, in all our experiments and simulations, and for this reason we expect that the user would not have to fine-tune the SharedBneck parameters before running the tool in practice.

7. RELATED WORK

We discuss related work in two contexts:

Spectral analysis of Internet traffic: He et al. analyzed the spectral characteristics imposed by bottleneck links on aggregate traffic [11, 12]. Specifically, they applied frequency-domain analysis on the interarrivals of a packet trace. The basic idea is that a bottleneck link imposes distinct signatures on the underlying traffic; their method detects those signatures at a downstream monitoring point, attempting to estimate the capacity of upstream bottlenecks. Research at BBN also used spectral analysis techniques to extract timing information, such as TCP connection RTTs or topological information from aggregated traffic traces [8]. This work uses the SP framework's notion of crosstalk to detect mobility in a wireless network. Note that the previous approaches are based on offline analysis of passive measurements, while our method is based on active probing in the frequency domain. A 2003 patent from Partridge and Cousins [21] proposed covert channels in packet networks using hidden frequencies in the packet interarrivals.

Broido et al. used spectral techniques to detect periodicities in the timeseries of DNS updates for private (RFC-1918) addresses received by an authoritative nameserver [4]. Signal processing methods have also been used for efficient detection of traffic anomalies such as flash crowds, denial-

of-service attacks and network outages [2]. In [14], Hussain et al. used spectral analysis to detect and classify denial-of-service attacks as single and multi-source attacks. Related work in this problem includes [7] and [13].

San-qi Li and his group have worked extensively on the spectral analysis of queueing processes (see [18] and references therein). An important result of their analytical work is that, in any queueing system, there is a break frequency, under which the low-frequency traffic stays intact as it crosses the queueing system. The characteristics of this low-frequency traffic can have a significant impact on downstream queues. In some sense, our work is related to this result given that the probing signals we create can go through a sequence of queues as long as their frequency is sufficiently low (see Equation 5).

Tomography and the shared congestion problem: Network tomography aims to infer internal link characteristics using end-to-end measurements. Scalar tomography attempts to infer metrics, such as the loss probability or the delay of a network link. Boolean tomography aims to infer a binary state ("good" versus "bad") for each network link. A recent work focused on the identifiability problem in scalar tomography [6], and it used a Fourier domain technique to infer link delays. Rabbat et al. proposed a multiple source tomography scheme that can be used to detect sharing (at any store-and-forward device) among two paths [22]. Their technique is based on the timing characteristics which which packet-pairs arrive at the two destinations. That method is based on time-domain analysis, while ours is based on frequency-domain analysis.

The problem of identifying *shared congestion* among two or more paths has received significant attention in the last decade [10, 15, 16, 17, 24]. The basic idea in most related techniques is to examine the cross-correlation of end-to-end delays, losses or interarrival measurements between paths to infer whether those paths share the same bottleneck. Our approach is significantly different because the shared link does not need to be congested in order to detect path sharing. The only constraint is that the prober should be able to cause queueing at the shared queue when it sends probing trains.

8. CONCLUSIONS - FUTURE WORK

The contributions of this paper are twofold. First, we introduced the Spectral Probing framework in which fundamental frequency-domain concepts from signal processing and analog communications are applied in active measurements and network tomography. The concepts we explored here relate to frequency modulation of binary information, frequency multiplexing and crosstalk. In future work, we will investigate more advanced spectral methods for signal transmission, modulation, and detection in the presence of noise, such as channel coding and spread-spectrum techniques, all in the context of queueing delay variations in IP paths.

Second, we proposed and evaluated one application of the SP framework, namely the detection of shared bottlenecks between two or more paths. This is an important application in practice as it can reveal the presence of shared store-and-forward devices (switches, routers, middleboxes) between end-to-end paths. The proposed method can detect sharing even if the shared device is not congested, as long as the prober(s) can cause short-term queueing at the corresponding shared buffers. Our Internet experiments have shown that the method is effective in discovering shared bottlenecks between paths, even when those links are invisible to traceroute.

In future work, we will apply the SP framework in more problems and applications. Next, we briefly describe two applications we are currently working on. First, we can use SP to create *covert channels* between hosts that are not allowed to exchange any data traffic or that are under surveillance. Consider paths P(A,B) and P(C,D) and suppose that they have one or more shared bottleneck queues. A can generate delay signals of different frequencies, which can be detected at the delay measurements from C to D. Notice that A does not send any traffic to C or D. D detects A's signal simply from the spectral characteristics of the delay variations in path P(C,D). We have verified with Internet experiments that this application is feasible in practice.

Another application is based on the notion of *frequency modulation* in analog communications. Recall that this type of modulation translates the spectrum of a *base signal* in the frequency domain, so that the modulated signal is more robustly transmitted (and multiplexed) on a given channel. Similarly, the base signal in our context can be any periodic packet stream, such as a voice or video stream. Such applications typically generate one packet every few tens of milliseconds. That frequency, however, may not be ideal for a given IP path. Queueing delay variations (jitter), because of other traffic between A and B or because of delay crosstalk from other paths, can be detrimental to the performance of a voice or video stream. One possibility is to modify the video transmission rate (at the source itself or at an application gateway at the edge of the source network) to a frequency in which path P(A,B) experiences less jitter (spectral noise). Frequency increase can be achieved by segmenting video packets into smaller packets, while a frequency decrease can be achieved by aggregating multiple consecutive video packets.

Appendix: Simulation Setup

The NS2 simulation topology is shown in Figure 17. Unless if stated otherwise, there is a single shared bottleneck with 50Mbps capacity. The topology includes N prober paths, a sampler path and several cross-traffic paths that are aggregated in the shared bottleneck. Unless if stated otherwise, the N prober paths have a shared bottleneck with the sampler.

The capacity of all links that feed into the shared bottleneck is 500Mbps. The access capacity of the probers and the sampler is 1Gbps. The access capacity of the servers is chosen uniformly as either 100Mbps or 1Gbps. The access capacity of the users (clients) is also chosen uniformly as 1, 10 or 100Mbps. The buffer size of the shared bottleneck is set to the bandwidth-delay product of that link (250 packets).

The cross-traffic in the forward path is generated by U=100 users. Each user goes through a cycle of "download" and "think" phases. In the download phase, the user receives a file through a TCP connection from a randomly selected server. We use TCP NewReno with the SACK option enabled. The receive window is set to a large value so that the flows are only limited by the congestion window. The file size distribution is Pareto with mean 80KB and shape parameter 1.5, creating LRD traffic. The duration of the think

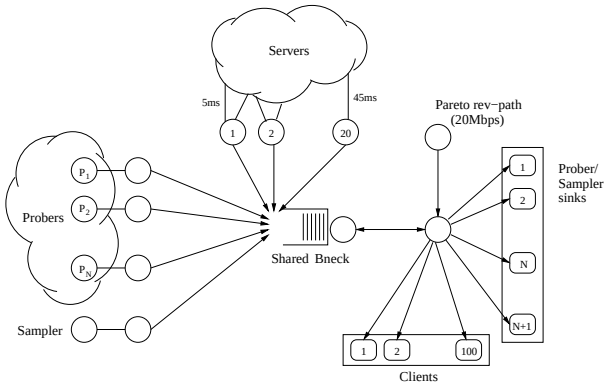


Figure 17: Simulation topology.

phase is exponentially distributed. Its duration is controlled to achieve a desired utilization at the shared bottleneck. The RTTs of the TCP connections vary between 30ms to 110ms, due to the heterogeneity in the propagation delays of the server access links.

We also generate some cross-traffic in the reverse path (same direction with the client ACKs) using a packet-level Pareto renewal process. The average rate of that Pareto source is 20Mbps.

Acknowledgments

We are grateful to Alan Clark and Paul McMenamin from Telchemy for their input during the early phases of this work.

9. REFERENCES

[1] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris Traceroute. In *Proc. ACM SIGCOMM IMC*, Rio de Janiero, Brazil, Oct. 2006.

[2] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proceedings of ACM SIGCOMM IMW*, pages 71–82, New York, NY, USA, 2002. ACM.

[3] P. Bloomfield. *Fourier Analysis of Time Series: An Introduction*. Wiley-Interscience, 2nd edition, 2000.

[4] A. Broido, E. Nemeth, and kc claffy. Spectroscopy of private DNS update sources. In *Proceedings of the Third IEEE Workshop on Internet Applications*, page 19, Washington, DC, USA, 2003. IEEE Computer Society.

[5] A. B. Carlson, P. Crilly, and J. Rutledge. *Communication Systems (4th edition)*. McGraw-Hill, 2004.

[6] A. Chen, J. Cao, and T. Bu. Network tomography: Identifiability and Fourier domain estimation. pages 1875–1883, 2007.

[7] C.-M. Cheng, H. Kung, and K.-S. Tan. Use of spectral analysis in defense against DoS attacks. In *Proceedings of IEEE GLOBECOM*. IEEE, 2002.

[8] D. Cousins, C. Partridge, K. Bongiovanni, A. Jackson, R. Krishnan, T. Saxena, and W. Strayer. Understanding Encrypted Networks Through Signal and Systems Analysis of Traffic Timing. In *Proceedings of IEEE Aerospace Conference*, Mar. 2003.

[9] I. Csabai. 1/f Noise in Computer Network Traffic. *Journal of Physics A*, A27(L417-421), 1994.

[10] W. Cui, S. Machiraju, R. H. Katz, and I. Stoica. SCONE: A tool to estimate shared congestion among Internet paths. *UCB Technical Report UCB/CSD-04-1320*.

[11] X. He, C. Papadopoulos, J. Heidemann, and A. Hussain. Spectral characteristics of saturated links. Technical Report USC/CS-TR-2004-827, USC, June 2004.

[12] X. He, C. Papadopoulos, J. Heidemann, U. Mitra, U. Riaz, and A. Hussain. Spectral analysis of bottleneck traffic. Technical Report USC/CS-TR-2005-853, USC, June 2005.

[13] P. Huang, A. Feldmann, and W. Willinger. A non-instrusive, wavelet-based approach to detecting network performance problems. In *Proceedings of ACM SIGCOMM IMW*, pages 213–227, New York, NY, USA, 2001. ACM.

[14] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of ACM SIGCOMM*, pages 99–110, New York, NY, USA, 2003. ACM.

[15] D. Katabi and C. Blake. Inferring congestion sharing and path characteristics from packet interarrival times. Technical Report MIT-LCS-TR-828, MIT, 2002.

[16] M. S. Kim, T. Kim, Y. Shin, S. S. Lam, and E. J. Powers. A wavelet-based approach to detect shared congestion. In *Proceedings of ACM SIGCOMM*, pages 293–306, New York, NY, USA, 2004. ACM.

[17] M. S. Kim, T. Kim, Y. J. Shin, S. S. Lam, and E. J. Powers. Scalable clustering of Internet paths by shared congestion. *Proceedings of IEEE INFOCOM*, pages 1–10, 2006.

[18] S. Li and J. Pruneski. The linearity of low frequency traffic flow: An intrinsic i/o property in queueing systems. *IEEE/ACM TON*, 5(3):429–443, jun 1997.

[19] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of IEEE INFOCOM*, 1999.

[20] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 1989.

[21] C. Partridge and D. Cousins. Systems and methods for creating covert channels using packet frequencies. United States Patent Application 20030091064, May 2003.

[22] M. Rabbat, M. Coates, and R. D. Nowak. Multiple-source Internet tomography. *IEEE JSAC*, 24(12):2221–2234, 2006.

[23] R.S.Tsay. Outliers, Level Shifts, and Variance Changes in Time Series. *Journal of Forecasting*, 1988.

[24] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM TON*, 10(3):381–395, 2002.

[25] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *ACM SIGCOMM IMW*, pages 197–211, Nov. 2001.