A Wavelet-based Approach to Detect Shared Congestion*

Min Sik Kim[1], Taekhyun Kim[2], YongJune Shin[2], Simon S. Lam[1], and Edward J. Powers[2]

[1]Department of Computer Sciences, The University of Texas at Austin
[2]Department of Electrical and Computer Engineering, The University of Texas at Austin

TR–03–51
Department of Computer Sciences
The University of Texas at Austin

November 2003

Abstract

Congestion control has been performed at a per-flow level to provide fairness and efficiency in sharing network resources. Better utilization of network resources is achievable if it is known that two different packet flows share a congested link. Such knowledge can be used to implement cooperative congestion control or improve the overlay topology of a P2P system. Previous techniques to detect shared congestion have limitations, namely: they assume a common source or destination node, drop-tail queueing, or a single point of congestion. We propose in this paper a novel technique, applicable to any pair of paths on the Internet, without such limitations. Our technique employs a signal processing method, *wavelet denoising*, to separate queueing delay caused by network congestion from various other delay variations. Our wavelet-based technique is evaluated through both simulations and Internet experiments. We show that for paths with a common synchronization point, our technique provides faster convergence and higher accuracy while using fewer packets than previous techniques. Furthermore, we show that our technique is robust and accurate without a synchronization point; more specifically, it can tolerate a synchronization offset of up to one second between two packet flows.

1 Introduction

Congestion control has been performed at a per-flow level; each flow adjusts its sending rate according to feedback regarding the congestion status of the network. The stability of today's Internet is mainly due to such congestion control, especially the additive increase and multiplicative decrease approach of TCP.

Better utilization of network resources is achievable with cooperation between flows. For example, Congestion Manager [3] examines all flows of the host where it resides, and groups flows passing through the same bottleneck link into a single flow aggregate. By performing congestion control over flow aggregates, rather than over each individual flow separately, Congestion Manager could improve fairness and efficiency significantly.

Recent proliferation of overlay systems poses a new challenge in cooperative congestion control. There are many applications of overlay systems that would benefit from cooperative congestion control, including end system multicast, file download from multiple servers, and overlay QoS routing. Such systems usually consist of a large number of end hosts and unicast flows between them. Unlike flows controlled by Congestion Manager, these unicast flows have different source and destination nodes, but still may interfere with each other by sharing one or more intermediate links. If the system can tell which flows are sharing a bottleneck link, it can improve overall performance by changing overlay topology to avoid such interference.

The basic primitive required for cooperative congestion control is to decide whether two flows are sharing a bottleneck link or not. Techniques for inferring shared congestion use two kinds of information from feedback: packet loss and delay. Techniques based on packet loss assume bursty packet loss [8, 15]. Thus they work well with drop-tail queues and lossy links. However they are slow and inaccurate with low loss rate or with other queueing disciplines such as RED. Techniques based on delay [11, 15] show more robust behavior in such an en-

*Research sponsored by National Science Foundation ANI–0319168 and Texas Advanced Research Program 003658–0439–2001.

vironment. They are adequate for the case where two flows have a common source or a common destination. The major weakness of both kinds of techniques is that they require a synchronization point, usually at a source. Thus they cannot be used for general overlay networks.

We propose a novel technique to detect shared congestion between two Internet paths. It is based on a simple observation: two paths sharing congested links have high correlation between their one-way delays. However, measuring correlation in a naive way may be inaccurate due to random fluctuation of queueing delay and mild congestion on non-shared links. In our technique, these interfering delay variations are filtered out with wavelet denoising, a signal processing method to separate signal from noise.

We evaluate our technique through extensive simulations and Internet experiments. When two paths have a common source, for which previous approaches can also detect shared congestion, our technique shows faster convergence and better accuracy with fewer packets. It takes at most 10 seconds to reach near 100% accuracy with both drop-tail and RED queues, while previous techniques take longer or fail. We also show that our technique maintains its accuracy without a synchronization point; more specifically, it tolerates a synchronization offset between flows of up to one second, which is achievable on the Internet.

The remainder of this paper is organized as follows. Section 2 describes our basic technique using cross-correlation. Section 3 introduces wavelet denoising and explains how to apply it to our technique. Section 4 addresses implementation issues, and Section 5 presents results of simulations and Internet experiments. We conclude in Section 6.

2 Basic Technique

We first present a basic technique to detect shared congestion using cross-correlation. This technique is effective when the clocks of the nodes measuring delay are synchronized and there is only one point of congestion. With this as a basis, we will develop a general technique that tolerates a large synchronization offset and allows multiple points of congestion in Section 3.

2.1 Model

Two paths sharing links on the Internet are illustrated in Figure 1. Paths X from X_{src} to X_{dst} and Y from Y_{src} to Y_{dst} are sharing links between S and T. Let the one-way delay of path X be D_X, and that of path Y be D_Y. Each of them has two components: d_S, the delay from S to T, and the remainder denoted by d_X or d_Y.

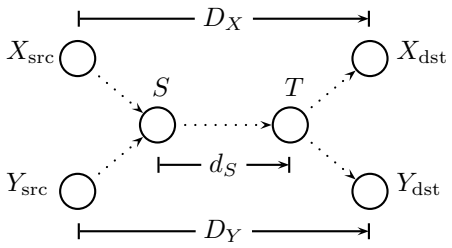

Figure 1: Two paths sharing links

$$\begin{aligned} D_X &= d_S + d_X \\ D_Y &= d_S + d_Y \end{aligned} \tag{1}$$

A key observation is that the delay of a congested link has large fluctuations due to queueing delay changes, while the delay of a link with light load is relatively stable. A persistently congested link may have stable delay with its queue always full. However, a measurement study shows that packet loss processes caused by congestion are better thought of as spikes rather than persistent congestion periods, and that most spikes are shorter than 220 ms [19]. It confirms that a congested link shows large fluctuations in delay. In order to detect shared congestion, we need to tell whether such fluctuations occur between S and T.

2.2 Cross-correlation

Suppose that paths X and Y in Figure 1 are sharing one or more congested links between S and T, and that the other links are lightly loaded. Then, since d_X and d_Y remain stable, both D_X and D_Y vary as d_S changes, showing strong similarity between them. On the other hand, if congestion occurs on links other than the links between S and T, D_X and D_Y become independent.

Our basic technique uses the cross-correlation coefficient to measure such similarity. Let $\{X_i\}$ and $\{Y_i\}$ be one-way delay sequences of paths X and Y, respectively, assuming X_i and Y_i are sampled at the same time. Then their cross-correlation coefficient $XCOR_{XY}$ is defined as follows.

$$XCOR_{XY} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 \cdot \sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{2}$$

$XCOR_{XY}$ is one if both d_X and d_Y are constant and d_S is not constant (shared congestion), and zero if d_S is constant and d_X or d_Y varies independently (no shared congestion).

One of the properties of the cross-correlation coefficient is that its value is independent of any constant

component of $\{X_i\}$ or $\{Y_i\}$ and dominated by components with large fluctuations. It matches well with our purpose to determine if any of the shared links has large delay fluctuations. Also note that, due to this property, no clock synchronization between the source and destination nodes of paths X and Y is required in measuring one-way delay between them. However, clock skew may affect measurement. We assume that the skew is minimized by other means [13].

2.3 Basic technique implementation

The basic technique consists of two stages: sampling and processing. In the sampling stage, X_{src} sends to X_{dst} a sequence of UDP packets with a timestamp, starting at time t_0 with its own clock. Each such UDP packet is called a *probe packet*. Probe packets are sent at a constant rate until $t_0 + T$, where T is the probe interval. On receiving a probe packet, X_{dst} calculates one-way delay and sends it with the original timestamp back to X_{src}. Then X_{src} records the one-way delay together with the timestamp as a delay sample. Missing samples are interpolated from neighboring samples. The sampling stage ends when the last delay sample from X_{dst} is received (or upon timeout if the last probe or the reply to it is lost). Y_{src} and Y_{dst} also collect delay samples in the same way.

In the processing stage, the cross-correlation coefficient of two sequences of delay samples is computed as defined in Eq. 2. The actual procedure to gather delay sequences collected by different nodes is application-dependent. For example, in application-layer multicast, a common ancestor node of X_{src} and Y_{src} in the multicast tree can gather and process delay sequences.

2.4 Limitations

Applicability of the basic technique is limited because of its strong requirements. Specifically, it makes two assumptions that generally do not hold for the Internet.

The first assumption is that the two delay sequences are synchronized. Ideally, the basic technique expects packets measuring X_i and Y_i to pass through S at the same time. However, it is unachievable for two reasons: the delay from X_{src} to S is likely to be different from the delay from Y_{src} to S, and the clocks of X_{src} and Y_{src} are not synchronized. There is no way for an end host to measure the one-way delay to a router (S) without support from the network. Although the clocks can be synchronized loosely by exchanging packets between two nodes, it may not be accurate enough to apply the basic correlation technique because it still allows errors up to half of the round-trip time between the nodes [12]. To quantify such synchronization errors, we define *synchronization offset* as the time difference between arrivals of two probe packets at S, one sent by X_{src} at time t with X_{src}'s clock and the other by Y_{src} at time t with Y_{src}'s clock. As the synchronization offset resulting from the two reasons stated above increases, the delay sequences collected by the two nodes show less and less correlation. Figure 2 illustrates the average cross-correlation coefficient over 300 simulations and the corresponding 5th and 95th percentile values, versus synchronization offset between delay sample sequences from two paths sharing a congested link. In each simulation, two delay sample sequences are collected for 100 seconds on the topology shown in Figure 3 using ns-2.[1] The bandwidth of every link is 1.5 Mb/s, and its delay is chosen randomly between 20 ms and 30 ms for each simulation. The delay sequences represent one-way delays of two paths, from X_{src} to X_{dst} and from Y_{src} to Y_{dst}. The congestion level is controlled by the number of background ON-OFF flows. The loss rate of the shared link is about 10%, and the other links do not have any loss. (Refer to Section 4 for ON-OFF flow parameter settings.) Without synchronization offset, the cross-correlation between them is about 0.9. However, the cross-correlation drops as synchronization offset increases so that a 600 ms synchronization offset results in half of the cross-correlation without offset.

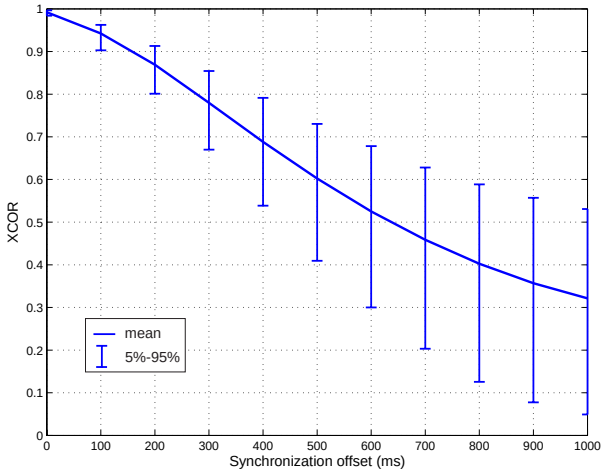

Figure 2: Cross-correlation coefficient between two delay sequences versus synchronization offset

The second assumption is that queueing delay variation on non-congested links is close to zero. If such delay variation is not negligible, it confuses the basic technique and will give an obscure cross-correlation coefficient not close to zero or one. Then it is difficult to determine the threshold to differentiate shared congestion and independent congestion cases.

In the next section, we propose wavelet denoising to

[1] http://www.isi.edu/nsnam/ns/

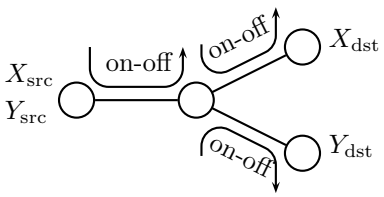

Figure 3: Simple topology with a common source

enhance the basic technique. It effectively filters out delay variations in non-congested links and short-term fluctuations that confuse the basic technique, as well as negative effects of synchronization offset. With the combination of wavelet denoising and cross-correlation, our new technique can detect shared congestion for paths with a large synchronization offset and delays at non-congested links.

3 Wavelet Denoising

In order to provide efficient solutions to network problems, various types of signal processing techniques have been employed for modeling [14] and analysis [1, 4, 10] of Internet traffic. However, they are mainly used to infer static or long-term network information from a large set of data collected over a long time span. In order to obtain dynamic information such as congestion status in a timely manner, techniques capable of on-line processing and fast response are required.

In this section, we first examine the time series of packet delay in a flow and its characteristics. Based on these characteristics, we introduce a signal processing technique—wavelet denoising [6]—that overcomes the limitations of the basic cross-correlation technique in Section 2.4. Wavelet denoising takes the original delay time series, and generates another time series with reduced interfering fluctuations that might affect cross-correlation adversely. Finally, we discuss a procedure to find the wavelet basis that maximizes the effect of the wavelet denoising technique.

3.1 Nature of delay data in time and frequency domain

Figure 4 demonstrates an example set of time series of packet delay for a link with two different traffic congestion levels. The source and destination nodes are connected through a 1.5 Mb/s link on ns-2. The delay between them was measured using UDP packets as explained in Section 2.3. The time series in Figure 4-(a) is the one-way delay under light traffic load (76 ON-OFF background flows) while the time series in Figure 4-(b) is the delay under heavy traffic load (92 ON-OFF background flows). ON-OFF flow parameter settings are identical to those described in Section 4. Observe that the one-way delay with light traffic is noise-like waveform with smaller amplitude, while the delay with heavy traffic shows an irregular pulse pattern with higher amplitude. Such pulses result from the congestion of network traffic.

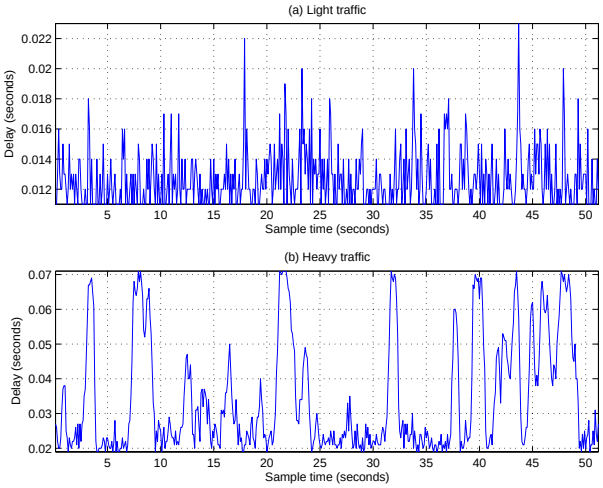

Figure 4: Time series of one-way delay of a single-hop path

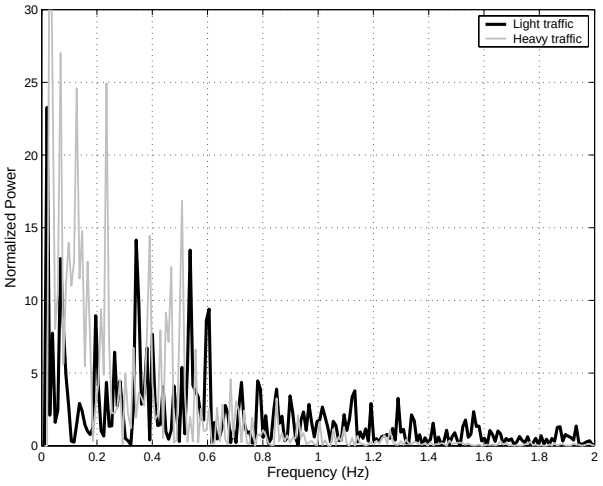

Figure 5: Power spectral densities of time series of delay data with light traffic and heavy traffic

The corresponding frequency domain power spectral densities, normalized to unity area, of the individual time series are provided in Figure 5. In the frequency domain, the delay with heavy traffic shows larger amplitude at low frequencies than the delay with light traffic. Such large amplitude components at low frequencies correspond to the irregular pulses in Figure 4-(b), caused by

congestion, while others are introduced by the randomness of the queue behavior, well-demonstrated in Figure 4-(a). Therefore, for a proper assessment of network traffic via delay data, it is necessary to reduce the effects associated with the randomness of queue behavior which corrupts the traffic delays in both the time and frequency domains. In addition, if a synchronization offset is introduced in delay sampling, the measure of network traffic via delay will be less reliable. Therefore, it is necessary to introduce a signal processing technique—wavelet analysis. It provides time *and* scale localized information of the signal with $O(N)$ complexity, where N is the number of delay samples, and thus permits us to mitigate the interfering effects of queue behavior in real time.

3.2 Wavelet transform and denoising

The wavelet transform is a signal processing technique that represents a transient or non-stationary signal in terms of time and scale distribution with light computational complexity. Therefore, the wavelet transform is an excellent tool for data compression, analysis, and denoising [5].

Assume that a signal $f(t)$ is contaminated by an additive noise $n(t)$; then the measured data is $x(t) = f(t) + n(t)$. The measured time series $x(t) \in L^2$ can be represented as an orthonormal expansion with wavelet basis $\psi_{i,j}(t) = 2^{-i/2}\psi(2^{-i}t - j)$ as follows [5]:

$$x(t) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} X_j^i 2^{-i/2} \psi(2^{-i}t - j) \qquad (3)$$

where the wavelet coefficients are calculated from

$$X_j^i = \langle x, \psi_{i,j} \rangle = \int_{-\infty}^{\infty} x(t) 2^{-i/2} \psi(2^{-i}t - j)\, dt\,. \qquad (4)$$

Note that double-indexed X_j^i is the discrete wavelet transform of signal $x(t)$ at scale i and at translation j, and represents how $x(t)$ is correlated with the i scaled and j translated basis function.

In order to achieve more robust and reliable assessment results for the delay data from cross-correlation, the slowly varying congestion information (at high scale) should be extracted from the delay data, which are corrupted by synchronization offset and random queue behavior. Wavelet denoising is a type of nonlinear approximation of the signal $f(t)$ corrupted by a certain type of noise $n(t)$ based on the wavelet coefficients of the measured data $x(t)$. The wavelet coefficients for the signal, $x(t) = f(t) + n(t)$, becomes $X_j^i = F_j^i + N_j^i$, where the coefficients are obtained by the inner product operation in Eq. 4 with the wavelet basis $\psi_{i,j}$ as follows:

$$X_j^i = \langle x, \psi_{i,j} \rangle, F_j^i = \langle f, \psi_{i,j} \rangle, N_j^i = \langle n, \psi_{i,j} \rangle \qquad (5)$$

From the wavelet coefficients of the signal $x(t)$, the signal $f(t)$ can be approximated as $\tilde{F}$ by suppressing the noise part ($n(t)$) with a nonlinear thresholding function d_T. In this paper, we employ a soft thresholding operation on d_T with following definition [6]:

$$d_T(x) = \begin{cases} x - T & \text{if } x \geq T \\ x + T & \text{if } x \leq -T \\ 0 & \text{if } |x| < T \end{cases} \qquad (6)$$

The soft thresholding in the wavelet denoising plays a key role in the approximation of the traffic congestion delay data. The dominant low frequency term, which corresponds to the true traffic congestion information, will exhibit relatively large wavelet coefficients value at high scale (low frequency) so that true traffic information will remain after thresholding operations. Meanwhile, the high frequency components, which can be assumed to be the effect of the randomness of queue behavior, will have relatively small wavelet coefficients at low scale (high frequency), and is subject to being filtered out by the thresholding operations. Also, soft thresholding has the effect of smoothing the transient irregular peaks in the delay data. In the basic cross correlation technique, randomly occurred peaks in the delay data could have a dominant deleterious effect on the cross correlation value. However, by smoothing these irregular peaks, the value of the cross correlation will be more robust to the irregular peaks than without wavelet denoising.

The value of the threshold T is determined by the variance of the noise σ^2 and the number of samples N such that $T = \sigma\sqrt{2\log_e N}$, as proposed by Donoho [7]. This threshold value can be adapted for different characteristics of the data and noise, which is called the "selective wavelet reconstruction" [7]. By applying the threshold to the wavelet coefficient X_j^i, the signal obtained after wavelet denoising, $\tilde{F}$ is equivalent to time and scale localized nonlinear averaging of the measured signal x in terms of $\psi_{i,j}$ as follows:

$$\tilde{F} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} d_T(X_j^i) \psi_{i,j} \qquad (7)$$

where I and J are chosen to be large enough to ensure $X_j^i < T$ for all $i \geq I$ and all $j \geq J$. In order to maximize the effect of wavelet denoising such that $\tilde{F} \approx f(t)$, the selection of a proper wavelet basis, which will be discussed in Section 3.3, is to be considered. The wavelet basis that is most similar to the time and scale localized properties of the signal $f(t)$ and most different from $n(t)$ will maximize the effects of wavelet denoising. Afterward, the effects of random queue behavior are expected to be reduced as much as possible by minimizing the

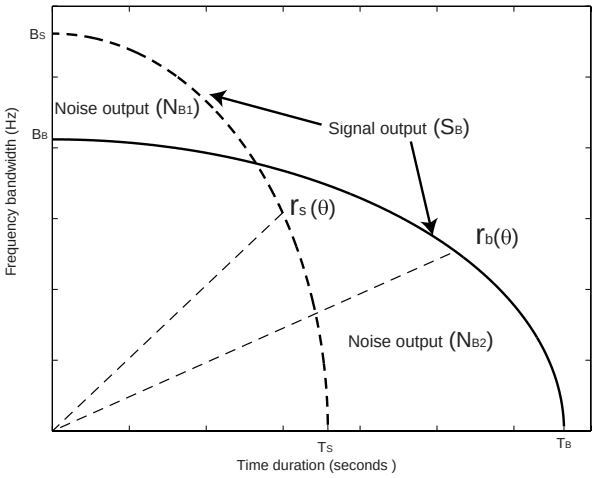

Figure 6: A schematic description of localized time and frequency characteristics for data signal (dotted curve) and wavelet basis (solid curve)

contributions of the noise, N_j^i, in the approximation of the function $\tilde{F}$ provided in Eq. 7.

3.3 Selection of best basis

The wavelet transform provides time and scale localized information of the signal; however, the time and scale resolution of the representation depends on the selection of a wavelet basis. Hence, in order to get the most robust and reliable results from wavelet analysis including wavelet denoising, selecting the best basis function for wavelet decomposition is crucial [16]. In this paper, selection of a wavelet basis is confined to within the Daubechies family of wavelets, which is widely used due to its simplicity of implementation.

Note that the correlation between a data signal and wavelet basis is determined by the time and frequency localized characteristics. Such characteristics of a data signal and wavelet basis can be represented by the time and frequency localized moments, which enable the approximation of the individual time-frequency signal elements as a Gabor logon [18]. Then the trace of the signal elements on the time-frequency plane will be an elliptic curve as shown in Figure 6. B_S and B_B represent frequency bandwidth of a data signal and wavelet basis, and T_S and T_B represent time duration of those two respectively [16]. The figure provides a schematic description of localized time and frequency characteristics for a data signal and wavelet basis. The dotted curve ($r_s(\theta)$) represents the localized time-frequency characteristics of the data signal, and the solid curve ($r_b(\theta)$) represents those of the wavelet basis. The common area determined by the two elliptic curves corresponds to the similarity between the data signal and wavelet basis, while the discrepancy area corresponds to the mismatch between them. Hence, "signal" (S_B) can be defined as the union of a data signal and wavelet basis, and "noise" (N_B) can be defined as the difference between those two terms as follows.

$$S_B = \int_0^{\pi/2} \int_{r=0}^{\max(r_b, r_s)} r \, dr \, d\theta \tag{8}$$

$$N_B = N_{B_1} + N_{B_2} = \int_0^{\pi/2} \int_{\min(r_b, r_s)}^{\max(r_b, r_s)} r \, dr \, d\theta \tag{9}$$

Based on (8) and (9), we postulate the following transient resolution index named "instantaneous SNR" whose dimension is dB/sec as follows:

$$\text{ISNR} = \frac{1}{T_B} 10 \log_{10} \frac{S_B}{N_B} \tag{10}$$

The index provides a measure of similarity between the data signal and wavelet basis within the time frame of the wavelet basis function. In our application, we have two parts of the data signal to be considered in terms of the wavelet basis: the slowly varying congestion information and the interference from synchronization offset and random queue behavior. As mentioned in the previous section, the randomness of queue behavior can be mitigated by employing a soft thresholding technique in wavelet denoising, but synchronization error should be treated differently. Synchronization error in the delay data can be interpreted as the difference of the time shifted version of delay data and the original one. Therefore, the synchronization error depends on the characteristics of the original data, so it could introduce a dominant low frequency term in the spectrum domain similar to those introduced by congestion. Hence, the basis $\psi_{i,j}(t)$ should be chosen to maximize the ISNR of $f(t)$ and $\psi_{i,j}(t)$, and minimize the ISNR of $n(t)$ and $\psi_{i,j}(t)$, where $f(t)$ is the delay changes caused by network congestion and $n(t)$ is the interference caused by the synchronization offset. Therefore, it suffices to find the basis that maximizes the difference of the two ISNR's, which will be referred as the *differential ISNR* later. However, since the true $f(t)$ and $n(t)$ are not available directly, an approximation is required; we used the delay data of a congested path with perfect synchronization as $f(t)$, and the difference between the delay data and its shifted version as an approximation of $n(t) = f(t) - f(t - \Delta_{max})$, where Δ_{max} is the maximum possible synchronization error (1 second in this paper). More discussion on the maximum possible synchronization error is presented in Section 4.3.

In Figure 7, we plot the differential ISNR for a set of Daubechies wavelets 2 through 10. The delay sequences were obtained by repeating the simulation used to draw Figure 4-(b) 120 times to approximate $f(t)$, and the interference $n(t)$ is directly computed from $f(t)$. Each

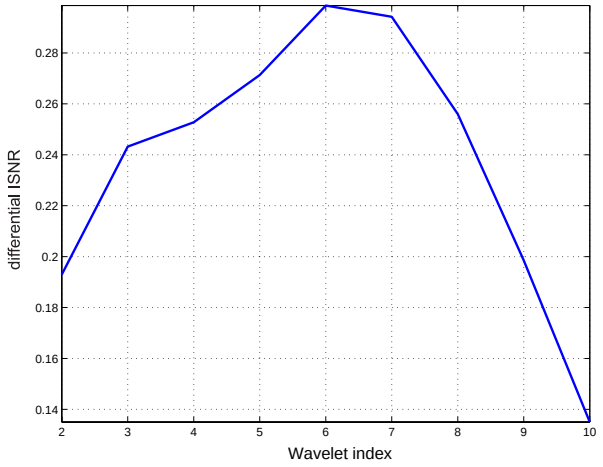

Figure 7: Differential ISNR between congestion signal and other noise for Daubechies wavelets

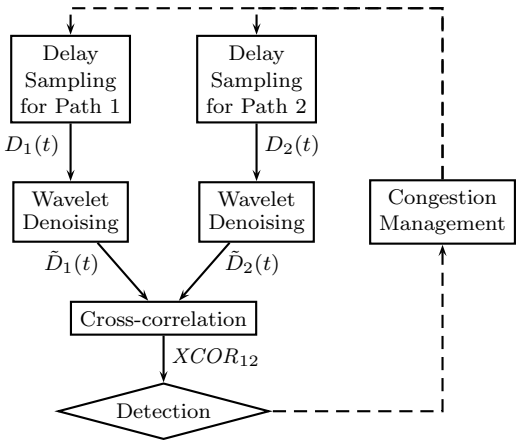

Figure 8: Shared congestion detection procedure

point in Figure 7 is the mean value of the differential ISNR for the 120 sequences. As shown in the figure, Daubechies wavelet 6 has the highest differential ISNR, which implies that it is most correlated with congestion information and least correlated with the synchronization offset. Therefore, the Daubechies wavelet 6 basis will be employed for wavelet denoising in this paper.

4 Implementation

The procedure of our wavelet-based technique is illustrated in Figure 8. The wavelet-based technique has the same sampling stage as described Section 2.3. After the sampling stage, two sequences of delay samples, $D_1(t)$ and $D_2(t)$, are obtained. In the processing stage, $D_1(t)$ is converted into $\tilde{D}_1(t)$ and $D_2(t)$ into $\tilde{D}_2(t)$, through wavelet denoising explained in Section 3.2. The cross-correlation coefficient $XCOR_{12}$ is computed from $\tilde{D}_1(t)$ and $\tilde{D}_2(t)$. An (application-dependent) appropriate action will be taken based on the $XCOR_{12}$ value. As in the basic technique, the procedure to gather delay sequences for different paths is application-dependent and out of the scope of this paper.

There are three issues to discuss in implementing the wavelet-based technique: the delay sampling rate, synchronization offset between delay sequences, and threshold for binary decision. We will discuss each of them in subsequent sections.

4.1 Sampling rate

There is a trade-off in choosing the sampling rate of a delay sequence. High-rate sampling is more accurate but incurs a large overhead on the network. On the other hand, low-rate sampling has little overhead while being slow in convergence. To investigate the effect of sampling rate on performance, we perform simulations with different sampling rates on the topology shown in Figure 3. The sequence of delay samples for each path is processed with our wavelet denoising method. To minimize effects from synchronization offset, we use a topology with a common source. The source nodes are co-located and their clocks are synchronized. A full evaluation involving synchronization offset will be presented in Section 5. Each link has a bandwidth of 1.5 Mb/s, and Pareto ON-OFF CBR flows as background traffic. The average ON and OFF times are selected uniformly between 0.2 and 3 seconds. The CBR rate is selected uniformly between 20 and 40 kb/s, and its Pareto shape parameter is 1.2. To simulate shared congestion, we put 100 ON-OFF flows on the shared link, and 60 on the other two links. With 60 flows, no packet loss was observed. The loss rate with 100 flows varies between 2% and 12%. For independent congestion, we put 60 ON-OFF flows on the shared link, and 100 on the others.

Given a sampling rate, an experiment is repeated 500 times for each of shared and independent congestion. Figure 9 plots the cross-correlation coefficient with the sampling rate of 10 Hz as time elapses. Each curve is the mean cross-correlation coefficients over 500 experiments, and a vertical bar represents the interval between the 5th and 95th percentile values at a specific time.

Figure 10 plots the mean cross-correlation coefficient over 500 experiments for five different sampling rates. The behavior consistent over all sampling rates is that the coefficients converge either to one or to zero as more and more samples are collected. With all sampling rates except 1 Hz, the cross-correlation coefficient converges within 10 seconds. Their variance is also small; after 5 seconds, the interval between the 5th and 95th percentile

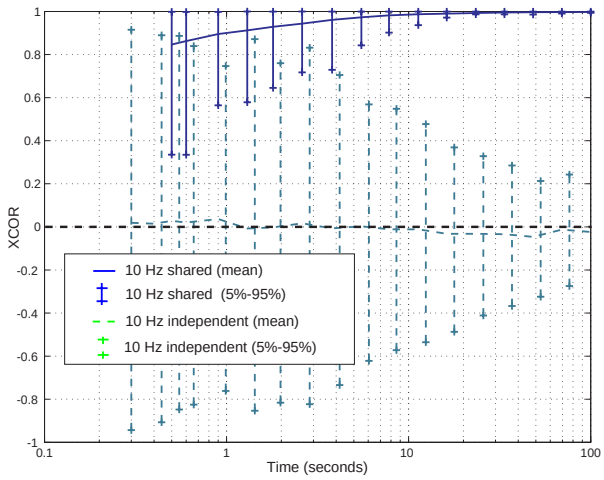

Figure 9: Cross-correlation coefficients with sampling rate of 10 Hz

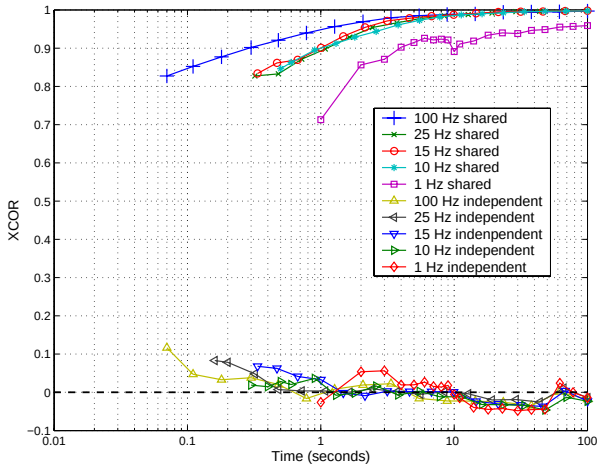

Figure 10: Effect of sampling rate

values with shared congestion never overlaps with the corresponding interval with independent congestion for every such sampling rate.

Since our technique is implemented in user space, the granularity of a timer in an operating system kernel should also be taken into account. Though recent operating systems provide clock rate of 100 Hz, older ones have only 10 Hz. From the figure, we conclude that a sampling rate of 10 Hz is fast enough in convergence and feasible to implement on most operating systems.

4.2 Limiting synchronization offset

There is a synchronization offset in the two sequences of delay samples collected. However, using simple techniques, the synchronization offset between any two paths on the Internet can usually be limited to 1 second. In Figure 1, the synchronization offset of two paths, from X_{src} to X_{dst} and from Y_{src} to Y_{dst}, is caused by (i) the difference of the delay from X_{src} to S and the delay from Y_{src} to S, and (ii) the clock difference between X_{src} and Y_{src}. (i) is bounded by the maximum one-way delay on the network, and (ii) by half the round-trip time between X_{src} and Y_{src} since the clocks in these two nodes can be synchronized by exchanging packets. So the maximum offset is roughly the maximum round-trip time on the network. Measurement studies including one by CAIDA[2] confirm that round-trip time is less than 1 second for the majority of paths on the Internet.

4.3 Threshold for binary decision

Though cross-correlation itself is a reasonable measure of shared congestion, in situations where a binary answer is preferred, a threshold should be set. Since cross-correlation converges to one (or zero) for shared (or independent) congestion as in Figure 10, our technique is not sensitive to the threshold in such cases. However, because synchronization offset reduces correlation of paths sharing a congested link (as shown in Figure 2), it is still important to investigate an appropriate value for the threshold.

When cross-correlation coefficients of delay sample sequences with shared and independent congestion are close to each other, two types of errors may occur: false positive and false negative. The former is the case where the technique reports shared congestion when there is no shared congested link, and the latter is the case where it reports non-shared congestion when there is one or more congested links shared by two paths. The error rate of each type can be estimated from distributions of cross-correlation coefficients for shared and independent congestion. Then the threshold can be adjusted to minimize the total cost of errors using Bayesian testing. Our implementation assumes that the cost of false positive and the cost of false negative are equal, and minimize the total error rate, which is the sum of the false positive ratio and the false negative ratio. Actual costs may differ from application to application.

To determine the best threshold value, we need an estimate of the synchronization offset for any two paths on the Internet. According to measurements by CAIDA, most paths from the F DNS root server to its customers have round-trip time less than 300 ms. Considering that customer hosts of a DNS root server are close to the server, we take 600 ms as the target synchronization offset to optimize the threshold for. More investigation is needed on the actual distribution of round-trip time, and

[2]Available at http://www.caida.org/tools/measurement/skitter/RSSAC/.

the relationship between the target offset and the accuracy of binary decision.

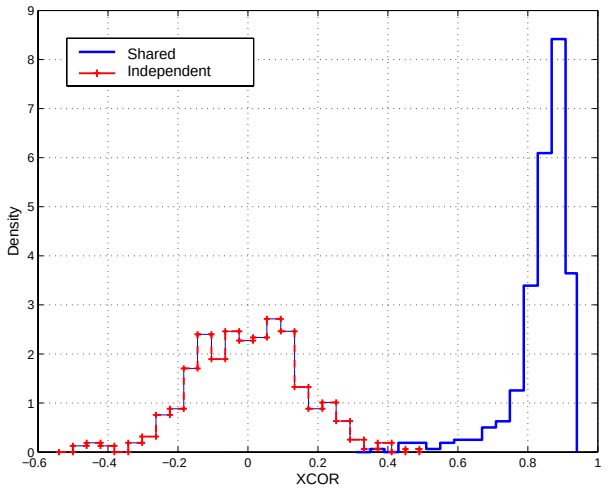

Figure 11: Cross-correlation coefficient distributions

Figure 11 shows the distributions of cross-correlation coefficients with 600 ms synchronization offset. The distributions were obtained from the same delay sequences used in Section 4.1. We used the delay samples collected during the first 10 seconds, with the sampling rate of 10 Hz. The left histogram represents the distribution for independent congestion, and the right one for shared congestion. If we approximate the histograms with normal distributions, they intersect when the cross-correlation coefficient (XCOR) is 0.512, which would be the threshold value that minimizes the total error rate. We use this value as the threshold in later experiments, unless stated otherwise. We will also investigate the effect of the threshold on false positive and false negative ratio in Section 5.2.

5 Performance Evaluation

In simulations, we compare our technique against two representative techniques: a delay-based approach and a loss-based one. We briefly summarize them below.

Markovian probing Rubenstein et al. proposed two techniques using Poisson processes [15]; one is delay-based and the other is loss-based. Since the delay-based technique was better in all their simulations, we compare our technique against their delay-based one. This technique uses a Poisson process with average interval of 40 ms to collect a sequence of delay samples. When two sequences are obtained for different paths, auto-measure M_a is computed from the delays of pairs of adjacent packets in the first delay sequence. Cross-measure M_x is computed from a new sequence obtained by merging the two delay sequences. Only adjacent pairs with the preceding element from the first sequence and the following element from the second sequence are used in computing M_x. If $M_a < M_x$, two paths are sharing a bottleneck.

Bayesian probing A common source sends a packet pair back to back at 15 Hz. The probability that only the second packet is lost is computed from packet losses. If the probability exceeds the threshold (0.4), two paths are sharing a bottleneck [8, 9].

Since both techniques give us a binary answer, we define *Positive Ratio* as a metric to represent accuracy of each technique.

$$\text{Positive Ratio} = \frac{\text{\# of answers indicating shared congestion}}{\text{\# of experiments}}$$

If an experimental setup involves shared congestion, Positive Ratio should be close to one; otherwise, it should be close to zero. We use 0.512 as the threshold to provide a binary answer and then compute the Positive Ratio in the same manner.

We first compare our technique with Markovian probing and Bayesian probing when paths share a common source node and have either shared congestion or independent congestion only, which are included in the design space of both techniques. Then we investigate how they performs in more challenging environments involving paths not sharing a common source or destination and multiple points of congestion. Finally, the performance of our technique on the Internet is presented.

5.1 Probing with synchronization point

Both Markovian probing and Bayesian probing assume that there is a common source (or a common destination for Markovian probing). For such a topology, clocks for the two paths can be synchronized and two samples can be merged into one in chronological order. This is a critical requirement for both techniques. In fact, Bayesian probing requires the stronger condition that two probe packets with different destinations must be sent back-to-back.

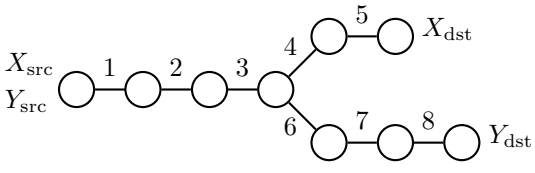

Figure 12: Topology with a common source

Figure 12 shows a network topology where two paths share a source node. A similar topology was used in simulations for Markovian probing [15]. To simulate both low-multiplexing and high-multiplexing environments, we use two kinds of background traffic. For a

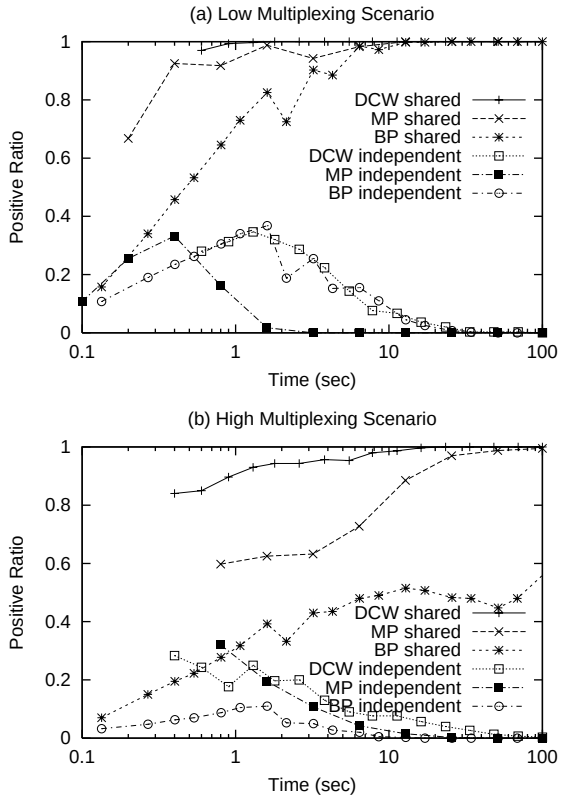

Figure 13: Convergence with synchronization point and drop-tail queues

low-multiplexing environment, a small number of TCP flows are used to cause congestion, and non-congested links are left idle. In shared congestion cases, a link is chosen from links 1 through 3, and 20 TCP flows are created to traverse the link. In independent congestion cases, links 1 through 3 are idle, and the other links have TCP flows, of which the number is chosen uniformly between 0 and 20.

For a high-multiplexing environment, a large number of ON-OFF CBR flows are used. Congestion level is controlled with the number of such flows. For shared congestion, a link chosen from links 1 through 3 has 100 ON-OFF flows. The number of ON-OFF flows on the other links is chosen uniformly between 31 and 70. For independent congestion, links 1 through 3 have ON-OFF flows between 31 and 70, and the other links between 61 to 100. The same parameter settings of ON-OFF flows as in Section 4 are used.

Figure 13 plots Positive Ratio of each technique over 500 experiments as time progresses when links are using drop-tail queues. In the legend, DCW refers to our delay correlation technique with wavelet denoising, and MP and BP refer to Markovian probing and Bayesian probing, respectively. In a low-multiplexing environment, MP is fast in detecting both shared and independent congestion, while BP is relatively slow in both cases. DCW is slightly faster than MP for shared congestion, but as slow as BP for independent congestion. Overall, every technique works well and reaches accuracy over 90% within 10 seconds. In a high-multiplexing environment, however, all three techniques are slower in detecting shared congestion than in a low-multiplexing environment. For DCW and MP, it is because non-congested links have small queueing delay fluctuations, which add noise to delay samples for DCW, and change the order in the merged samples and decrease cross-measure M_x for MP. Nevertheless, since DCW removes most noise through wavelet denoising, its degradation is not as severe as MP's. BP experiences the most notable degradation among the three; though it is the fastest for independent congestion, its Positive Ratio for shared congestion is still less than 0.6 after 100 seconds. It is because our ON-OFF background flows include those with very short ON/OFF time, while all ON-OFF flows in the simulations of [8] have relatively long ON time—2 seconds. BP requires the probability that both packets in a packet pair are lost to be high to detect shared congestion. A longer ON time means a queue remains full for a long time causing both packets in the pair to be dropped. However, it is less likely with short ON time. That leaves DCW to be the only technique that reaches 90% accuracy after 10 seconds in high-multiplexing environment.

Figure 14 presents the same simulation results when links use RED. DCW and MP show similar performance as with drop-tail queues. However, BP does not work at all with RED queues. Its problem with RED was already pointed out by their authors [8] using ON-OFF background flows, but the problem is more serious here because their simulation setup has a higher loss rate and smaller queues, which means a RED queue's behavior is similar to that of a drop-tail queue. Neither DCW nor MP has such a problem; they maintain performance as good as with drop-tail queues.

5.2 Probing without synchronization point

The topology in Figure 15 is an extended version of that in Figure 12. The paths have different source and destination nodes. Delay samples collected at different nodes cannot be synchronized because of two reasons. First, the clocks of node X_{src} and node Y_{src} are not synchronized. Second, delay from X_{src} to S is different from delay from Y_{src} to S.

To investigate the effect of synchronization offset between two paths, we plot in Figure 16 Positive Ratio with shared congestion as we increase the synchronization offset for both low and high multiplexing scenarios. The

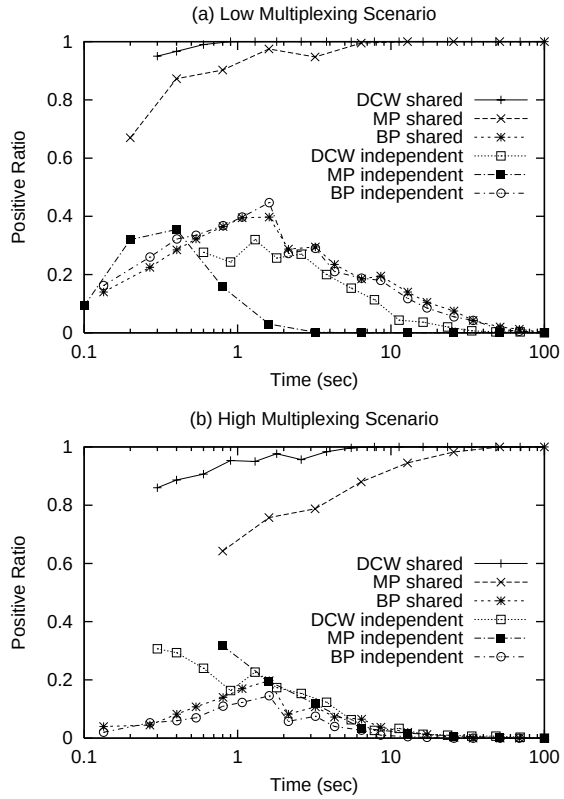

Figure 14: Convergence with synchronization point and RED queues

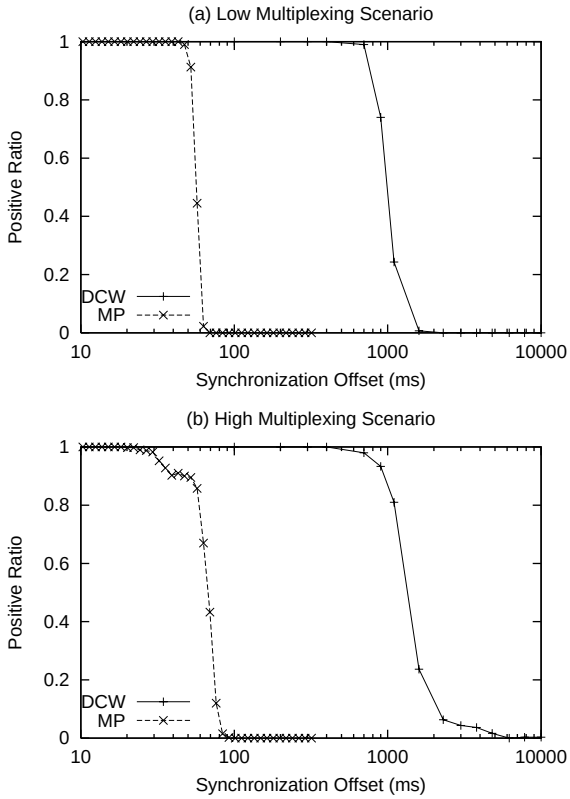

Figure 16: Effect of synchronization offset

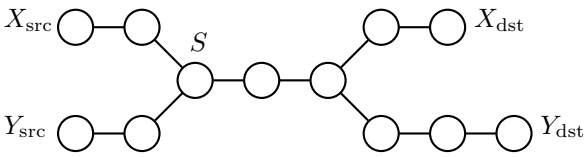

Figure 15: Topology without Synchronization Point

original sets of delay samples are obtained from the two paths on the topology in Figure 12, and the synchronization offset is added to the set of delay samples between Y_{src} and Y_{dst}. Only the overlapping portion between two sets are used by the techniques investigated. BP is excluded because it is already known that Positive Ratio of BP with shared congestion is 0.2 or less even with 10 ms offset [8] due to its requirement that two packets (for different paths) be sent back-to-back. Because MP is slower than DCW in Positive Ratio convergence, MP may exhibit lower performance not because of synchronization offset but because of low accuracy if the number of delay samples are not large. Thus in this simulation, detection is made with delay samples belonging to the first 100 seconds of the overlapping period to ensure that both DCW and MP have near-100% accuracy. Positive Ratio drops to zero between 30 ms and 70 ms for MP, and between 1 sec and 2 sec for DCW. The sharp decrease of MP happens in the [30 ms, 70 ms] interval because the average interval between two packets in MP is 40 ms. Therefore, if the offset exceeds that value, most packets in a merged sequence are out of order, and the cross-measure M_x becomes low. Though we plot the results for drop-tail queues only, the results for RED queues are similar.

Next, we examine how wavelet denoising helps our technique in tolerating a large synchronization offset. The dotted curve and vertical bars crossing it in Figure 17 are copied from Figure 2, which shows the cross-correlation coefficients without wavelet denoising. We process the data used in Figure 2 with our wavelet denoising, and plot cross-correlation coefficient versus synchronization offset. The solid curve represents the mean cross-correlation coefficients, and the vertical bars indicates the 5th and 95th percentile values. Without wavelet denoising, the cross-correlation of the delay sequences decays very fast with increase of synchronization offset; with a 600 ms offset, the mean coefficient approaches the solid line representing the threshold (0.512). This means that the cross-correlation technique without denoising is only as good as random decision with the

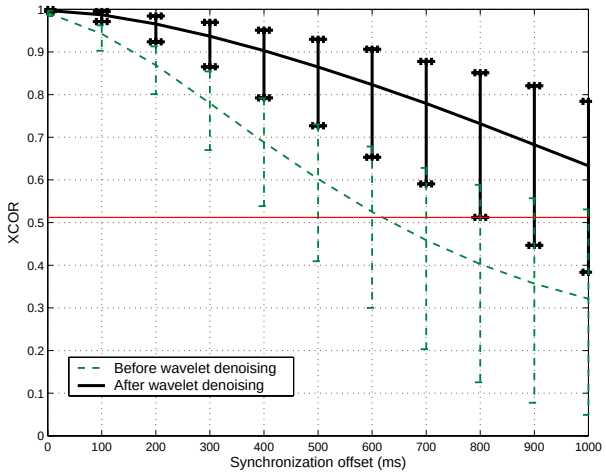

Figure 17: The effect of wavelet denoising on synchronization offset

offset. However, the cross-correlation of the delay sequences after wavelet denoising is less sensitive to the synchronization offset so that one can properly determine the state of congestion even with a fair amount of synchronization offset between the data. The wavelet denoising described above results in a smoothing of the delay data curve in a time- and scale-localized manner so that evaluation of the cross-correlation after wavelet denoising becomes a more robust estimation.

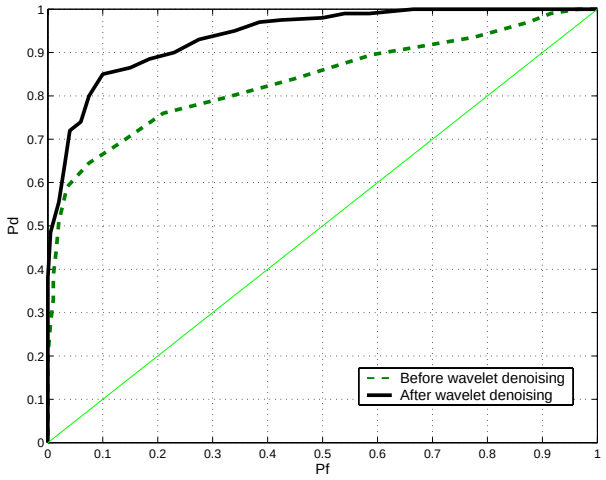

Figure 18: Comparison of receiver operating characteristic with versus without wavelet denoising

Figure 18 shows the effect of the threshold value on false positive and false negative ratio using the receiver operating characteristic (ROC) curves in the presence of synchronization offset. ROC is a performance test methodology in terms of probability of detection P_D against the probability of false positive P_F [17]. In

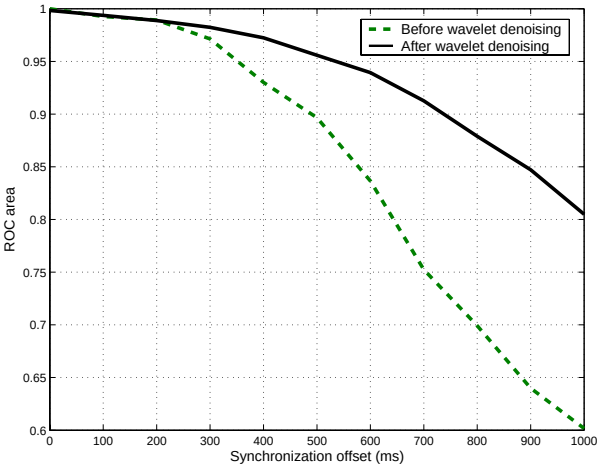

Figure 19: Comparison of receiver operating characteristic performance in synchronization offset with vs. without wavelet denoising

our application, they are defined as follows for a certain threshold value of cross-correlation T_{XCOR}.

$$\begin{aligned} P_D &= P(XCOR \geq T_{XCOR} \mid \text{shared congestion}) \\ P_F &= P(XCOR \geq T_{XCOR} \mid \text{independent congestion}) \end{aligned}$$

The performance can be graphically depicted for all possible values of threshold T_{XCOR}; as we move along an ROC curve from the lower-left corner to the upper-right corner, the threshold varies from 1 to -1. The dashed straight line is the characteristics of the worst case whose probability of detection P_D is linearly proportional to the false positive probability P_F, which is the same as a random decision maker.

Figure 18 has two ROC curves drawn using the DCW simulation data for Figure 13. An offset of 600 ms was added to one of the delay sequences of each experiment. The dotted curve is an ROC curve before wavelet denoising, and the solid curve is after denoising. Since our technique converges in 10 seconds as Figures 13 and 14 show, delay samples for the first 10 seconds were used to compute the cross-correlation coefficient. With wavelet denoising, our technique shows an improved curve (higher detection probability P_D with the same false positive probability P_F) compared with the curve without denoising. Note that the area under the curve called the ROC area provides a quantitative measure of performance for comparison of different curves; the area of an ideal curve is 1, while the area of a random decision maker is $\frac{1}{2}$.

Figure 19 demonstrates the effect of wavelet denoising for different synchronization offsets using the ROC area of our technique. Two curves show the ROC area with and without wavelet denoising as the synchronization offset increases. With tight synchronization, wavelet denoising makes little difference. As the offset increases,

however, the basic technique curve drops to 0.6 at an offset of 1 second, becoming close to random decision. On the other hand, the technique with denoising degrades smoothly, maintaining 0.8 at the 1 second offset.

5.3 Multiple points of congestion

So far, queueing delay variation on non-congested links was filtered out with wavelet denoising. However, if non-congested links have significant queueing delay variation, or there are more than one point of congestion, the delay variation on such links cannot be eliminated and makes shared congestion detection more difficult. In fact, it is unclear what 'shared congestion' should mean under such conditions. Therefore, instead of deciding whether a technique detects shared congestion correctly, we investigate how the technique respond as the degree of shared congestion changes. One possible metric to represent the degree of shared congestion is how large the loss rate on shared links is compared with that on non-shared links. Hence, we define a new quantity called *shared loss rate ratio*. Let the loss rate of the shared portion of two paths be L_{shared}, and the loss rate of the non-shared portion of the first path to be L_1 and the second path L_2. Then the shared loss rate ratio is defined as follows.

$$L_s = \frac{L_{\text{shared}}}{L_{\text{shared}} + \max(L_1, L_2)} \tag{11}$$

If loss occurs only on shared portion, L_s becomes 1, and if loss occurs only on non-shared portion, L_s becomes 0. If there is no loss at all, then L_s is defined as 0, indicating no shared congestion.

In the following simulation, we use the topology in Figure 3. The number of ON-OFF background flows on each link is chosen uniformly between 81 and 100, resulting in loss rate between 0 and 12%, and delay samples are collected for 100 seconds. L_s is computed from the actual loss rates of the links. 1000 experiments are classified into 10 groups depending on the interval their L_s belongs to. If L_s of an experiment is in $[0, 0.1)$ then it is in the first group, if in $[0.1, 0.2)$ then the second, and so on. If $L_s = 1$, the experiment is in the same group as those with L_s in $[0.9, 1)$. Positive Ratio is calculated over all experiments in the same group. The results for DCW, MP, and BP are presented in Figure 20.

Positive Ratio of DCW is only about 0.1 when $L_s < 0.1$, but 0.8 or larger when $L_s \geq 0.3$. It means DCW has a cut-off at $L_s = 0.2$ differentiating shared and independent congestion. MP shows very different behavior. Positive Ratio is 0 for most intervals, and only 0.1 for the last one. Since we know that Positive Ratio of MP reaches 1 after 100 seconds if $L_s = 1$, this indicates that MP answers positively (meaning shared congestion) only when L_s is very close to 1. In other words, MP always gives

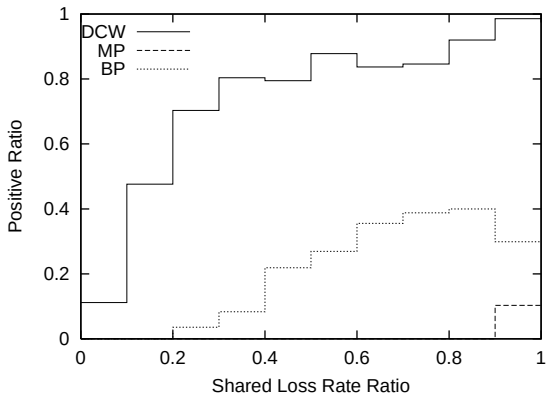

Figure 20: Positive Ratio with multiple points of congestion

a negative answer if there are multiple points of congestion, regardless of the degree of shared congestion. BP gives more and more positive answers as L_s increase, but does not have any sharp increase as DCW has. Therefore, for those applications requiring a cut-off in shared congestion detection, DCW is preferred. However, the preferred cut-off value depends on applications. DCW can be customized for applications with different needs on the cut-off value by adjusting the threshold. Some applications need to determine whether two paths share *all* congested links [2], which corresponds to $L_s = 1$. In this case, MP would also be a good choice.

5.4 Internet Experiments

We apply our technique to a large-scale network, the Internet. Six end hosts were used in our Internet experiments. Figure 21 shows their abstract topology. Note that each hop in the figure may consist of multiple physical hops. Three hosts, A_1, A_2, and A_3, are located in North America. The other three hosts, K, T, and H, are located in Korea, Taiwan, and Hong Kong, respectively.

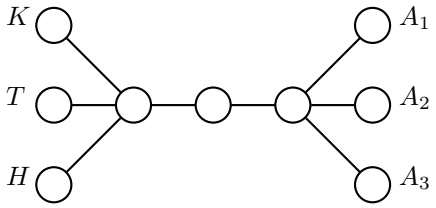

Figure 21: Experimental topology on the Internet

Delay samples were collected from the paths from A_1 to K and from A_2 to T between October 28 and November 2, 2003. We can reasonably conclude that there was

no congested link because no probe packet was lost during measurement. In order to create a shared bottleneck, we opened 40 TCP sessions between H and A_3. The loss rate was about 5% while they are running. Since both paths experienced a similar loss rate, we conclude that the congestion occurred on one of the shared links.

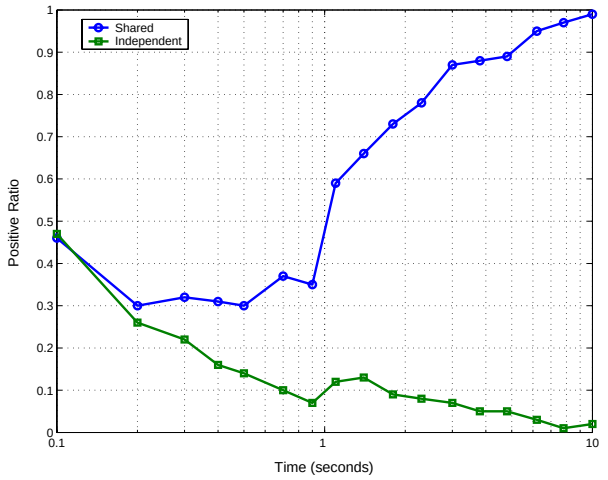

Figure 22: Convergence with Internet traces

Positive Ratio for shared congestion and independent congestion (or no congestion in this case) is shown in Figure 22. The delay samples were collected for 15 seconds, and time is adjusted with measured clock difference between A_1 and A_2 by exchanging packets between them. Each experiment is repeated 100 times to calculate the Positive Ratio. The result resembles what we obtained through simulations. The accuracy of our technique exceeds 80% using the samples for the first 3 seconds, and reaches 98% after 8 seconds.

6 Conclusion

Network resources are better utilized when multiple flows cooperate; we can implement cooperative congestion control and improve the overlay topology of a P2P system. However, such cooperation is feasible only when we can identify flows sharing a congested bottleneck. Techniques proposed previously addressed this problem with limitations including a common synchronization point and drop-tail queues. But they are not effective under other conditions, such as RED queueing, multiple points of congestion, or paths with different sources and destinations.

We proposed a robust technique that performs well with various settings. It is based on wavelet denoising and cross-correlation. The denoising process effectively removes noise and makes our technique more resilient to synchronization offset that confuses other techniques. The proposed technique achieves faster convergence and broader application than previous ones, using less probe packets. We believe that many applications constructing topology in the application layer can benefit from our technique applying wavelet transform.

References

[1] P. Abry, R. Baraniuk, P. Flandlin, R. Riedi, and D. Veitch. Multiscale nature of network traffic. *IEEE Signal Proessing Magazine*, 19(3):28–46, May 2002.

[2] A. Akella, S. Seshan, and H. Balakrishnan. The impact of false sharing on shared congestion management. In *Proceedings of the 11th IEEE International Conference on Network Protocols*, Nov. 2003.

[3] H. Balakrishnan, H. Rahul, and S. Seshan. An integrated congestion management architecture for Internet hosts. In *Proceedings of ACM SIGCOMM '99*, Sept. 1999.

[4] M. Coates, A. O. Hero III, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Proessing Magazine*, 19(3):47–65, May 2002.

[5] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005, Sept. 1990.

[6] D. L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, May 1995.

[7] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455, 1994.

[8] K. Harfoush, A. Bestavros, and J. Byers. Robust identification of shared losses using end-to-end unicast probe. In *Proceedings of the 8th IEEE International Conference on Network Protocols*, Nov. 2000.

[9] K. Harfoush, A. Bestavros, and J. Byers. Robust identification of shared losses using end-to-end unicast probe. Technical Report BUCS–TR–2001–001, Computer Science Department, Boston University, Massachusetts, U.S.A., Jan. 2001. Errata to the previous reference.

[10] P. Huang, A. Feldmann, and W. Willinger. A non-instrusive, wavelet-based approach to detecting network performance problems. In *Proceedings of the First ACM SIGCOMM Internet Measurement Workshop*, pages 213–227, Nov. 2001.

[11] D. Katabi, I. Bazzi, and X. Yang. A passive approach for detecting shared bottlenecks. In *Proceedings of the 10th IEEE International Conference on Computer Communications and Networks*, Oct. 2001.

[12] D. L. Mills. Network time protocol (version 3) specification, implementation and analysis. RFC 1305, Mar. 1992.

[13] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of IEEE INFOCOM '99*, Mar. 1999.

[14] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk. A multifractal wavelet model with application to network traffic. *IEEE Transactions on Information Theory*, 45(3):992–1018, 1990.

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

[16] Y. Shin, E. J. Powers, W. M. Grady, and S. C. Bhatt. Optimal Daubechies' wavelet bases for detection of voltage sag in electric power distribution and transmission systems. In *Wavelet Applications in Signal and Image Processing VII, SPIE*, pages 873–883, July 1999.

[17] H. L. V. Trees. *Detection, Estimation, and Modulation Theory.* John Wiley & Sons, Dec. 1968.

[18] W. Williams. Uncertainty, information, and time-frequency distributions. In *Advanced Signal Processing Algorithms, Architectures and Implementations II, SPIE.*, pages 144–156, July 1991.

[19] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path properties. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.