A Wavelet-Based Approach to Detect Shared Congestion*

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

[1]Dept. of Computer Sciences
The University of Texas at Austin
{minskim,lam}@cs.utexas.edu

[2]Dept. of Electrical and Computer Engineering
The University of Texas at Austin
{thkim,june,ejpowers}@ece.utexas.edu

ABSTRACT

Per-flow congestion control helps endpoints fairly and efficiently share network resources. Better utilization of network resources can be achieved, however, if congestion management algorithms can determine when two different 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 either 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, when detecting shared congestion of paths with a common endpoint, our technique provides faster convergence and higher accuracy while using fewer packets than previous techniques, and that it also accurately determines when there is no shared congestion. Furthermore, we show that our technique is robust and accurate for paths without a common endpoint or synchronized clocks; more specifically, it can tolerate a synchronization offset of up to one second between two packet flows.

Categories and Subject Descriptors

C.2.3 [**Computer Systems Organization**]: COMPUTER-COMMUNICATION NETWORKS—*Network Operations*

General Terms

Measurement

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

Keywords

shared congestion, wavelet denoising

1. INTRODUCTION

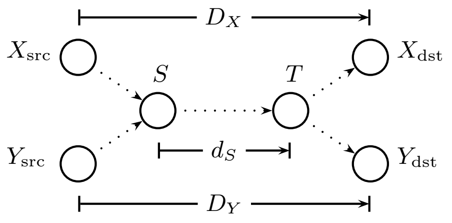
Congestion control has been performed at a per-flow level; each flow adjusts its sending rate according to feedback regarding the network's congestion status. The stability of today's Internet is mainly due to congestion control, especially the additive increase/multiplicative decrease approach of TCP.

Better utilization of network resources is achievable with cooperation between flows. For example, the 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, the Congestion Manager could improve fairness and efficiency significantly.

The 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 the 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 the 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, but 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 environment. 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 that the two tested paths share an endpoint, usually at the source. Thus, they cannot be used for general overlay networks.

We propose a novel technique (delay correlation with wavelet denoising or DCW) to detect shared congestion between two



$$d_X = D_X - d_S, \quad d_Y = D_Y - d_S$$

Figure 1: Two paths sharing links

Internet paths. Like previous techniques, it is based on a simple observation: two paths sharing congested links have high correlation between their one-way delays. However, naive correlation measurements 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 fast convergence with fewer packets. It takes at most 10 seconds to reach near 100% accuracy with both drop-tail and RED queues, while previous techniques often take longer or fail. We also show that our technique maintains its accuracy without a common endpoint; 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 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.

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

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 because its queue is persistently 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 loss runs of 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 determine whether such fluctuations occur between S and T.

2.2 Cross-correlation

Our basic technique is based on the observation that measured delays of two paths show strong correlation if the paths share one or more congested links, and little correlation if they do not share any congested links [15]. Suppose that paths X and Y in Figure 1 are sharing congested links between S and T, and that the other links are lightly loaded. Then D_X and D_Y will show strong similarity, since the only strongly varying component d_S is shared by both paths. On the other hand, if congestion occurs on links other than the links between S and T, D_X and D_Y become independent.

We use 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 that each $\langle X_i, Y_i \rangle$ pair was measured at the same time. Then their cross-correlation coefficient $XCOR_{XY}$ is defined as follows.

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

Note that $XCOR_{XY} = 1$ if both d_X and d_Y are constant and d_S is not constant (shared congestion), and $XCOR_{XY} = 0$ if d_S is constant and d_X or d_Y varies independently (no shared congestion). Of course, other network effects could make $XCOR_{XY} = 1$ in the absence of shared congestion, or make $XCOR_{XY} = 0$ in the presence of shared congestion. We follow earlier work by assuming that this rarely happens [15]; further Internet experimentation is required.

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 such 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 linearly interpolated from neighboring samples, because if missing samples are discarded, X_i and Y_i are very likely out of synchronization from then on. The sampling stage ends when the last delay sample from X_{dst} is received (or upon timeout if the last probe or the reply

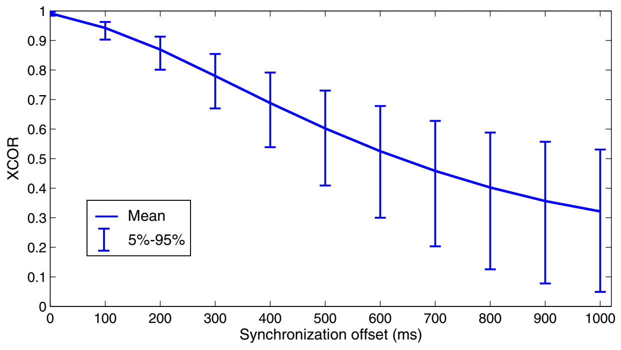


Figure 2: Cross-correlation coefficient between two delay sequences vs. synchronization offset

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 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. To achieve this, the endpoints would need precisely synchronized clocks, and to predict the delays from X_{src} and Y_{src} to S. However, one-way delays cannot be measured without network support, and network clock synchronization protocols are not accurate enough for our purposes, since they still allow 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 increases, the delay sequences collected by the two nodes show less and less correlation.

Figure 2 illustrates this; it plots the cross-correlation coefficient for two paths sharing a congested link as synchronization offset rises from 0 to 1 second. Each point is the mean coefficient over 300 simulations; the bars show 5th and 95th percentiles. In each simulation, two delay sample sequences were collected for 100 seconds on the topology shown in Figure 3 using ns-2.[1] The bandwidth of every link was 1.5 Mb/s, and its propagation delay was 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}. Pareto ON-OFF CBR (constant bit rate) flows were used as background traffic, because then the congestion level could be controlled easily by changing the number of flows. The average ON and OFF times were

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

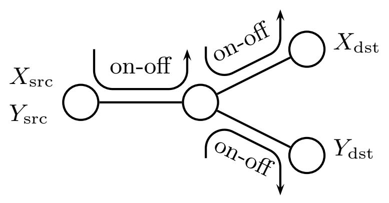


Figure 3: Simple topology with a common source

selected uniformly between 0.2 and 3 seconds. The CBR rate was selected uniformly between 20 and 40 kb/s, and its Pareto shape parameter was 1.2. The loss rate of the shared link was about 10%; the other links did not have any loss. Without synchronization offset, the mean cross-correlation between the two delay sequences is about 0.99. However, the mean cross-correlation drops as synchronization offset increases so that a 600 ms synchronization offset results in half of the mean cross-correlation without offset.

The second assumption required by the basic technique 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 Section 3, we propose wavelet denoising to 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 varying delays at non-congested links. It also determines quickly when there is no shared congestion.

2.5 Related work

Previous approaches to detect shared congestion using probe packet streams are also based on the assumption of strong correlation between packet delays or losses of two paths that share a bottleneck. Thus these approaches have the same limitation as our basic technique, i.e., two probe packet streams should be synchronized for such technique to be effective.

Rubenstein et al. proposed two techniques, one based on one-way delays and the other based on packet losses [15]. Both techniques send probe packets for each path, using a Poisson process with a rate of 25 Hz. The delay-based technique computes cross-correlation coefficients of one-way delays, but in a different way from our basic technique. Given two delay sequences obtained for different paths, auto-measure M_a is computed as the cross-correlation coefficient between the second sequence and its shifted version obtained by removing its first sample. 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 selected. Then M_x is calculated as the cross-correlation between the samples belonging to the first sequence and those belonging to the second. If $M_a < M_x$, the two paths share a bottleneck. In their loss-based technique, M_a and M_x are conditional probabilities that a packet is lost when its following packet is lost. Since the loss-based technique requires such conditional probabilities to be high, it doesn't work well with queueing disciplines other than drop-tail queueing. Even with drop-tail queues, the delay-based technique was more robust than the loss-based one in all their simulations. Both techniques require that two paths share either a common source or a common destination node to synchronize probe packet streams.

Harfoush et al. [8, 9] proposed a loss-based technique that outperforms the loss-based technique of Rubenstein et al. In their technique, 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. However, like the loss-based technique of Rubenstein et al., this technique also assumes that back-to-back packets are likely lost together when a link is congested, and thus requires drop-tail queueing. Furthermore, due to the requirement of sending packets back to back, it is applicable only when two paths share a common source node.

The technique of Katabi et al. [11] is an entropy-based technique, which doesn't depend on delay or loss correlation. Rather than detecting shared congestion between two paths, its objective is to group flows. More specifically, a node observes inter-packet arrival times of multiple flows passively, and partitions flows into groups such that all flows in each group share the same bottleneck. It is based on the assumption that correct grouping minimizes the entropy of inter-arrival times. This technique is more scalable than others when grouping a large number of flows because no probe packet is required and pair-wise comparison is unnecessary. However, it requires that all flows reach the same destination node, and that they occupy a significant fraction of the bottleneck bandwidth.

3. WAVELET DENOISING

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 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 a wavelet basis that minimizes negative effects of synchronization offset.

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 congestion levels. The source and destination nodes were 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 CBR flows, no packet loss) while the time series added in Figure 4(b) is the delay under heavy traffic load (92 ON-OFF CBR flows, loss rate between 2% and 10%). ON-OFF CBR flow parameter settings were identical to those described in Section 2.4. The 95th percentile of loss run length for heavy traffic load was about 180 ms, which is close to the Internet measurement result (220 ms) in [19].[2] Observe that the one-way delay with light traffic is a noise-like waveform with small amplitude, while the delay with heavy traffic shows an irregular pulse pattern with larger amplitude. Such pulses result from network congestion.

The corresponding frequency domain power spectral densities of the individual time series, normalized to unity area, 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 queue behavior, well-demonstrated in Figure 4(a). Therefore, for a proper assessment of network traffic under congestion via delay data, it is necessary to reduce the effects associated with random 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.

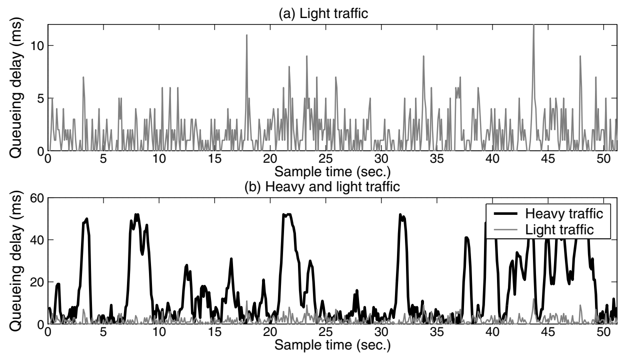


Figure 4: Time series of one-way delay of a single-hop path

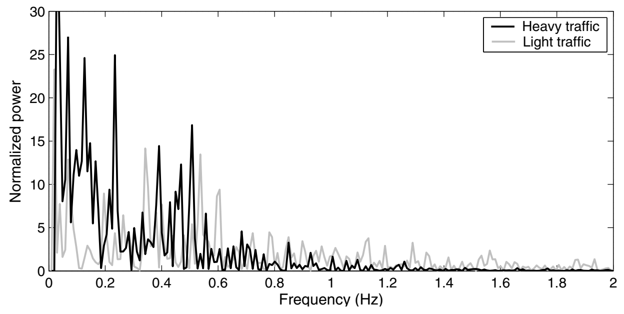


Figure 5: Power spectral densities of time series of delay data with light traffic and heavy traffic

[2]Loss run lengths were measured using a Poisson packet stream with a rate of 50 Hz.

If we are only interested in extracting the large amplitude components at low frequencies, a simple low-pass filter seems to be an intuitive solution. Low-pass filtering would smooth the delay signals, increasing cross-correlation when there is shared congestion. On the other hand, low-pass filtering may fail to diagnose non-shared congestion cases. Consider the extreme case that there is no congestion on either path. In such a case, near-zero cross-correlation is expected since the delay signals will be dominated by random queue behavior. However, simple low-pass filtering may over-smooth the signal, resulting in an inappropriately high value of cross-correlation. This is because the frequency spectrum in network delay data varies in a dynamic fashion due to the fact that network traffic changes in time. Therefore, any attempt to mitigate the interference effects should include an approach based on both time and frequency (or scale) analysis, e.g., the wavelet transform. Hence, we use wavelet denoising rather than simple filtering.

We will show that wavelet denoising is highly effective for the purpose of detecting shared congestion. A major advantage of wavelet denoising is that it preserves the dominant characteristics of one-way delay and filters out non-dominant ones in a time and scale localized manner, thus it can deal with the time-varying spectrum of network delay data. Therefore, even when there is no congestion, wavelet denoising preserves strong transients at high frequencies and thus maintains low cross-correlation between denoised signals.

There may exist other signal processing techniques that perform as well as or better than wavelet denoising. Much more investigation is needed to evaluate the large number of signal processing techniques in the literature.

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. Due to its light computational complexity, the wavelet transform is an excellent tool for on-line data compression, analysis, and denoising.

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 $x(t)$ 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]:

$$x(t) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} X_j^i \psi_{i,j}(t) \tag{3}$$

where the wavelet coefficients are calculated from

$$X_j^i = \int_{-\infty}^{\infty} x(t)\psi_{i,j}(t)\, dt\,. \tag{4}$$

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

Two cases should be taken into account to achieve robust and reliable cross-correlation results. When there is congestion, 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. Without congestion, strong random transients should be extracted to ensure a low correlation. Wavelet denoising is capable of selecting the desired signal while removing others in each case.

Wavelet denoising lets us build a nonlinear approximation of the signal $f(t)$ using the wavelet coefficients of the measured data $x(t)$. The wavelet coefficients for the measured data $x(t) = f(t) + n(t)$ become $X_j^i = F_j^i + N_j^i$, where $F_j^i = \int_{-\infty}^{\infty} f(t)\psi_{i,j}(t)\, dt$ and $N_j^i = \int_{-\infty}^{\infty} n(t)\psi_{i,j}(t)\, dt$. Then $\tilde{f}(t)$, an approximation of the signal $f(t)$, is obtained from the wavelet coefficients of the measured data $x(t)$ by suppressing noise with a nonlinear thresholding function, d_T. In this paper, we employ a soft thresholding operation on d_T with the following definition [6]:

$$d_T(x) = \begin{cases} x - T & \text{if } x \geq T \\ x + T & \text{if } x \leq -T \\ 0 & \text{if } |x| < T\,. \end{cases} \tag{5}$$

The value of the threshold T is determined by the variance of the noise σ^2 [6] and the number of samples N using $T = \sigma\sqrt{2\log_e N}$, as proposed by Donoho [7]. Then the denoised signal $\tilde{f}(t)$ is obtained by applying the threshold to the wavelet coefficients X_j^i in Eq. 3.

$$\tilde{f}(t) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} d_T(X_j^i)\psi_{i,j}(t) \tag{6}$$

Soft thresholding plays a key role in the approximation of the traffic delay data under congestion. If there is shared congestion, the dominant low frequency term, which corresponds to the true traffic congestion information, will exhibit relatively large wavelet coefficient values at high scale (low frequency) so that true traffic information will remain after the thresholding operation. Meanwhile, the high frequency components, which can be assumed to be the effects of random queue behavior, will have relatively small wavelet coefficients at low scale (high frequency), and will be filtered by the thresholding operation. Soft thresholding also has the effect of smoothing the transient irregular peaks in the delay data. In the basic cross-correlation technique, randomly occurring peaks in the delay data could have a dominant deleterious effect on the cross-correlation value. Wavelet denoising smooths these irregular peaks, making the cross-correlation value more robust. On the other hand, when there is no congestion, delay variations caused by random queue behavior will have relatively large wavelet coefficient values, and thus will be preserved by soft thresholding.

3.3 Selection of wavelet basis

The wavelet transform provides a time and scale localized representation of a measured time series; 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, it is crucial to select the best basis function for wavelet decomposition [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. Other wavelets and their tradeoffs between performance and complexity need more investigation.

The correlation between a data signal and a wavelet basis is determined by time and frequency localized characteristics. Such characteristics of a data signal and wavelet basis can be represented by the time and frequency localized mo-

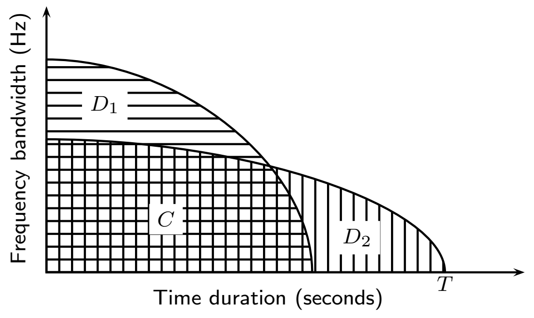


Figure 6: A schematic description of localized time-frequency characteristics for a data signal (horizontally hatched area) and a wavelet basis (vertically hatched area)

ments, 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 is defined as an elliptic curve as shown in Figure 6. In this section, we define a metric, instantaneous SNR (signal-to-noise ratio), to indicate how closely a wavelet basis matches a data signal on the time-frequency plane. Then the metric is used to select a wavelet basis that minimizes adverse effects of synchronization offset.

3.3.1 *Instantaneous SNR*

Figure 6 provides a schematic description of localized time and frequency characteristics for a data signal and wavelet basis. The quarter ellipse including C and D_1 represents the localized time-frequency characteristics of the data signal, and the quarter ellipse including C and D_2 represents those of the wavelet basis. For the two quarter ellipses to be well-matched, the size of the common area C should be large while the discrepancy $D = D_1 + D_2$ should be small. To quantify how closely the time-frequency characteristics of a data signal and wavelet basis match, we postulate a transient resolution index named "instantaneous SNR" whose dimension is dB/sec:

$$\text{ISNR} = \frac{1}{T} 10 \log_{10} \frac{C}{D}\,. \tag{7}$$

T is the time duration of the wavelet basis [16] shown in Figure 6. ISNR provides a measure of similarity between the data signal and wavelet basis within the time frame of the wavelet basis function.

3.3.2 *Minimizing adverse effects of synchronization offset*

In our application, the measured data consists of two parts, slowly-varying congestion information and interference from random queue behavior and synchronization offset; such interference can be mitigated by employing a soft thresholding technique in wavelet denoising. We can further reduce the interference from synchronization offset by choosing a wavelet basis carefully.

Synchronization offset 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 offset depends on the characteristics of the original data. 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 between the two ISNRs, which we call the *differential ISNR*. 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 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 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 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 best matched with congestion information and least matched with the noise due to synchronization offset on the time-frequency plane. Therefore, the Daubechies wavelet 6 basis will be employed for wavelet denoising in this paper.

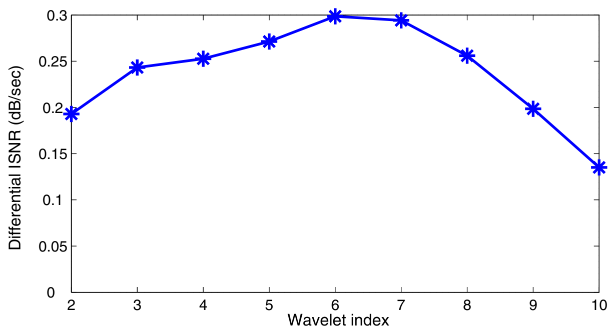


Figure 7: Differential ISNR between congestion signal and other noise for Daubechies wavelets

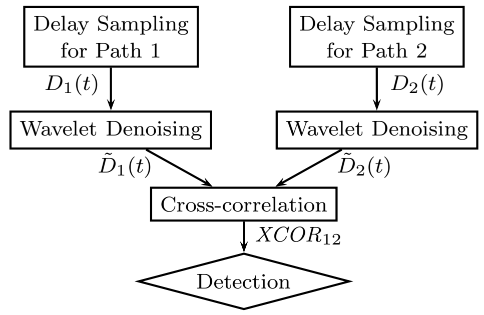


Figure 8: Shared congestion detection procedure

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 in Section 2.3. The sampling stage produces two sequences of delay samples, $D_1(t)$ and $D_2(t)$. The processing stage uses wavelet denoising to

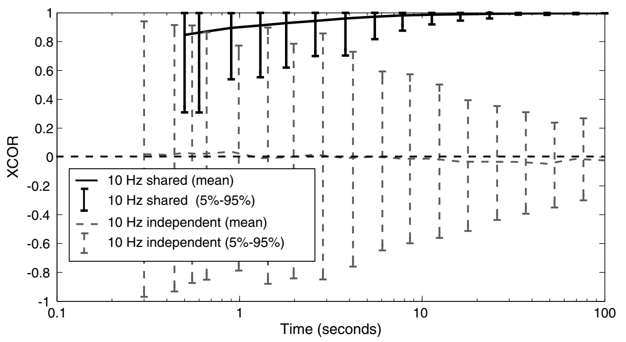


Figure 9: Cross-correlation coefficients with sampling rate of 10 Hz

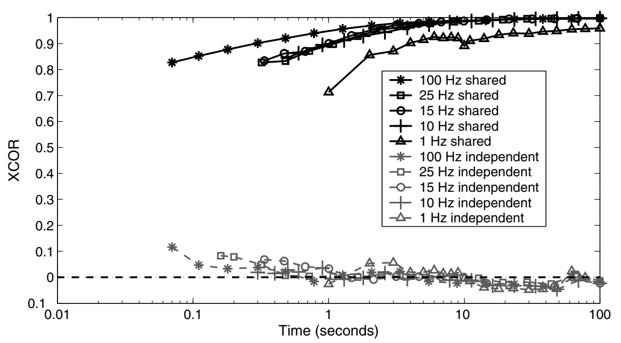


Figure 10: Effect of sampling rate

produce new, denoised sequences $\tilde{D}_1(t)$ and $\tilde{D}_2(t)$, as explained above. The cross-correlation coefficient $XCOR_{12}$ is computed from $\tilde{D}_1(t)$ and $\tilde{D}_2(t)$. (The computational overhead of these operations is very low. We found that when delay samples were collected at 10 Hz for 100 seconds for each of two paths, a machine with a 2.53 GHz Intel Pentium 4 CPU took only a few milliseconds to finish the operations.) 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.

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 performed simulations with different sampling rates on the topology shown in Figure 3. The sequence of delay samples for each path was processed with our wavelet denoising method. To minimize effects from synchronization offset, we used a topology with a common source. The source nodes were co-located and their clocks were synchronized. A full evaluation involving synchronization offset will be presented in Section 5. Each link had a bandwidth of 1.5 Mb/s, and ON-OFF CBR flow parameter settings were identical to those in Section 2.4. To simulate shared congestion, we put 100 ON-OFF CBR 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 varied between 2% and 12%. For independent congestion, we put 60 ON-OFF CBR flows on the shared link, and 100 on the others.

Given a sampling rate, an experiment was 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 values with shared congestion never overlaps with the corresponding interval with independent congestion for every rate but 1 Hz.

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[3] confirm that round-trip time is less than 1 second for the vast 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

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

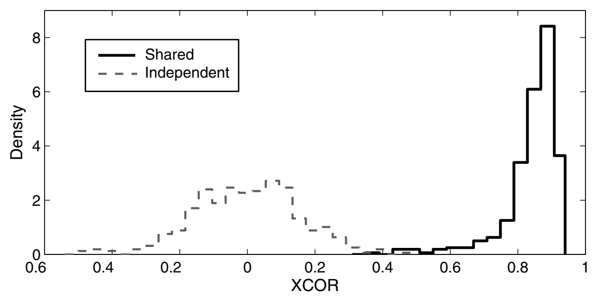


Figure 11: Cross-correlation coefficient distributions

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 minimizes 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 times, and the relationship between the target offset and the accuracy of binary decision.

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. (The error rate is not sensitive to the choice of the threshold value as long as the threshold is between 0.3 and 0.6, because XCOR is rarely close to 0.512.) We use this value as the threshold in later experiments, unless stated otherwise. We will 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 of Rubenstein et al. [15] and a loss-based one of Harfoush et al. [8, 9]. Below we refer to them respectively as MP (Markovian probing) and BP (Bayesian probing). See Section 2.5 for descriptions of both techniques.

We define *Positive Ratio* as a metric to represent accuracy of each technique.

$$\text{Positive Ratio} = \frac{\#\text{ of answers indicating shared congestion}}{\#\text{ of experiments}} \tag{8}$$

If an experimental setup involves shared congestion, Positive Ratio should be close to one; otherwise, it should be close to zero.

We first compare our technique with MP and BP when paths share a common source node and have either shared congestion or independent congestion only. Then we investigate how they perform in more challenging environments involving paths not sharing a common source or destination and multiple points of congestion. Finally, we present initial results on the performance of our technique on the Internet.

5.1 Probing with a common source

Both MP and BP assume that there is a common source (or a common destination for MP). 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, BP requires the stronger condition that two probe packets with different destinations must be sent back-to-back.

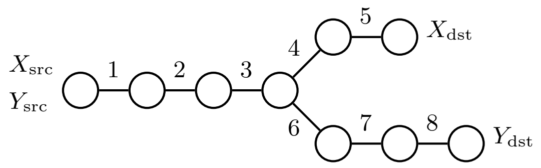


Figure 12: Topology with a common source

Figure 12 shows a network topology where two paths share a source node. Each link has a bandwidth of 1.5 Mb/s. A similar topology was used in simulations for MP [15]. We ran experiments for the following three scenarios depending on the type of background traffic.

Long-lived TCP flows A small number of long-lived 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.

ON-OFF CBR flows A large number of ON-OFF CBR flows are used as background traffic. 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 CBR flows. The number of ON-OFF CBR flows on the other links is chosen uniformly between 31 and 70. For independent congestion, links 1 through 3 have ON-OFF CBR flows between 31 and 70, and the other links between 61 to 100. The same parameter settings of ON-OFF CBR flows as in Section 2.4 are used.

Short-lived TCP flows A large number of short-lived TCP flows, created by ns-2's web traffic generator, are used as background traffic. The generated traffic consists of many

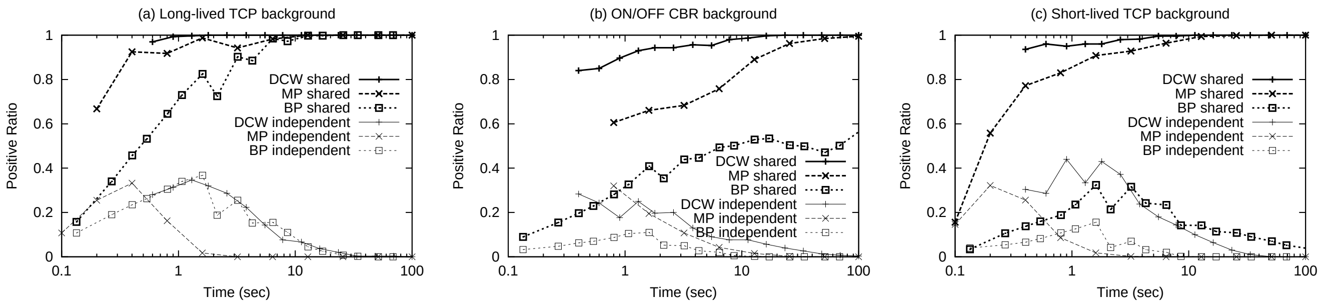


Figure 13: Convergence with a common source and drop-tail queues

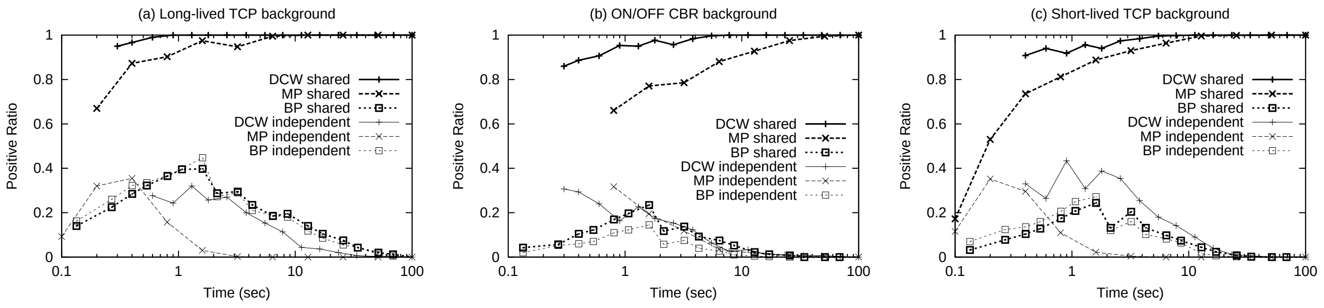


Figure 14: Convergence with a common source and RED queues

"web sessions," in each of which a client node continually downloads from a server a web page containing multiple objects. For shared congestion, a link chosen from links 1 through 3 has 250 web sessions created by 25 web servers and 250 clients. The number of web sessions on the other links is chosen uniformly between 1 and 25. For independent congestion, links 1 through 3 have web sessions between 1 and 25, and the other links have web sessions between 151 and 250.

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. With long-lived TCP background, 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. MP is the fastest in detecting independent congestion, in this case and many others below, because relatively small delay fluctuations on independent links can make the cross-measure M_x smaller than M_a. We will show this in Sections 5.2 and 5.3. Overall, every technique works well and reaches accuracy over 90% within 10 seconds.

With ON-OFF CBR background, however, all three techniques are slower in detecting shared congestion than with long-lived TCP background. For DCW and MP, this is because non-congested links have small queueing delay fluctuations. For DCW, such fluctuations add noise to delay samples; for MP, they change the order in the merged samples and thus decrease M_x. 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. This is because our ON-OFF CBR background flows include some with very short ON/OFF time, while all ON-OFF CBR 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 with ON-OFF CBR background. Degradation of BP is even more pronounced with short-lived TCP background, because a loss period is even shorter in that scenario. As a result, BP fails to detect shared congestion. On the other hand, DCW and MP are not affected much.

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 using ON-OFF CBR flows [8], 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 close 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.

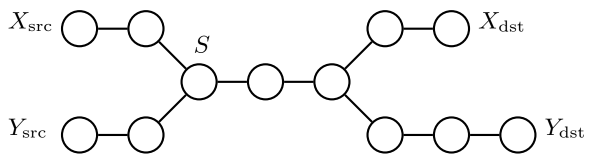


Figure 15: Topology with no common endpoint

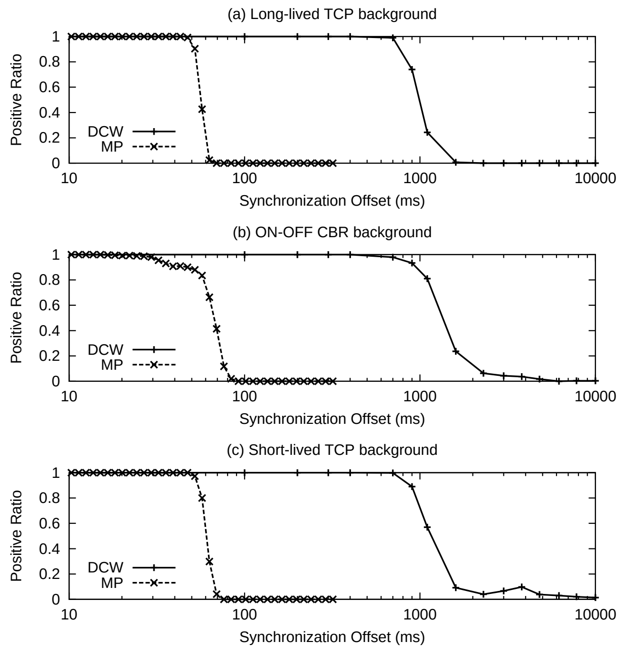


Figure 16: Effect of synchronization offset

5.2 Probing with no common endpoint

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, the delay from X_{src} to S is different from the delay from Y_{src} to S.

5.2.1 *Effects of synchronization offset*

To investigate the effect of synchronization offset between two paths, we plot, in Figure 16, the Positive Ratio for experiments with shared congestion as we increase the synchronization offset for all three types of background traffic. The original sets of delay samples were obtained from the two paths on the topology in Figure 12; the synchronization offset was added to the set of delay samples between Y_{src} and Y_{dst}. Only the overlapping portions were used. BP is excluded; its Positive Ratio 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 for shared congestion, MP may exhibit lower performance because of low accuracy if the number of delay samples is not large. Thus, detection used delay samples belonging to the first 100 seconds of the overlapping period to ensure that both MP and DCW had 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 probe rate in MP is 25 Hz, equivalent to 40 ms inter-departure time. 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 processed the data used in Figure 2 with our wavelet denoising, and plotted cross-correlation coefficient versus synchronization offset. The solid curve represents the mean cross-correlation coefficients, and the vertical bars indicate 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 horizontal line representing the threshold (0.512). This means that the cross-correlation technique without denoising is only as good as random decision at this point. 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. On the other hand, for independent congestion, the mean cross-correlation coefficients are not affected by wavelet denoising and are almost zero regardless of the synchronization offset.

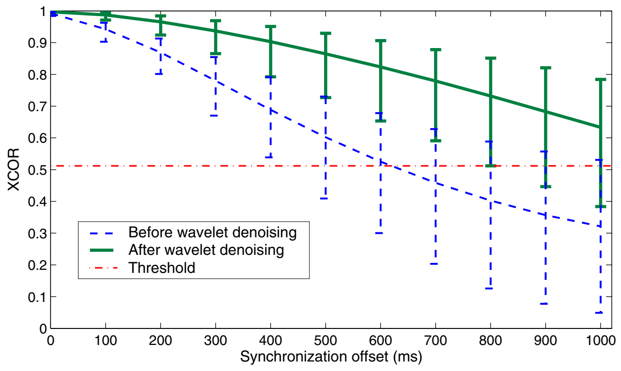


Figure 17: The effect of wavelet denoising on cross-correlation with synchronization offset

Since synchronization offset may vary during delay measurements, we also performed an experiment with a randomized synchronization offset. For a given value of average synchronization offset m, the actual synchronization offset for a particular pair of packets in the two sequences of an experiment was chosen randomly over the interval $[0, 2m]$. The mean cross-correlation results were almost the same as those in Figure 17; the variances were larger due to the presence of randomized synchronization offsets.

5.2.2 *Threshold value and false positive/negative*

We use the receiver operating characteristic (ROC) curves to show the effect of the threshold value on false positive and false negative ratio in the presence of synchronization offset. ROC is a performance test methodology that measures the probability of detection P_D against the probability of false positive P_F [17]. In our application, they are defined as follows for a certain threshold value of cross-correlation T_{XCOR}.

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

ROC performance can be graphically detected for all possible values of threshold T_{XCOR}; as we move along an ROC

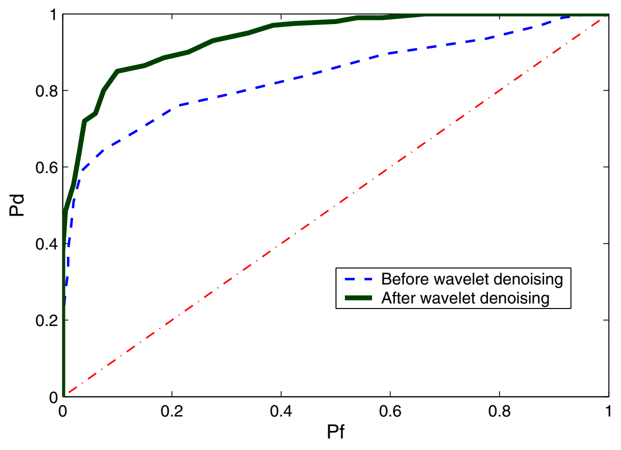


Figure 18: ROC with and without wavelet denoising

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, where the detection probability P_D equals the false positive probability P_F.

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, 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 ROC area. 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.

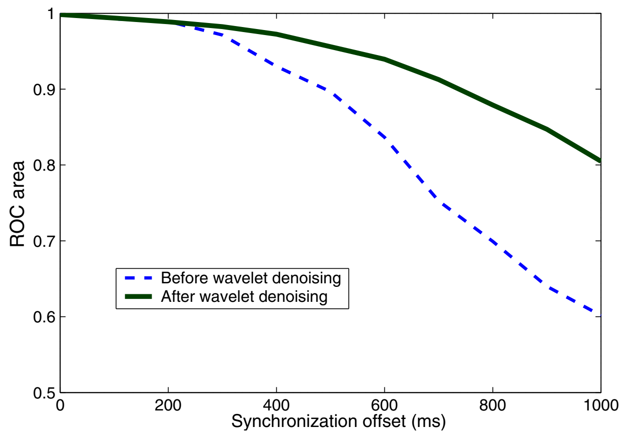


Figure 19: ROC performance versus synchronization offset with and without wavelet denoising

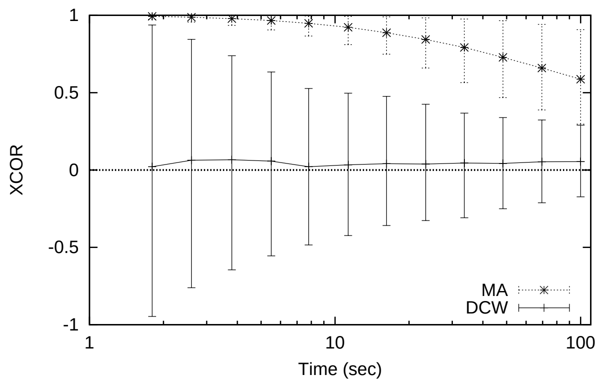


Figure 20: Convergence of low-pass filtering and wavelet denoising for independent congestion

5.2.3 *Comparison with low-pass filtering*

When congestion occurs on shared links, wavelet denoising makes cross-correlation evaluation more robust by smoothing delay data curves. We tested a simpler mechanism to achieve this smoothing, namely a simple low-pass filter. With suitable parameters, a moving average was able to provide similar improvement as wavelet denoising for cases with shared congestion. (We set the span of the moving average to 1.1 sec, which provides the same improvement as wavelet denoising for the experiments of Figure 17.) The problem with this filter appears in experiments with independent congestion. Figure 20 shows the convergence of the cross-correlation coefficient for the moving average (MA) and DCW when there is independent congestion in the experiment of Figure 13(b). Each point is the mean coefficient over 500 simulations; the bars show 5th and 95th percentiles. The mean coefficient of the moving average at 100 seconds is still 0.6, while that of DCW is almost zero from the beginning. That is, a simple low-pass filter may over-smooth transients at small scales, and thus require more delay samples to detect independent congestion. The ability of wavelet denoising to preserve strong transients at both small and large scales is critical for fast convergence in both shared and independent congestion scenarios.

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 is 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 responds 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.

$$L_s = \frac{L_{\text{shared}}}{L_{\text{shared}} + \max(L_1, L_2)} \qquad (9)$$

If $L_{\text{shared}} > 0$ and $L_1 = L_2 = 0$, then L_s becomes 1; if $L_{\text{shared}} = 0$ and at least one of L_1 and L_2 is not zero, then 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 used the topology in Figure 3. The number of ON-OFF CBR background flows on each link was chosen uniformly between 81 and 100, resulting in loss rate between 0 and 12%, and delay samples were collected for 100 seconds. L_s was computed from the actual loss rates of the links. 1000 experiments were classified into 10 groups depending on the interval their L_s belonged 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 (defined in Eq. 8) was calculated over all experiments in the same group. The results for DCW, MP, and BP are presented in Figure 21.

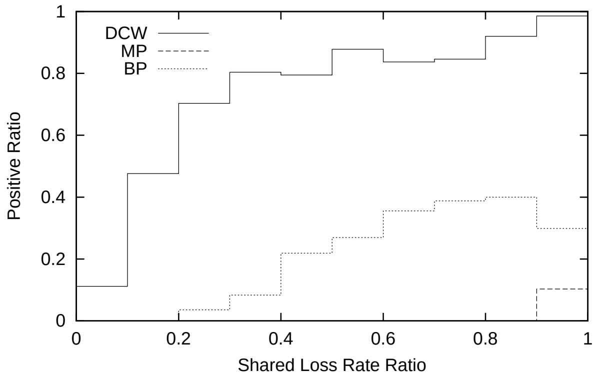


Figure 21: Positive Ratio with multiple points of congestion

Positive Ratio of DCW is only about 0.1 when $L_s < 0.1$, but 0.8 or larger when $L_s \geq 0.3$. Thus, 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 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 the application. DCW can be customized for applications with different needs by adjusting its 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 be a good choice.

5.4 Internet Experiments

We applied our technique to a large-scale network, the Internet. Our preliminary Internet experiments involved six end hosts. Figure 22 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 Austin, Texas, U.S.A. The other three hosts, K, T, and H, are located in Korea, Taiwan, and Hong Kong, respectively.

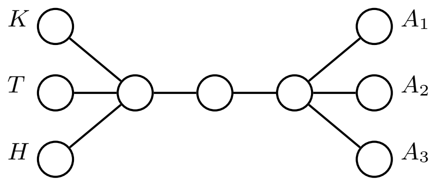


Figure 22: Experimental topology on the Internet

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 were running. Since both paths experienced a similar loss rate, we conclude that the congestion occurred on one of the shared links.

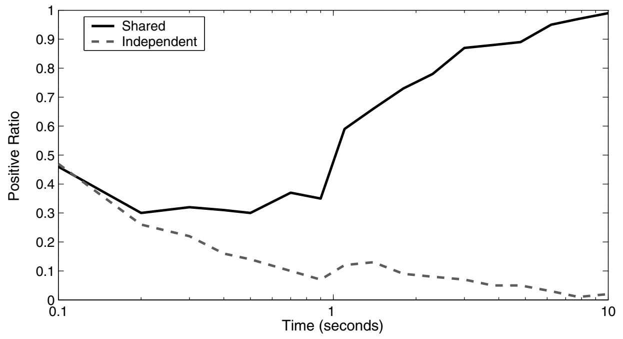


Figure 23: Convergence with Internet traces

Positive Ratio for shared congestion and independent congestion (or no congestion in this case) is shown in Figure 23. The delay samples were collected for 15 seconds, and time was adjusted with measured clock difference between A_1 and A_2 by exchanging packets between them. Each experiment was 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.

This experiment shows that our technique works well with real background traffic, and also that it diagnoses non-shared congestion correctly even when there is no congestion. However, the experiment was performed with limited settings, which included long-delay transpacific links and proximity of source nodes. Additional experiments are needed for more diverse environments.

6. CONCLUSION AND FUTURE WORK

Network resources are better utilized when multiple flows cooperate. However, such cooperation is feasible only when we can identify flows sharing a congested bottleneck. Previously proposed techniques had limitations, including a common endpoint and (sometimes) drop-tail routers. 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 based on wavelet denoising and cross-correlation, namely DCW. The denoising process effectively removes noise and makes our technique more resilient to synchronization offset, which confuses other techniques. In simulations with shared congestion, DCW achieves faster convergence and broader application than previous techniques, while using fewer probe packets. Preliminary experiments on the Internet confirmed the simulation results. We believe that applications requiring topology construction in the application layer can benefit from our delay correlation technique with wavelet denoising.

To validate DCW further, more extensive Internet experiments are necessary. We are building an overlay network to convey multimedia data, which will provide a large-scale testbed for DCW. An actual deployment of DCW for peer-to-peer applications running on this network is also under study. In addition to validating DCW through experiments, we are exploring other signal processing approaches to improve robustness for different traffic patterns and network scales; alternative denoising approaches and different wavelets will be investigated.

7. ACKNOWLEDGMENTS

We thank our shepherd, Eddie Kohler, and the anonymous reviewers for their constructive comments.

8. 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.