Remote Detection of Bottleneck Links Using Spectral and Statistical Methods

Xinming He [a], Christos Papadopoulos [b], John Heidemann [c], Urbashi Mitra [d], Usman Riaz [d],

[a] *Cisco Systems, Inc., 725 Alder Drive, Milpitas, CA 95035, USA*

[b] *Colorado State University, Computer Science Department, Fort Collins, CO 80523, USA*

[c] *University of Southern California, Computer Science Department, Los Angeles, CA 90089, USA*

[d] *University of Southern California, Electrical Engineering Department, Los Angeles, CA 90089, USA*

Abstract

Persistently saturated links are abnormal conditions that indicate bottlenecks in Internet traffic. Network operators are interested in detecting such links for troubleshooting, to improve capacity planning and traffic estimation, and to detect denial-of-service attacks. Currently bottleneck links can be detected either locally, through SNMP information, or remotely, through active probing or passive flow-based analysis. However, local SNMP information may not be available due to administrative restrictions, and existing remote approaches are not used systematically because of their network or computation overhead. This paper proposes a new approach to remotely detect the presence of bottleneck links using spectral and statistical analysis of traffic. Our approach is *passive*, operates on *aggregate traffic* without flow separation, and supports *remote detection* of bottlenecks, addressing some of the major limitations of existing approaches. Our technique assumes that traffic through the bottleneck is dominated by packets with a common size (typically the maximum transfer unit, for reasons discussed in Section 5.1). With this assumption, we observe that bottlenecks imprint periodicities on packet transmissions based on the packet size and link bandwidth. Such periodicities manifest themselves as strong frequencies in the spectral representation of the aggregate traffic observed at a downstream monitoring point. We propose a detection algorithm based on rigorous statistical methods to detect the presence of bottleneck links by examining strong frequencies in aggregate traffic. We use data from live Internet traces to evaluate the performance of our algorithm under various network conditions. Results show that with proper parameters our algorithm can provide excellent accuracy (up to 95%) even if the traffic through the bottleneck link accounts for less than 10% of the aggregate traffic.

Key words: Spectral Analysis, Bottleneck Detection, Traffic Analysis

1. Introduction

A link is *saturated* when the offered load at the link exceeds its capacity. A saturated link is very likely to be the *bottleneck* link for traffic passing through it. Links may become saturated for brief moments during normal operation in the Internet. However, with the exception of expensive, highly utilized links (e.g., satellite or deep space links), a *sustained*, saturated link typically implies an abnormal condition in the network. Examples of such links that are of interest to network operators include: (a) an under-provisioned link that may spend most of its time saturated as many users share its capacity; (b) an under-provisioned access link that may indicate a customer's need to purchase more bandwidth; (c) a denial-of-service (DoS) attack that saturates a link near the victim; (d) individual attackers in a distributed DoS attack that saturate their access links as they try to send packets as fast as possible; and (e) links that may become accidentally saturated due to faulty software or hardware.

These examples are of interest for several reasons. First, information about saturated links is necessary to influence long-term decisions such as capacity planning and traffic matrix estimation, and in short-term response to external attacks or internal bugs. More importantly, bottlenecks represent a performance problem for users of the network. Network operators would like to systematically monitor bottlenecks and report on them to their users. If traffic is limited by an access link, it shows that a user needs to purchase greater capacity. If within the ISP, a bottleneck link represents a problem that may affect multiple users and must be diagnosed. If outside the ISP, the bottleneck link demonstrates that the problem is external.

Email addresses: `xhe@cisco.com` (Xinming He), `christos@cs.colostate.edu` (Christos Papadopoulos), `johnh@isi.edu` (John Heidemann), `ubli@usc.edu` (Urbashi Mitra), `uriaz@usc.edu` (Usman Riaz).

Currently, saturated links can be detected through direct observation, e.g., network monitoring with SNMP. While network monitoring tools are widely used, they are not a panacea, particularly because access to SNMP data is often administratively limited and does not provide visibility into a customer's behavior or an external network's performance. In addition, SNMP reports are typically averaged over long intervals (5 minutes or more) and so they miss short but recurring saturation events. Finally, in some cases SNMP data may not be collected or processed because of economic or bandwidth costs.

Detecting bottlenecks can also be done remotely by active probing (Kim et al., 2004; Hu et al., 2004) or per-flow analysis and traffic correlation (Katabi and Blake, 2001). However, such techniques either require additional traffic to be inserted into the network exacerbating the congestion on the bottleneck link, or incur high computational cost as packets have to be separated according to flows and then correlated to detect sharing of bottlenecks. Ideally a network operator would like to install a network monitoring system that can detect saturated links quickly by looking at the aggregate traffic level, and then resort to detailed per-flow analysis only if a problem is detected.

We propose an approach that can remotely detect the presence of bottleneck links in *aggregate* traffic without flow separation. Note that we are interested in *transient* bottlenecks, which are hard to detect with standard methods such as SNMP monitoring Our key observation is that when links are saturated, they send packets out as fast as possible, resulting in regular back-to-back packet transmissions. We call this regular, back-to-back packet stream *bottleneck traffic* as it is rate-limited by the capacity of the saturated (bottleneck) link. If we observe the bottleneck traffic in the frequency domain, the back-to-back packet transmissions exhibit strong periodicities regulated by the bottleneck link capacity and the packet size. While there are potentially an infinite number of combinations of link speed and packet size, resulting in many potential bottleneck frequencies, in practice, both are constrained to a relatively few, common values. Links typically come in discrete capacities (e.g., 1.5Mbps, 10Mbps, 45Mbps, 100Mbps, 1Gbps, etc) corresponding to WAN technologies. Packet sizes exhibit strong modes governed by protocol design, including 60B for TCP acknowledgments, 572B for the minimum supported Internet datagram, 1500B for maximum size Ethernet segment. Moreover, transient bottlenecks often result from large file transfers; such transfers use the maximum available packet size. Throughout this paper we assume that the bottlenecks are caused by large flows, dominated by packets with a common size (typically near the maximum transfer unit, for reasons discussed in Section 5.1). We observe that the strong periodicities in the packet transmissions along the bottleneck link can manifest themselves as strong frequencies in the spectral representation of the aggregate traffic observed at a downstream monitoring point, and we show that bottlenecks can be detected despite interference and noise created by irregular packet transmissions of other flows. Thus, we can detect the presence of bottleneck links by detecting the existence of bottleneck traffic in the aggregate traffic in the spectral domain. Our approach builds on top of prior work of spectral analysis of network traffic (Barford et al., 2002; Partridge et al., 2002; Cheng et al., 2002; Hussain et al., 2003).

The main contribution of this paper is the development of a novel approach to detect bottleneck links through the periodic packet transmissions in network traffic. Our approach is completely passive and incurs no additional network overhead. It can detect the presence of bottleneck links without flow separation even if the traffic through the bottleneck link accounts for less than 10% of the aggregate traffic at the monitoring point. We also investigate the sensitivity of our approach under different network conditions such as different bottleneck bandwidths. We have not investigated deeply the performance of our algorithms when only partial bottleneck traffic is observed. Such scenarios reduce the bottleneck signal and make detection harder. Further investigation is part of our future work.

The rest of the paper is organized as follows. First, we describe potential applications of bottleneck link detection in Section 2 and give an overview of our detection system in Section 3. Then, in Section 4 we present our technique for calculating spectral representation of network traffic. After applying this technique to visually demonstrate spectral characteristics of bottleneck links in Section 5, we propose an automatic detection algorithm based on Maximum Likelihood Detection in Section 6 and evaluate its performance using real Internet traffic in Section 7. Finally we conclude the paper in Section 9.

2. Applications of Bottleneck Detection

We are not the first to explore the problem of detecting bottlenecks by passively monitoring traffic. Others have used the regularities in the packet transmission along the bottleneck link to detect bottleneck sharing among multiple flows (Katabi and Blake, 2001). However, network operators today rarely explore traffic at this level for at least two reasons. First, current approaches to bottleneck traffic detection are computationally expensive, requiring splitting traffic into flows and then combining different flows into groups sharing the same bottleneck, an expense we avoid in this paper. Second, perhaps because of this expense, there has been limited exploration of how useful early detection of bottleneck traffic might be in network operations. We explore several possibilities below.

2.1. *Capacity Planning and Traffic Engineering*

In capacity planning and traffic engineering it is important to understand which parts of the network are bottleneck limited. Bottleneck information is useful in capacity planning to inform decisions about link upgrades; in traffic engineering it can help operators make informed decisions to fine-tune routing and label switching.

Current approaches to these problems typically use network monitoring (with SNMP (Case et al., 1990)) and traffic matrix estimation (Zhang et al., 2003). SNMP runs on individual interfaces in an ISP's network. While essential, these approaches are limited to ISP links only (customer links are typically inaccessible) and information is often aggregated at coarse timescales (5 minutes or more). Our proposed approach is able to detect the presence of a bottleneck from afar (e.g., an upstream ISP) without the use of SNMP. It does so by monitoring incoming or outgoing traffic at an ISP ingress or egress. If a bottleneck is detected at the incoming traffic, the bottlenecked is clearly external to the ISP. If the bottleneck is detected at the outgoing traffic, the bottleneck's is internal to the ISP network, and perhaps it was not detected by SNMP because it happened on a non-SNMP capable interface, or traffic from a customer. Moreover, our approach is able to detect bottleneck traffic in a manner of seconds rather than minutes. While planning and even traffic engineering decisions may be done infrequently, trend analysis of the frequency of bottlenecks is important information for the network operator. Our system does not reveal the exact location of the bottleneck link, but only the presence of one. However, it can be used as early-warning to trigger further actions such as costly active probing techniques to locate the bottleneck.

2.2. *Preemptively Diagnosing Performance Problems for Customers*

For customers it is challenging to diagnose network performance problems. Performance limitations may exist at the application, the access link, the receiver or "somewhere in the network". Our technique can be the basis for a potential new service for ISPs, where they continuously monitor customer traffic and preemptively determine if the bottleneck is inside or outside the ISP. This can be done by monitoring traffic at the ISP egress point, and using our techniques to determine whether incoming or outgoing traffic is bottlenecked on not. In addition to the value-added service to the customer, such diagnosis provides several attractive opportunities for the ISP. A frequently bottlenecked client access link might help persuade the client to upgrade. Problems inside the ISP's network can be detected and resolved quickly, before users notice. Problems tracked outside the ISP it can avoid open tickets and tarnish to the ISP's reputation.

2.3. *Detecting DDoS attacks and other network anomalies*

Although "smart" denial-of-service (DoS) attacks have been described in (Kuzmanović and Knightly, 2003), many such attacks harm their victims by simply saturating links. Our system is perfectly suited for detecting flooding attacks such as distributed DoS attacks (DDoS), where many machines (called zombies) generate small streams that result into a damaging aggregate at the target. Individual zombie attack traffic could be also be detected because it typically saturates the zombie's access link (Hussain et al., 2003). While our focus here is not on zombie detection, our techniques are directly applicable to this problem, complementing signature-based schemes and obviating the need for signature discovery and distribution. We do not evaluate the effectiveness of such techniques here and it is possible that they would generate many false positives, if focused on an individual stream. However, an ISP could look for the appearance of multiple bottlenecks at the same time, with the same destination. Such correlation would most likely drastically reduce false positives.

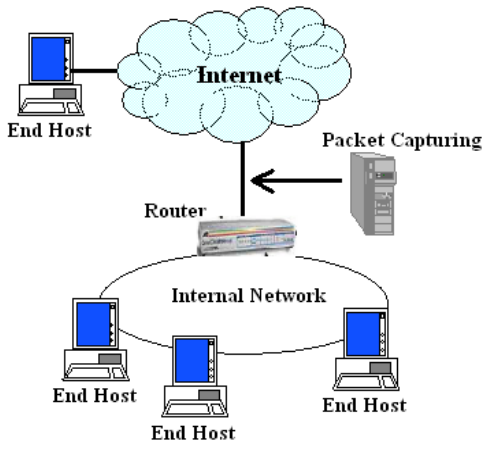


Fig. 1. A Typical System Deployment

3. System Overview

Figure 1 depicts a typical deployment of our system for detecting the presence of bottleneck flows, rate-limited by the capacity of one or more bottleneck links. Note that we expect the network administrator to deploy our system at the access link of the network to maximize traffic visibility.

Figure 2 illustrates the processing steps in our system. First, we capture packet streams from the network and record the arrival times of each packet. Then, we map the packet arrival-time sequence to a uniformly sampled time series. Each number in the series is the number of packets that arrive during a fixed interval. A Fourier Transformation converts the time domain representation into the frequency domain, where the Power Spectral Density (PSD) shows the strength (or energy) at each individual frequency. Finally, we employ an algorithm based on Maximum Likelihood Detection (Trees, 1968) to examine the PSD of the aggregate traffic and detect bottleneck flows using the probability density functions (PDFs) estimated from training data (a step that must be performed infrequently, when traffic changes drastically – we will describe this shortly).

Next, we briefly describe capturing network traffic. The remaining steps are covered in Sections 4 and 6.

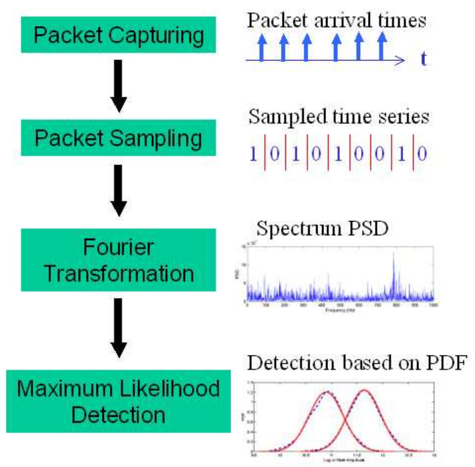


Fig. 2. Processing Steps

3.1. *Packet Capture*

Packet capture is typically done by *capture machines* exposed to network traffic via *port mirroring* or *in-line tapping*.

Port mirroring is commonly available in most enterprise level routers and switches. Here routers forward a copy of the traffic through one or more ports to a mirror port connected to the capture machine. While this method is common, it adds a burden to the router and may add jitter, caused by queuing at the router.

In-line tapping uses a network tap physically attached to the monitored link. In-line tapping does not affect network operation, except for a brief interruption when it is first installed. It works at line speed and does not distort packet timing. A disadvantage is the need for additional hardware.

Once packets reach the capture machine, they can be captured with normal network cards and commonly available sniffing tools (e.g., tcpdump). This inexpensive approach, however, may drop packets at high speeds and suffer from inaccurate timestamps due to interrupt coalescing. More expensive specialized devices such as the Endace DAG network monitoring cards (Endace, 2005), support line speed capture and nanosecond timestamp resolution, as well as GPS synchronized clocks.

4. Spectral Representation of Network Traffic

While there are many different methods to analyze network traffic once it is captured, ours differs from most in that our analysis is in the spectral domain. This section briefly reviews the techniques used by our system for calculating the spectral representation of network traffic. In Section 5 we use our techniques to visually observe the spectral characteristics of bottleneck links to build intuition.

Other researchers have explored spectral representations of network traffic (Barford et al., 2002; Partridge et al., 2002; Cheng et al., 2002; Hussain et al., 2003). Here we adopt the technique proposed by Hussain et al. (Hussain et al., 2003) with minor modifications, to obtain the spectral representation of network traffic.

Our approach has three main steps. First, we capture packet traces from the network (see top box, Figure 1) using tcpdump or other tools. The only information we need from the trace is the packet arrival time. We divide the captured trace into segments of ℓ-second long before further processing. The length of each segment ℓ is a configurable parameter; we discuss its selection in the next subsection.

Second, we sample each segment with a sampling rate p (see second box, Figure 1; we discuss in Section 4.1 how to select a proper p) to obtain a time series X, where $X(i)$ is the number of packets that arrive in the time period $[\frac{i}{p}, \frac{i+1}{p})$. Time is relative to the start of a segment, and i varies from 0 to $\ell \times p - 1$. This results in $N = \ell \times p$ number of samples for each segment. We then subtract the mean arrival rate before proceeding with spectral transformation in the next step, since the mean value results in a DC component in the spectrum that does not provide useful information for our purposes.

Third, the Power Spectral Density (PSD) is computed as the discrete Fourier transform of the auto-covariance function (ACF) of the time series data (see third box, Figure 1). Auto-covariance is a measure of how similar the stream is to itself shifted in time by offset k (Box et al., 1994; Bracewell, 1986). When $k = 0$ we compare the packet stream to itself, the auto-covariance is maximum and equals to the variance of the packet stream. When $k > 0$ we compare the packet stream with a version of itself shifted by lag k. The auto-covariance sequence $c(k)$ at lag k is

$$c(k) = \sum_{t=0}^{N-k-1} (X(t) - \bar{X})(X(t+k) - \bar{X}); \tag{1}$$

where $\bar{X}$ is the mean of $X(t)$, N is the number of samples, and k varies from $-N$ to N.

The PSD is obtained by applying discrete-time Fourier transform to the auto-covariance sequence of length M. While the PSD contains both phase information and amplitude information, we are mostly interested in the amplitude information calculated as follows.

$$S(f) = \left| \sum_{k=0}^{M-1} c(k) e^{-\imath 2\pi f k} \right| \tag{2}$$

The spectrum amplitude $S(f)$ captures the power, or strength, of the individual observable frequencies embedded in the time series. The overall computational complexity of calculating PSD for a segment is $O(N + N^2 + N * log(N)) = O(N^2)$.

4.1. *Parameter Selection*

There are two important parameters to be selected for PSD calculation. The first one is the trace segment length ℓ. If the segment length ℓ is too short, we may not have enough samples to reveal interesting patterns inside the packet trace from the spectrum. What it shows may be temporary or transient phenomena on the network. If it is too long, patterns inside the packet stream may have changed during this long period. For the detection of bottleneck traffic, we suggest using a segment length on the order of seconds. Here we use a default value of 1 second and investigate the impact of segment length in Section 7.5.

The sampling rate p is another important parameter. Given a sampling rate p, the highest frequency that is observable is $\frac{p}{2}$ according to the Nyquist Theorem. If the sampling rate is too low, aliasing can occur. If it is too high, it will increase both storage and processing overhead unnecessarily. A compromise has to be made between reducing the overhead and obtaining a better spectral representation. In this paper, we select a conservative sampling rate of 200kHz, which is sufficiently high to capture the periodic patterns of transmitting 1500-byte packets over an 100Mbps Ethernet link (8333 packets per second). A more thorough exploration of the trade-off for selecting a proper sampling rate is the subject of future work.

5. Building Intuition: Spectra of Bottleneck Traffic

The final step in Figure 1 is to detect bottleneck links based on the traffic PSD. In this section, we apply the techniques described in Section 4 to visually demonstrate the spectral characteristics of bottleneck traffic. The goal is to develop intuition behind our automated detection algorithm. While in this section we focus on graphical representations of the spectrum, in Section 6, we build upon intuition to design a quantitative algorithm to detect bottleneck traffic. We begin with experiments in a simple controlled lab environment and then proceed to more complex wide-area experiments.

5.1. *Signatures in Controlled Lab Experiments*

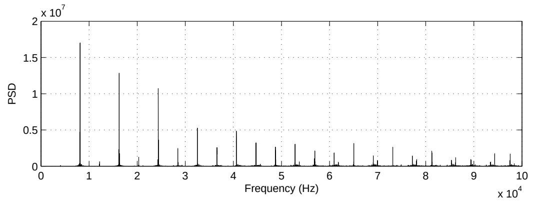
When a link is congested, it sends packets out back-to-back. Assuming for now that all packets are of the same size, this translates to a single periodic pattern in the spectral domain, with a period equal to the packet interarrival time. The packet interarrival time is calculated as follows.

$$Interarrival\ Time = \frac{Packet\ Size}{Link\ Bandwidth} \tag{3}$$

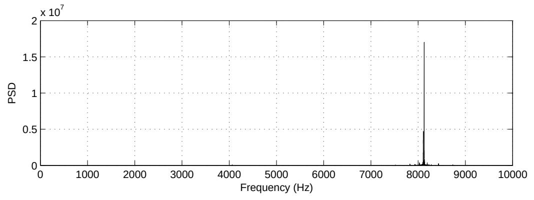
The frequency of this pattern is the inverse of the interarrival time and it can be calculated as follows.

$$Base\ Frequency = \frac{Link\ Bandwidth}{Packet\ Size} \tag{4}$$

Our approach assumes that the bottleneck traffic is dominated by packets of a single size. It is well established that a few common packet sizes dominate Internet traffic (see, for example, (Claffy et al., 1998), (Katabi and Blake, 2001), and (Sinha et al., 2006)). We believe bottlenecks will be particular prone to a few packet sizes, since they are typically caused by either denial-of-service attacks or large or many TCP flows. Denial-of-service attacks today typically use fixed packet sizes, partially because random packet sizes would make them vulnerable to entropy analysis (Lakhina et al., 2005). TCP ensures that for bulk traffic, the majority of segments will be as large as the the maximum transmission unit. Only a mix of many short TCP flows will show significant variability in packet sizes, but studies show that most Internet traffic (by bytes) is in longer flows (for example, (Thompson et al., 1997)). For these reasons we focus on detecting the effects of a single dominant packet size, and the experiments in this paper primarily use the packet size 1500-byte for the bottleneck traffic. However, our approach becomes less effective if our assumption about packet size is relaxed, and it would work poorly if the bottleneck traffic consists of packets with random sizes.



(a) Complete Spectrum



(b) Partial Spectrum

Fig. 3. Spectral signature of a 100Mbps link saturated with a TCP flow

Our experiments use a very simple topology, where the sender and the receiver are directly connected through an Ethernet switch. We select a switch instead of a hub because a switch forwards traffic more efficiently and switches are far more widely used today. We use tcpdump on the receiver side to gather packet traces. We also use different Ethernet bandwidth (10 or 100Mb/s) and vary the transport protocol (TCP or UDP) to get the spectra under different scenarios. We use Iperf (Tirumala et al., 2003) to generate TCP and UDP packet streams, which mimic TCP file downloads or UDP CBR (Constant-Bit-Rate) traffic. Each experiment lasts for 30 seconds. We divide the trace into individual segments, each 1-second long and use the techniques in Section 4 to calculate the PSD for each segment. With the exception of the first segment, which includes startup effects, we observe little variation in the spectra among the remaining 29 segments. Thus, we only present results from an arbitrarily selected segment from the 29 segments.

5.1.1. *TCP Traffic through a 100Mbps Bottleneck*

In the first experiment, we use a single Iperf TCP flow to saturate a 100Mbps switched Ethernet link with 1500-byte

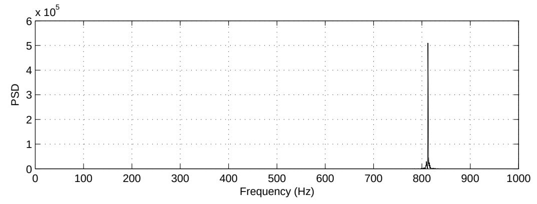


Fig. 4. Spectral signature of a 10Mbps link saturated with a TCP flow

packets. We set the TCP socket buffer size to 128K bytes to ensure there is no window starvation and RTT was less than 1ms. Under these conditions, TCP utilizes virtually the entire link bandwidth; we measured actual bit rates of around 99.9Mbps (including Ethernet header overhead).

Figure 3(a) illustrates the complete measured spectrum of the packet stream along the Ethernet link. A complete measured spectrum is the spectrum that shows the energy (or strength) at all observable frequencies from 0Hz up to the Nyquist limit (half the sampling rate), 100KHz in our case since we sample at 200KHz. The amplitude at each frequency represents the power or strength of the frequency. In Figure 3(a), we observe spikes around the 8130Hz base frequency and at multiples (harmonics) of this frequency. The amplitude at 8130Hz in the PSD reaches nearly 17,000,000, while the amplitudes at other frequencies are much lower, especially at frequencies other than 8130Hz and its harmonics. The important conclusion of this simple experiment is that the periodic patterns are very distinct and clearly visible.

We refer to the 8130Hz base frequency as the *bottleneck frequency*. To fully understand why there is strong energy at 8130Hz, we precisely calculate the frame size on the wire. The Ethernet frame format, specified in IEEE 802.3 2002 Standard (IEEE, 2002), specifies that each Ethernet frame has a 38-byte overhead, including an 8-byte preamble, a 6-byte destination MAC address, a 6-byte source MAC address, a 2-byte type/length field, a 4-byte CRC, and a 12-byte "Inter-Packet Gap". The max length of the Ethernet data payload is 1500-byte, which is what we used with the Iperf TCP flow in our experiment. Hence the packet interarrival time for the Iperf flow according to Equation 3 is (1500 + 38) * 8 bit / 100Mbps = 0.12304ms. The inverse of 0.12304ms is a frequency of 8127.44Hz. However, since tcpdump (our packet capture tool) has a time resolution of 1 microsecond, the majority of the packet interarrival times stamped at 0.123ms, resulting in strong energy around 8130Hz (the inverse of 0.123ms), in the actual spectral representation.

From this experiment, we can see that high energy around the 8130Hz base frequency and its harmonics is a strong indication of the presence of traffic through a 100Mbps bottleneck link. We explain the presence of harmonics in Section 4.1. Since harmonics do not provide additional information, we focus mainly on the partial spectrum in the range 0Hz..10kHz in Figure 3(b).

5.1.2. *TCP Traffic through a 10Mbps Bottleneck*

In the second experiment, we repeat the prior experiment but with a 10Mbps switched Ethernet link. TCP traffic can also nearly fully utilize the link bandwidth with an actual bit rate very close to 10Mbps (including the Ethernet header overhead).

Figure 4 shows the corresponding spectrum of the packet stream. As expected, we see strong energy around 813Hz, a tenth of the previous experiment. The PSD amplitude at 813Hz reaches 510,000, which is much lower than the amplitude in the previous experiment. The lower amplitude is not surprising since the packet rate here is much lower. Note that we excluded harmonics from this graph.

5.1.3. *UDP Traffic through a 10Mbps Bottleneck*

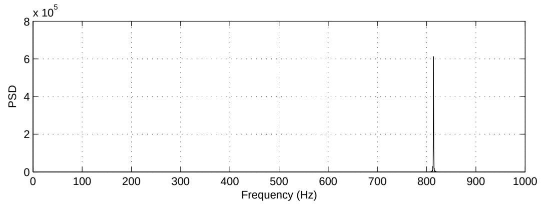
To investigate how the spectrum changes with CBR traffic, we use Iperf to saturate the 10Mbps link. We configure Iperf to send out 1472-byte UDP packets with a sending rate of 10Mbps. The Ethernet frame data payload length is 1500 bytes after adding 28-byte UDP/IP headers. We measure 10Mbps (including Ethernet packet overhead), verifying that the UDP flow can fully utilize the link bandwidth.

Figure 5(a) depicts the spectrum of the Iperf UDP flow. we observe a single peak at the 813Hz base frequency (again, we exclude harmonics). The PSD amplitude at 813Hz is 612,000, which is a bit higher than the amplitude for TCP in Figure 4. We believe the reason to be that packet transmission with UDP is more regular than TCP due to UDP's lack of a window mechanism. However, the difference in our experiments is not significant due to lack of competing cross traffic.

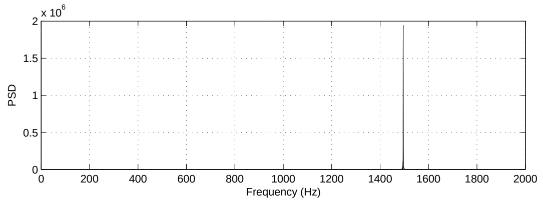
In the next experiment, we try a different packet size. We again use Iperf to send 772-byte UDP packets at 10Mbps. The Ethernet frame data payload length is 800 bytes after adding the 28-byte UDP/IP header. The bit rate along the Ethernet link is still 10Mbps (including the Ethernet header overhead). The spectrum in Figure 5(b) shows a spike around 1492Hz, which agrees with Equation 4, with 10Mbps link bandwidth and 838-byte packet size (800-byte data payload plus 38-byte Ethernet overhead). In the PSD, the amplitude at 1492Hz reaches 1,944,000, which is more than three times the amplitude at 813Hz for the earlier experiment with 1500-byte packets in Figure 5(a).

5.1.4. *A Multi-Flow Bottleneck*

In the above experiments the bottleneck was saturated with a single Iperf flow. In practice, however, high-bandwidth links are often saturated by many flows. Our approach detects the characteristics of a saturated link, whether saturated by one flow or many, but becomes less effective if we can only observe some of the bottleneck traffic. To understand this scenario better we conducted controlled lab experiments with the following two scenarios: (a) we observe all flows through the bottleneck, and (b) we observe only a fraction of the flows through the bottleneck.



(a) With 1472-byte UDP Packets



(b) With 772-byte UDP Packets

Fig. 5. Spectral signature of a 10Mbps link saturated with different size UDP packets

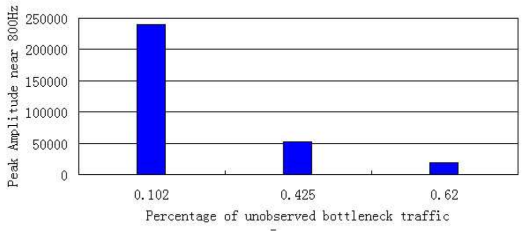


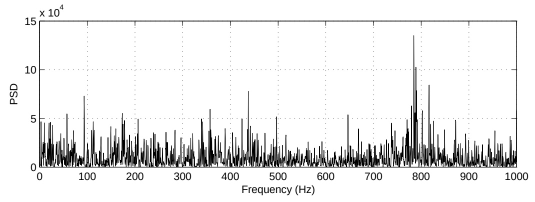
Fig. 6. Peak Amplitude with increasing percentage of unobserved bottleneck traffic

In case (a) the bottleneck traffic spectrum is similar to the case where a single flow saturates the bottleneck link. The reason is that the bottleneck shapes packet transmission based on bandwidth and packet size, regardless of how many flows are saturating the link. Experiments supporting this observation are omitted here due to space limitations, but can be found in our technical report (He et al., 2005).

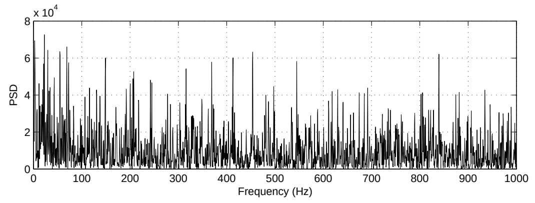
Case (b) is more challenging as we do not observe all packets that carry the bottleneck signal. We call the flows that traverse the bottleneck link but do not reach the observation point *unobserved bottleneck traffic*; the fraction of unobserved traffic greatly affects our ability to detect the bottleneck.

Figure 6 shows the impact of unobserved bottleneck traffic. In the test, we use one Iperf TCP flow and multiple web flows generated by a tool called Surge (Barford and Crovella, 1998) to saturate a 10Mbps switched Ethernet link. Only the Iperf TCP flow reaches the observation point, while the web flows do not reach the observation point and thus serve as unobserved bottleneck traffic. We vary the number of simulated web users in Surge to change the unobserved bottleneck traffic volume.

The X-axis of the graph shows the percentage of unobserved bottleneck traffic in terms of packet count out of all packets through the bottleneck in a 1-second window. The Y-axis shows the peak amplitude of observed bottleneck traffic around the 800Hz frequency. The high-level observation is that as the percentage of unobserved bottleneck traffic increases the bottleneck signal (i.e., the peak amplitude) becomes weaker and hence more difficult to detect. We provide additional results and more detailed discussion on the effects of unobserved bottleneck traffic elsewhere (He, 2006). In this paper, we focus on case (a) where we observe all bottleneck traffic. Refinement to improve sensitivity to partial observation is an area of future work.



(a) with an Iperf flow through a 10Mbps bottleneck link



(b) without the Iperf flow

Fig. 7. Power spectra of aggregate traffic at USC Internet II link

5.2. *Wide-area Network Experiments*

The previous examples show that strong periodicities of bottleneck traffic can be easily observed from the spectra in simple lab experiments without cross traffic. However, the situation is more complicated in wide-area network environments with real Internet traffic. Differences include a much more complex network topology, the monitoring point being far away from the actual bottleneck link, and much richer cross traffic, which will interact with the bottleneck traffic and affect the spectra.

We attempt to observe some of these challenges by carrying out experiments on a live, wide-area network. We place the capture machine close to an Internet II router at the edge of our university's network. We mirror traffic at the router to a separate port and capture it using a Endace DAG network monitoring card (Endace, 2005). These purpose-built cards are capable of nanosecond resolution timestamps and can keep up with the 1Gbps links without dropping any packets. We introduce an artificial 10Mb/s bottleneck traversed by an Iperf TCP flow, which flows through the capture point.

Figure 7(a) shows the spectrum of aggregate traffic observed at the capture machine, which includes the bottleneck flow. The throughput of the TCP flow is around 9.03Mbps, and the aggregate traffic volume is 30.9Mbps. We see a spike around 790Hz in the PSD. Compared with the spectrum in Figure 4 we observe three main differences. First, the peak amplitude in the spectrum appears at 793Hz, which is close, but different than the 813Hz observed for the spectrum without cross traffic. Second, the peak amplitude is 140,000, only 27.5% of the peak amplitude of the previous case. We believe that these differences are due to interference from cross traffic. While it is hard to precisely quantify how cross traffic affects the spectrum of the aggregate traffic as the underlying process is non-linear in nature, we have two general observations. First, cross traffic competes with the bottleneck traffic and distorts the regularity in the periodic nature of the bottleneck traffic. This results in a lower peak amplitude and a slightly different location of the peak frequency. Second, background traffic at the monitored link introduces its own frequency components to the observed spectrum resulting into noise. We expect that the presence of noise will make if harder to detect the signal from the bottleneck traffic. We investigate the effect of noise on detection in later sections.

In Figure 7(a) we observe the spectra from a different experiment, this one between our university and the University of Santa Barbara. Here we still see a prominent spike near the 813Hz base frequency despite the the presence of noisy cross traffic. This suggests that there are cases where the bottleneck traffic could be detected even when *mixed* with aggregate traffic. Finally, Figure 7(b) shows the spectrum of aggregate traffic at a time when our artificial TCP bottleneck flow was not present. Note that the strongest energy around 790Hz is only 1/3 of the peak amplitude when the bottleneck flow is present. The aggregate traffic volume is about 17.8Mbps.

From the above experiments we conclude that despite noise introduced by background traffic the bottleneck traffic spectra may still be detected by examining the PSD of the aggregate traffic. This is encouraging and motivates the work in the next section, where we present a quantitative approach to detecting the presence if bottleneck traffic by examining the aggregate.

6. Detection of Bottleneck Traffic

In the previous section we have visually demonstrated the presence of a spectral signature of bottleneck traffic in both controlled lab experiments and wide-area network experiments. In this section we propose and investigate the performance of an automated process to detect bottleneck traffic based on a classic statistical method, Maximum Likelihood Detection. We will first describe how to apply Maximum Likelihood Detection with a generalized framework suitable for any detection feature. Then, we will propose a specific algorithm, namely the Top-Frequency Algorithm, which extracts the peak amplitude in a frequency window as the detection feature. we will also simplify the matching operation by approximating the data with normal distributions. We will explore alternative detection features later, in Section 6.5.

6.1. *Maximum Likelihood Detection*

Maximum Likelihood Detection (Trees, 1968) is a mature technique that has been applied to many problems. We apply it to the detection of bottleneck traffic by treating the detection problem as a binary-hypothesis-testing problem where hypothesis H_B corresponds to the presence of bottleneck traffic of a given type in aggregate traffic, and hypothesis H_0 corresponds to the absence of such bottleneck traffic.

For these two hypotheses, we select a feature drawn from the aggregate traffic as the random variable. We discuss potential features in Sections 6.4 and 6.5. For example, one possible feature can be the highest amplitude in a frequency window of the spectrum of the aggregate traffic. We denote the PDFs (probability density functions) for the feature under these two hypotheses as $p(x|H_0)$ and $p(x|H_B)$, respectively. Formally, if a random variable has a probability density function $f(x)$, then it has a probability of $f(x)dx$ to have a value in the infinitesimal interval $[x, x + dx]$.

We now first introduce Maximum Likelihood Test Rule that determines which hypothesis is more likely to be true for a given trace segment using the PDFs of the designated feature in Section 6.2. Then, we describe the two phases of the overall scheme, the training phase and the detection phase, in Section 6.3.

6.2. *Maximum Likelihood Test Rule*

Assuming we know the PDFs for both hypotheses H_0 and H_B, we can determine which hypothesis is more likely to be true for a given aggregate trace segment, i.e. if the aggregate trace segment contains bottleneck traffic of a given type or not, by comparing the values of the two PDFs at X, where X is the value of the selected feature in the aggregate trace segment. The test rule is defined as follows:

$$\begin{aligned} &\text{if } p(X|H_B) \geq p(X|H_0), \text{ select } H_B \\ &\text{if } p(X|H_B) < p(X|H_0), \text{ select } H_0 \end{aligned} \tag{5}$$

We do not consider the prior probabilities of the two hypotheses H_0 and H_B in our test rule because they are hard to obtain. If the prior information were available, we could improve the accuracy of the test by forming Maximum a Posteriori test, which compares $P[H_0]p(X|H_0)$ and $P[H_B]p(X|H_B)$, where $P[H_0]$ and $P[H_B]$ are the prior probability for hypothesis H_0 and H_B.

6.3. *Two Phases in Maximum Likelihood Detection*

In order to apply Maximum Likelihood Testing in Section 6.2, we need to know the PDFs of the designated feature for each of the two hypotheses. Thus, our overall scheme has two phases: training and detection. We describe each of them in detail below.

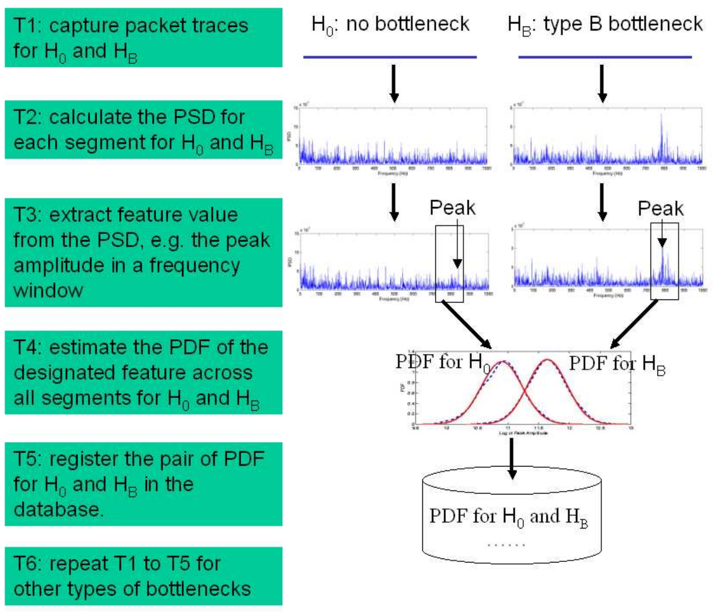

T1: capture packet traces for H0 and HB
T2: calculate the PSD for each segment for H0 and HB
T3: extract feature value from the PSD, e.g. the peak amplitude in a frequency window
T4: estimate the PDF of the designated feature across all segments for H0 and HB
T5: register the pair of PDF for H0 and HB in the database.
T6: repeat T1 to T5 for other types of bottlenecks
H0: no bottleneck
HB: type B bottleneck
Peak
Peak
PDF for H0
PDF for HB
PDF for H0 and HB
......

Fig. 8. Steps in the training phase

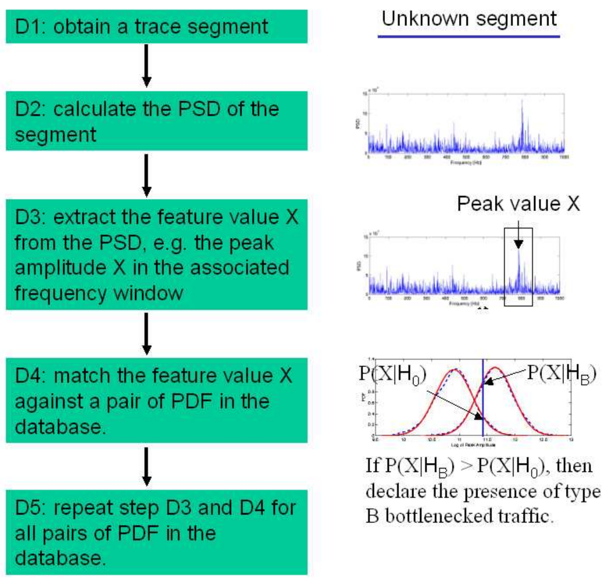

D1: obtain a trace segment
D2: calculate the PSD of the segment
D3: extract the feature value X from the PSD, e.g. the peak amplitude X in the associated frequency window
D4: match the feature value X against a pair of PDF in the database.
D5: repeat step D3 and D4 for all pairs of PDF in the database.
Unknown segment
Peak value X
P(X|H0)
P(X|HB)
If P(X|HB) > P(X|H0), then declare the presence of type B bottlenecked traffic.

Fig. 9. Steps in the detection phase

In the *training phase*, we estimate the probability density functions for the designated feature for each hypothesis. The steps are illustrated in Figure 8. In the first step, T1, we capture a training trace that we know contains no bottleneck traffic and another trace that contains bottleneck traffic of a given type using existing trace capturing techniques. Then, in step T2, we segment both traces and calculate the spectrum for each segment according to the equations in Section 4. Next, we extract the value of the designated feature from the spectrum of each segment in step T3. Our preferred feature is the peak amplitude in a frequency window. We present it formally in the next subsection, and explore alternatives in Section 6.5. In step T4, we estimate the PDFs of the designated feature based on its values across all segments for each of the two hypotheses. As a result, we get the PDF for H_0 and the PDF for H_B. We then register this pair of PDFs into a database together with the associated bottleneck traffic type in step T5. We repeat the same steps T1 - T5 to get the pairs of PDFs for other types of bottleneck traffic and register them in the database. Note that we always form a binary-hypothesis-testing problem for each type of bottleneck traffic, and we may use different features, e.g., peak amplitude in different frequency windows, for different types of bottleneck traffic.

In the *detection phase*, the algorithm uses the PDFs obtained in the training phase to determine if an unknown trace segment contains bottleneck traffic or not. The detailed steps are illustrated in Figure 9. We first obtain a trace segment in step D1. Then, we calculate the spectrum for the trace segment in step D2 using the equations in Section 4. In step D3, we extract the value of the feature from the spectrum of the trace segment. Then, in step D4, we match the feature value against a pair of PDFs registered in the database by applying the Maximum Likelihood Test Rule developed in Section 6.2 to determine which hypothesis is more likely. If H_B is more likely, then we declare the presence of bottleneck traffic of the given type in the trace segment. In step D5, we repeat step D3 and D4 to test the trace segment with other pairs of PDFs registered in the database to see if it contains other types of bottleneck traffic.

It is important to note that, theoretically, the detection space for different types of bottleneck traffic can be infinite. In practice, however, there are a few common discrete values that variables such as bandwidth and packet size can take. For example, link bandwidth is typically discrete with very few possible values (e.g., 10Mbps, 100Mbps, 1Gbps, etc.). Common packet sizes include 64B, 576B, or 1500B. Transport protocols are mostly limited to TCP and UDP, although other protocols such as SCTP are beginning to emerge. This means that the search space can be greatly reduced by only looking at combinations of common values.

6.4. *Top-Frequency Detection Algorithm*

As alluded to previously, there are a number of possible features upon which to base our detection algorithm. Here we describe an algorithm that uses the peak amplitude in a given frequency window as the detection feature. We review alternative features briefly in the following subsection.

Figures 3 to 5 suggest that in the absence of background traffic, the periodicity exhibited by a bottleneck reveals itself by a strong peak near the key frequency determined by the bottleneck traffic bandwidth and packet size. Even when obscured by background traffic, we can still see a modestly sized peak in Figure 7(a). Thus, we conjecture that the peak amplitude near the associated key frequency is a feature of interest. In addition, due to the impact of background traffic, the exact location of the peak amplitude varies slightly from time to time, suggesting that we should

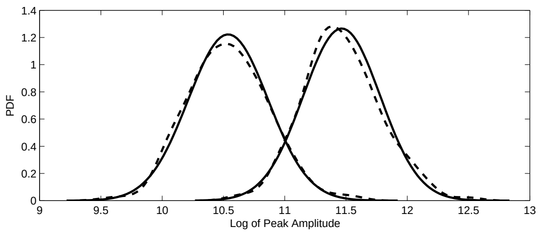


Fig. 10. PDFs of the peak amplitude (after log) in [780Hz, 800Hz] for H_0 (left dashed line) and H_B (right dashed line)

search in a window near the key frequency to catch the bottleneck signal.

To take advantage of these observations, we design the *Top-Frequency Detection Algorithm*, which uses the peak amplitude within a frequency window as the feature to distinguish between H_0 and H_B. The frequency window is selected according to a number of factors, most importantly the bottleneck bandwidth and packet size. For example, we would check the window around 800Hz for a 10Mbps link saturated by 1500-byte packets as opposed to the 8kHz for the 100Mbps link by the same 1500-byte packets. The window size should not be too small or we will miss the strong energy associated with the bottleneck for some instances. Neither should it be too wide as this may include strong energy caused by other network phenomena. We will discuss the proper selection of window location and size in Section 7.3 and 7.4 using real Internet traffic.

Rather than directly compare empirical probability density functions, we simplify the operation by first fitting a mathematical distribution to the data. This approach allows us to estimate the observed behavior parsimoniously and simplifies the hypothesis testing. As we do not have concrete intuition for a particular distribution model, we have tested a number of distribution models to fit them to the actual data. The log-normal distribution is selected for a balance between simple description and goodness of fit with the data.

The log-normal distribution means the log of the peak amplitude follows a Gaussian distribution. Thus, in the log-domain, the parameters which completely characterize the two distributions are the means and standard deviations, *i.e.*(μ_0, σ_0) for H_0 and (μ_B, σ_B) for H_B. Formally, the two distributions are given by,

$$p(x|H_0) = \frac{1}{\sigma_0\sqrt{2\pi}} e^{-(x-\mu_0)^2/2\sigma_0^2} \tag{6}$$

$$p(x|H_B) = \frac{1}{\sigma_B\sqrt{2\pi}} e^{-(x-\mu_B)^2/2\sigma_B^2} \tag{7}$$

where x is the *log of the peak amplitude* in the selected frequency window. The mean and standard deviation under each hypothesis are determined from the training data.

Although the log-normal distribution does not always offer the best fit to the data, it provides an excellent tradeoff between fit and a parsimonious modeling of the feature. Figure 10 shows the distributions of the peak amplitude in the frequency window [780Hz, 800Hz] for the training data associated with the two hypotheses, H_0 and H_B, in one experimental trace set involving a 10Mbps TCP bottleneck flow. All data is presented in the log domain. The dashed lines are the empirical PDFs for H_0 and H_B, while the solid lines are the normal distributions with the mean and standard deviation derived from the experimental set for H_0 and H_B. We can see the two empirical PDFs can be closely approximated by normal distributions in the log domain represented by the solid lines. We have also verified that these two distributions follow normal distributions at the 5% significance level through the Lilliefors test (Lilliefors, 1967).

With the log-normal distribution, we can simplify the Maximum Likelihood Test Rule in Section 6.2 by solving the equation $p(x|H_0) = p(x|H_B)$ first (recall that x here is the log of the data). This equation yields a quadratic function which has two roots if $\sigma_0 \neq \sigma_B$, and one root if $\sigma_0 = \sigma_B$. In general, we expect trace segments in H_B to have higher peak amplitudes than trace segments in H_0 as we have seen in Figure 10. Thus, if the quadratic equation has two roots, we discard the root which contradicts this expectation, i.e. classifies segments with lower peak amplitudes into H_B and segments with higher peak amplitudes into H_0. We select the root that agrees with the expectation as the cut-off threshold. If there is only one root for the quadratic function, then this root should agree with the expectation and it will be selected as the cut-off threshold.

In both cases, the detection rule can be simplified to a direct comparison between the cut-off threshold and the peak amplitude of the input trace in the selected window. If the latter is larger, then the input trace will be classified as H_B, having the associated bottleneck traffic. Otherwise, we select H_0 and declare that there is no such bottleneck traffic. The cut-off threshold can be visually seen in Figure 10 as the point where the two PDFs cross.

The computational cost for the Top Frequency Algorithm can be calculated as follows. Assuming that we build a database of N types of bottleneck traffic, for each type we gather M trace segments with the bottleneck traffic and M trace segments without the bottleneck traffic, the size of the frequency window used in the Top Frequency Algorithm is W, then the complexity in the training step would be $O(N*(O(M*PSD)+O(M*W))) = O(N*M*PSD)$, where PSD is the cost to calculate the PSD of a trace segment which is greater than O(W). The computational cost for detecting bottleneck in a trace segment would be $O(N*(PSD+O(W))) = O(N*PSD)$.

6.5. *Detection Using Other Features*

We have designed and investigated the performance of a suite of detection algorithms using different kinds of features selected from the trace spectra. Table 1 summarizes their differences. In the Single-Frequency Algorithm, we examine the amplitude at a particular frequency (*e.g.* 800 Hz). We also use log-normal distribution to model the PDFs

Table 1
Comparison of detection features

Algorithm	Feature	PDF Model
Top-Frequency	peak amplitude in a frequency window (after log)	normal distribution
Single-Frequency	amplitude at a single frequency (after log)	normal distribution
Top-M-Frequencies	M peak amplitudes in a frequency window (after log)	multi-variate normal distribution
All-Frequencies	all amplitudes in a frequency window (after log)	multi-variate normal distribution

of H_0 and H_B and calculate a cut-off threshold for classifying new unknown traces in the same way as in the Top-Frequency Algorithm. The Top-M-Frequencies Algorithm is a generalized variation of the Top-Frequency Algorithm. It considers the M highest amplitudes in a frequency window. The method assumes that the vector comprised of the log of these amplitudes follow a multi-variate Gaussian distribution. Finally, we consider all amplitudes within a frequency window in the All-Frequencies Algorithm. It also assumes a multi-variate Gaussian distribution of the log of these amplitudes.

Our experimental results show that the Top-Frequency Algorithm performs much better than the Single-Frequency Algorithm, since it accommodates the possible shift of the peak amplitude associated with the bottleneck traffic in a small frequency window. Its performance is also comparable to the two multi-variate detection algorithms while having much lower computational overhead since it only models the distribution of a single random variable. Due to space limitations we only present the results for the Top-Frequency Algorithm in this paper. Details of other methods can be found in (He et al., 2005).

7. Evaluation with Real Internet Traffic

To systematically evaluate the performance of our detection algorithm we artificially introduce bottleneck traffic into a wide-area network and observe it in packet traces gathered at the Internet-2 access link at our university. Overall, our primary goal is to understand the performance of our detection algorithms under different network conditions and with different algorithm parameters. We first study the impact of algorithm parameters in Section 7.3, 7.4, and 7.5. We then consider the stability of the results with different transport protocols in Section 7.6 and the selection of a proper frequency window in Section 7.7. Finally, we investigate the effect of using different training data in Section 7.8 and how the signal-to-noise ratio (defined as the ratio of bottleneck traffic volume to background traffic volume) affects algorithm performance in Section 7.9.

Table 2
Experiment Scenarios

Scenario	Bottleneck Traffic Type	Background Traffic Volume
T10L	an Iperf TCP flow through a 10Mbps bottleneck	low (24Mbps to 59Mbps)
T10H	an Iperf TCP flow through a 10Mbps bottleneck	high (94Mbps to 186Mbps)
U10L	an Iperf UDP flow through a 10Mbps bottleneck	low (19Mbps to 57Mbps)
T100H	an Iperf TCP flow through a 100Mbps bottleneck	high (112Mbps to 204Mbps)

7.1. *Experiment Setup*

In our experiments, we use the same wide-area network experiment setup as in Section 5.2. The actual load measured during our experiments varies from 19Mbps to 204Mbps.

We create bottleneck traffic from an outside source to a destination inside the university. The source is nine hops away from the destination. The link between the source and the first hop is the bottleneck link and is over 90% utilized. No other flow shares this link during the experiment. The bottleneck flow traverses the Internet-2 link and is observed by the capture machine together with the other background traffic.

During our experiments there are three variables: the transport protocol of the bottleneck traffic (TCP or UDP), the bandwidth of the bottleneck link (10Mbps or 100Mbps), and the amount of background traffic at the monitored link. Among all possible permutations, we have investigated four specific scenarios as shown in Table 2. These four scenarios are selected because they capture key differences. For example, U10L differs from T10L in the transport protocol of the bottleneck traffic; T10L differ from T10H in background traffic volume; and T10H differs from T100H in the bandwidth of the bottleneck link.

For each of the four scenarios we gather a pair of packet traces at the monitored link every two hours for 24 hours. Each pair consists of a 5-minute long trace gathered when there is no intentionally introduced bottleneck flow (H_0) and another 5-minute long trace gathered when we intentionally introduce a bottleneck Iperf flow (H_B). In all cases we use a default segment length ℓ of 1 second and a default sampling rate p of 200kHz. Thus, each trace pair has 300 trace segments without a bottleneck flow and 300 trace segments with the bottleneck flow.

There are many other variations to the above four scenarios we could try, to evaluate the performance of our detection algorithm. For example, we could vary the traffic through the bottleneck by using multiple flows with different packet size distributions. We could also vary the portion of bottleneck traffic seen at the observation point. Preliminary results show that varying the observed portion of bottleneck traffic has a greater effect than varying the number of flows through the bottleneck. We plan to explore these scenarios in future work.

7.2. *Detection Accuracy*

We use *detection accuracy* to measure the performance of our detection algorithm, defined as the probability that the algorithm gives the correct answer about the existence of bottleneck traffic in the trace. To calculate detection accuracy, we adopt the following procedure.

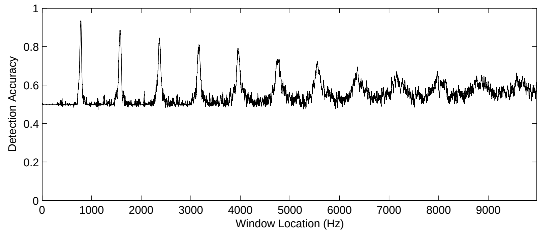
First, we select a pair of traces with and without bottleneck flow as the training data set and train the algorithm on it. Next, we use the training result to classify the trace segments in the training data and other trace pairs under the same scenario. We compare the answer by the algorithm with the ground truth to obtain detection accuracy. In our experiments, the detection accuracy is equal to the average value of the true positive rate and the true negative rate, since we have equal number of trace segments with and without the bottleneck flow.

We use two types of detection accuracies to better capture the performance of the algorithm on the training data and on other traces under the same scenario. The first is the accuracy on the training data, which we call *training accuracy*. It measures the ability of the algorithm to distinguish trace segments in the training data, where the algorithm knows the truth whether the segment has bottleneck traffic or not. The second is the average accuracy on all other trace pairs gathered under the same scenario as the training set, which we call *average accuracy* in short. The average accuracy represents the performance of the algorithm on traces where it has no prior knowledge if the segment has bottleneck traffic or not.

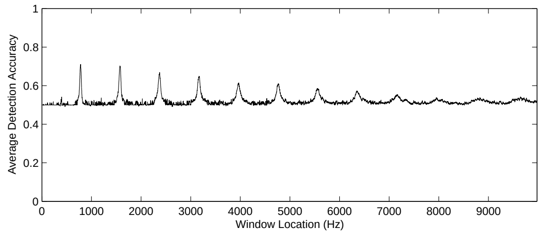
In this paper, we use training accuracy to tune the algorithm parameters, such as selecting the proper detection window. We present the corresponding average accuracy to show the expected performance of the algorithm on unknown traces (traces where the algorithm has no prior knowledge if the segment has bottleneck traffic or not). As observed in Section 7.3 and 7.4, there is a strong correlation between these two types of accuracies in their response to the changes in the algorithm parameters. This strong correlation gives credence to our methods for parameter tuning based on examining the training accuracy.

7.3. *Frequency Window Location*

The first question we consider is where the frequency window should be located to detect specific bottleneck traffic. We expect the algorithm to perform best with the window located around the base frequency calculated according to the Equation 4. For example, it should be around 813Hz for 10Mbps bottlenecks congested with 1500-byte packets. For 100Mbps bottlenecks with 1500-byte packets it should be around 8130Hz. To validate our hypothesis, we run the detection algorithm with the center of the frequency window W_c varying from near 0Hz to near 10kHz. Here, we focus on the results under two scenarios: T10L, which targets detecting a 10Mbps bottleneck; and T100H, which targets detecting a 100Mbps bottleneck.

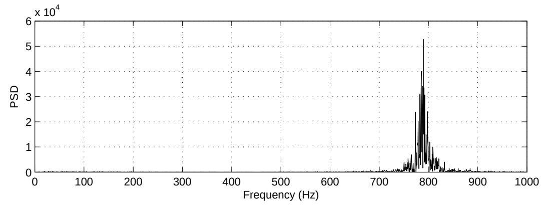


(a) Training Accuracy with 7am trace pair as the training data

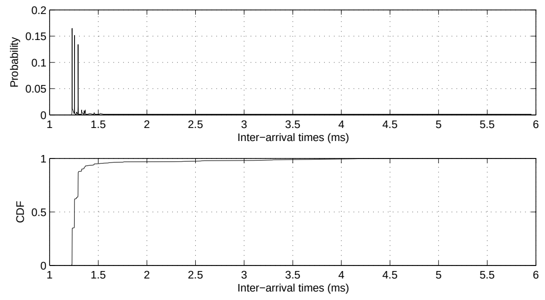


(b) Average Accuracy on other traces

Fig. 11. Accuracy in detecting the 10Mbps bottleneck (T10L) as a function of window location, $W_s = 20$Hz



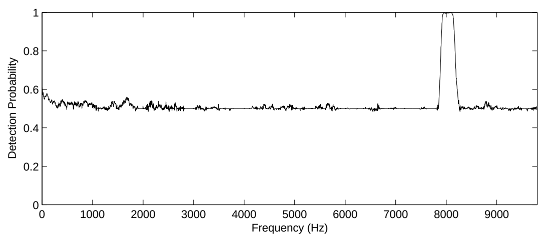
(a) Spectrum



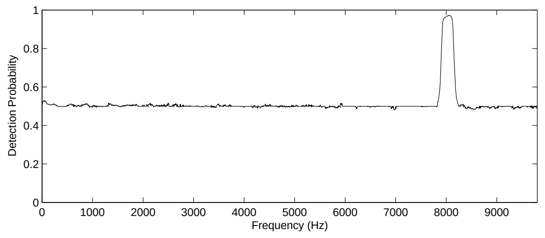
(b) PDF and CDF of packet interarrival times

Fig. 12. Spectrum and packet interarrival time distribution of the isolated Iperf TCP flow (T10L)

Figure 11 shows the impact of window location on detecting a 10Mbps bottleneck under the T10L scenario. The top subgraph 11(a) shows the training accuracy using the 7am trace pair as the training data, while the bottom subgraph 11(b) depicts the corresponding average accuracy on other trace pairs. In both subgraphs, we use a fixed window size of 20Hz and vary the window center from 10Hz to 9990Hz. We can see that the center of the frequency window has a strong effect on both types of detection accuracies. Both types of accuracies reach peak values with windows around 790Hz (close to the predicted 813Hz frequency) and its harmonics. For example, training accuracy can reach as



(a) Training Accuracy with 7am trace pair as the training data



(b) Average Accuracy on other traces

Fig. 13. Accuracy in detecting the 100Mbps bottleneck (T100H) as a function of window location, $W_s = 200$Hz

high as 93% when the window is around 790Hz, but it drops to 50% when the window is around 100Hz.

Furthermore, there is a strong correlation of the two types of accuracies in terms of their response to the changes in window location. As the window location changes, the average accuracy on other trace pairs will increase as training accuracy increases, and decrease as the latter decreases. However, in general the former is lower than the latter. For example, average accuracy is only 71% while training accuracy reaches 93% with the window around 790Hz. This indicates some mismatch between the statistics of the training trace pair and other trace pairs.

We have similar observations when we train the algorithm with different trace pairs under the T10L scenario. The correlation between these two types of accuracies demonstrates that a window with a high training accuracy will generally lead to a high accuracy on other trace pairs.

To find out why the detection accuracies peak near 790Hz instead of 813Hz as we have expected, we isolate the Iperf bottleneck flow from the aggregate and plot its spectrum in Figure 12(a) for one trace segment. We can see that the spectrum suggests the existence of significant cross traffic which triggers the TCP congestion control for the bottleneck flow resulting in less periodic packet transmissions. In the spectrum the peak amplitude appears around 790Hz, a frequency lower than 813Hz for the case without cross traffic in Figure 4. In addition, the peak amplitude is only 55,000, much weaker than the peak amplitude of 510,000 for the case without cross traffic. The weaker amplitude makes it hard to detect the bottleneck flow. The CDF and PDF of packet interarrival times for the isolated Iperf flow in Figure 12(b) show a spike bump pattern, also suggesting that the bottleneck experiences significant cross traffic in a downstream link according to the findings in (Katabi and Blake, 2001).

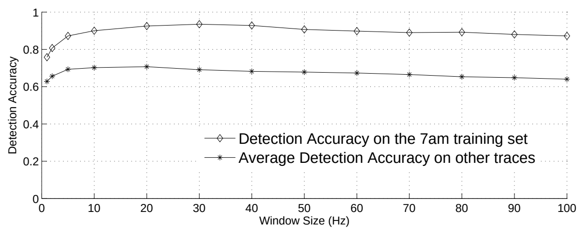


Fig. 14. Detecting the 10Mbps bottleneck (T10L) with varying window sizes, $W_c = 790$Hz

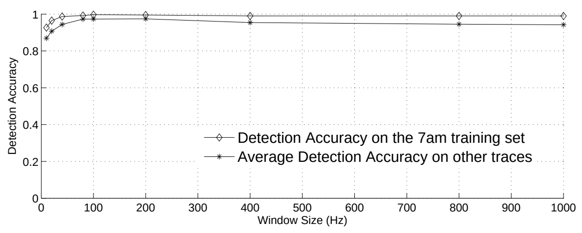


Fig. 15. Detecting the 100Mbps bottleneck (T100H) with varying window sizes, $W_c = 8100$Hz

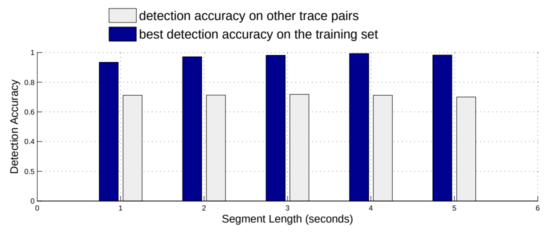


Fig. 16. Detection accuracy as a function of segment length

Figure 13 reveals the impact of frequency window location on the accuracies of detecting a 100Mbps bottleneck under the T100H scenario. We fix the window size to 200Hz here, as the predicted base frequency for a 100Mbps bottleneck link is ten times of the value for a 10Mbps bottleneck link. Again we see a strong correlation of the two types of accuracies in terms of their response to the changes in window location. Both types of accuracies peak around 8100Hz, close the predicted 8130Hz base frequency. For example, when the window is around 8100Hz, training accuracy with the 7am training trace pair can reach 100%, while the corresponding average accuracy on other trace pairs reaches 97%. These values are significantly higher than the accuracies for detecting the 10Mbps bottleneck in Figure 11, because we have much higher signal-to-noise ratios. We will discuss the impact of signal-to-noise ratio in more detail in Section 7.9. Both experiments agree with our expectation of the location of the frequency window for the best detection accuracy.

7.4. *Frequency Window Size*

In the previous section, we considered the impact of window location with fixed window sizes. In this section, we consider the impact of window size while fixing the window location. Our intuition behind the choice of window size is that the window should be neither too narrow nor too wide, since narrow windows might fail to capture the strong signal for the bottleneck that is shifted to another location due to noise, and wide windows might result in increasing false positives due to noise from unrelated events. Another reason against choosing a large window is due to the increase in the processing overhead as a larger window means more frequencies to be examined.

To systematically study the impact of window size, we run the Top-Frequency algorithm using the trace pair gathered at 7am in scenario T10L as the training set while varying the window size from 1Hz to 100Hz with fixed window center location W_c at 790Hz. Figure 14 shows the detection results. The upper line is the training accuracy on the 7am training set while the bottom line is the corresponding average accuracy on other trace pairs under Scenario T10L. We see that the two types of accuracies react in a similar way to the changes in window size. For both types of accuracies, we observe that window sizes between 20Hz to 30Hz (about 2.5% to 4% of the predicted 813Hz base frequency) give the best result. In addition, smaller sizes show much lower accuracies because with smaller windows, it becomes possible to miss a signal that is shifted outside the window due to noise. On the other hand, larger sizes also result in lower accuracies but the penalty is not as dramatic as smaller sizes.

We have also conducted a similar investigation for detecting the 100Mbps bottleneck in the T100H scenario with windows centered at 8100Hz. Figure 15 shows the detection results using the trace pair gathered at 7am as training set while varying the window size from 1Hz to 1000Hz. Again we observe a strong correlation of the two types of accuracies in terms of their response to the changes in window size. Both types of accuracies reach their peak values with the window size in the range of 100Hz to 200Hz (about 1.2% to 2.5% of the predicted 8133Hz base frequency). In addition, smaller sizes show much lower accuracies than larger window sizes.

7.5. *Sampling Rate and Segment Length*

Segment length and sampling frequency are also important parameters of the algorithm. A sampling frequency that is too low will result into aliasing; one that is too high increases processing overhead. In future work, we will investigate the optimum balance between the two. For this work we choose a conservative frequency of 200kHz. Assuming 1500-byte packet size, 200kHz sampling frequency is sufficiently high to capture the signatures of bottleneck links with bandwidth up to 100Mbps.

We desire a relatively short segment length to allow rapid detection of bottleneck traffic. However, as discussed in Section 4.1, segment length cannot be too short or we cannot compute an accurate spectrum. In addition, the algorithm may become too sensitive to transient flows.

To study the impact of segment length, we vary it from 1 second to 5 seconds. Again we use the trace pair gathered at 7am in scenario T10L as the training set. Here, we use a fixed window size of 20Hz and a fixed window location at 790Hz since results in Section 7.3 and 7.4 show they are good choices for detecting the 10Mbps bottleneck in the T10L scenario. Figure 16 shows that training accuracy and average accuracy with different segment length. We only see small changes for both types of detection accuracies when we vary the segment length from 1 second to 5 seconds. In future work we will explore other trace lengths. One practical constraint for using longer trace segment is that we need longer traces so that we have enough number of segments to form a distribution and estimate its parameters.

7.6. *Transport Protocol*

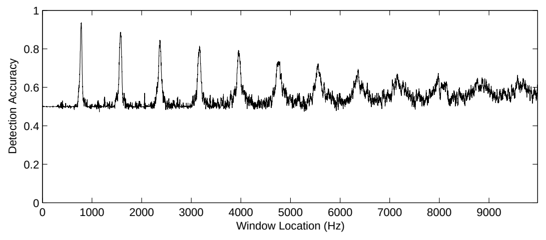
From the controlled experiments in Section 5.1, we see that UDP flows are more regular and provide stronger signals than TCP flows. Here we evaluate how that translates into the detection accuracy.

Figure 17 compares the accuracy for detecting a 10Mbps link saturated by a TCP flow (scenario T10L) and the accuracy for detecting the same 10Mbps link saturated by a UDP flow (scenario U10L). Again we vary the window center from 10Hz to 9990Hz with fixed window size of 20Hz. The two top subgraphs 17(a) and 17(b) compare training accuracy on the 7am training trace between TCP and UDP. We can see that the spikes for TCP are around 790Hz and its harmonics, and the spikes for UDP are around 810Hz and its harmonics. The 810Hz base frequency for UDP is closer to the predicted 813Hz base frequency for 10Mbps bottleneck links than the 790Hz base frequency for TCP. In addition, UDP has higher detection accuracies (up to 98%) around its base frequency than TCP (up to 93%). UDP also maintains high detection accuracies around the harmonics, while TCP has a decay of detection accuracies among harmonics as they move further away from the base frequency.

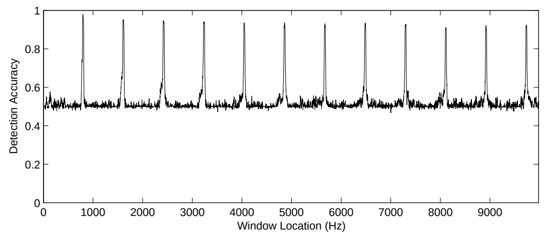
The two bottom subgraphs 17(c) and 17(d) compare the corresponding average accuracy on other traces between TCP and UDP. Again we see that UDP has spikes around 810Hz and its multiples, while TCP has spikes around 790Hz and its multiples. In addition, UDP has higher average accuracies (up to 75%) around its base frequency than TCP (up to 71%). UDP also maintains high detection accuracies around the harmonics while TCP has a decay of detection accuracies among harmonics.

The differences for the detection accuracies between TCP and UDP are due to the fact that TCP adjusts packet transmissions by considering feedback from the network while UDP does not in our experiments. This results in less periodic packet transmissions for the TCP flow, which translates in lower detection accuracies and a decay of detection accuracies among harmonics.

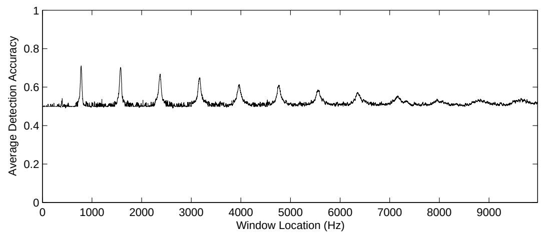
To affirm that UDP has more regular packet transmis-



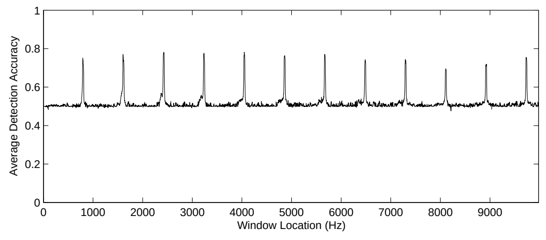
(a) Training Accuracy on 7am training set with the bottleneck saturated by TCP, $W_s = 20\text{Hz}$



(b) Training Accuracy on 7am training set with the bottleneck saturated by UDP, $W_s = 20\text{Hz}$

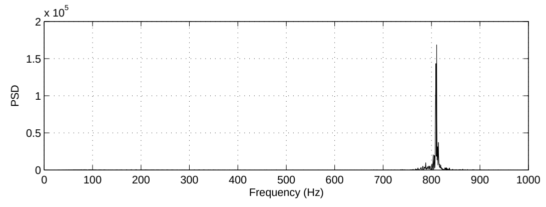


(c) Average Accuracy on other trace pairs with the bottleneck saturated by TCP, $W_s = 20\text{Hz}$

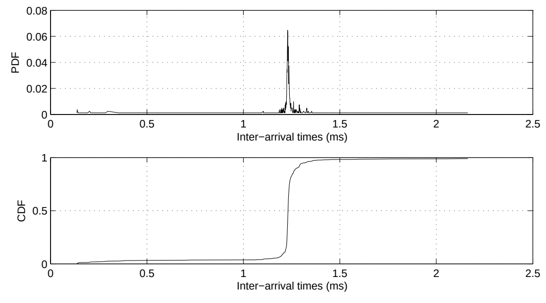


(d) Average Accuracy on other trace pairs with the bottleneck saturated by UDP, $W_s = 20\text{Hz}$

Fig. 17. Detecting the 10Mbps bottleneck saturated with different transport protocols



(a) Spectrum



(b) PDF and CDF of packet interarrival times

Fig. 18. Spectrum and packet interarrival time distribution of the isolated Iperf UDP flow (U10L)

sions than TCP, we isolate the UDP bottleneck flow from the aggregate and plot its spectrum in Figure 18(a) and the distribution of its packet interarrival times in Figure 18(b). We can see that the UDP flow exhibits much less impact from cross traffic compared with the Iperf TCP flow as shown in Figure 12(a) and 12(b). For example, the peak amplitude for the UDP flow is 170,000, more than 2 times stronger than the TCP flow, and it appears around 810Hz instead of the 790Hz for the TCP flow. The PDF and CDF of packet interarrival times for UDP show a single spike at 1.23ms, while for TCP the interarrival times has a spike bump pattern. These figures demonstrate that the UDP flow has more periodic packet transmissions than the TCP

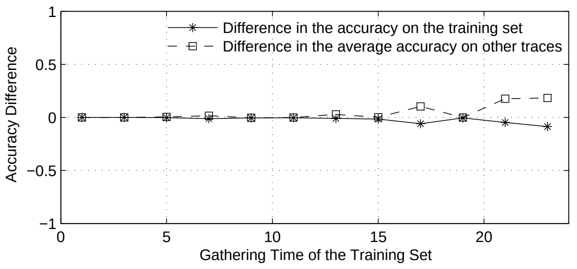


Fig. 19. Differences between using the fixed [780Hz, 800Hz] window and the best training windows in T10L scenario

flow, which makes it easier to detect UDP flows than to detect TCP flows.

7.7. *Guidelines for Selection of Frequency Windows*

Based on our investigation on the window location, window size, and transport protocol, we opt to use the following fixed detection windows for the four scenarios rather than using the window that yields the best training accuracy by exhaustively searching all possible window locations and sizes. We call the latter the *best training window*. For scenario T10H and T10L we use the 20Hz wide window centered at 790Hz. For scenario U10L we select the same size 20Hz wide window but centered at 810Hz. For scenario T100H we choose the 200Hz wide window centered at 8100Hz.

Results show that the detection accuracies with these fixed windows are comparable to the detection accuracies achieved by the best training window, and in some cases the fixed windows can even yield better average accuracy on other traces than the best training window, because the latter only guarantees the best accuracy on the training set, but not the best average accuracy on other traces.

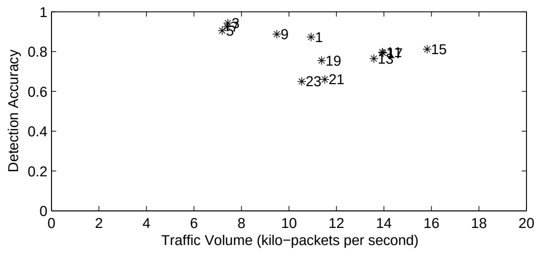
For example, Figure 19 shows the differences between using the fixed [780Hz, 800Hz] window and using the best training window in the T10L scenario. The x axis represents the time of the day when the training trace pair was gathered. For each training trace pair, we find the best training window that yields the best training accuracy by exhaustively searching all possible window locations and sizes. Then we calculate training accuracy and average accuracy on other traces using this window. In comparison, we also calculate the corresponding detection accuracies using the fixed [780Hz, 800Hz] window. The differences plotted in Figure 19 are obtained by deducting the accuracies for the best training windows from the accuracies for the fixed window. We can see that for most training traces, the differences between using the fixed window and the best training windows are small. The largest difference happens with the training trace pair gathered at 11pm. With the best training window, training accuracy is 8.7% higher than the value using the fixed window, but the average accuracy on other traces is 18% lower than the value using the fixed window.

In general, we should follow the following principles in selecting the detection window. First, we should use a detection window whose center is located near the predicted base frequency for the bottleneck link (e.g. 813Hz for 10Mbps links and 8133Hz for 100Mbps links). Second, the detection window size should be around 1% to 5% of the predicted base frequency. Third, the exact location and size of the detection window to be used in real operation should be adjusted slightly according to the specific network. Network operators should do some training to select the proper detection window.

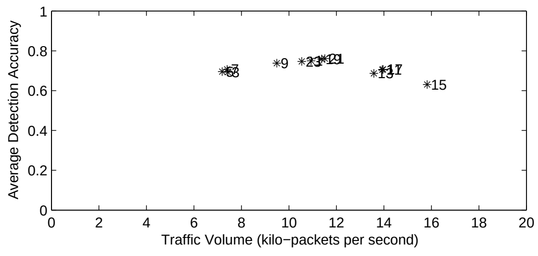
7.8. *Training Data Variation*

The accuracy of any training-based detection algorithm is influenced by the quality of the training data. In this section we investigate the impact of using different training data on the two types of detection accuracies, training accuracy and average accuracy on other trace pairs. We select the T10L scenario for illustration.

Figure 20 shows the impact of using different training data on the two types of detection accuracies. The top subgraph shows training accuracy as we vary the trace pair using for training algorithm. The x axis is the average aggregate traffic volume of the training trace pair. The number beside each point in the graph indicates the time of the day when the training trace pair was gathered. For example, the point denoted by "9" shows the training accuracy on the 9am training trace pair. In the bottom subgraph, each point represents the average accuracy on all other trace pairs using the cut-off threshold learned from the corresponding training trace pair. For example, the point denoted by "9" reflects the average accuracy on all other trace pairs using the cut-off threshold learned from the training trace pair gathered at 9am. The x axis for the bottom subgraph is also the average aggregate traffic volume of the training trace pair.

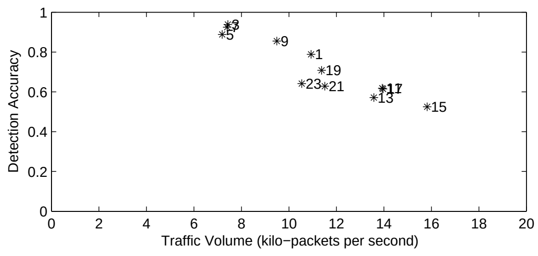


(a) Training Accuracy with 7am training set under T10L

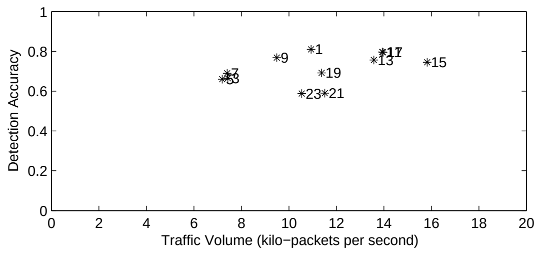


(b) Average Accuracy on other traces under T10L

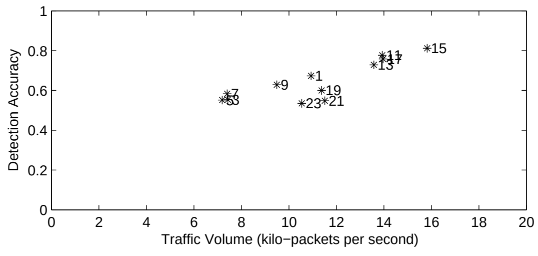
Fig. 20. Impact of training data on detection accuracies.



(a) training with low traffic volume trace (7am)



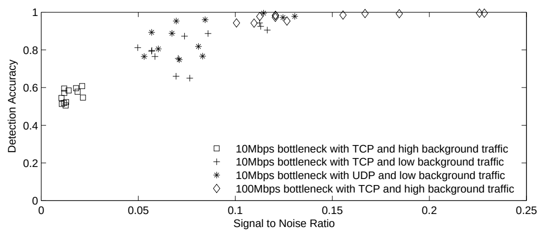
(b) training with medium traffic volume trace (11am)



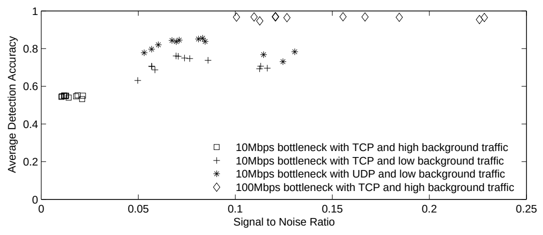
(c) training with high traffic volume trace (3pm)

Fig. 21. Training with different traffic volume traces

We see that, in general, training on trace pairs with lower aggregate traffic volume yields higher accuracies on the training set itself, with the exception of the points denoted by "19", "21" and "23" (flow-based analysis suggests that



(a) Training Accuracy



(b) Average Accuracy on other traces

Fig. 22. Detection accuracy as a function of signal-to-noise ratio using fixed frequency windows specified in Section 7.7

three points have been tainted with presence of other bottleneck traffic when we believe there is no bottleneck traffic). An intuition behind the general trend is that lower aggregate traffic means less interference to the bottleneck traffic by the background traffic on the monitored link, and this translates into clearer signal for the bottleneck traffic making it easier to detect the presence of the bottleneck flow. On the other hand, the average accuracy on other trace pairs has high values when we train on trace pairs with medium aggregate traffic volume. This agrees with the intuition that it is better to train with the common cases (middle traffic volume) instead of extreme cases (either low or high traffic volume).

As the average accuracy reported in Figure 20(b) may hide the variation among the accuracy on individual trace pairs used for evaluation, we dissect it into individual graphs, each representing the result with one training set. Figure 21 shows the detection accuracy when training with three different trace pairs corresponding to low, medium, and high aggregate traffic volume at different periods of the day. In each graph, a point represents the accuracy on the indicated trace pair using the cut-off threshold learned through the training trace pair.

We see a strong correlation between the accuracy on a trace pair and its distance to the training trace pair in terms of traffic volume. For example, in Figure 21(a), trace pairs at 3am and 5am have very close traffic volume to the training trace pair gathered at 7am. Both have accuracies very close to the accuracy on the training set at 7am. As the distance to the training set increases in terms of packet rate, the detection accuracy on the corresponding trace pair decreases. This observation suggests that we could take advantage of the similarity in terms of traffic volume between the trace pair used for training and the unknown trace to get high accuracies on the unknown trace. We could use the cut-off threshold learned through a training set that has similar traffic volume to the unknown trace for detecting bottleneck traffic in the unknown trace. Further exploration along this direction is part of our future work.

7.9. *Effect of Signal-to-noise Ratio*

Detection theory tells us that the detection accuracy is related to the signal-to-noise ratio (SNR). When detecting bottleneck traffic in aggregate network traffic, the signal is the intensity of the bottleneck traffic, and the noise is the level of background traffic. Since the spectral representation is obtained through the processing of packet arrivals, we define signal-to-noise ratio as the ratio of bottleneck traffic packet rate to background traffic packet rate. Although it is hard to quantify how SNR affects the detection accuracy as their relationship is non-linear, in general, we expect to see better detection accuracy with higher SNR.

Figure 22 shows how the detection accuracy varies as a function of signal-to-noise ratio using fixed frequency windows specified in Section 7.7. The top subgraph shows the detection accuracy on the training trace pair, while the bottom subgraph shows the detection accuracy on all other trace pairs using the cut-off learned from the corresponding training trace pair. The x axis is the signal-to-noise ratio for the training trace pair. Both subgraphs include all trace pairs under the four experiment scenarios listed in Table 2 (different scenarios are indicated with different symbols).

As expected, we see a general trend that a higher SNR leads to a better detection accuracy. The T100H scenario for 100Mbps TCP bottleneck traffic with high background traffic shows the highest detection accuracies because the bottleneck traffic (around 7.5kpps)is quite large relative to background traffic (32 - 74 kpps) for SNRs of (0.1 - 0.23). Both training accuracy and average accuracy on other traces can reach over 94% in the T100H scenario. The U10L scenario for 10Mbps UDP bottleneck traffic with low background traffic (denoted by "*") also has pretty good detection accuracies even though the SNRs are lower (0.053 - 0.125). Training detection accuracy ranges from 75% to 99%, while the average accuracy on other traces ranges from 76% to 86%. The T10L scenario for 10Mbps TCP bottleneck traffic with low background traffic shows generally lower detection accuracies (around 63% to 94%) than the U10L scenario with similar SNRs for the reason that we have discussed in Section 7.6. Finally, the T10H scenario for 10Mbps TCP bottleneck traffic with high background traffic gets the lowest detection accuracies (around 53% to 61%) as the SNRs are the lowest (0.01 - 0.02).

8. Related Work

Recently, a number of researchers have used spectral techniques to analyze network traffic for various purposes. Hussain et al. apply spectral techniques to packet arrival time series to distinguish single-source and multi-source DDoS attacks (Hussain et al., 2003). More recently, they have proposed a spectral approach for attack re-identification (Hussain et al., 2006), after detecting and isolating attack packets from background traffic. Barford et al. use wavelets to analyze IP flow-level and SNMP information to detect DoS attacks and other network anomalies (Barford et al., 2002). Cheng et al. also apply spectral analysis to separate normal TCP traffic from DDoS traffic as the former exhibits strong periodicities around its round-trip time (Cheng et al., 2002). Magnaghi et al. propose a wavelet-based framework to proactively detect network misconfigurations. It utilizes the TCP retransmission timeout events during the opening phase of the TCP connection (Magnaghi et al., 2004). Partridge et al. apply Lomb periodograms to retrieve periodicities in wireless communication, including CBR traffic and FTP traffic (Partridge et al., 2002). Kim et al. apply wavelet denoising to improve the accuracy of detecting congestion among different flows (Kim et al., 2004). It requires active probing to measure the one-way-delay.

There are also a number of non-spectral techniques for detecting congestion sharing and estimating link capacity and available bandwidth. Katabi et al. propose to use packet inter-arrival times to infer the path characteristics such as bottleneck capacity and bottleneck sharing among flows based on entropy (Katabi and Blake, 2001). Hu et al. (Hu et al., 2004) presented a tool to detect the location of a bottleneck in a path using active probing. This tool, however, would be too costly for routine bottleneck monitoring. Examples of non-spectral techniques for estimating link capacity and available bandwidth include Pathchar (Jacobson, 1997), Pchar (Mah, 1999), CapProbe (Kapoor et al., 2004), Cprobe (Carter and Crovella, 1996), Pathload (Jain and Dovrolis, 2002, 2003), IGI and PTR (Hu and Steenkiste, 2003), and Spruce (Strauss et al., 2003).

Unlike the above techniques, our approach is *passive* (i.e., does not require probing packets), operates on *aggregate traffic* (so that individual traffic flows do not have to be separated), transforms traffic into a suitable *spectral domain representation* (which is powerful to reveal periodic patterns), and makes use of *more rigorous statistical methods* rather than relying on qualitative visual evidence.

9. Conclusions and Future Work

Given the size and complexity of the Internet today, tools that can help understand and diagnose problems with the network are very important. Spectral techniques have great potential in creating powerful tools to extract hidden patterns in the network to understand phenomena ranging from network anomalies to application and protocol behavior, and detection theory provides the background to translate these "pretty pictures" into quantitative algorithms.

In this work, we presented a methodology to apply these techniques to the detection of bottleneck traffic. Applications include troubleshooting, capacity planning, traffic estimation, DDoS detection, application monitoring, etc. While we cannot pinpoint the exact location of the bottleneck, our techniques can be used to determine if the bottleneck is inside or outside the network. In addition to visually demonstrating the spectral signature imposed by bottleneck links, we proposed an algorithm based on the Maximum Likelihood Detection to automatically detect the bottleneck signature embedded in aggregate traffic, and evaluated its performance using real-world Internet traces.

Our results show that we can detect the presence of a bottleneck link several hops away from the monitoring point without flow separation, even if the traffic through the bottleneck link accounts for less than 10% of the aggregate traffic observed at the monitoring point. Our techniques are completely passive, suitable for routine network monitoring, and can detect bottlenecks remotely without the need for direct observation, which is useful when bottlenecks are outside our administrative domain. Our analysis investigated the effects of several factors on detection performance, including the effect of Signal-to-Noise ratio, the selection of the detection window, and the variation using different training data.

In the future we plan to strengthen our methodology as follows. First, we want to model the underlying processes that govern the generation of bottleneck traffic signature and how it is shaped by competing traffic, and use the model to design more sophisticated detection algorithms that take traffic load and other time-varying factors into consideration. For example, we can adjust the cut-off threshold according to the traffic volume to improve the detection accuracy since higher traffic volume will generally increase the peak amplitude in a window. Future work includes determining the relationship between traffic load and the cut-off threshold and use this relationship to improve detection accuracy.

Second, we want to apply the detection methods in more diversified environments, including different monitoring points, different bottleneck locations, different types of traffic composition (e.g., different packet size distribution and single flow versus multiple flows), and different types of cross traffic, to gain a more thorough understanding of performance.

Finally, we would like to extend our techniques into a framework that can be applied to study other periodic traffic patterns, such as protocol behavior and network anomalies. This will expand the applicability of our methodology and help gain insight into other network phenomena.

10. Acknowledgments

This paper has benefited greatly from the valuable feedback from Dr. Antonio Ortega and Dr. Alefiya Hussain.

References

Barford, P., Crovella, M., June 1998. Generating Representative Web Workloads for Network and Server Performance Evaluation. In: Proceedings of the ACM SIGMETRICS'98. Madison, Wisconsin, USA.

Barford, P., Kline, J., Plonka, D., Ron, A., November 2002. A Signal Analysis of Network Traffic Anomalies. In: Proceedings of the ACM SIGCOMM Internet Measurement Workshop. Marseilles, France.

Box, G., Jenkins, G., Reinsel, G., 1994. Time series analysis: forecasting and control. Prentice-Hall.

Bracewell, R., 1986. The Fourier Transform and Its Applications. McGraw-Hill.

Carter, R., Crovella, M., 1996. Measuring bottleneck link speed in packet-switched networks. Tech. Rep. 1996-006.

Case, J., Fedor, M., Schoffstall, M., Davin, J., May 1990. A Simple Network Management Protocol (SNMP). Request For Comments (RFC) 1157.

Cheng, C.-M., Kung, H., Tan, K.-S., Nov. 2002. Use of spectral analysis in defense against DoS attacks. In: Proceedings of the IEEE GLOBECOM. Taipei, China, pp. 2143 – 2148.

Claffy, K., Miller, G., Thompson, K., July 1998. The nature of the beast: Recent traffic measurements from an internet backbone. In: Proceedings of International Networking Conference (INET)'98. Geneva, Switzerland.

Endace, 2005. Endace DAG network capture cards, http://www.endace.com/.

He, X., 2006. Detecting periodic patterns in Internet traffic with spectral and statistical methods. Ph.D. thesis, University of of Southern California.

He, X., Papadopoulos, C., Heidemann, J., Mitra, U., Riaz, U., Hussain, A., June 2005. Spectral analysis of bottleneck traffic. Tech. Rep. USC/CS-TR-2005-853, University of Southern California Computer Science Department.

Hu, N., Li, L. E., Mao, Z. M., Steenkiste, P., Wang, J., August 2004. Locating Internet Bottlenecks: Algorithms, Measurements and Implications. In: Proceedings of the ACM SIGCOMM 2004. Oregon, USA, pp. 41–54.

Hu, N., Steenkiste, P., Aug. 2003. Evaluation and characterization of available bandwidth probing techniques. IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling 21 (6), 879 – 894.

Hussain, A., Heidemann, J., Papadopoulos, C., August 2003. A Framework for Classifying Denial of Service Attacks. In: Proceedings of the ACM SIGCOMM'2003. Karlsruhe, Germany, pp. 99–110.

Hussain, A., Heidemann, J., Papadopoulos, C., April 2006. Identification of Repeated Denial of Service Attacks. In: Proceedings of the IEEE Infocom 2006. Barcelona, Spain.

IEEE, March 2002. IEEE 802.3-2002 standard.

Jacobson, V., April 1997. Pathchar: A Tool to Infer Characteristics of Internet Paths.

Jain, M., Dovrolis, C., March 2002. Pathload: A measurement tool for end-to-end available bandwidth. In: Proceedings of Passive and Active Measurements (PAM) Workshop 2002. Fort Collins, CO, USA.

Jain, M., Dovrolis, C., August 2003. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. ACM/IEEE Transactions on Networking 11 (4), 537 – 549.

Kapoor, R., Chen, L.-J., Lao, L., Gerla, M., Sanadidi, M. Y., August 2004. CapProbe: A Simple and Accurate Capacity Estimation Technique. In: Proceedings of the ACM SIGCOMM'2004. Oregon, USA, pp. 67 – 78.

Katabi, D., Blake, C., 2001. Inferring congestion sharing and path characteristics from packet interarrival times. Tech. Rep. MIT-LCS-TR-828, Massachusetts Institute of Technology, Laboratory for Computer Science.

Kim, M. S., Kim, T., Shin, Y., Lam, S. S., Powers, E. J., August 2004. A Wavelet-Based Approach to Detect Shared Congestion. In: Proceedings of the ACM SIGCOMM 2004. Portland, Oregon, USA, pp. 293 – 306.

Kuzmanović, A., Knightly, E. W., Aug. 2003. Low-rate tcp-targeted denial of service attacks (the shrew vs. the mice and elephants). In: Proceedings of the ACM SIGCOMM Conference. Karlsruhe, Germany.

Lakhina, A., Crovella, M., Diot, C., 2005. Mining anomalies using traffic feature distributions. In: In Proceedings of ACM SIGCOMM'2005. Philadelphia, Pennsylvania, USA, pp. 217–228.

Lilliefors, H., 1967. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. Journal of the American Statistical Association 62, 399–402.

Magnaghi, A., Hamada, T., Katsuyama, T., August 2004. A Wavelet-Based Framework for Proactive Detection of Network Misconfigurations. In: Proceedings of ACM workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality. Portland, Oregon, USA, pp. 253 – 258.

Mah, B. A., Feb 1999. Pchar: A tool for measuring internet path characteristics.

Partridge, C., Cousins, D., Jackson, A., Krishnan, R., Saxena, T., Strayer, W. T., Sep. 2002. Using Signal Processing to Analyze Wireless data Traffic. In: Proceedings of ACM workshop on Wireless Security. Atlanta, GA, pp. 67–76.

Sinha, R., Papadopoulos, C., Heidemann, J., 2006. Fingerprinting internet paths using packet pair dispersion. Tech. Rep. USC/CS-TR-2006-876, University of Southern California Computer Science Department.

Strauss, J., Katabi, D., Kaashoek, F., 2003. A measurement study of available bandwidth estimation tools. In: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement. ACM Press, Miami Beach, FL, USA, pp. 39–44.

Thompson, K., Miller, G. J., Wilder, R., Nov/Dec 1997. Wide-area internet traffic patterns and characteristics (extended version). IEEE Network Magazine 11 (6), 10–23.

URL http://www.vbns.net/presentations/papers/MCItraffic.pdf

Tirumala, A., Qin, F., Dugan, J., Ferguson, J., Gibbs, K., 2003. Iperf. http://dast.nlanr.net/Projects/Iperf/.

Trees, H. L. V., 1968. Detection, Estimation and Modulation Theory. John Wiley & Sons Inc.

Zhang, Y., Roughan, M., Lund, C., Donoho, D., August 2003. An information-theoretic approach to traffic matrix estimation.