Elsevier Editorial System(tm) for Computer Networks
Manuscript Draft

Manuscript Number:

Title: OnTimeDetect: Dynamic Anomaly Notification Tool and Method in perfSONAR-instrumented Networks

Article Type: Special Issue: Complex Dynamic Networks

Keywords: plateau detector, dynamic thresholds for change detection, perfSONAR framework, anomaly analysis tool

Corresponding Author: Dr. Prasad Calyam,

Corresponding Author's Institution: The Ohio State University

First Author: Prasad Calyam

Order of Authors: Prasad Calyam; Prasad Calyam; Mukundan Sridharan; Jialu Pu; Ashok Krishnamurthy

Abstract: To monitor and diagnose bottlenecks in complex dynamic networks, there is an increasing trend to deploy multi-domain measurement frameworks such as perfSONAR. These deployments use web-services to expose vast data archives of current and historic measurements, which can be queried across end-to-end multi-domain network paths. Consequently, there has arisen a need to develop automated techniques and intuitive tools that help analyze these measurements for detecting and notifying prominent network anomalies such as plateaus in both real-time and offline manner. In this paper, we present our ``OnTimeDetect'' tool that we developed to query measurements and analyze network anomalies in measurements of complex dynamic networks instrumented via perfSONAR. The query module extends the standardized web-services schema of perfSONAR measurment archives, and the analysis module features a novel dynamically adaptive plateau detection (APD) scheme for accurate detection of prominent network anomalies. We show that notifying anomalies and identifying the causes of anomalies in complex dynamic networks requires integration of tools such as OnTimeDetect with social networking tools such as Twitter. In addition, we show that our APD scheme can be applied either: (a) directly on raw measurement data of one-way delay and throughput in path-level anomaly detection, or (b) to the transformed measurement data of principal component analysis for network-wide correlated anomaly detection. We empirically evaluate our APD scheme in terms of accuracy, agility and scalability by using measurement traces collected by OnTimeDetect tool from worldwide perfSONAR deployments in diverse networking communities. Further, we analyze the measurement sampling characteristics in the perfSONAR measurement traces to study the challenges associated with achieving low anomaly detection times in multi-domain networks.

Suggested Reviewers: Khaled Salah
khaled.salah@kustar.ac.ae

Graciela Perera
gcperera@cis.ysu.edu

Martin Swany
swany@cis.udel.edu

Sonia Fahmy
fahmy@cs.purdue.edu

Les Cottrell
cottrell@slac.stanford.edu

Opposed Reviewers:

OnTimeDetect: Dynamic Anomaly Notification Tool and Method in perfSONAR-instrumented Networks

Prasad Calyam, Mukundan Sridharan, Jialu Pu, Ashok Krishnamurthy

Ohio Supercomputer Center/OARnet, The Ohio State University,
{pcalyam, jpu, ashok}@osc.edu; sridhara@oar.net

Abstract

To monitor and diagnose bottlenecks in complex dynamic networks, there is an increasing trend to deploy multi-domain measurement frameworks such as perfSONAR. These deployments use web-services to expose vast data archives of current and historic measurements, which can be queried across end-to-end multi-domain network paths. Consequently, there has arisen a need to develop automated techniques and intuitive tools that help analyze these measurements for detecting and notifying prominent network anomalies such as plateaus in both real-time and offline manner. In this paper, we present our "OnTimeDetect" tool that we developed to query measurements and analyze network anomalies in measurements of complex dynamic networks instrumented via perfSONAR. The query module extends the standardized web-services schema of perfSONAR measurment archives, and the analysis module features a novel dynamically adaptive plateau detection (APD) scheme for accurate detection of prominent network anomalies. We show that notifying anomalies and identifying the causes of anomalies in complex dynamic networks requires integration of tools such as OnTimeDetect with social networking tools such as Twitter. In addition, we show that our APD scheme can be applied either: (a) directly on raw measurement data of one-way delay and throughput in path-level anomaly detection, or (b) to the transformed measurement data of principal component analysis for network-wide correlated anomaly detection. We empirically evaluate our APD scheme in terms of accuracy, agility and scalability by using measurement traces collected by OnTimeDetect tool from worldwide perfSONAR deployments in diverse networking communities. Further, we analyze the measurement sampling characteristics in the perfSONAR measurement traces to study the challenges associated with achieving low anomaly detection times in multi-domain networks.

1. Introduction

Scientific communities involved in major simulation and experimental activities in projects such as the Teragrid [1] and Large Hadron Collider (LHC) [2] have been generating massive data sets on a regular basis. These data sets are transferred to various labs and universities at high-speeds on multi-domain networks that span across continents. Given the real-time consumption demands of the data and the substantial investments made for the network infrastructure to support the data transfers, there is a rapidly increasing trend to deploy instrumentation and measurement frameworks such as perfSONAR [3]. These frameworks assist in measurement data collection, storage and dissemination for monitoring and diagnosing bottlenecks that hinder end-to-end data transfer speeds. In addition, the frameworks help in provisioning end-to-end network measurements that can be used to understand network performance changes in complex dynamic networks due to causes such as end-user behavioural patterns (e.g., high volume flows, flash crowds), network fault events (e.g., misconfigurations, outages, malicious attacks) and cross-traffic congestion impact end-application and protocol behavior.

The emergence of perfSONAR deployments in scientific communities that rely on diverse high-performance networking technologies is revolutionary in the area of network performance troubleshooting. This is because the deployments support web-services to publish and subscribe multi-domain network measurements of various network health metrics such as utilization, throughput, delay, jitter and loss. The web-service schemas are being standardized in the Open Grid Forum [4] and have been integrated in frameworks such as OSCARS [5], Cricket-SNMP [6], and PingER [7].

Numerous perfSONAR deployments are sampling both active and passive measurements of various metrics several times a day. They are exposing these collected measurements via web-services in the form of vast data archives of current and historic measurements on national and international backbones (e.g., ESnet, Internet2, GEANT, SWITCH, RNP). The consumers of these measurements (e.g., network operators, researchers in scientific disciplines) are faced with the challenge to analyze and interpret the vast measurement archives across end-to-end multi-domain network paths with minimal human inspection. They direly need automated techniques and intuitive tools to query, analyze, detect and notify prominent network performance anomalies that hinder data transfer speeds. Anomalies of interest can be detected at the network-path level using one-way delay and throughput measurements from OWAMP and BWCTL tools [3] used in perfSONAR deployments. Also, there is a need to detect correlated network-wide anomalies by combining several network-path level measurements in order to localize the change-cause to a particular path/link. Accurate and timely notification of anomalies using effective techniques and tools at the network-path level or network-wide level can lead to quicker resolution of network faults. Consequently, service providers can proac-

tively resolve cases where end-users may experience annoyingly slow data transfers on high-speed networks.

There have been several network anomaly detection studies that utilize various statistical and machine learning techniques such as: online change-point detection [12] [13], principal component analysis [8], wavelet analysis [9], Kalman filter [10], and covariance matrix analysis [11]. Of these schemes, plateau-detector scheme [12] that belongs in the change-point detection category has been found to be relatively more effective [14] for path-level anomaly detection. Moreover, its low complexity, ease of implementation and effectiveness have led to its adoption by large-scale monitoring deployments such as NLANR AMP [12] and SLAC IEPM-BW [13] (predecessors to the perfSONAR deployments). In comparison, q-statistic test scheme that belongs in the principal component analysis category has been found to be suited for network-wide correlated anomaly detection.

In both the network-path and network-wide levels, existing plateau-detection threshold selection approaches have limitations for perfSONAR users. The NLANR AMP and SLAC IEPM-BW deployments used *static* configurations of "sensitivity" and "trigger elevation" threshold parameters for increasing the probability of anomaly event detection while at the same time, decreasing the probability of false alarms. The sensitivity parameter is used to specify the magnitude of plateau change that may result when an anomaly event on a network path is to be triggered. The trigger elevation parameters are used to temporarily increase the thresholds to avoid repeated triggers for a brief period after an anomaly event has been detected. Due to the static configurations in earlier frameworks, the sensitivity and trigger elevation threshold parameters of the *static plateau-detection* (SPD) schemes need to be manually calibrated to *accurately* detect plateaus in different measurement sample profiles on network paths. Such a static threshold selection is similarly an issue for plateau-detection accuracy in filtering techniques used for network-wide correlated anomaly detection. Obviously, such a laborious process for accurate anomaly detection is impractical for large-scale network monitoring with dynamically changing traffic characteristics. Another limitation for perfSONAR users is that there are no tools to notify real-time anomaly events or perform drill-down analysis of anomaly events at multi-resolution timescales by leveraging the web-service enabled access to perfSONAR measurement archives.

In this paper, we present our "OnTimeDetect" tool that we developed to query measurements and analyze network anomalies in measurements of complex dynamic networks instrumented via perfSONAR. The query module extends the standardized web-services schema of perfSONAR measurment archives by allowing raw perfSONAR measurements to be transformed to detect and notify anomalies via graphical user and command-line interfaces. We show that notifying anomalies and identifying the causes of anomalies in complex dynamic networks requires integration of tools such as OnTimeDetect with latest social networking tools such

as Twitter. The analysis module in OnTimeDetect tool features a novel dynamically adaptive plateau detection (APD) scheme for accurate detection of prominent network anomalies. To dynamically configure thresholds for the "sensitivity" and "trigger elevation" parameters in our APD scheme, we apply *reinforcement learning* that guides the learning process based on partially available Markov Decision Processes [11]. Our premise is that the raw measurements just after an anomaly event provide direct intelligence about the anomaly event itself, and leveraging them for reinforcement of the machine learning (to compare statistics of historic and current measurement samples for detecting change points) can make the anomaly detection more robust and accurate. Based on a systematic study of anomaly events in real and synthetic measurement traffic traces, we observe that *variance* of the network performance just after an anomaly event is the critical statistic that can be used for reinforcement to accurately trigger an anomaly. Leveraging this observation, we derive closed-form expressions using statistical curve-fitting principles for dynamically determining thresholds of sensitivity and trigger elevation parameters in our APD scheme. Thus, our APD scheme avoids manual calibration of sensitivity and trigger elevation threshold parameters used in SPD schemes for different profiles of measurement samples on network paths. It achieves low false alarm rates at the cost of a fractional increase in online detection time that is needed for the reinforcement learning.

In addition, we show that our APD scheme can be applied either: (a) directly on raw measurement data of one-way delay and throughput in path-level anomaly detection, or (b) to the transformed measurement data of principal component analysis for network-wide correlated anomaly detection. We empirically evaluate our APD scheme in terms of accuracy, agility and scalability by using measurement traces collected by OnTimeDetect tool from worldwide perfSONAR deployments in diverse networking communities. Further, we analyze the measurement sampling characteristics in the perfSONAR measurement traces to study the challenges associated with achieving low anomaly detection times in multi-domain networks.

The remainder paper organization is as follows: Section 2 describes related work. Section 3 discusses our OnTimeDetect tool implementation. Section 4 describes the trace data used in this paper. Section 5 overviews existing SPD schemes and highlights limitations. Section 6 details our APD scheme for network-path level anomaly detection. Section 7 details our APD scheme for network-wide correlated anomaly detection. Section 8 discusses the APD scheme performance evaluation with perfSONAR datasets. Section 9 concludes the paper.

2. Related Work

It is common practice even today to rely on user-defined monitor thresholds to detect and notify anomalies [6]. The thresholds can be exact values (e.g., notify

anomaly if delay exceeds 50ms on a network path) or relations that use differences between temporally distinct values for the same data source or a secondary data source (e.g., notify anomaly if difference in TCP throughput changes by 25%). Obviously, such user-defined threshold based methods do not consider any network path's inherent behavior, which has unique variability at any given time instant. Hence, they are prone to high false alarm rates. To adapt the anomaly detection to handle the unique variability on a network path, mean $\pm$ standard deviation (MSD) methods have been employed, where the thresholds are determined based on a moving window summary of raw measurements. However, such anomaly detection methods are not robust to outliers that are common in network measurements. The outliers are typically caused by intermittent spikes, intermittent dips, and bursts in network performance, which are typically not of interest as anomaly events.

Works such as [15] propose anomaly detection methods based on pre-analysis of network norms given as templates or signatures, which are calculated by constructing time-series of standard deviations of salient metrics such as utilization and packet counts. Given that network traffic is non-stationary, estimates in such works that rely on pre-determined standard deviations are obviously prone to produce too many false alarms similar to MSD methods.

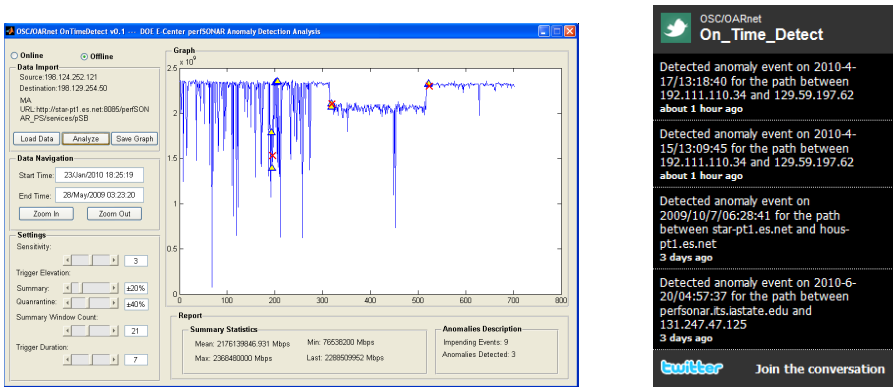
The authors in [8] use principal component analysis principles and the authors in [9] uses wavelet analysis for detecting network performance anomalies. Both these studies involve offline analysis and focus on detecting anomalies in passive measurements at a network link basis. The authors in [10] attempt to overcome the link basis limitation by creating a traffic matrix of all links within an enterprise network domain. They employ a Kalman-filter based anomaly detection scheme that filters out the network norm by comparing future predictions of the traffic matrix state to an inference of the actual traffic matrix obtained from recent measurements. The residual filtered process is examined for detecting a specific kind of anomalies viz., volume anomalies (e.g., flash crowds, outages, denial-of-service attacks) using different methods.

In [16], an adaptive fault detector algorithm is proposed that determines the baseline for network norm in local area networks using a stochastic approximation of the maximum likelihood function of recent performance samples. Abrupt jumps in means of local area network metrics (e.g., ARP broadcast packets, TCP connections to a web-server) are treated as anomalies. This scheme that is based on online change-point detection is not suitable for wide area network monitoring because it assumes that the data sets comprise of k-variate Gaussian distributions that are common in local area network data sets. Authors in [17] also use sequential change point detection principles to detect anomalies when correlated changes occur in various network health metrics collected along a network path.

There are also several related studies that use machine learning techniques for unsupervised anomaly detection [18] [19] [12] [13]. Frameworks such as NLANR

AMP [12] and SLAC IEPM-BW [13] have used variants of the plateau-detector because it is well suited for real-time monitoring of anomalies pertaining to active measurements across end-to-end network paths. Specifically, it uses unsupervised machine learning that relies on raw measurements and dynamic network norm estimation. It also has low complexity, and is relatively easy to implement in large-scale network monitoring frameworks. Moreover, it has been shown to be reasonably effective for monitoring anomalies of active measurements comprising of end-to-end paths across multi-domain paths in the NLANR AMP and SLAC IEPM-BW frameworks for measurements collected over several years.

3. OnTimeDetect Tool Implementation



(a) GUI dialog (b) Twitter integration

Figure 1: Screenshots of the OnTimeDetect tool for anomaly notification

We have developed and made openly available (for both Windows and Linux platforms) an OnTimeDetect tool [20] whose capabilities include: (i) selection of network paths within perfSONAR measurement topologies to be monitored for anomaly events, and (ii) drill-down analysis of anomaly events in vast measurement archives in both real-time and offline manner at multi-resolution timescales. Figure 1 shows example screenshots of the OnTimeDetect GUI dialog in the offline mode, and the real-time notifications of detected anomalies via integration of the OnTimeDetect command-line utility with the Twitter social networking tool. The command-utility also has been integrated to function via a web-interface that has integration with Google charts. In the offline modes with the GUI dialog or web-interface, a measurement trace with timeseries and measurement tuple can be loaded into the tool to obtain an annotated graph with anomalies. Various timeseries portions of the graphs can be zoomed in-and-out to analyze trends at different

time granularities. Salient anomaly detection threshold parameters can also be adjusted to observe anomaly detection results. For any user setting of the parameters, the tool reports summary statistics of the measurement data being analyzed as well as the number of impending and detected anomaly events. The tool also allows a user to save the analyzed graphs. The graphs and anomaly report are intended to help in communicating network anomaly information in traces amongst end-users and network operators. The real-time notifications via Twitter are another important feature that the perfSONAR community direly requires. This is because, anomaly notifications can be sent to network operations personnel using a variety of mobile devices. More importantly, the notification information allows "ground truth" diagnosis for the cause of the anomaly event when placed along common timelines with network events excerpted from network maintenance logs.

Figure 2 shows the workflow between the OnTimeDetect tool and a perfSONAR deployment. In the query module, standardized perfSONAR web-services schema are used to identify the measurement archive (MA) accessible at an address e.g., http://testproject.example-university.edu:8085. The MA stores the measurement timeseries for a network path to be analyzed for anomaly events. The MA web-service calls are routed through a global lookup service (gLS) hosted by the perfSONAR development community. The gLS registers with multiple home lookup services (hLS) that are hosted by individual measurement domains. The hLS inturn maintains information of all the MAs registered to be collecting measurements. Details of the perfSONAR lookup service can be found in [21].

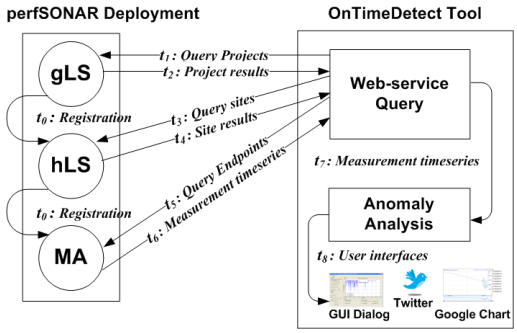


Figure 2: Workflow between OnTimeDetect tool and a perfSONAR deployment

4. Trace Data

Using our OnTimeDetect tool, we have collected real-network measurement traces for studying anomaly events from several worldwide perfSONAR deployments [3]. More specifically, we queried current and historic measurement data

archives from 65 sites that monitor approximately 480 network paths connecting various HPC communities (i.e., universities, labs, HPC centers) over diverse networks that include ESnet, Internet2, GEANT, CENIC, KREONET, LHCOPN and many others. To study network-wide correlated anomaly detection, we analyzed measurement traces collected from the SCinet-2010, the high-performance network setup to support the IEEE/ACM Supercomputing Conference, 2010 in New Orleans. Network measurements were collected between the conference site and 28 other sites across the United States for one-week duration of the conference. The SCinet-2010 traces have both temporal and spatial correlation; temporal correlation exists due to the common monitoring period, and spatial correlation exists because measurements originated from a single conference site to other geographically distributed sites with common intermediate links. Due to the common intermediate links, anomalies on a common link affects network-wide performance.

We queried active measurements of TCP throughput and one-way delay pertaining to the BWCTL and OWAMP tools, respectively. The BWCTL tool is a scheduling and policy daemon that wraps popular data throughput testing tools such as Iperf [22]. The OWAMP tool is an implementation of the protocol specified in IETF RFC 4656. We chose to query and analyze only these 2 metrics in this paper because: (a) they are commonly-used by consumers of perfSONAR meaurements, and (b) they have direct relevance to monitoring end-to-end network path performance affecting large-scale file transfers.

To obtain greater flexibility in studying how plateau-detector parameters are affected by a wide-range of characteristics of anomaly events, we also use synthetic measurement traces. For generating the synthetic traces that mimic the behavior of real active measurement traffic traces, we leverage results from our earlier study on modeling real-network active measurement traces [23]. In this study, we analyzed large data volumes of active and passive measurements collected over campus, regional and national network backbones with time series trends that are: (i) routine, and (ii) event-laden [24]. Routine data set is one where the data time series exhibits normal behavior without any notable events over extended periods of time. Event-laden data set is one where the data time series contains several noticeable events such as persistent increase/decrease, persistent variations, intermittent spikes, intermittent dips, and bursts over timescale resolutions on the order of several months. Our modeling was based on the Box-Jenkins method for time-series analysis. Using diagnostics, 95% confidence interval checking, and prediction, we showed that it is reasonable to characterize active measurements on wide area network paths using Auto-Regressive Integrated Moving Average (ARIMA) model class parameters. We also showed that first-order differencing is sufficient to remove inherent trends and thus filter the data sets. Further, we concluded that active measurement time series have "too much memory", because of which they generally follow MA(q) process (even when they contain events with AR characteristics)

with low q values. Hence, we generated synthetic traces for the round-trip delay metric following the *ARIMA* (0, 1, q) model with q values in the range of 1 to 3. Next, we injected the traces with a wide-range of anomaly events that had different white noise standard deviations in the anomaly time-series regions.

One of the significant challenges in dealing with all the measurement traces is to decide what kind of network events need to be labeled as anomaly events that have to be triggered by a detection scheme under test. For this task, we leveraged the established fact that plateau anomalies are of most interest since they have in most cases indicated reasons for changes in data transfer speeds on high-speed network paths [13]. In addition, network paths with high performance variability behave similar to noticeable plateau anomalies in terms of their statistical nature. Intermittent spikes, intermittent dips, and bursts generally are caused due to user-behavior during normal network operation, i.e., users generating various application traffic. Thus, intermittent spikes, intermittent dips, and bursts are not of interest as anomaly events and should not be notified. The anomalies we categorized as worthy of notification are based on our own experiences as network operators [24**?**], and are based on our extensive discussions with other network operators supporting HPC communities (e.g., ESnet, Internet2, GEANT). We remark that the causes for the performance patterns seen in the real-network measurement traces can be due to several reasons. It could be due to large-application flows, misconfigurations of network elements, cross-traffic congestion, or even due to changes in the test platforms (e.g., kernel driver, OS image, TCP flavor, auto-tuned buffers versus fixed buffers) of the active measurement probes.

5. Plateau Anomaly Detection

5.1. Plateau-Detector Overview

The description of the basic plateau-detector algorithm implemented in [12] and [13] is as follows. The algorithm is an enhanced version of the MSD method. In the MSD method, the first step is to determine the network health norm by calculating the mean μ for a set of measurements sampled recently into a "summary buffer". The sampled measurements correspond to network health metrics such as throughput, and one-way delay on a network path. The number of samples in the summary buffer is user-defined and is specified using a "summary window count" swc. In the next step, an anomaly is triggered if the value of the most recent measurement sample x_t crosses either of the $\mu \pm \sigma$ thresholds of the summary buffer measurements; note that σ corresponds to standard deviation of the measurements in the summary buffer. In comparison to the MSD method, the plateau-detector requires two additional user-defined inputs called "trigger duration" td and "sensitivity" s. The trigger duration td specifies the duration of the anomaly event before

a trigger is signaled. The sensitivity s specifies the magnitude of the plateau change that may result when an anomaly event on a network path is to be triggered.

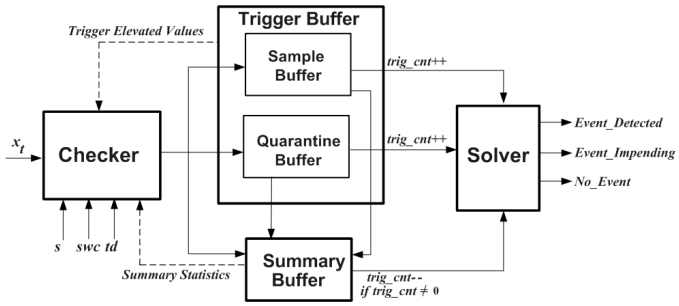


Figure 3: Plateau-detector block diagram

Figure 3 shows the different plateau-detector components. The values of x_t are first input to a "checker" which compares if the most recent x_t value lies within the upper and lower threshold set $ts(.) = \{T_{SU}, T_{QU}, T_{SL}, T_{QL}\}$ of: (i) "summary buffer" *sumbuff* i.e., T_{SU} and T_{SL} or (ii) "quarantine buffer" *qbuff* i.e., T_{QU} and T_{QL}. These thresholds are illustrated in Figure 5 and are calculated using mean μ and standard deviation σ of the summary window as shown in Equations (1) - (4).

$$T_{SU} = \mu + s * \sigma \quad (1)$$
$$T_{QU} = \mu + 2 * s * \sigma \quad (2)$$
$$T_{SL} = \mu - s * \sigma \quad (3)$$
$$T_{QL} = \mu - 2 * s * \sigma \quad (4)$$

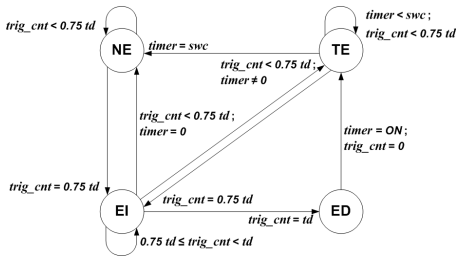


Figure 4: Plateau detector states and state-transitions

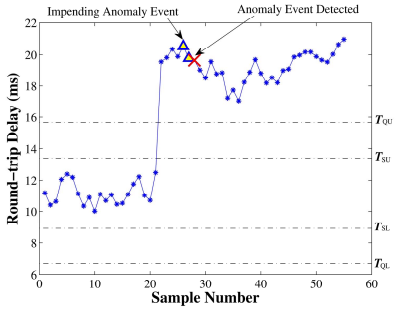


Figure 5: Plateau-detector thresholds illustration

If x_t values lie within these thresholds, the plateau-detector will be in the no event (NE) state shown in Figure 4. In this state, x_t values are put into the *sumbuff*.

If x_t values go below T_{QL} or exceed T_{QU}, they are put into the quarantine buffer *qbuff*. Similarly, if x_t values cross T_{SL} and T_{SU}, they are put into the "sample buffer" *sampbuff*. If x_t is put into either *qbuff* or *sampbuff*, a trigger count $trig\_cnt$ counter is incremented. Whereas, if x_t is put into *sumbuff*, $trig\_cnt$ is decremented as long as $trig\_cnt$ is non-zero. If $trig\_cnt$ exceeds 0.75*td due to increasing number of x_t values going into *qbuff* or *sampbuff*, then the plateau-detector enters into an event impending (EI) state. If the $trig\_cnt$ drops below 0.75*td, then the plateau-detector returns to NE state. Otherwise, the plateau-detector stays in the EI state until $trig\_cnt$ equals td, after which it enters into an event detected (ED) state. Figure 5 shows an anomaly event being triggered after x_t crosses the $ts(.)$ thresholds. The EI state x_t values are marked as triangles, and the triggered event in the ED state is marked by the cross mark. At this point, the $trig\_cnt$ is reset, and a *timer* is turned ON. The plateau-detector now goes into a trigger elevated (TE) state, where the values in the upper and lower threshold set $ts'(.)$ = $\{T'_{SU}, T'_{QU}, T'_{SL}, T'_{QL}\}$ are calculated as shown in Equations (5) - (8).

$$T'_{SU} = 1.2 * max(x_t) \text{ in } \textit{trigbuff} \quad (5)$$

$$T'_{QU} = 1.4 * max(x_t) \text{ in } \textit{trigbuff} \quad (6)$$

$$T'_{SL} = 0.8 * min(x_t) \text{ in } \textit{trigbuff} \quad (7)$$

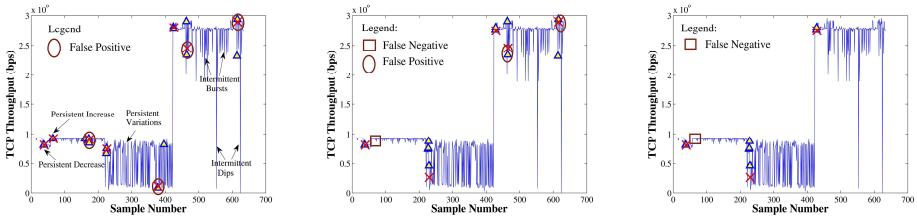
$$T'_{QL} = 0.6 * min(x_t) \text{ in } \textit{trigbuff} \quad (8)$$

Until the *timer* equals swc, the elevated thresholds are used for comparing x_t. The reason for the trigger elevation is to avoid reporting of repeated triggers for the already detected anomaly. Once *timer* equals swc, and x_t does not cross the elevated thresholds, the plateau-detector returns to the NE state. Referring back to the Figure 3, we can see that the "solver" tracks the plateau-detector state transitions and outputs the *No_Event*, *Event_Impending*, and *Event_Detected* signals.

5.2. Plateau Detector Parameters

We now discuss selection of values used for the following plateau-detector parameters: summary window count swc, trigger duration td, sensitivity s, NE state threshold set $ts(.)$, and TE state threshold set $ts'(.)$. Based on this discussion, we motivate the need for dynamically adaptive determination of s, $ts(.)$, and $ts'(.)$.

The value of swc is chosen depending upon the number of recent history samples that are sufficient to obtain a rough yet reliable estimate of the network norm. Choosing a small value for swc has the risk of allowing network noise such as intermittent spikes, intermittent dips, or bursts that distort the network norm estimation. Alternately, choosing a large value for swc has the risk of smoothening out trends of impending anomalies that inturn increases detection time or leads to false negatives. Both the earlier plateau-detector implementations [12] and [13] have used a setting of swc = 20.



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

Figure 6: Impact of choosing static s values on anomaly detection accuracy

The value of td is chosen to be relatively smaller than the swc. The smaller the value of td compared to swc, the faster a trigger will be signaled in the event of an anomaly. However, the td must be chosen to be large-enough such that intermittent spikes, intermittent dips, or bursts i.e., noise events in network health do not influence the trigger signaling and cause false alarms. Given the fact that samples that don't cross the $ts(.)$ thresholds reduce the $trig\_cnt$, values on the order of td 5 to 10 are suitable. In the earlier plateau-detector implementations, the td is assumed to be approximately 1/3rd swc (i.e., td = 7). Based on our systematic study of anomaly events in real and synthetic measurement traffic traces (we provide detailed descriptions of the trace sources in Section 6), we have found that the assumptions of swc = 20 and td = 7 are reasonable, and minor modifications to these settings have negligible influence in triggered false alarms.

However, we found that the s, $ts(.)$, and $ts'(.)$ parameters are relatively more salient, and minor modifications in their values selection significantly influences the anomaly detection accuracy. Chosing the s value needs consideration of tradeoffs i.e., a small s value results in triggers for slight variations in network performance magnitudes, whereas a large s value could overlook actual anomalies that should be detected. Both the NLANR AMP [12] and SLAC IEPM-BW [13] frameworks choose s = 2, and analysis with a large number of real measurement traces in [14] has shown that s settings in the range of 2 to 3 is effective. In essence, setting of the threshold values to $\mu \pm 2{*}\sigma$ is based on an inherent assumption that network norm data follows normal distribution. It is known that if any data follows normal distribution, approximately 95% of the data points fall within this $\mu \pm 2{*}\sigma$ range. Since this assumption is not completely true, static setting of s = 2, which also directly impacts $ts(.)$ settings (see Equations (1) - (4)) causes false alarms.

Figures 6(a) - (c) show the impact of choosing static s values of 2, 3 or 4 for detecting anomaly events on a conjoined set of real-network traces. We conjoined several perfSONAR traces to illustrate how the anomaly detection accuracy is affected by the static s values for different kinds of network events such as persistent

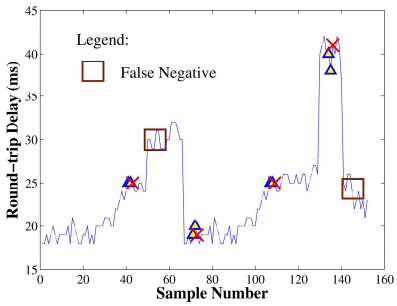


Figure 7: Impact of choosing static $ts'(.)$ settings on anomaly detection accuracy

increase/decrease, persistent variations, intermittent spikes, intermittent dips, and bursts. As we can see, increasing the sensitivity from 2 to 3 reduces the number of false positives, and causes a false negative. Increasing the sensitivity to 4 can avoid the triggering of all false positives but the false negative remains. These observations motivate us to develop an adaptive scheme (described in detail in Section 6) for dynamically configuring s values, which avoids triggering such false alarms that occur when using the SPD scheme.

Similarly, we found that choosing static $ts'(.)$ settings as shown in Equations (5) - (8) can result in false alarms. Recall from Section 5.1 that the $ts'(.)$ settings use elevated thresholds based on the $max(x_t)$ and $min(x_t)$ values in the *trigbuff* until the $timer$ equals swc. Figure 7 shows how the static $ts'(.)$ settings do not detect consecutive anomalies of the same magnitude of either the persistent increase/decrease kinds when the $timer$ is lesser than swc. These observations motivate us to develop an adaptive scheme (described in detail in Section 6) for dynamically configuring $ts'(.)$ values, which avoids triggering false negatives in consecutive anomaly event scenarios when using the SPD scheme.

6. Adaptive Plateau-Detector for Network-path level Anomalies

In this section, we present our novel, dynamically adaptive plateau-detection (APD) scheme based on reinforcement learning to overcome the SPD scheme limitations described in Section 5.2 for network-path level anomalies.

6.1. Dynamic Sensitivity Selection

In our analysis of the traces described in Section 4, we found that s in the range of 2 to 4 consistently detected the labeled anomalies. We noted that although the SPD scheme frequently detected the labeled anomaly events, the false alarms i.e., false positives and false negatives were caused primarily due to the s values

not changing for various anomaly events. If the sensitivity at the time point of a false alarm was modified, the anomaly event was successfully detected. Hence, we concluded that s needs to be re-evaluated at each time step to avoid false alarms. Further, we observed that false alarms were triggered in the SPD scheme due to the nature of persistent variations in the time series after an anomaly event is detected. Hence, we concluded that variance σ_f^2 of the raw measurements just after an anomaly event provides direct intelligence about the anomaly event itself, and leveraging it for reinforcement of the machine learning (to compare statistics of historic and current measurement samples for detecting change points) can make the anomaly detection more robust and accurate. We remark that this idea of using σ_f^2 as the critical statistic for reinforcement to detect anomaly events robustly and accurately is formally referred to in the machine learning literature as "reinforcement learning". Also, finite-state Markov decision processes are commonly used to formulate the reinforcement learning, however such a formulation for our APD scheme is beyond the scope of this paper.

Based on these observations, we use the $\frac{\sigma_f^2}{\sigma_c^2}$ relation in our APD scheme to determine the sensitivity s_{t-swc} dynamically at a time step t. Here, σ_f^2 refers to the variance of the swc number of measurement samples in the future, and σ_c^2 refers to the variance of the swc number of most current measurement samples. We set s_{t-swc} to a higher value (closer to 4) based on how much the $\frac{\sigma_f^2}{\sigma_c^2}$ ratio is higher than 1. Similarly, we set s_{t-swc} to a lower value (closer to 2) based on how much the $\frac{\sigma_f^2}{\sigma_c^2}$ ratio is lower than 1. Equations 9 - 12 show the expressions to calculate the σ_f and σ_c values.

$$\mu_c = \frac{1}{swc}\sum_{i=1}^{swc} swd[i] \tag{9}$$

$$\sigma_c = \sqrt{\frac{1}{swc-1}\sum_{i=1}^{swc}(swd[i]-\mu_c)^2} \tag{10}$$

$$\mu_f = \frac{1}{swc}\sum_{i=1}^{swc} rbd[i] \tag{11}$$

$$\sigma_f = \sqrt{\frac{1}{swc-1}\sum_{i=1}^{swc}(rbd[i]-\mu_f)^2} \tag{12}$$

Note that swd refers to summary window data and rbd refers to reinforcement buffer data. Figure 8 shows the "Learner" and "Reinforcement Buffer" blocks needed in the plateau-detector shown in Figure 3 for dynamically determining the

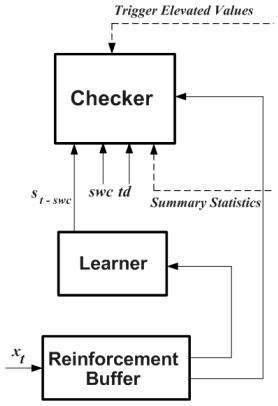


Figure 8: Additional blocks in plateau-detector for APD scheme

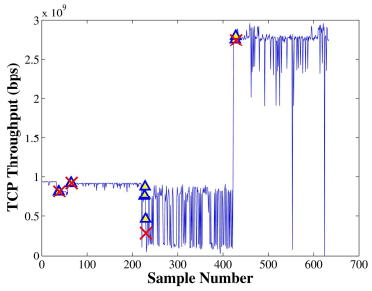


Figure 9: Illustration to show choosing dynamic s values avoids false alarms

sensitivity parameter. The Learner implements our APD scheme logic to output instantaneous values of sensitivity s_{t-swc} to the Checker. The Reinforcement Buffer stores the measurement samples x_t as they arrive to the plateau-detector, which are subsequently used by the Learner for the reinforcement learning.

Using this setup of the plateau-detector, we record the $\frac{\sigma_f^2}{\sigma_c^2}$ values between sensitivity 2 to 4 that detect the different kinds of anomaly events accurately in the traces. Next, using statistical curve-fitting principles, we best fit the recorded values to derive the closed form linear expression as shown in Equation 13 for calculating the dynamic sensitivity s_{t-swc}.

$$s_{(t-swc)} = 0.4 * \frac{\sigma_f^2}{\sigma_c^2} + 2 \tag{13}$$

Figure 9 shows how the APD scheme accurately detects the anomaly events and avoids false alarms for the example trace described in Section 5.2. Correspondingly, Figure 10 shows how the sensitivity configuration changes dynamically between the range of 2 to 4 during the robust and accurate anomaly detection of the APD scheme.

6.2. Dynamic Trigger Elevation Selection

In our analysis of the synthetic traces, we found that choosing static $ts'(.)$ settings does not detect consecutive anomalies of the same magnitude of either the persistent increase/decrease kinds when the $timer$ is lesser than swc. We noted that using the $ts'(.)$ settings based on the $max(x_t)$ and $min(x_t)$ values in the SPD scheme was the cause for the false negatives. In order to understand how we

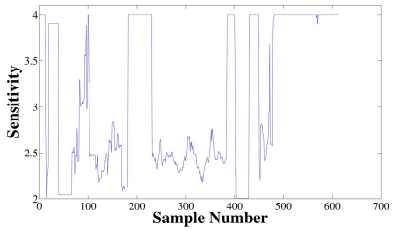


Figure 10: Instantaneous s value selections in APD scheme

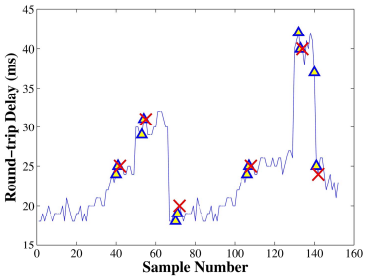


Figure 11: Illustration to show choosing dynamic s values avoids false alarms

overcame this limitation, let us consider the measurement sample x_d that arrived at the time when the anomaly was detected i.e., x_d refers to the x_t point on which we annotate the cross mark on our graphs. By using x_d as the network norm in the trigger elevated (TE) state for calculating the thresholds, we were able to avoid the false negatives in consecutive anomaly event scenarios. Hence, we use Equations (14) - (17) to calculate the $ts'(.)$ upper and lower threshold set values.

$$T'_{SU} = x_d + s_t * \sigma_c \quad (14)$$

$$T'_{QU} = x_d + 2 * s_t * \sigma_c \quad (15)$$

$$T'_{SL} = x_d - s_t * \sigma_c \quad (16)$$

$$T'_{QL} = x_d - 2 * s_t * \sigma_c \quad (17)$$

Figure 11 shows how the false negatives that were marked in the Figure 7 for the SPD scheme are detected using the dynamic $ts'(.)$ settings in the APD scheme.

7. Adaptive Plateau-Detector for Network-wide Correlated Anomalies

In this section, we study the application of our APD scheme for network-wide correlated anomaly detection. More specifically, we analyze the effectiveness of our APD scheme to the transformed measurement data of principal component analysis (PCA). We choose PCA because it is best suited to combine and extract the common features from multiple measurement timeseries. In addition, it is relatively easy for users to configure PCA as a "black-box" without knowing its internals compared to other related schemes [10] [9]. This study is relevant because existing schemes based on PCA such as [8] also rely on static threshold schemes, and their anomaly detection accuracy can be improved with the APD scheme.

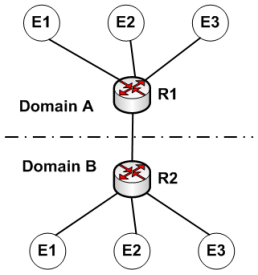


Figure 12: Measurement topology to illustrate network-wide correlated anomaly detection

Figure 13: PCA-with-APD block diagram

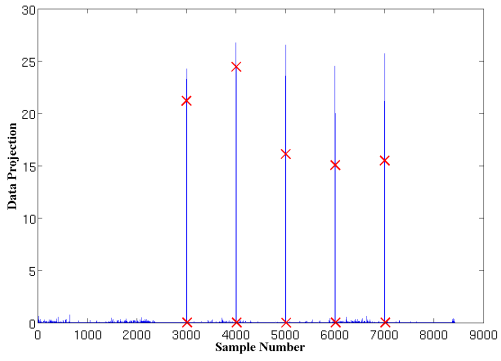


Figure 14: Data projection of first eigen vector showing just the correlated anomalies

To understand the utility of PCA in the context of network-wide correlated anomaly detection, let us consider Figure 12 that has network-paths with spatial correlation with common intermediate links. Assuming measurements $x_t^1, x_t^2, \ldots$ are collected between end points {E1,E2,E3} in say, Domain A and {E4,E5,E6} in say, Domain B during the same time interval, then all of the measurements would reflect anomaly events on link R1↔R2 that are captured by techniques such as PCA. The output of PCA as shown in Figure 13 is the fused reoriented data comprising of eigen vectors, where the first eigen vector captures maximum variability and the last is left with minimum variability. What this translates into in reality is that - the data projection using the first eigen vector has variability that is common to most of datasets and the last eigen vectors have the variability that is least

common in the dataset (e.g., variability present in only one dataset amongst all). Next, data dimension selection is performed on the fused reoriented data. For example, if we are interested only in the common anomalies, we select only the first eigen vector. After the data dimension (number of eigen vectors) is selected, the data is projected using the eigen vectors and the APD scheme is used to detect the anomalies.

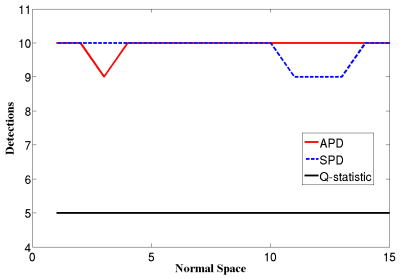


Figure 15: Detections of correlated anomalies by Q-statistic, SPD and APD on PCA output

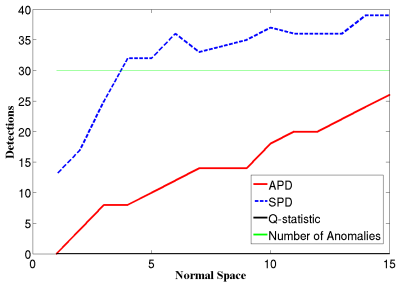


Figure 16: Detections of uncorrelated anomalies by Q-statistic, SPD and APD on PCA output

We now illustrate how the APD scheme can accurately detect anomalies in the output of PCA with low number of false alarms. For this, we use the SCinet-2010 measurement traces described in Section 4. Five plateaus (i.e. 10 anomalies due to rising and falling edges) were injected in all traces at the same time instances to create correlated anomalies. Similarly, a plateau anomaly was injected in each trace at different time instances to create 30 uncorrelated anomalies. Figure 14 shows the data projection using the first eigen vector and anomalies detected by the APD scheme on the data with 5 conspicious spikes i.e., correlated anomalies. The high magnitude of the spikes is due to the correlated nature of the anomalies in the dataset. While this could be an advantage in terms of detecting correlated anomalies, for detectors using static thresholds, the detection of uncorrelated anomalies of smaller magnitudes would result in a large number of false alarms.

To demonstrate this fact, we plot the anomaly detection performance of the APD, SPD and Q-statistic schemes with increasing number of PCA eigen vectors used for projecting the data. To convert the Q-statistic scheme from [8] into a plateau detector, we look for 7 (same as the trigger count in APD and SPD schemes) consecutive threshold detections to classify it as a plateau event. Figures 15 and 16 show the detection performance for correlated and uncorrelated anomalies for the three schemes. Figures 17 and 18 show the number of detections, false positive and false negatives of SPD and APD schemes for the data projections using all the eigen vectors (i.e., all anomalies in the data). The Q-statistic

scheme finds the rising edge of the correlated plateaus, but not the falling edges since it looks only for values above a threshold. It also completely misses all the uncorrelated anomalies due to the conspicious spikes of correlated anomalies that cause threshold to be high. In contrast, the SPD scheme detects most of the anomalies, but produces 7 false positives and 3 false negatives. This is because it does not account for the change in the variance in the data. However, the APD scheme adapts to both the changes in mean and variance and has the best false alarm rates amongst the three. As shown in Figure 18, the APD scheme detects all 10 correlated anomalies and detects 27 out of 30 (3 false negatives) uncorrelated anomalies with 2 false positives. Thus, we can conclude that the APD scheme accurately detects anomalies in the transformed measurement data of principal component analysis (PCA).

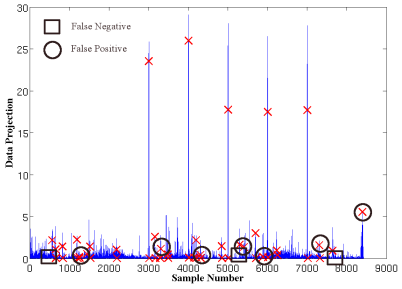


Figure 17: Detection accuracy of correlated and uncorrelated anomalies by SPD

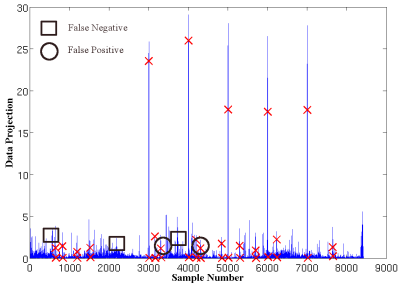


Figure 18: Detection accuracy of correlated and uncorrelated anomalies by APD

8. Performance Evaluation

In this section, we evaluate our APD scheme in terms of accuracy, agility and scalability.

8.1. Accuracy Evaluation

We have extensively investigated the accuracy of our APD scheme on measurement traces collected by our OnTimeDetect tool from 65 sites that monitor approximately 480 network paths connecting various HPC communities. To illustrate the findings of our investigation, we choose to show the accuracy performance for a representative set of 8 BWCTL traces shown in Table 1 with unique time series characteristics. We use three metrics shown in Equations 18, 19 and 20 respectively, to evaluate the anomaly detection accuracy of the APD scheme in

Table 1: Traces description

Trace ID	Source ↔ Destination	Time Range (Start - End)	Time Series Characteristics
1	psmsu02.aglt2.org ↔ psum02.aglt2.org	2009-10-9 15:03:19 - 2010-4-7 17:28:05	Persistent Decrease, Burst Decrease, Intermittent Dips
2	bwctl.ucsc.edu ↔ bwctl.atla.net.internet2.edu	2010-1-16 06:51:22 - 2010-4-7 20:36:05	Persistent Decrease, Persistent Increase, Intermittent Dips
3	bwctl.ucsc.edu ↔ bwctl.wash.net.internet2.edu	2010-1-16 08:50:36 - 2010-4-7 20:37:43	Persistent Decrease, Persistent Increase, Intermittent Bursts, Intermittent Dips
4	wtg248.otctest.psu.edu ↔ perf-sonar.dragon.maxgigapop.net	2010-2-8 14:08:31 - 2010-4-7 21:25:57	Persistent Variations
5	chic-pt1.es.net ↔ anl-pt1.es.net	2009-7-2 20:04:41 - 2010-1-9 12:32:48	Persistent Increase, Persistent Decrease, Persistent Variations
6	nersc-pt1.es.net ↔ wash-pt1.es.net	2009-5-18 22:48:13 - 2010-1-9 16:46:47	Persistent Increase, Intermittent Bursts, Intermittent Dips
7	hous-pt1.es.net ↔ pnwg-pt1.es.net	2009-5-19 04:05:12 - 2010-4-7 13:39:31	Persistent Increase, Persistent Variations, Intermittent Dips
8	nettest.boulder.noaa.gov ↔ wtg248.otctest.psu.edu	2009-10-6 20:41:22 - 2010-4-7 21:27:05	Persistent Decrease, Persistent Increase, Intermittent Bursts, Intermittent Dips

Table 2: Accuracy evaluation results

Trace ID	$SPD_{s=2}$			$SPD_{s=3}$			$SPD_{s=4}$			$APD_{s=2...4}$		
No.	R_s	R_{f+}	R_{f-}	R_s	R_{f+}	R_{f-}	R_s	R_{f+}	R_{f-}	R_s	R_{f+}	R_{f-}
1	1	0	**2**	1	0	1	1	0	0	1	0	0
2	1	0	**1.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	0	1	0	0
3	1	0	0	**0.67**	**0.33**	**0.33**	1	0	0	1	0	0
4	**0.5**	**0.5**	**5**	1	0	0	1	0	0	1	0	0
5	1	0	**0.5**	1	0	0	**0.5**	**0.5**	0	1	0	0
6	**0**	0	**3**	1	0	**2**	1	0	0	1	0	0
7	1	0	**0.5**	**0.5**	**0.5**	0	**0.5**	**0.5**	0	1	0	0
8	1	0	**0.5**	1	0	0	1	0	0	1	0	0

comparison with the SPD scheme: *success ratio* (R_s), *false positive ratio* (R_{f+}), and *false negative ratio*(R_{f-}).

$$R_s = \frac{\textit{number of true triggers detected}}{\textit{number of true triggers}} \tag{18}$$

$$R_f+ = \frac{\textit{number of false triggers detected}}{\textit{number of true triggers}} \tag{19}$$

$$R_f- = \frac{\textit{number of true triggers missed}}{\textit{number of true triggers}} \tag{20}$$

Note that higher values of R_s and lower values of R_{f+} and R_{f-} denote superior performance. In addition, the value of the expression R_s+R_{f-} is always equal to 1. Table 2 shows the accuracy evaluation results for the APD scheme in comparison with the SPD scheme with static s settings of 2, 3 and 4. We can observe that the APD scheme outperforms atleast one of the static SPD schemes in

detecting anomaly events and avoiding false alarms in all of the 8 traces. However, in some cases the APD scheme performs similar to a particular SPD scheme. The cases where the APD scheme outperforms the SPD scheme variants are shown in bold font. We can note that the SPD scheme with s=2 causes most false negatives indicating that it is least robust than the other schemes.

8.2. Agility Evaluation

Agility evaluations presented herein are for the purpose of showing the anomaly detection time results from our real-network measurement trace analysis. Our definition of detection time of an anomaly refers to the time difference between the instant the plateau-detector enters the impending (EI) state and the instant it is in the event detected (ED) state. As expected, low detection times indicates a superior anomaly detection scheme. Figure 19 shows the detection times for the representative set of 8 BWCTL traces shown in Table 1. We can observe that the anomaly detection times using our APD scheme varies across perfSONAR deployments, and is on the order of one or more days for BWCTL measurements. The reason for these variations and long detection times is mainly due to the currently chosen average periodic (a.k.a. stratified random) sampling patterns and frequencies in the perfSONAR deployments. The average periodic pattern and frequency indicates that there is not a strict sampling at periodic time points (e.g., every hour on the hour sampling), and for a given time period (e.g., a day), there are a fixed number of measurement samples (we observed anywhere from 3 to 12 samples per day) at a deployment site.

To improve the detection times of our APD scheme in the BWCTL traces, we evaluate using adaptive sampling instead of the average periodic sampling. In the case of adaptive sampling, once the plateau-detector enters the EI state, the sampling frequency at a site *sf* is increased to collect measurement samples at smaller inter-sampling times. Figure 20 shows the detection times when using the adaptive sampling with the sampling frequencies 2**sf*, 4**sf*, and 8**sf*. To increase the number of samples available at higher frequencies at a deployment site, we use the values of the consecutive samples in the future. We do so with the assumption that they follow the same process of the anomaly event time series region just as the additional samples would have if they were adaptively sampled in reality. We can see that by using adaptive sampling measurement data, the APD scheme detection times reduce to the range of only a few hours versus the earlier observed ranges of several days. In addition, we can see that 4**sf* and 8**sf* cases would result in oversampling (i.e., measurement traffic consumes the network path bandwidth that could have been used by actual application traffic), and do not provide any further improvement to just using 2**sf*. Nevertheless, the reduced detection times lessen the online detection time tradeoff that is needed for reinforcement learning in our APD scheme.

Given the fact that BWCTL measurements are resource intensive in terms of CPU, memory and bandwidth consumption, the sampling frequencies are quite low in comparison to low-overhead OWAMP measurements. In the OWAMP measurement traces, we observed 50% of the cases had 2-3 samples per minute, and another 50% of the cases had 0.5-1 samples per minute. Consequently, we found detection times to be on the order of minutes, mostly $<$10 minutes. Hence, adaptive sampling is not a major necessity for agile detection of OWAMP anomaly events for routine network monitoring purposes.

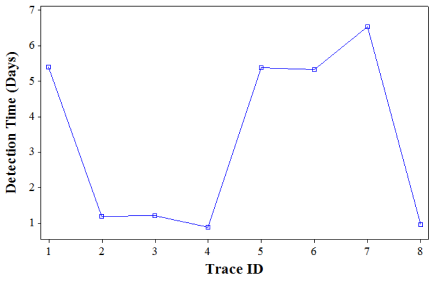


Figure 19: Effect of average periodic sampling on anomaly detection times

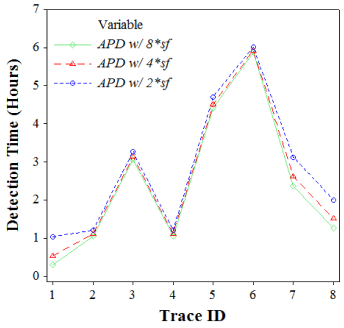


Figure 20: Effect of adaptive sampling on anomaly detection times

8.3. Scalability Evaluation

Lastly, we evalute the scalability of the APD scheme if used for notifying anomalies in large-scale measurement topologies comprising of several hundred of network paths. For this, we calculate the average analysis time per-site to sequentially detect anomalies on over 480 perfSONAR monitored paths whose measurement data we queried from 65 sites using our OnTimeDetect tool. We define analysis time for a measurement archive at a site as the sum of the times taken for perfSONAR web-service processing in the gLS and hLS, MA measurement timeseries transfer, and run time of the APD scheme on the measurement timeseries (i.e., t_7 - t_1 in Figure 2). Figure 21 shows the average analysis time per-site for sequential queries spanning multi-timescale resolutions shown on the x-axis. We remark that the average analysis times shown have been calculated from 75 query iterations spanning several days. We can observe that the analysis times are on the order of tens of seconds. Upon further analysis, we found that some sites have analysis times that are less than a second, whereas some sites have analysis times on the order of several seconds. Based on this observation, we experimented with a parallel query mechanism where the OnTimeDetect concurrently queries all the sites in the site-list. The speed up results in the average analysis time per-site for

the parallel query compared to the sequential query from 75 iterations are shown in Figure 22. The speedup value is calculated as a percentage ratio of the sequential query analysis time over the parallel query analysis time. We can see that the parallel-query mechanism in the OnTimeDetect tool can speedup the average analysis time per-site by (40 - 60)%, and inturn improve the OnTimeDetect GUI dialog user experience during drill-down analysis of measurement timeseries.

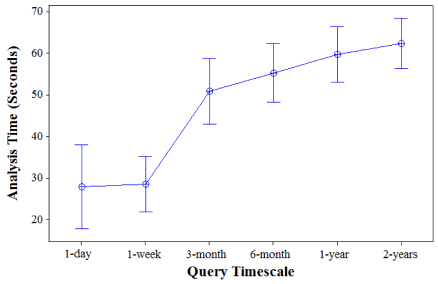


Figure 21: Per-site analysis times for sequential queries spanning multi-timescale resolutions

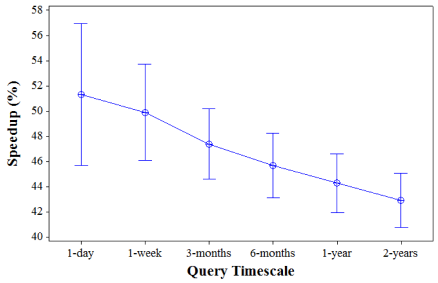


Figure 22: Speedup in per-site analysis times for parallel queries spanning multi-timescale resolutions

9. Conclusion

In this paper, we presented a dynamically adaptive plateau-detection (APD) scheme and its implementation in our "OnTimeDetect" tool to notify network anomaly events that are of interest to network operators and end-users. We leveraged the vast measurement data archives available via perfSONAR web-services at several worldwide sites, and also synthetic measurement traces with anomaly events to develop our APD scheme. We showed that notifying anomalies and identifying the causes of anomalies in complex dynamic networks requires integration of tools such as OnTimeDetect with social networking tools such as Twitter. In addition, we showed that our APD scheme can be applied either: (a) directly on raw measurement data of one-way delay and throughput in path-level anomaly detection, or (b) to the transformed measurement data of principal component analysis for network-wide correlated anomaly detection. Through our performance evaluations, we showed that the APD scheme is robust and accurate in anomaly notifications. In addition, we showed that it is possible to minimize anomaly detection times in resource intensive BWCTL measurements throughput by using adaptive sampling instead of the currently employed average periodic sampling in perfSONAR deployments. Further, we showed that it is possible to substantially speedup anomaly analysis times when dealing with large number of paths using a parallel-query mechanism instead of a sequential-query mechanism.

References

[1] Teragrid Project - http://www.teragrid.org

[2] The Large Hadron Collider (LHC) Project - http://lhc.web.cern.ch/lhc

[3] A. Hanemann, J. Boote, et. al., "PerfSONAR: A Service Oriented Architecture for Multi-Domain Network Monitoring", *Proc. of Service Oriented Computing, Springer Verlag, LNCS 3826*, pp. 241-254, 2005. (http://www.perfsonar.net)

[4] J. Zurawski, M. Swany, D. Gunter, "Scalable Framework for Representation and Exchange of Network Measurements", *Proc. of IEEE TRIDENTCOM*, 2006.

[5] C. Guok, D. Robertson, et. al., "Intra and Interdomain Circuit Provisioning Using the OSCARS Reservation System", *Proc. of IEEE/ICST Conference on Broadband Communications, Networks, and Systems*, 2006.

[6] J. Allen, "Driving by the Rear-View Mirror: Managing a Network with Cricket", *Proc. of USENIX Network Administration Conference*, 1999.

[7] W. Matthews, L. Cottrell, "The PingER Project: Active Internet Performance Monitoring for the HENP Community", *IEEE Communications Magazine*, 2000.

[8] A. Lakhina, M. Crovella, C. Diot, "Diagnosing Network-Wide Traffic Anomalies", *Proc. of ACM SIGCOMM*, 2004.

[9] P. Barford, J. Line, D. Plonka, A. Ron, "A Signal Analysis of Network Traffic Anomalies", *Prof. of ACM SIGCOMM Internet Measurement Workshop*, 2002.

[10] A. Soule, K. Salamtian, N. Taft, "Combining Filtering and Statistical Methods for Anomaly Detection", *Proc. of Internet Measurement Conference*, 2005.

[11] M. Thottan, G. Liu, C. Ji, "Anomaly Detection Approaches for Communication Networks", *Book Chapter in Algorithms for Next Generation Networks*, Springer, 2010.

[12] A. McGregor, H-W. Braoun, "Automated Event Detection for Active Measurement Systems", *Proc. of Passive and Active Measurement Workshop*, 2001.

[13] C. Logg, L. Cottrell, "Experiences in Traceroute and Available Bandwidth Change Analysis", *Proc. of ACM SIGCOMM Network Troubleshooting Workshop*, 2004.

[14] L. Cottrell, M. Chhaparia, F. Haro, F. Nazir, M. Sandford, "Evaluation of Techniques to Detect Significant Network Performance Problems using End-to-end Active Network Measurements", *Proc. of IEEE/IFIP NOMS*, 2006.

[15] F. Feather, R. Maxion, "Fault Detection in an Ethernet Network using Anomaly Signature Matching", *Proc. of ACM SIGCOMM*, 1993.

[16] H. Hajji, "Statistical Analysis of Network Traffic for Adaptive Faults Detection", *IEEE Transactions on Neural Networks*, 2005.

[17] M. Thottan, C. Ji, "Anomaly Detection in IP Networks", *IEEE Transactions on Signal Processing*, 2003.

[18] D. Gunter, B. Tierney, A. Brown, M. Swany, J. Bresnahan, J. Schopf, "Log Summarization and Anomaly Detection for Troubleshooting Distributed Systems", *Proc. of IEEE/ACM Grid Computing*, pp. 226-234, 2007.

[19] L. Yang, C. Liu, J. Schopf, I. Foster, "Anomaly Detection and Diagnosis in Grid Environments", *Proc. of ACM/IEEE Supercomputing*, 2007.

[20] OnTimeDetect: Anomaly Notification Tool for perfSONAR Deployments - http://ontimedetect.oar.net

[21] J. Zurawski, J. Boote, et. al., "Hierarchically Federated Registration and Lookup within the perfSONAR Framework", *Proc. of IFIP/IEEE IM*, 2007.

[22] A. Tirumala, L. Cottrell, T. Dunigan, "Measuring End-To-End Bandwidth with Iperf Using Web100", *Proc. of Passive and Active Measurement Workshop*, 2003.

[23] P. Calyam, A. Devulapalli, "Modeling of Multi-resolution Active Network Measurement Time-series", *Proc. of IEEE Workshop on Network Measurements*, 2008.

[24] P. Calyam, D. Krymskiy,M. Sridharan, P. Schopis, "Active and Passive Measurements on Campus, Regional and National Network Backbone Paths", *Proc. of IEEE ICCCN*, 2005.

***Author Biography**

Dr. Prasad Calyam received the BS degree in Electrical and Electronics Engineering from Bangalore University, India, and the MS and PhD degrees in Electrical and Computer Engineering from The Ohio State University, in 1999, 2002, and 2007 respectively. He is presently a Principal Investigator and Senior Systems Developer/Engineer at the Ohio Supercomputer Center/OARnet, The Ohio State University. His current research interests include multimedia networking, cyber security, cyberinfrastructure systems, and network management.

Mukundan Sridharan received the BS degree in Electronics and Communication Engineering from Madras University, India, and the MS and PhD degrees in Computer Science and Engineering from The Ohio State University, in 2000, 2002, and 2011 respectively. He is currently a Postdoctral Fellow at OARnet, The Ohio State University. His current research interests include wireless sensor networks, multimedia networking and distributed systems.

Jialu Pu is currently pursuing his BS degree in Mathematics at The Ohio State University, and is a research assistant student at the Ohio Supercomputer Center, The Ohio State University. His current research interests include statistical data analysis, network performance measurements, and software engineering.

Dr. Ashok K. Krishnamurthy earned his BS degree in Electrical Engineering from the Indian Institute of Technology in Madras, India in 1979. He received his MS and PhD degrees in Electrical Engineering at the University of Florida in 1981 and 1983, respectively. He is currently the Co-Executive Director of the Ohio Supercomputer Center, and also is an associate professor in The Ohio State University Electrical and Computer Engineering Department. His research interests include signal/image processing, high performance computing, parallel high level language applications and computational models of hearing.

***Author Photo**

Click here to download high resolution image