SLAC-PUB-11653

Evaluation of Techniques to Detect Significant Network Performance Problems using End-to-End Active Network Measurements

by R.L. Cottrell, et al.

Contributed to 2006 IEEE/IFIP Network Operations
& Management Symposium (NOMS 2006), April 3-7, 2006, Vancouver, Canada

Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309

Work supported by Department of Energy contract DE–AC02–76SF00515.

Evaluation of Techniques to Detect Significant Network Performance Problems using End-to-End Active Network Measurements

R. Les Cottrell[1], Connie Logg[1], Mahesh Chhaparia[1], Maxim Grigoriev[2]
SLAC Computing Services(SCS)
Stanford Linear Accelerator Center(SLAC)[1]
Menlo Park, CA 94025 USA
Fermilab[2]
Batavia, IL 60510, USA
{cottrell,cal,maheshkc}@slac.stanford.edu, maxim@fnal.gov

Felipe Haro[3], Fawad Nazir[4], Mark Sandford[5]
Pontificia Universidad Catolica de Chile[3],
Santiago, Chile
NUST Institute of Information Technology[4]
Rawalpindi, Pakistan
Department of Electronic Engineering[5],
Leicestershire, LE11 3TU, UK
felipeharo@gmail.com, fawad.nazir@niit.edu.pk,
J.M.Sandford@lboro.ac.uk

***Abstract*—.End-to-End fault and performance problems detection in wide area production networks is becoming increasingly hard as the complexity of the paths, the diversity of the performance, and dependency on the network increase. Several monitoring infrastructures are built to monitor different network metrics and collect monitoring information from thousands of hosts around the globe. Typically there are hundreds to thousands of time-series plots of network metrics which need to be looked at to identify network performance problems or anomalous variations in the traffic. Furthermore, most commercial products rely on a comparison with user configured static thresholds and often require access to SNMP-MIB information, to which a typical end-user does not usually have access. In our paper we propose new techniques to detect network performance problems proactively in close to real-time and we do not rely on static thresholds and SNMP-MIB information. We describe and compare the use of several different algorithms that we have implemented to detect persistent network problems using anomalous variations analysis in real end-to-end Internet performance measurements. We also provide methods and/or guidance for how to set the user settable parameters. The measurements are based on active probes running on 40 production network paths with bottlenecks varying from 0.5Mbits/s to 1000Mbit/s. For well behaved data (no missed measurements and no very large outliers) with small seasonal changes most algorithms identify similar events. We compare the algorithms' robustness with respect to false positives and missed events especially when there are large seasonal effects in the data. Our proposed techniques cover a wide variety of network paths and traffic patterns. We also discuss the applicability of the algorithms in terms of their intuitiveness, their speed of execution as implemented, and areas of applicability. Our encouraging results compare and evaluate the accuracy of our detection techniques when applied to step down/up, diurnal changes and congestion effects.**

This work was supported in part by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical and Computational Sciences Division under the U.S. Department of Energy. SLAC is operated by Stanford University for the U.S. Department of Energy under contract DE-AC02-76SF00515. Fermilab is operated by Universities Research Association, Inc. for the U.S. Department of Energy under contract DE-AC02-76CH03000.

Keywords- Anomalous event detection, forecasting, network monitoring, network performance, performance analysis, persistent anomalies, trouble shooting, Kolmogorov-Smirnov, Holt-Winters, Plateau algorithm.

I. INTRODUCTION

Management of wide area networking from an end user/administrator point of view is increasingly hard as the complexity of the paths, the diversity of the performance, and the dependency on the network increases. Several monitoring infrastructures have been built [1], [2], [3], [4], [5], [6], [7] to assist by addressing the measurement, archiving, analysis, and presentation aspects of end-to-end performance monitoring. Each of these infrastructures consists of tens to hundreds of monitoring hosts. Each of these monitoring hosts, can make measurements of multiple metrics e.g. delays (both Round Trip Time (RTT) and one way delay), loss, jitter, TCP achievable throughput, available bandwidth, and applications' performance (e.g. file transfers or web requests) to hundreds of monitored (remote) hosts. Typically for every pair of hosts (monitor and remote host) there will be a time series plot for each metric, amounting to hundreds to thousands of plots that need to be reviewed to look for anomalous changes in performance. The network administrator can, at best, review some of these reports reactively upon being presented with a problem by a user. We need to enable the network administrator to be pro-active and spot the problem before the user. This in turn requires automating reliable (few false positives and most events detected) detection of persistent (lasts for at least a few hours), anomalous (unusual and significant[1]) changes (events) in performance

[1] For our purposes we roughly defined an anomalous event as having a relatively quick (fall time ≤ 3hours) step down in performance, where the magnitude of the step was over 10%, and the reduction in performance lasted for a duration of over 4 hours. The fall time and duration depend on the needs of the network administrator and the frequency of measurements

and reporting them in an efficient way to the network administrator.

Most current commercial products rely on a comparison with a user configured static threshold value and often require access to SNMP MIB information from network devices that the end-user does not have permission to view. Our intent is to dynamically derive the threshold from the end-user accessible data so that it automatically tracks the network's performance.

In this paper we report on several open source approaches to make forecasts and automatically detect persistent anomalies in end-to-end network performance metrics using active end-to-end network performance measurements from an instantiation of the IEPM-BW [2] measurement infrastructure. The requirements are to detect decreases in performance that are sufficiently large and persist for sufficient time that, upon notification, the local network administrator is able to review the change and report the problem to the up stream provider's Network Operations Center.

The rest of the paper is organized as follows. Section II describes how the measurements were made, Section III describes previous work and the parameter setting we used for the various techniques used to extract anomalous events, Section IV describes the results, Section V presents the conclusions and section VI describes in progress and possible future work.

II. MEASUREMENTS

We use measurements from the ABwE [8] lightweight bandwidth estimation tool that uses the packet pair dispersion technique, and from the more intrusive [9] iperf [10] achievable throughput estimation tool. We are also applying the techniques to bbftp [11] and GridFTP [12] measurements made at 60 to 120 minute intervals [13], though the results from this are not reported here. ABwE was chosen since it quickly (< 1 second) and with low impact (it uses only twenty packets per direction to make a measurement) provides both RTT and rough dynamic bandwidth estimates, that are important to many applications such as bulk data transfer, while it imposes a light network load. The frequency of the measurements used for the current work is one to three minute intervals. For each interval, three metrics are measured: dynamic bottleneck capacity (*Cap*) by analyzing the minimum packet pair separation; Cross Traffic (*Xtr*) by analyzing the packet pair dispersion; and the Available Bandwidth (*Abw*) = *Cap* – *Xtr*. ABwE also simultaneously provides *Cap*, *Xtr* and *Abw* measurements for the reverse direction.

The *Abw* measurements are probably of most interest to a user, however they are more sensitive to cross-traffic over which we have little control. Changes in *Cap* on the other hand are more likely to reflect route changes or operator errors etc. and thus may be easier to address. *Cap* estimates are thus generally preferred for our work. Since only 20 packet pairs are used for each bandwidth estimate, the statistical variability of the estimates is quite high. Estimates can thus vary dramatically from minute to minute and have large outliers. Therefore, ABwE also provides smoothed data using an Exponential Weighted Moving Average (EWMA).

Single stream iperf measurements of achievable TCP throughput were made for 15 second periods at 90 minute intervals. Potentially the measurements can utilize a significant fraction of the available network bandwidth for small RTTs (less than say 20 milliseconds). At higher RTTs the standard TCP algorithm's congestion control recovers slowly enough following congestion that the aggregate utilization is not significant.

The measurements are made to about 40 hosts in 13 countries. The static bottlenecks vary from 0.5Mbits/s to 1000Mbits/s. The paths traverse about 50 Autonomous Systems (ASs) and over 15 major Internet Service Providers (ISPs). The topology of the remote hosts is seen in Fig. 1. The main ISPs that the paths cross are identified as shaded boxes. For Abilene and ESnet the major Points of Presence (PoPs) are also identified. The remote host sites are also noted, as well as the capacity bottlenecks (*Cap*) for the paths. Five of the remote hosts (identified in Fig. 1 by "I2" and "Host") are at ISP PoPs, the remainder are at end user sites.

The measurements also suffer from gaps in the observations due to problems with the measurement host, the paths and/or the remote (measured host).

III. RELATED WORK AND ANALYSIS

A. Plateau Algorithm

The "Plateau" bandwidth change detection algorithm is described in [14]. It is a modification of the algorithm described in [15] that was successfully used to detect step changes in a time series set of measurements of RTT. Here we use it to analyze both the *Abw* and *Cap* measurements. Currently, missing measurements (e.g. because there is no functioning path between the monitor and monitoring host) are ignored and not assumed to be zero.

The Plateau algorithm basically divides the measurements into two buffers: a history buffer (h) for base-lining, or into a trigger buffer (t), when a measurement meets a specific requirement. The specific requirement is that the current measurement is less than $\boldsymbol{\beta}$ (we use bold face to indicate a user settable parameters) standard deviations (o_h) below the current mean of the history buffer m_h. If the measurement is placed in h then the oldest entry is removed from t. The buffers have maximum durations of $\boldsymbol{\lambda}$ (history) and $\boldsymbol{\tau}$ (trigger). Given a requested buffer duration, the number of items in a buffer (length) is calculated using the median time

available. Less frequent measurements will necessarily increase the fall time and duration in order to accumulate sufficient data to be statistically meaningful.

separation of the data points. When τ is reached the mean of the trigger buffer m_t is compared with m_h and if the relative difference $\Delta = (m_h - m_t) / m_h$ is greater than the threshold δ then an event is deemed to have occurred.

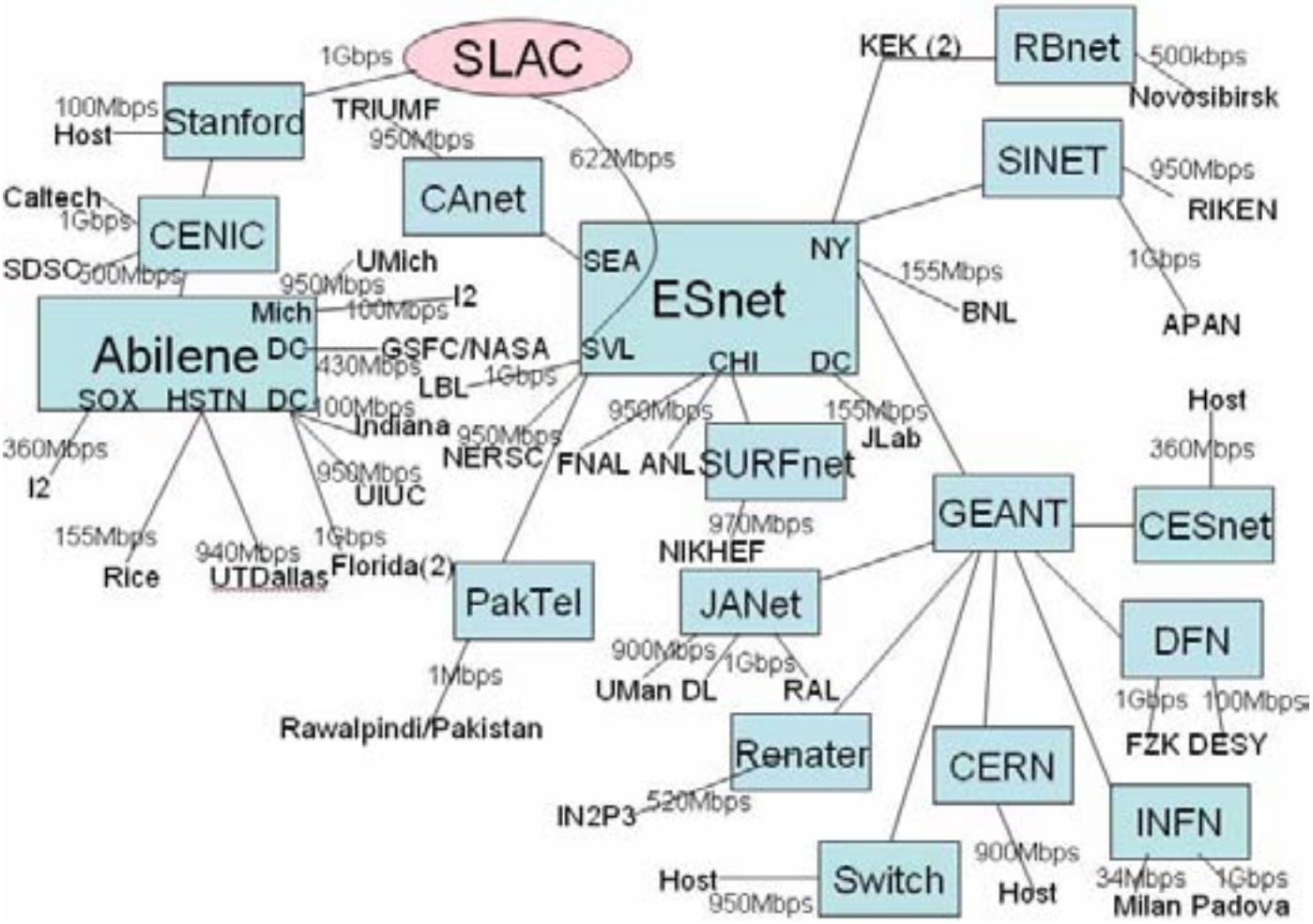

Figure 1: Topology of the remote hosts measured from SLAC

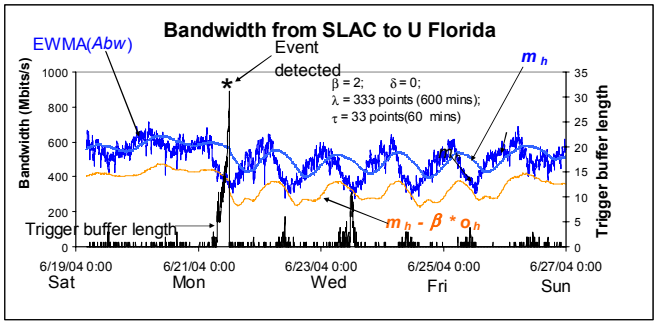

Figure 2 : ABwE bandwidth estimates from SLAC to U. Florida with a history buffer duration (λ) of 10 hours.

Sensible values of β are between 2 and 3 [16], we used β = 2. To minimize the effects of diurnal changes we used λ = 1 day. Less frequent measurements will require λ to be longer. In general we believe λ should be aligned with any seasonality in the data (e.g. an integral number of days) the length of the history buffer should be > 100 points and λ should be >> τ. Larger values flatten the time series behavior of m_h, shorter values will yield less statistically accurate values of m_h. Values that are not aligned with the seasonality (in our case the diurnal behavior) of the data will result in the sinusoidal-wave like curve of m_h being out of synchronization with the diurnal changes (see for example Fig. 2 (where λ = 10 hours, τ = 3 hours, δ = 0%), and is seen to trail the EWMA(*Abw*) by several hours). Note that missing data points can also cause loss of synchronization. Since we were only interested in long term changes we typically use τ = 3 hours. For measurements at 3 minute intervals this gives a

trigger buffer length of 60 that gives sufficient statistical accuracy. We currently use δ = 33%. Larger values of δ are likely to miss more real events; lower values are likely to lead to more false positives.

B. Kolmogorov-Smirnov (KS)

The KS test [17] is the best known of several distribution free techniques that test general differences between distributions. The technique makes no assumption about the underlying distribution of the measurements. It compares the observed and expected Cumulative Distribution Functions (CDF) for ***M*** data points before (expected) and after (observed) each measurement being evaluated. The KS statistic is calculated taking the vertical difference between the two CDFs as a test statistic. We define the KS parameter (***K***) as the threshold value of the KS statistic above which an event is deemed to have occurred. ***M*** in the current work was chosen to be 100 (5 hours for measurements separated by 3 minutes) as a reasonable compromise between the accuracy obtained (sufficient points for the distributions) and the time to wait for a response or the analysis time. If response time is deemed less important it may well be worth using a larger value of ***M*** as this will provide a larger sample for each distribution. For previous work [18] with less frequent measurements, we chose ***M*** to be 24 hours worth of data to minimize the diurnal effects as discussed earlier in the current paper (although one still gets false positives associated with weekends, public holidays etc.)

C. Holt-Winters (HW) Algorithm

The Holt-Winters (HW) [19] [20] algorithm uses a triple EWMA approximation to characterize the time series behavior as a superposition of three components: a baseline, a linear trend and a seasonal effect (e.g. diurnal changes). We developed two implementations of the HW technique and also used the RRD implementation [19] to compare our results against, and to understand the technique. We will focus our discussion on the implementation developed at SLAC (based on the formulation in [20]) since it has the most flexibility for our needs.

HW is critically dependent on having regularly spaced data with no missing points, so the first step is to bin the data into regularly spaced time bins and use similar data to interpolate for bins with no data. For bins with no data in the first week we use data from following weeks for the same day and time bin. For the following weeks we use the previous week's interpolated data. For our data with bin widths of 3 minutes we found that having about five to seven weeks of data enabled us to successfully interpolate the data and fill in missing bins. Once we have the first week's interpolated data, new data can be quickly merged onto the existing interpolated data without having to go back through all the data.

Due to the noisiness of the data, we also set the maximum forecast = maximum of all observed values.

We used the following two methods to choose the initial HW parameters.

1. We chose the initial HW parameters using the guidelines in [18]. 99% of the contribution for the baseline EWMA came from measurements made in the last 24 hours; 99% of the seasonal EWMA contribution came from the last week, 50% of the trend EWMA contribution came from the last 24 hours.
2. For each path, we minimized the sum of the squares of the residuals ($R^2 = \Sigma r_i^2$, where $r_i = y_i - f_i$, the sum is over all interpolated data, y_i is the interpolated observation at time i and f_i is the forecast at the same time) as a function of the HW parameters. This method always resulted in the trend parameter being set to very close to 0 (< 0.00001).

We have settled on using the second method. It provides good forecasts, works for a wider range of paths and requires minimal user input.

HW is a forecasting technique, and needs to be complemented with a method to identify events. The following techniques analyze HW residuals in different ways to raise events:

- The residual (r_i) at each point was examined to see if it was a trigger, i.e. outside the standard deviation of the forecast for the last say 100 points. If 70% of the points in a window of 2.5 hours were triggers then an event was generated (referred to as HWR).
- The residual was compared to the EWMA of the absolute deviation [18]. If over 82% of the residuals were outside twice the EWMA of the absolute deviation in the last 84 minutes then an event was generated (referred to as HWE).
- With a moving window sized to cover 12 hours we calculated $X^2 = \Sigma r_i^2 / f_i$, and using tabulated X^2 values for $N-1$ degrees of freedom, where N is number of points per time window, we set a threshold to generate our triggers and more than 50% of points in the time window generated triggers then an event was raised (referred to as HWX).
- We applied the Plateau algorithm on HW residuals (PHR technique) and KS on HW residuals (KHR technique).

D. Mark Burgess (MB) Technique

The Mark Burgess (MB) technique introduces a two dimensional time approach [21] to classify a periodic, adaptive threshold for service level anomaly detection. An iterative algorithm is applied to history analysis on this periodic time to provide a smooth roll-off in the significance of the data with time. This method was originally designed to detect anomalous behavior on a single host, with the aim of using the information for self-regulation, by initiating a counter response. An anomaly is indicated by a code indicating the state of the given statistic, as compared to an average of equivalent earlier times.

IV. RESULTS

A. Creating a Canonical Dataset

To provide a canonical set of measurements to evaluate and compare the detection methods against we used *Cap* measurements for ~100 days from June through September 2004 from SLAC to 30 remote hosts at sites shown in Fig. 1.

To first characterize the potential events seen in the canonical data, we used the Plateau algorithm, since it is the most intuitive of the algorithms and allows direct variation of parameters representing the size and duration of an event. It captures all true events and may generate false positives which were reviewed and classified. We set δ to 0 (i.e. we detect all events that fill the trigger buffer) and the other user parameters were set as described above. About 25 of the 40 paths manifested one or more events in this period. We carefully reviewed each of these events and created a library of interesting events. We observe three general types of events that trigger our Plateau algorithm.

- Step down changes in bandwidth ("step")
- Diurnal changes ("diurnal")
- Changes caused by actions causing congestion, e.g. a regularly scheduled cron job ("host"), or network bandwidth test, flash crowds etc.

Three paths out of 30 (Caltech, NIIT and U. Florida) exhibited marked diurnal changes that triggered "diurnal" Plateau events, especially following a weekend. To study the diurnal behavior more carefully we binned the bandwidth data by hour of day and calculated the percentiles to identify the daily bandwidth patterns. An example is shown in Fig. 3 where there is a quick decrease in bandwidth when people arrive to work (20:00-23:00 PDT = 08:00 – 11:00 Pakistan time). This in turn causes an abnormally high number of events to be detected by the Plateau and KS algorithms during these hours (see Fig. 4). Usually these changes are not as sudden as the typical step change which may also help in separating the two types. For our purposes, these "diurnal" events are false positives that need to be eliminated.

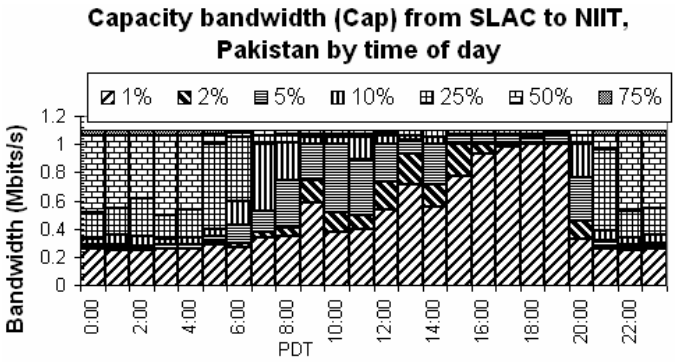

Figure 3: Percentiles of capacity bandwidth (*Cap*) seen on the SLAC-NIIT path as a function of time of day

One host (ANL) exhibited regular "host" type events that were tracked down to a `cron` job running on the host that used (via NFS) the network heavily. This host was eliminated from further non-seasonal analyses. Events for a given host typically have a small range for Δ (standard deviation (Δ) / mean (Δ) ~ 0.11± 0.1) indicating that the backup routes or diurnal behavior is consistent. This manifests itself in a multi-modal Distribution Function for Δ with a small number of modes.

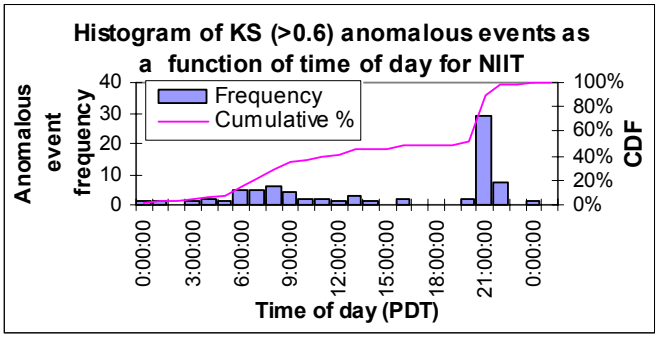

Figure 4: Distribution of events seen by KS in the *Cap* data as a function of time of day for the SLAC-NIIT path. We observed similar results using the Plateau algorithm.

Table 1: Comparison between different techniques based on various parameters.

Algorithm / Technique	Detects Step Up or Down	Tolerance to Seasonal Changes	Number of Parameters	CPU Utilized[2]	Regularized Data Required[3]	Data Requirement[4]	Applicability
Plateau	Step Down only[5]	Poor	Medium	4	No	Low	Suitable for finding long term persistent step changes, but can be fooled by seasonal changes.
KS	Both	Poor	Low	2	No	Medium	More valuable for applications which are sensitive to data distribution (e.g. interactive voice which depends on jitter). Can be fooled by seasonal changes.
HWE	Both	Medium	Medium	3	Yes	High	Basically a forecasting technique, needs a more sophisticated event detection mechanism.
MB	Both	Medium	Medium	1	No	Low	Aimed at real time identification of changes
PHR	Both	High	High	5	Yes	High	Identifies step up/down events and absorbs seasonal changes
KHR	Both	Medium	Medium	3	Yes	Medium	Less effective than PHR for seasonal changes.

By careful examination of ~ 120 Plateau candidate events detected[6] with $\delta = 0$ (and ignoring whether the events are diurnal) we classify all candidates as to whether they are events we are interested[7] in or not (i.e. exhibit sharp drop in bandwidth (e.g. 90% of change occurs in < 220 mins), persist for a long term (>> 3 hours) and are large enough (e.g. $\delta > 10\%$)). With $\delta = 10\%$ and restricting the duration of 90% of the trigger buffer to 220 minutes, we miss 8% of the events and see 16% false positives. Increasing δ to 33% we get 32% misses and 2% false positives. In this case 15% of the events are caused by diurnal changes.

[2] Large numbers indicate more CPU utilization. These relative values are for our implementations which have not been optimized for CPU utilization. Also, as noted in the text the CPU utilization can depend on the parameters chosen.

[3] HW requires data at regular intervals to quickly identify data from similar phases of the seasonal cycles. Regularized data also helps for the Plateau algorithm since it can ensures the history buffer is for a fixed time interval.

[4] HW requires much more data (e.g. several weeks) to be able to see the effects of seasonal changes.

[5] Modifying our Plateau algorithm to accommodate steps in both directions is fairly simple and has been implemented in a more recent version.

[6] Less than 10% of these candidate events were associated with noticeable traceroute changes.

[7] Others may have very different criteria, in particular the duration of the change or the magnitude of the drop. With the exception of the Plateau algorithm, the duration and magnitude filters need to be applied as a separate step.

B. Comparisons of the Various Techniques

We now present observations mostly on data patterns that have traditionally posed a challenge to the event detection techniques (e.g. diurnal changes) and other interesting scenarios. Table 1, presents a summary of this comparison between different techniques on various useful parameters

We applied KS and HW (with various anomalous event detection methods described in section III) algorithms to the canonical data. Fig. 5 shows a visualization of the Plateau, KS and HW algorithms applied to capacity bandwidth observations with step changes in performance from SLAC to BINP. As expected all three algorithms detect marked changes. KS detects both increases and decreases in the data and thus detects roughly twice as many events (229:116) as the Plateau algorithm which was tuned for negative changes only. Detecting both steps is valuable since, for example, it enables determining the duration of a change and/or taking some action when the original performance is restored. KS also provides the most accurate time estimate[8] for when the event occurred which is important when correlating the event with other time-dependent information.

[8] The time estimates for Plateau and HW could be improved, for example for the Plateau algorithm by identifying the time of the change when the event occurred as the time at which the trigger buffer reached say 10% full.

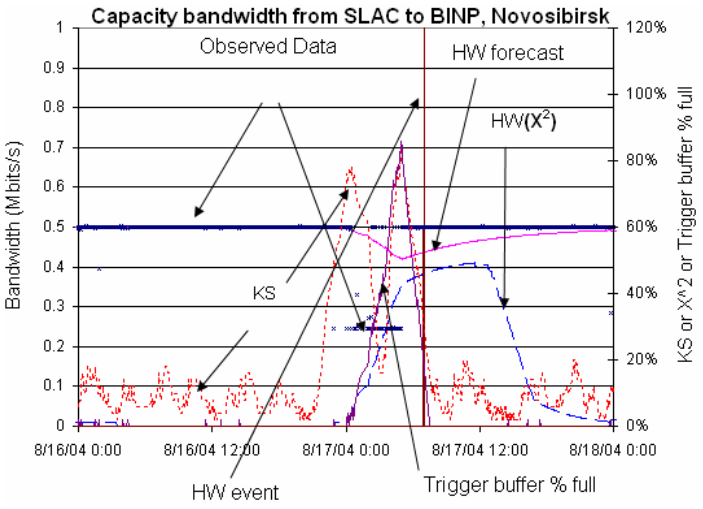

Figure 5: Plateau, KS and HW algorithms applied to an observed step down and back up performance change

In Fig. 5 the KS statistic (dotted) is seen to reach a value of about 80%, on both the step up and down. The Plateau algorithm's trigger buffer (solid line) only reaches ~ 80% full (since the step down's duration is too short, in this case ~ 4 hours) so no event is triggered. The HW X^2 (dashed line) triggers an event after the performance has recovered.

Increasing the threshold value (***K***) of the KS statistic that defines an event reduces the false positives at the cost of increasing the missed events as seen in Fig.6. Currently we are using a value of $K = 0.7$ to trigger an event. Most (69%) of the false positives come from four paths.

Minimizing R^2 to estimate the initial HW parameters results in a fairly wide range of values of the local smoothing parameter ($a = 0.0001$ to 0.95, median = 0.0024 ± 0.12) and the seasonal parameter ($b = 0.0023$ to 0.999, median = 0.22 ± 0.2). About 50% of the paths have $0.0008 < a < 0.02$ and $0.18 < b < 0.4$. There is very weak correlation between a and b or R^2 and a or **b**, which is suggestive that there may not be a suitable single set of parameters for all paths.

Using HW to forecast, and with the 70% of triggers in 2.5 hours method to detect events, plus bunching together events separated by < 3 hours, successfully removed the diurnal events for Caltech, NIIT and U. Florida. It also succeeded in eliminating the effects of the ANL `cron` jobs that ran at regular times each morning. However, there appear to be similar host based effects that do not occur at regular intervals that make the ANL data problematic. A similar effect giving rise to false positives was seen with the SDSC path. In this case there was a step down of about 10% (65Mbits/s) lasting for 3-4 hours starting around 1am each day for about one week.

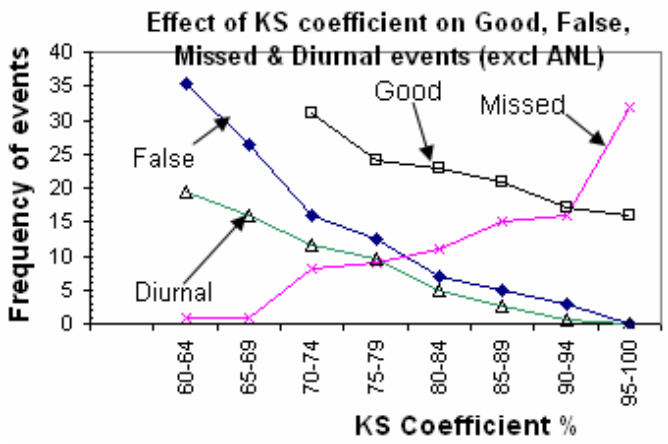

Figure 6: Cumulative step down event types as function of KS parameter

With the above HW event detection method, three paths (SOX, NASA and CESnet) with very small deviations in the observations, had small changes in bandwidth that resulted in events.

Eliminating the ANL and SDSC paths, and demanding a bandwidth change of at least 5% for an event, HW detected 23 true events, with 1 false positive and 6 missed events. Four of the misses were during the first week of data, when the HW algorithm performs poorly due to not having good initial estimates. A further miss was since the change happened slowly (it took over a day to get from the initial value to the new stepped down value). The final miss was for a step down in performance that only lasted four hours. Thus, with the caveat that HW cannot forecast (and thus is not amenable to event detection methods discussed above) for measurements affected by applications running at irregular times and causing congestion, the HW technique works well for our data. To understand such events better more measurements are needed on hosts and network equipment to isolate the cause.

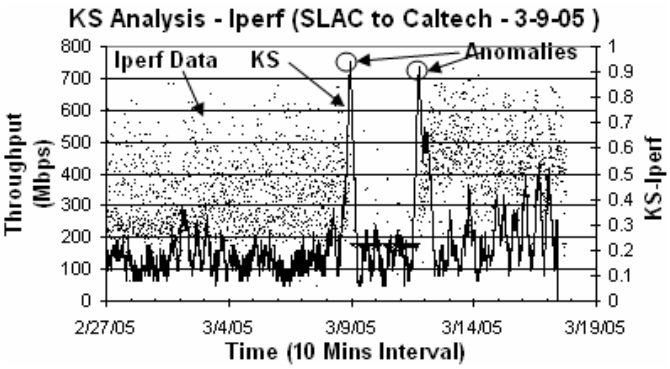

Figure 7: KS on Iperf data from SLAC to Caltech

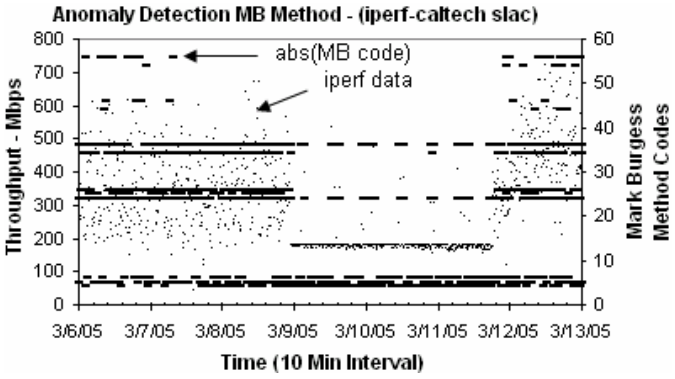

Figure 8: MB on Iperf Data SLAC to Caltech

Figs. 7 and 8 show the effects of applying the KS and MB techniques to the same iperf data from SLAC to Caltech March 6 – 13, 2005. Fig. 8 is a snapshot of the original graph showing the MB technique applied on iperf Data. This snapshot gives a good idea about the real-time behavior of MB technique. It is seen in Fig. 7 that the KS statistic (solid line) identifies the obvious long term step down and step up (both identified by circles) in the data on March 9, 2005 and March 11, 2005 respectively. In Fig. 8 on the other hand, the MB code (light dots) oscillates wildly as it tries to track the individual spikes in the noisy data and misses the important steps down and up.

We then applied Plateau and KS on the HW residuals for the *Cap* measurements from SLAC to the U. of Michigan which have both diurnal changes and one-off anomalous events. We find that although both are able to detect one-off step-downs, Plateau shows no false positives while KS raises several events on weekends. This happens because KS compares two frequency distributions irrespective of the relative change in values. During weekends, residuals are close to 0 as data values usually mirror past weekend's data and HW is able to make good forecasts based on its past week's seasonal cycle values. However due to higher network usage during weekdays, there are higher fluctuations and residuals are more spread out, although small in absolute value. So KS on weekend data effectively compares two very different distributions (past weekdays and weekend) thereby raising false events. This interesting observation highlights a weakness of KS for our current application.

C. CPU Utilization

Both the Plateau and HW algorithms are implemented on Linux systems as Perl scripts and so should be relatively easy to port. Currently no attempt has been made to optimize the speed of execution. For 43K data points on a dual Xeon 3GHz cpu host, it takes about 15 seconds to interpolate the data, 30-40 seconds to minimize R^2 to find the optimum HW parameters, and about 2 minutes for the HW analysis and reporting. The Plateau algorithm has mainly been used for exploring the data and thus has many extra tests and reporting which result in it taking about six times as long. The KS algorithm is implemented in C and takes about 1 minute with $\boldsymbol{M}$ = 100. The execution time of KS, however, is very dependent on $\boldsymbol{M}$. For example, increasing $\boldsymbol{M}$ from 100 to 400 increases the analysis time by a factor of 14.

V. CONCLUSIONS

End-to-End fault and performance problem detection is an important and challenging problem. The major challenges are due to the complexity of different network components, their working and interaction, diversity of the network performance and dependency on the networks. On the other hand, interpreting limited amount of end-to-end network monitoring data and detecting performance problems without having access to the intermediate network devices imposes another important challenge. To solve this problem most of the existing commercial network problem detection tools use statically set threshold based techniques that require human interaction and thus are not adaptive to network change. Keeping in consideration the dynamic nature of network end-to-end paths and above mentioned challenges, we proposed a new technique to detect network performance problems proactively in near real time. In our method we do not rely on the device specific information of the intermediate nodes like SNMP-MIB's as typically network end users don't have access to this type of information. The major techniques which we used in our approach were: the Plateau algorithm; Kolmogorov-Smirnov (KS) technique; the Holt-winters (HW) forecasting algorithms; and the Mark Burgess (MB) technique. We also integrated different techniques applying the Plateau algorithm to HW Residuals and KS to HW-Residuals. We applied these techniques on our active end-to-end measurements that we gathered by monitoring 40 production network paths with bottlenecks varying from 0.5 Mbits/sec to 1000Mbit/sec.

Our results show that for measurements with limited diurnal (or other seasonal) changes the Plateau, KS and HW algorithms work well. The HW technique explicitly incorporates seasonal changes and so event detection methods incorporating HW work better than the Plateau and KS methods on paths with significant diurnal changes. Among the different event detection methods discussed for HW, Plateau and KS applied to HW residuals are the most effective in identifying seasonal variations.

KS provides accurate identification of when the step up or down occurred. KS, compared to Plateau is sensitive to both the average values and the distribution. It may thus be more valuable if the concern is for applications which are distribution sensitive (e.g. real-time applications such as interactive voice which depends on jitter). MB is aimed at real-time identification of changes but is not suited to finding long-term persistent anomalous changes. On the other hand for our purposes we are more interested in changes in average performance and so prefer the Plateau algorithm applied to HW residuals.

The Plateau algorithm is easily understood by people with a non-statistical background and has easy to interpret user settable parameters.

The KS algorithm is the most statistically formal of the three algorithms. It has only two parameters: the number of data points (***M***) used to evaluate the distribution functions; and the threshold (***K***) of the KS parameter above which an event is assumed to have occurred. It provides the best estimates of when a step occurs.

For the HW techniques it is critical to provide complete data at regular time intervals. Also one needs a few weeks worth of data to get a reasonable estimate of the seasonal variations. Estimating the parameters using the minimization techniques appears to work well in most cases. The basic Holt-Winters algorithm is a forecasting technique, and so may also be used for providing forecast information to applications such as Grid middleware [22].

Each technique requires several parameters. We have provided guidelines for selecting the Plateau parameters. For KS there are only two parameters. We settled on using ***M*** = 100 data points and a threshold (***K***) of 70%. We have developed an automated technique (minimizing the residuals) to select the HW parameters to greatly simplify the use of the forecasting technique. This is particularly important since there is not a single set of selected HW parameters for all paths.

VI. Future Work

Since this is a production network that we do not administer, we are not comfortable with deliberately introducing known problems into the network to characterize their effects. On the other hand, now we have understood the techniques and optimized the parameters, we have reduced the number of potential events to a few per week. Each of these is now reported by email and carefully analyzed to see if it corresponds to a known cause (network maintenance, fibre cut, routing change, incorrect configuration etc.) Initial results are encouraging, but more work is needed to properly characterize the relationships. Our plan is to filter the events, gather relevant information gathered from network devices, further analyze this data and report it to network administrators in the email.

We are looking at using wavelet decomposition to eliminate outliers and white noise seen in real data.

To reduce the problems with occasional outliers in the ABwE measurements caused by inter-packet timing problems [23], we are evaluating using PathChirp [24][24], and/or Pathload [25] instead. Both PathChirp and Pathload increase the network traffic and the time to make a measurement by roughly a factor of ten and 100 respectively, they do appear to give more accurate results.

The subspace PCA analysis has been reported [26][26] to work well when applied to measurements from core routers. It is unclear how well it will work on less correlated end-to-end active Internet performance measurements. It has the advantage of being able to simultaneously look at measurements of multiple metrics (e.g. RTT, iperf throughput, *Cap, Xtr*, reverse and forward performance measures such as provided by ABwE, and/or hosts/system performance measures) and paths simultaneously. On the other hand it may be less intelligible to someone without a statistical background. We have implemented this algorithm and are currently applying it to ABwE, iperf, and PathChirp measurements.

A second, filter process may be applied once potential events have been identified using one of the above techniques. A filter may serve two purposes. Firstly unwanted events, or false positives may be removed and not reported. Secondly, events that are noteworthy may be classified to aid in further diagnostic processes. A neural network has been used as such a filter process in [18]. We are also looking at using neural networks to interpolate between infrequent, heavyweight measurements such as iperf by using more frequent lightweight measurements such as ABwE or RTT etc.

Once we have a robust, reliable anomalous event detection technique that works with the ABwE data, we will extend it to other measurements including GridFTP and observations of host/system performance measures.

Besides detecting anomalous events, we plan to make the forecasts available for Grid middleware such as a replication manager.

Acknowledgement

We gratefully acknowledge Jerrod Williams and the administrators at the remote hosts for their work in setting up the hosts and keeping the measurements running. We also thank Mark Crovella, Jerry Friedman, and Waqar Mahmood for useful discussions on the subspace PCA technique, and Alf Wachsman for help with the RRD version of HW and assistance with the CFETool version of MB. Also thanks are due to the reviewers(s) for many helpful suggestions.

References

[1] McGregor A, Braun H-W, and Brown J, "The NLANR network analysis infrastructure", IE*EE Communications Magazine*, May 2000

[2] Cottrell R. L, Logg C, and Mei I-H, "Experiences and Results from a New High Performance Network and Application Monitoring Toolkit", *PAM 2003*.

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

[4] "National Internet Measurement Infrastructure", available at http://www.ncne.nlanr.net/nimi/

[5] "E2E piPES", available at http://e2epi.internet2.edu/pipes/

[6] "RIPE NCC Test Traffic Measurements", available at http://www.ripe.net/ttm/

[7] "MonALISA: Monitoring Agents using a Large Integrated Services Architecture" available at http://monalisa.caltech.edu/

[8] Navratil J and Cottrell R. L, "ABwE: A practical Approach to Available Bandwidth Estimation", *PAM 2003* also SLAC-PUB-9622.

[9] Alok Shriram , Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, k claffy "Comparison of Public End-to-end Bandwidth Estimation Tools on High-Speed Links", PAM2005.

[10] "Iperf the TCP/UDP Bandwidth Measurement Tool", available http://dast.nlanr.net/Projects/Iperf/

[11] "BBFTP Large Files Transfer Protocols", available at http://doc.in2p3.fr/bbftp/

[12] "Globus Grid FTP Protocol and Software", available at http://www.globus.org/datagrid/gridftp.html
[13] Grigoriev M, Cottrell R. L. and Logg C., "Wide Area Network Monitoring System for HEP Experiments at FNAL", *Computing in High Energy Physics*, Interlaken Switzerland, Sep 2004.
[14] Logg C, Cottrell R. L. and Navratil J., "Experiences in Traceroute and Available Bandwidth Change Analysis", *SIGGCOMM'04 Workshops*, Aug 30 & Sep 3, 2004, Portland OR, USA
[15] McGregor A.J. and Braun H-W, "Automated Event Detection for Active Measurement Systems", Proceedings of PAM2001, Amsterdam, Netherlands, April 2001
[16] A. Ward, P. Glynn and K. Richardson, "Internet service performance failure detection" *Performance Evaluation Review*, 26:38-43, 1998
[17] Brockwell P. and Davis R, "Introduction to Time Series and Forecasting", *Springer* New York, 1996
[18] Sandford, J.M., Parish, D.J. and Phillips, I.W., "Neural approach to detecting communication network events", IEEE Proc. Communications, 1495 , October 2002, pp 257-264, ISSN 1350-2425.
[19] Brutlag J.D., "Aberrant Behaviour Detection in Time Series for Network Monitoring", Proceedings of LISA 2000, New Orleans, LA, USA, December 2000.
[20] NIST e-handbook of statistics, http://www.itl.nist.gov/div898/handbook.
[21] Mark Burgess, Two dimensional time-series for anomaly detection and regulation in adaptive systems, Proceedings of 13th IFIP/IEEE International Workshop on Distributed System, operations and management (DSOM 2002). "Management Technologies for E-Commerce and E-Business Applications" Springer 2002.
[22] Ian Foster, Carl Kesselman, "The Grid 2: Blueprint for a New Computing Infrastructure", Publisher: Morgan Kaufmann; 2 edition (November 18, 2003), ISBN: 1558609334
[23] Shiram A, Murray M, Young H, Brownless N, Broido A, Fomenkov M, Claffy K, "Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links, PAM 2005.
[24] Ribeiro V, Reidi R, Baraniuk R, Navratil J, Cottrell R. L, "pathchirp: Efficient Available Bandwidth Estimation for Network Paths", PAM 2003.
[25] Jain, M., Dovrolis, C, "Pathload: an available bandwidth estimation tool", PAM 2002.
[26] Lakhina A, Crovella M, Diot C, "Diagnosing Network-Wide Traffic Anomalies", *Sigcomm* 2004.