Communication-Efficient Online Detection of Network-Wide Anomalies

Ling Huang* XuanLong Nguyen* Minos Garofalakis† Joseph M. Hellerstein*
Michael I. Jordan* Anthony D. Joseph* Nina Taft†
*UC Berkeley †Intel Research Berkeley
{hling, xuanlong, hellerstein, jordan, adj}@cs.berkeley.edu {minos.garofalakis, nina.taft}@intel.com

***Abstract*—There has been growing interest in building large-scale distributed monitoring systems for sensor, enterprise, and ISP networks. Recent work has proposed using Principal Component Analysis (PCA) over global traffic matrix statistics to effectively isolate network-wide anomalies. To allow such a PCA-based anomaly detection scheme to scale, we propose a novel approximation scheme that dramatically reduces the burden on the production network. Our scheme avoids the expensive step of centralizing all the data by performing intelligent filtering at the distributed monitors. This filtering reduces monitoring bandwidth overheads, but can result in the anomaly detector making incorrect decisions based on a perturbed view of the global data set. We employ stochastic matrix perturbation theory to bound such errors. Our algorithm selects the filtering parameters at local monitors such that the errors made by the detector are guaranteed to lie below a user-specified upper bound. Our algorithm thus allows network operators to explicitly balance the tradeoff between detection accuracy and the amount of data communicated over the network. In addition, our approach enables real-time detection because we exploit continuous monitoring at the distributed monitors. Experiments with traffic data from Abilene backbone network demonstrate that our methods yield significant communication benefits while simultaneously achieving high detection accuracy.**

I. Introduction

Today's large distributed systems (e.g., server clusters, large Internet Service Providers (ISPs), and enterprise networks) employ distributed monitoring infrastructures to collect and aggregate information describing system status and performance. Remote monitor sensors are typically deployed throughout the network yielding numerous large and widely-distributed time-series data streams representing information from multiple vantage points; this information is continuously monitored and analyzed for a variety of purposes.

An example application that employs a distributed monitoring infrastructure is one that seeks to detect network-wide traffic anomalies. Recent work by Lakhina, *et al.* [13] proposes an anomaly detection scheme in which monitors ship observations to a central Network Operations Center (NOC), which in turn assembles and analyzes the data to perform anomaly detection. Specifically, they propose that local monitors continuously measure the total volume of traffic (in bytes) on each network link, and periodically push all recent measurements to the NOC. The NOC then performs Principal Component Analysis (PCA) on the assembled data matrix to reveal traffic anomalies that were not detectable in any single link-level measurements. Lakhina, *et al.* demonstrate that this technique is quite effective in detecting anomalies in traffic, in part due to the inherently low-dimensional nature of the underlying data.

However, such a "periodic push" approach suffers from two scalability limitations. The first limitation has to do with the time scale of operation and how fast anomalies can be detected. The work by Lakhina, *et al* was initially shown to work at 5 and 10 minute time scales [13]. However many anomalies occur on much smaller time scales. If the method were employed on a second or sub-second time scale, then the volume of measurement data transmitted through the network would increase dramatically because the monitoring data would need to be collected on a second (or sub-second) time scale.

The second scalability limitation has to do with the effect of increasing the number of monitoring sites. An approach in which *all* monitors upload *all* of their data to a central processing site regularly, creates two problems. It may overload the central processing site. Also, sending such large quantities of data through the network is a problem for certain kinds of networks such as sensor networks, many wireless networks, and enterprise networks (that do not overprovision inter-site connectivity). Although such measurement overhead may be supportable in today's ISPs, it may not in the future as we move towards petascale monitoring infrastructures that will monitor hundreds or thousands of network data features. The combined effect of increasing the number of monitors while simultaneously reducing the time scale of operation could lead to an explosion in the volume of data collected for this application.

It is a central premise of this work that backhauling all distributed monitoring data may be unnecessary, depending upon the particular monitoring task, and thus smart data-filtering or data-reduction at the local monitoring sites should be employed. This approach would enable distributed monitoring systems to scale more gracefully both as the number of monitors increase and as the time scale for data collection and anomaly detection decreases. The promising effectiveness of the Lakhina, *et al.* technique provides strong motivation for designing a significantly more communication-efficient PCA-based scheme for real-time anomaly detection.

We are thus motivated to study how well traffic anomalies can be detected if only a portion of the monitored data is shipped to the NOC. In this paper, we take the ideas of Lakhina, *et al.* and recast them in a communication-efficient

framework that detects anomalies at a desired accuracy level with low communication cost. It is communication efficient because we reduce the amount of data needed for anomaly detection. We engage the monitors in *local* filtering so that they only send data to the NOC on an "as-needed" basis. The NOC (often referred to as a *coordinator* hereafter) guides the monitors in how to do the filtering because it sees the global data and knows, via the triggering condition, the extent of dependencies across different monitors.

In our framework, the distributed monitors collect data continuously but each monitor only updates the coordinator with new data as needed (determined by the filtering parameter). Monitors can do so at any moment in time, and are not restricted to time window boundaries (such as every 5 minutes). Because the NOC will find out anything it "needs" to know immediately (ignoring network delays), the NOC is effectively doing continuous tracking, which in turn enables real-time detection.

Our Contributions. We propose a novel approach for communication-efficient online detection of network-wide traffic anomalies. Our solution is unusual in that it extends the power of the PCA-based method by coupling insights from *Stochastic Matrix Perturbation (SMP) theory* together with in-network processing ideas [4], [17]. Because we filter locally at the distributed monitors, the NOC's view of the global data (captured in a matrix) is approximate since elements in the matrix can become out-of-date. Thus the computation of the principle components is done on a *perturbed* data matrix. We appeal to Matrix Perturbation theory as it helps to quantify the effect of such perturbations on the computation of eigenvectors and eigenvalues. Out-of-date data can lead to errors that propagate through the anomaly detector including not only the eigenvalues, but also the anomaly trigger thresholds – because all of these are data-driven. This results in an anomaly detector that can make mistakes. Using SMP theory, we derive analytic bounds on the terms affected by error propagation. We design an algorithm that derives filtering parameters for the monitors, such that the errors made by the detector are bounded. Our algorithm combines many techniques together, SMP theory, binary search and monte carlo simulation.

Our evaluation using real-world data streams collected from a well known ISP network shows that our methods work very well. While sending less than 10% of the original time-series data (over an order-of-magnitude reduction in communication), we guarantee that the detection error would be no more than 4% bigger than when using full data. In fact, our system performs much better than these bounds; we find that the actual error rates are nearly indistinguishable from the full-data method. This results in a huge savings in communication overhead (e.g., typically 80 or 90% of the original data is no longer sent) with only a very small impact on errors. Put another way, within the same (fixed) communication budget, our algorithms can allow for a ten-fold increase in the time granularity of network-statistics collection. Finally, we show that our system can indeed scale gracefully as the number of monitors grows.

Prior Work. A number of techniques have been proposed to detect network traffic anomalies [1], [3], [13], [20], [25]. However, the goal of minimizing communication overhead in widely distributed Internet environments has not been addressed. Recent progress in distributed monitoring, profiling and anomaly detection [18], [23], [24] aims to share information and foster collaboration between widely distributed monitoring boxes to offer improvements over isolated systems. These systems are examples for which a distributed detection tool such as ours would be useful.

In a distributed online setting, Keralapura *et al*. [12] proposed solutions to detect threshold violations on sum functions with specified accuracy while minimizing communication overhead. Sharfman *et al*. [19] proposed protocols to detect general distributed functions exceeding thresholds using a geometric decomposition method. However, their method assumes preset thresholds, does not consider global matrix analysis queries, and cannot scale to large networks with high-speed data streams.

In our recent paper [6], we illustrated that SMP theory can be used to analytically relate the errors in detection performance as a function of the errors in data collection. However that work does not define an algorithm for controlling the errors in data collection (e.g., via filtering), because the mathematical relationship between data errors and detection errors is the inverse of what is needed to design a practical scheme. In this work, we design such an algorithm, by allowing the user to specify a tolerable detection error and working backwards to determine the kinds of errors in data collection that are allowed to achieve the target detection error. In [6], the tunable knob was an eigenvalue error metric - not a very intuitive knob. Our approach proposed herein is more appealing since it allows the tolerable detection error to become the tunable knob; because there is a tradeoff between the detection accuracy and the communications cost, our algorithm explicitly allows a network operator to control this tradeoff.

II. Problem description and background

We consider a monitoring system that includes a set of *distributed monitor nodes* $M_1, \ldots, M_n$, each of which collects a locally-observed time-series data stream (Fig. 1). For instance, the monitors may be attached to routers to collect the volume of traffic per second from each network link, participate in firewalls to log the number of TCP connection requests per second, or connect to servers to record the number of DNS transactions per minute. A central *coordinator node* seeks to observe the ensemble of these time series (i.e., the global network-wide data), and make global decisions such as those concerning matters of network-wide health. The application of detecting *volume anomalies* across a large network employs such a distributed monitoring infrastructure. A volume anomaly refers to unusual traffic load levels in a network that are caused by anomalies such as DDoS attacks, flash crowds, device failures, misconfigurations, and so on.

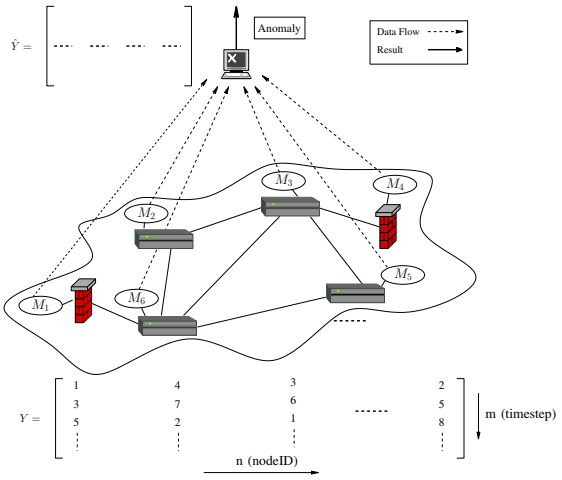


Fig. 1. The distributed monitoring system.

Each monitor M_i collects a new data point $\mathbf{Y}_i(t)$ at every time step and, assuming a naive, "continuous push" protocol, sends the new point to the coordinator. Based on these updates, the coordinator keeps track of a sliding time window of size m (i.e., the m most recent data points) for each monitor's time series, organized into a matrix $\mathbf{Y}$ of size $m \times n$ (where the i^{th} column $\mathbf{Y}_i$ captures the data from monitor i, see Fig. 1). The coordinator then makes its decisions based on this global $\mathbf{Y}$ matrix.

Centralized Subspace Method for Volume Anomaly Detection. We now briefly summarize the PCA-based anomaly detector in [13]. As observed by Lakhina et al., due to the high level of traffic aggregation on ISP backbone links, volume anomalies can often go unnoticed by being "buried" within normal traffic patterns. On the other hand, they observe that, although, the measured data is of seemingly high dimensionality (n = number of links), normal traffic patterns actually lie in a very low-dimensional subspace; furthermore, separating out this normal traffic subspace using PCA (to find the principal traffic components) makes it much easier to identify volume anomalies in the remaining subspace.

As before, let $\mathbf{Y}$ be the global $m \times n$ time-series data matrix, centered to have zero mean, and let $\mathbf{y} = \mathbf{y}(t)$ denote a n-dimensional vector of measurements (for all links) from a single time step t. Formally, PCA is a coordinate-transformation method that maps a given set of data points onto principal components ordered by the amount of data variance that they capture. The set of n principal components, $\{\mathbf{v}_i\}_{i=1}^{n}$, are defined as:

$$\mathbf{v}_i = \arg\max_{\|\mathbf{x}\|=1} \|(\mathbf{Y} - \sum_{j=1}^{i-1} \mathbf{Y}\mathbf{v}_j\mathbf{v}_j^T)\mathbf{x}\|$$

and are the n eigenvectors of the estimated covariance matrix $\mathbf{A} := \frac{1}{m}\mathbf{Y}^T\mathbf{Y}$. As shown in [13], PCA reveals that the Origin-Destination (OD) flow traffic matrices (i.e., the complete traffic demand across an entire network) of ISP backbones have low intrinsic dimensionality. Because the link traffic and the end-to-end traffic demands are linearly related, it turns out that the ensemble of all link traffic in a backbone network also exhibits low dimensionality. For example, in the Abilene network with 41 links, most data variance can be captured by the first $k = 4$ principal components. Thus, the underlying normal OD flows effectively reside in a (low) k-dimensional subspace of $\mathbb{R}^n$. This subspace is referred to as the *normal* traffic subspace $\mathcal{S}_n$. The remaining $(n - k)$ principal components constitute the *abnormal* traffic subspace $\mathcal{S}_a$.

Detecting volume anomalies relies on the decomposition of link traffic $\mathbf{y} = \mathbf{y}(t)$ at any time step into normal and abnormal components, $\mathbf{y} = \mathbf{y}_n + \mathbf{y}_a$, such that (a) $\mathbf{y}_n$ corresponds to modeled normal traffic (the projection of $\mathbf{y}$ onto $\mathcal{S}_n$), and (b) $\mathbf{y}_a$ corresponds to residual traffic (the projection of $\mathbf{y}$ onto $\mathcal{S}_a$). Mathematically, $\mathbf{y}_n(t)$ and $\mathbf{y}_a(t)$ can be computed as

$$\mathbf{y}_n(t) = \mathbf{P}\mathbf{P}^T\mathbf{y} = \mathbf{C}_n\mathbf{y} \text{ and } \mathbf{y}_a(t) = (\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{y} = \mathbf{C}_a\mathbf{y}$$

where $\mathbf{P} = [\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}]$, is formed by the first k principal components which capture the dominant variance in the data. The matrix $\mathbf{C}_n = \mathbf{P}\mathbf{P}^T$ represents the linear operator that performs projection onto the normal subspace $\mathcal{S}_n$, and, $\mathbf{C}_a$ projects onto the abnormal subspace $\mathcal{S}_a$.

As observed in [13], a volume anomaly typically results in a large change to $\mathbf{y}_a$; thus, a useful metric for detecting abnormal traffic patterns is the squared prediction error (SPE): $\mathbf{SPE} \equiv \|\mathbf{y}_a\|^2 = \|\mathbf{C}_a\mathbf{y}\|^2$ More formally, their proposed algorithm signals a volume anomaly if

$$\mathbf{SPE} = \|\mathbf{C}_a\mathbf{y}\|^2 > Q_\alpha \tag{1}$$

where Q_α denotes the threshold statistic for the $\mathbf{SPE}$ residual function at the $1-\alpha$ confidence level. Such a statistical test for the $\mathbf{SPE}$ residual function, known as the Q-statistic [9], can be computed as a function $Q_\alpha = Q_\alpha(\lambda_{k+1}, \ldots, \lambda_n)$, of the $(n-k)$ non-principal eigenvalues of the covariance matrix $\mathbf{A}$. With the computed Q_α, this statistical test can guarantee that the false alarm probability is no more than α (under certain assumptions).

Our Communication Efficient Detection Problem. The problem we address is how to do the filtering at the monitors, so as to send as little data as possible through the network but still allow the anomaly detector to work accurately. The idea is that monitors should send a description of their time series signal, and then not send any more measurements (or summaries) until a change happens that is either "sufficiently large" or likely to impact the global trigger condition being monitored. In our application, the trigger condition being monitored by the coordinator is that in Eqn (1). Because the monitors send data less frequently to the coordinator, the coordinator's view of the global network data can be out-of-date and perturbed. Thus, the statistics it computes for anomaly detection, such as the eigenvalues and projection matrix, will deviate from those of the true global state. This implies that the detection error at the coordinator (when triggering on condition (1)) will differ from that achieved using the full data.

Our solution includes the design of protocols used by the monitors and coordinator, and an algorithm to determine how

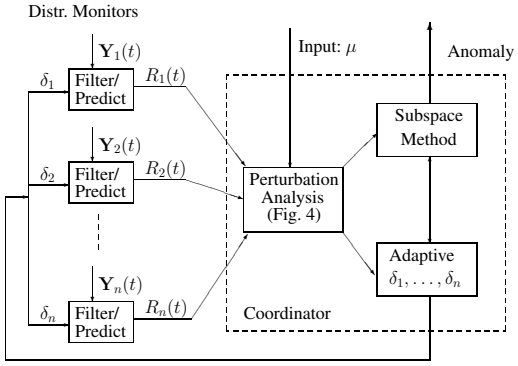


Fig. 2. Distributed detection system.

Symbol	Meaning
M_i	Monitor sites ($i = 1, \ldots, n$)
$\mathbf{Y}_i(t), \hat{\mathbf{Y}}_i(t)$	Data at monitor i and its approx. at the coordinator
$\mathbf{Y}, \hat{\mathbf{Y}}$	Data matrix at monitors and its approx. at the coordinator
$\mathbf{A}, \hat{\mathbf{A}}$	Cov. matrix at monitors and its approx. at the coordinator
$\lambda_i, \hat{\lambda}_i, \bar{\lambda}$	Eigenvalues of $\mathbf{A}$, $\hat{\mathbf{A}}$, and $\bar{\lambda} = \sum \hat{\lambda}_i / n$
$R_i(t)$	Most recent prediction model for $\mathbf{Y}_i(t)$
$\mathbf{W}_i(t), \mathbf{W}$	Filtering error time-series and matrix, $\mathbf{W} = \mathbf{Y} - \hat{\mathbf{Y}}$
$\mathbf{y}, \hat{\mathbf{y}}$	One time-step data from all monitors (one row of $\mathbf{Y}, \hat{\mathbf{Y}}$)
$\mathbf{C}_a, \hat{\mathbf{C}}_a$	The projection matrix of residual subspace
$\alpha, \hat{\alpha}$	False alarm (*i.e.*, false positive) probability
$Q_\alpha, \hat{Q}_\alpha$	The detection threshold
ϵ, ϵ^*	Tolerable and actual aggregate eigen-error
μ	Tolerable deviation of false alarm probability
δ, δ_i	Local monitor slack parameters

TABLE I
Notation.

to do the appropriate filtering. We allow the user (network operator) to input the tolerable *deviation μ of the false alarm probability*, a parameter that specifies how much the false alarm probability achieved by our approximation technique is permitted to deviate from the false alarm probability achieved by the centralized-data solution. We provide an algorithm for computing each monitor's filtering parameter that guarantees that our false alarm probability does not deviate by more than the specified deviation μ. In order to guarantee an error performance within μ, we need to track and limit the perturbations in the system caused by the local filtering at the monitors. This amounts to bounding the perturbations of the eigenvalues λ_i, projection matrix $\mathbf{C}_a$ and trigger threshold Q_α (all of which get perturbed due to error propagation that occurs with out-dated measurement data). In this paper, we show that all of these system component errors can be bounded, and thus excellent detection can still be achieved, even with a substantial reduction in data transmitted to the coordinator.

Our filtering parameters are both heterogeneous (across different monitors) and adaptive in time. Intuitively, the selection of the filtering parameter at a monitor should take into account two things: the variability of the local time series data itself, and the marginal impact this particular data has on the global trigger condition relative to other data streams. We thus aim to do filtering locally at monitors using parameters that are derived based upon global correlations across different data streams and their joint impact on the trigger condition being tracked.

III. Our Approach

The architecture of our system is depicted in Fig. 2. Our approach consists of two parts: (1) the monitors process their collected data by applying local filtering to suppress unnecessary message updates to the coordinator; and (2) the coordinator makes global decisions and provides feedback to the monitors (e.g., local filter parameter settings) based on the observed updates.

As mentioned earlier, $\mathbf{Y}_i(t)$ denotes the actual time series observed at monitoring node M_i, which is one column vector of data matrix $\mathbf{Y}$. We use $R_i(t)$ to denote the approximate representation of $\mathbf{Y}_i(t)$ that is sent to the coordinator. If no further data is sent shortly after time t, the coordinator assumes that $R_i(t)$ serves as a *prediction* of the true data at these latter time instances. A simple prediction model might set $R_i(t)$ to the latest $\mathbf{Y}_i(t)$ value communicated from the site, or an average of recent communications, but more sophisticated prediction models [4], [10] can be used. Our techniques remain applicable regardless of prediction-model specifics.

The coordinator has two principal tasks: (1) to carry out anomaly detection, based on the PCA subspace method, using the inputs $R_i(t)$ it receives, and (2) to compute the filtering parameters δ_i for each monitor. The inputs to the coordinator are the deviation of false alarm probability μ, and the filtered time series. The outputs of the coordinator are a trigger that is fired whenever the condition in Eqn (1) is true, and the filtering parameters δ_i, that are sent to the monitors whenever they change. We will informally call the filtering parameter at a node the "slack" for that node. The monitors use slacks when tracking the drift between the actual time series signal and the prediction function; whenever this drift exceeds the allowed slack, the monitor sends the coordinator an updated prediction, $R_i(t)$. Intuitively, these slacks are used to upper bound the difference between the coordinator's view of the data and the actual data.

The Local Monitor Protocol. Given a slack parameter δ_i, the protocol that runs at each monitor site M_i is fairly straightforward. Let $R_i(t)$ be the most recent prediction model for $\mathbf{Y}_i(t)$ sent to the coordinator. At any time t, monitor M_i continuously tracks the deviation of $\mathbf{Y}_i(t)$ from its prediction $R_i(t)$ as $\mathbf{W}_i(t) = \mathbf{Y}_i(t) - R_i(t)$, and checks the condition $|\mathbf{W}_i(t)| \leq \delta_i$. Whenever $|\mathbf{W}_i(t)| > \delta_i$, the monitor sends an update message to the coordinator that includes $\mathbf{Y}_i(t)$ and an up-to-date prediction $R_i(t)$, and resets $\mathbf{W}_i(t)$ to zero. (Table I summarizes our notation.)

The Coordinator Protocol. The coordinator maintains a perturbed version $\hat{\mathbf{Y}}$ of the accurate global data matrix $\mathbf{Y}$. The connection between the slacks and the coordinator's detection scheme comes from the following. The PCA at the coordinator is performed on a perturbed version of the covariance matrix, $\hat{\mathbf{A}} := \frac{1}{m}\hat{\mathbf{Y}}^T\hat{\mathbf{Y}} = \mathbf{A} + \mathbf{\Delta}$. The magnitude of the perturbation matrix $\mathbf{\Delta}$ is determined by the slack parameters δ_i ($i = 1, \ldots, n$). We can thus bound the perturbation of the covariance matrix through the control of the slack parameters.

Procedure Monitor(i, δ_i)
Input: Monitor index i, local slack parameter δ_i.
1. **while** (**true**) **do**
2. t := current time
3. $\mathbf{W}_i(t) := \mathbf{Y}_i(t) - R_i(t)$
4. **if** ($|\mathbf{W}_i(t)| > \delta_i$) **then**
5. Send update message $(i, \mathbf{Y}_i(t), R_i(t))$ to coordinator
6. Set $\mathbf{W}_i(t) := 0$
7. **if** (new slack δ_i^* is received from coordinator) **then**
8. Set $\delta_i := \delta_i^*$

Procedure Coordinator(μ)
Input: Deviation μ on false alarm probability.
1. **while** (**true**) **do**
2. Make a new row of data $\hat{\mathbf{y}} = [\ R_1(t) \quad \dots \quad R_n(t)\ ]$
3. Replace the oldest row of $\hat{\mathbf{Y}}$ using $\hat{\mathbf{y}}$, pointed to by $\hat{\mathbf{Y}}_i(t)$
4. **for each** (monitor update $(i, \mathbf{Y}_i(t), R_i^*(t))$ received) **do**
5. Set local prediction $R_i(t) := R_i^*(t)$
6. Set $\hat{\mathbf{Y}}_i(t) := \mathbf{Y}_i(t)$
7. Re-compute PCA on $\hat{\mathbf{Y}}$
8. Re-compute threshold $\hat{Q}_\alpha$, matrix $\hat{\mathbf{C}}_a$ and residual $\|\hat{\mathbf{C}}_a\hat{\mathbf{y}}\|^2$
9. **if** ($\|\hat{\mathbf{C}}_a\hat{\mathbf{y}}\|^2 > \hat{Q}_\alpha$) **then** **fire**("anomaly");
10. Compute new optimal settings for local slacks $\{\delta_i\}$ based on μ and maintained statistics *(Sec. IV)*
11. **if** (adaptive allocation) **then** **disseminate**($\{\delta_i\}$)

Fig. 3. Procedures for (a) local monitor update processing, and (b) distributed detection at the coordinator.

The coordinator protocol works as follows. Each time t, if a new input arrives at the coordinator from some or all of the monitors, it carries out the following steps:

1) Makes a new row of data $\hat{\mathbf{y}}$ as $\hat{\mathbf{y}} = [\ \hat{\mathbf{Y}}_1(t) \quad \hat{\mathbf{Y}}_2(t) \quad \dots \quad \hat{\mathbf{Y}}_n(t)\ ]$, where $\hat{\mathbf{Y}}_i(t)$ is defined as either the update received from monitor i (if one exists), or the corresponding prediction $R_i(t)$ otherwise.
2) Updates its view of the global data $\hat{\mathbf{Y}}$, by replacing the oldest row of $\hat{\mathbf{Y}}$ using $\hat{\mathbf{y}}$.
3) Re-computes PCA on $\hat{\mathbf{Y}}$, the residual projection matrix $\hat{\mathbf{C}}_a$, and the trigger threshold $\hat{Q}_\alpha$.
4) Performs anomaly detection using $\hat{\mathbf{C}}_a, \hat{Q}_\alpha$ and $\hat{\mathbf{y}}$; fires an alarm if $\|\hat{\mathbf{C}}_a\hat{\mathbf{y}}\|^2 > \hat{Q}_\alpha$.

The coordinator can recompute the monitor slacks either periodically or upon each monitor update. The coordinator only sends new slacks to the monitors if there is a substantial change. Due to lack of space, we do not expand on this issue herein. A high-level pseudo-code description of both the local-monitor and coordinator protocols is depicted in Fig. 3.

IV. Algorithm for Filtering Parameter Selection

We now describe our method for determining the parameters used for filtering δ_i (also called *slacks*) by the local monitors. Let α denote the false alarm probability that is guaranteed by the Q_α-statistic condition in Eqn (1) in the original push-all solution. Similarly, $\hat{\alpha}$ denotes the false alarm probability of our approximation algorithm. The false alarm deviation, μ, specifies the extent to which the $\hat{\alpha}$ is allow to increase compared to α. In particular, our goal is to determine δ_i values such that the false alarm probability $\hat{\alpha}$ of our technique satisfies $\hat{\alpha} - \alpha < \mu$, while minimizing communication cost

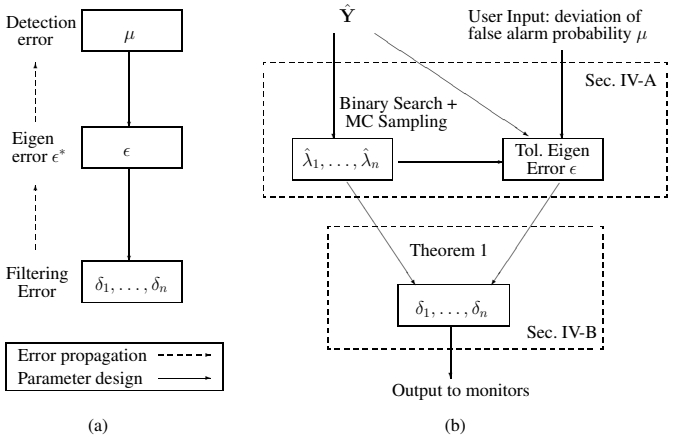


Fig. 4. Perturbation analysis: from deviation of false alarm to monitor slacks.

on the network[1]. To determine the δ_i values minimizing communication for a given μ, we need to be able to quantify the effects of local monitor filtering on the observed false alarm probability.

We remind the reader that because the monitors filter their data and thus often do not send updates to the coordinator, the coordinator's matrix of the global data can have elements that are out-of-date. This perturbed view of the data propagates errors forward through a PCA-based detector as follows. First there will be errors in the computation of the eigenvalues of the covariance matrix, and second there will be errors introduced into the projection matrix (projecting onto the anomalous subspace) as well as to the Q_α threshold. The errors in these last two terms, cause errors in the trigger condition, thus causing errors in the detection accuracy. The direction of error propagation is depicted in (Fig. 4(a)) via the dashed lines.

Because the goal of our algorithm is to take the tolerable deviation μ of false alarm probability as input, and produce the δ_i parameters as output, we need a model of error propagation in the inverse direction to which it naturally flows. This turns out to be a non-trivial task due to the complex dependencies across different parameters in our monitoring framework. The errors in the eigenvalues are critical in our methodology as they impact all parts of the PCA-based detector. We thus elect to control the errors introduced into the eigenvalues. Let λ_i and $\hat{\lambda}_i$ ($i = 1, \dots, n$) denote the eigenvalues of the covariance matrix $\mathbf{A} = \frac{1}{m}\mathbf{Y}^T\mathbf{Y}$, and its perturbed version $\hat{\mathbf{A}} = \frac{1}{m}\hat{\mathbf{Y}}^T\hat{\mathbf{Y}}$, respectively. We use the *L_2 aggregate eigen-error* ϵ^*, defined formally as $\epsilon^* := \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{\lambda}_i - \lambda_i)^2}$, as a metric of the errors across the set of eigenvalues. By limiting this quantity, we can limit error propagation. Our approach thus consists of a two step method: 1) given a false alarm deviation bound μ, determine an upper bound on eigen-error ϵ^*; 2) then for a given eigen-error ϵ^*, find monitor slacks δ_i such that eigen-error does not exceed its bound.

[1]Even though condition (1) is only a one-sided test, our experimental results demonstrate that our methods achieve very small missed-detection rates, similar to [13].

A. Step 1: From false alarm deviation to eigen-error

Unfortunately, there is no closed-form solution for determining the tolerable eigen-error ϵ given a desired bound on the false alarm probability $\hat{\alpha}$. As mentioned earlier, errors in the eigenvalues propagate through $\|\hat{\mathbf{C}}_a\hat{\mathbf{y}}\|^2$ and the threshold $\hat{Q}_\alpha$, thus affecting the trigger condition $\|\hat{\mathbf{C}}_a\hat{\mathbf{y}}\|^2 > \hat{Q}_\alpha$, which determines the false alarm deviation μ.

From our observations, μ is typically monotonically increasing in ϵ; this matches our intuition as larger perturbations to eigenvalues naturally imply higher false alarm probabilities. Thus, given an efficient method for computing μ for a given tolerable eigen-error ϵ, our strategy is to determine ϵ for a given μ using a binary search strategy. Our search starts with an initial guess for a tolerable ϵ, and then computes our estimate for the resulting μ^*. If this is too far from our target μ, then a standard binary search procedure can be used to iteratively find a better ϵ value that satisfies our requirements on μ. A pseudo-code description of our method for estimating the eigen-error ϵ corresponding to a desired μ is given in Fig. 5. Thus, in what follows, we focus on estimating μ for a particular ϵ. Our analysis relies on considering the following random variables:

$$X = \frac{\phi_1[(\|\mathbf{C}_a\mathbf{y}\|^2/\phi_1)^{h_0} - 1 - \phi_2 h_0(h_0-1)/\phi_1^2]}{\sqrt{2\phi_2 h_0^2}} \qquad (2)$$

where $h_0 = 1 - \frac{2\phi_1\phi_3}{3\phi_2^2}$, $\phi_p = \sum_{j=k+1}^{n} \lambda_j^p$ for $p = 1, 2, 3$. The X random variable essentially normalizes the random quantity $\|\mathbf{C}_a\mathbf{y}\|^2$ and is known to approximately follow a standard normal distribution [11]. To perform detection on $\|\mathbf{C}_a\mathbf{y}\|^2$ with false alarm α, the threshold Q_α can be determined as a high-order complex function of $\lambda_{k+1}, \ldots, \lambda_n$ [9]. Based on (2), we can express the false alarm probability (of the original PCA-based detector) as

$$\mathbf{Pr}\left[\|\mathbf{C}_a\mathbf{y}\|^2 > Q_\alpha\right] = \mathbf{Pr}\left[X > c_\alpha\right] = \alpha,$$

where c_α denotes the $(1-\alpha)$-percentile of a standard normal distribution.

In our approximation setting, the normalized quantity of $\|\hat{\mathbf{C}}_a\hat{\mathbf{y}}\|^2$ is denoted by $\hat{X}$ rather than X. Let η_X denote an upper bound on $|\hat{X} - X|$. Then, the deviation of the false alarm probability in our approximate detection scheme can be estimated as

$$\mu = \mathbf{Pr}\left[c_\alpha - \eta_X < N(0,1) < c_\alpha\right] \qquad (3)$$

where $N(0,1)$ denotes a standard normal random variable. A key issue here is how to estimate the η_X upper bound on $|\hat{X} - X|$. Our approach is to use a *Monte Carlo (MC) sampling* technique to obtain observations of the $|\hat{X} - X|$ random variable, and use the maximum of these observations as an estimate of η_X. (Due to space constraints, the details can be found in [7].)

B. Step 2: From tolerable eigen-error to monitor slacks

Let $\mathbf{W}$ denote the error matrix that arises due to filtering; in other words $\hat{\mathbf{Y}} = \mathbf{Y} - \mathbf{W}$. Due to our filtering methods, all the elements of the column vector $\mathbf{W}_i$ are bounded within the interval $[-\delta_i, \delta_i]$. To keep the analysis tractable, we make the following assumptions (both fairly standard in SMP theory) on this filtering error matrix $\mathbf{W}$:

1) The column vectors $\mathbf{W}_1, \ldots, \mathbf{W}_n$ are independent and radially symmetric m-dimensional random vectors, i.e., their projections on a sphere is uniformly distributed.
2) For each $i = 1, \ldots, n$, all elements of column vector $\mathbf{W}_i$ are i.i.d. random variables with mean 0 and variance σ_i^2.

Note that the independence assumption is on the single-monitor errors only – this by no means implies that the signals received by different monitors are statistically independent. The *error* variance $\sigma_i^2 := \sigma_i^2(\delta_i)$ is a function of the corresponding monitor slack because the slack determines the size of the allowed drift (or discrepancy), between the true data and the coordinator's view of the data, before the coordinator needs an update.

Let $\bar{\lambda} := \frac{1}{n}\sum \hat{\lambda}_i$ denote the average of the perturbed eigenvalues of $\hat{\mathbf{A}}$. Based on the statistical analysis on the Frobenius norm of $\mathbf{\Delta}$, we can prove the following theorem relating monitor slacks δ_i to an upper bound of the aggregate eigen-error ϵ^*.

Theorem 1 *Under the above assumptions on filtering errors, setting $\delta_1, \ldots, \delta_n$ to satisfy*

$$2\sqrt{\frac{\bar{\lambda}}{m}\cdot\sum_{i=1}^{n}\sigma_i^2} + \sqrt{\left(\frac{1}{m}+\frac{1}{n}\right)\sum_{i=1}^{n}\sigma_i^4} = \epsilon \qquad (4)$$

guarantees that $\epsilon^* \le \epsilon$ with probability $\ge 1 - o(\frac{1}{m^3})$.

(We refer to ϵ as *tolerable eigen-error* in what follows.) [7] contains the proof of this theorem as well as similar results for the eigen-subspace $\mathbf{C}_a$ and individual eigenvalues. Given a tolerable eigen-error ϵ as input, we can then solve for the slacks δ_i using the equation in Theorem 1. However to do so, we need to quantify the relationship between error variances σ_i and local slacks δ_i. We now discuss different techniques employed in our system for this purpose.

Homogeneous Slack Allocation: Uniform Distribution Method. A simple method that often works well in practice is to assume that filtering errors are independently and uniformly distributed in $[-\delta_i, \delta_i]$. This gives a closed form for local

---

Procedure FalseAlarmToEigenError(μ, err)
Input: Deviation μ of false alarm probability; desired approximation factor (err) for eigen-error ϵ.

```
1. ε_l := 0.0;  ε_u := λ̄          // search range for ε
2. while ( (ε_u − ε_l) > err · ε_l ) do
3.     ε := 0.5 · (ε_l + ε_u)
4.     η_X := MonteCarloSampling(ε)
5.     μ* := Pr[c_α − η_X < N(0,1) < c_α]
6.     if (μ* > μ) then ε_u := ε else ε_l := ε
7. return(ε)
```

Fig. 5. Procedure for estimating eigen-error given a false alarm probability deviation μ using binary search.

variances: $\sigma_i = \frac{\delta_i^2}{3}$. Assuming homogeneous slack allocation, that is, all monitors share the same slack $\delta_i = \delta$, we can directly solve Eqn. (4) for δ:

$$\delta = \frac{\sqrt{3\bar{\lambda}n + 3\epsilon\sqrt{m^2 + m \cdot n}} - \sqrt{3\bar{\lambda}n}}{\sqrt{m+n}}$$

Homogeneous Slack Allocation: Local Variance Estimation Method. In some cases, the uniform-distribution assumption for filtering errors may be unrealistic. A more accurate method is to estimate local error variances $\sigma_i(\delta)$ directly from the data. Variance estimation is performed locally (at each monitor) by fitting a (e.g., quadratic) function of δ using a recent window of observations. These local functions are sent to the coordinator (either periodically or on-demand), and plugged into Eqn. (4) to solve for a new δ. While imposing some additional overhead on the network and local monitors, this method avoids possibly unrealistic uniformity assumptions on the monitor data.

Heterogeneous Slack Allocation. We now consider allowing the local slacks $\delta_1, \ldots, \delta_n$ to differ from one another, and to dynamically adapt to local stream characteristics. Let the message update frequency (a direct measure of communication cost) of each monitor M_i be a function $f_i(\delta_i)$. Then, assuming each slack takes on a random value uniformly in the range $[-\delta_i, \delta_i]$, we can formalize heterogeneous slack allocation as the following optimization problem:

$$\text{Minimize} \quad \sum_i^n f_i(\delta_i) \quad \text{such that} \quad 2\sqrt{\frac{\bar{\lambda}}{m} \cdot \sum_{i=1}^n \frac{\delta_i^2}{3}} = \epsilon,$$

where the second summand in Eqn. (4) is ignored, since it is typically an order of magnitude smaller than the first. We used the method in [17], based on Lagrangian multipliers to solve for the optimal slack allotments. Although heterogeneous slack values are intuitively appealing, they often bring marginal benefit over homogeneous allocations. We will see this to be the case in our evaluations as well.

V. Evaluation

A. Evaluation Methodology and Metrics

We implemented our system and developed a trace-driven simulator to validate our methods. The real world traffic, used as input to our simulator, comes from the Abilene network. We used four one-week traces of router-to-router origin-destination (OD) traffic matrices. The traces contain OD-flow traffic loads measured every 10 minutes, for all 121 flows of the Abilene network, from which we can compute the per-link traffic loads for all 41 links, using its provided routing matrix. With a time unit of 10 minutes, data was collected for 1008 time units for each week.

To evaluate the detection accuracy of our approach, we synthetically injected 60 anomalies and 60 non-malicious bursts[2] into the dataset using the method described in [13], so that we would have sufficient anomaly data to compute error rates. We used a threshold Q_α corresponding to an $1-\alpha = 99.5\%$ confidence level. In the detection process, when any anomaly is missed, we count it as a missed detection; when any non-malicious burst is detected, we count it as a false alarm. To make the results intuitive, we define the *false alarm rate* as the fraction of false alarms over the total number of injected bursts, which is α (defined in Sec. III) re-scaled as a rate rather than a probability. We define *missed detection rate* as the fraction of missed detections over the total number of injected anomalies.

[2]In [13] the authors use the term "small anomaly" to refer to events that should be ignored (not flagged) and whose detection counts as false alarms. While we use their same method for synthetic anomalies, we change the terminology to be more intuitive.

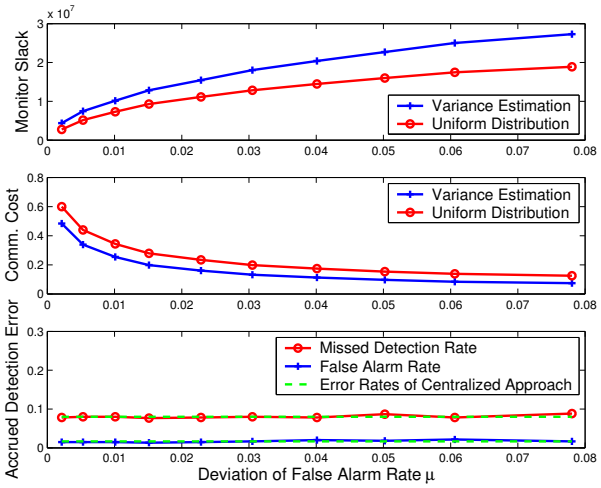


Fig. 6. Monitor slacks, communication cost and accrued detection. The dashed line is the detection error of centralized approach with complete data.

In order to evaluate the scalability of our method, we had to generate synthetic traffic matrices because no traffic matrix datasets with thousands of links and tens of thousands of OD flows exist. We used the BRITE topology generator [15] to generate both sample topologies and their associated routing matrices. We considered a number of networks with anywhere from 100 to 1000 links, and up to 500×500 pairs of OD flows. For each of the $250,000$ OD flows, we generate four weeks of data based on the method discussed in [16], by extracting the relevant statistics (e.g., mean distribution, noise level, etc.) from the Abilene network traffic matrices.

We compute the communication cost as follows. Let num be the number of messages exchanged between monitors and the coordinator, including both the signal updates from monitors to coordinator as well as the slack updates from the coordinator to the monitors. Let n be the number of monitors and m the number of values in each monitor's time series. Then communication cost is calculated as $num/(n \cdot m)$ which gives the per-monitor communication cost.

B. Detection accuracy vs. communication cost

We now evaluate the performance and tradeoffs of our protocols and algorithm for computing the monitor slacks. We implemented both methods of homogeneous allocation for computing the monitor slack δ: the closed-form solution

relying on uniform assumptions and the variance measurement solution.

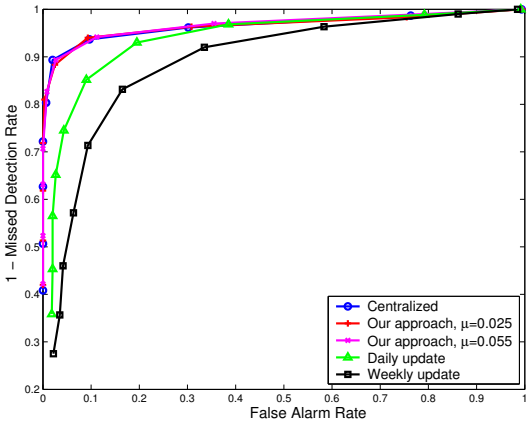
In Fig. 6 we consider a whole range of possible inputs on the tolerable false alarm rate deviation μ (the probability Eqn (3) is re-scaled to a rate). We show in the top plot the relationship between μ and the filtering slack δ, in the middle plot the relationship between μ and communication cost. These results make intuitive sense. As we allow more error tolerance μ, we can use greater slack and filter out more data at the monitors, and consequently reduce the amount of data sent to the coordinator over the network. For example, when the tolerable deviation of false alarm is 5%, our algorithm reduces the data sent through the network by more than 90% when using the variance estimation method.

The bottom plot shows the actual accrued detection errors. The curve with circles depicts the missed detection rate; the curve with pluses depicts the false alarm rate; the dashed lines depict the corresponding detection errors of the centralized approach. First we point out that in all cases, the actual false alarm rate with our protocols is always smaller than the guaranteed bound. In other words, although we may input that we can tolerate an additional μ = 5% errors, in fact we don't have to incur this reduced performance, because the lower plot illustrates that our method performs nearly identically to the original subspace method in terms of false alarms and missed detections. Moreover, this nearly identical error performance can be achieved with far less data; values such as 80% or 90% less data (depending upon the particular value of μ) are typical. These results show, that for our dataset, the reduction in communication costs can be enormous whereas the tradeoff in terms of increased detection error is very small. These promising results confirm our hypothesis that it is not necessary to back-haul all the data for an anomaly detection such as [13].

In comparing the variance estimation and the uniform distribution methods for slack estimation, we see that the measurement-based variance estimation method always performs better. The absolute difference in communication cost varied from 5% to 10% for tight requirements on μ (with μ=0.006 to 0.08, respectively). The advantage of the closed-form method is its simplicity and low computational overhead. Since, for this dataset, its performance is quite close to the measurement-based method, we conclude that such solutions might be "good enough" for many datasets.

We now compare our method and the original subspace method using an ROC curve [20]. The y-axis plots the true positives (one minus the missed detections) and the x-axis depicts the false alarms. ROC curves allow one to compare two methods over a range of detection thresholds; each point on each curve corresponds to a different cutoff threshold for signaling an alarm. In general, if one curve lies entirely above and to the left of another [20], then that method is superior in that it handles the tradeoff between missed detections and false alarms better.

Because in [13] they do not indicate how often they update their PCA transform, we tested 3 variants of their method.



Approach	Communication Cost
Centralized, daily and weekly update	1.000
Our approach, $\mu = 0.025$	0.159
Our approach, $\mu = 0.055$	0.097

Fig. 7. ROC curve: benefit and cost of data update approaches.

The "centralized" version updates the principal components each time interval (upon the arrival of new data). The "daily update" version updates the principal components once a day (based on the previous 24 hours); the "weekly update" version updates the components once a week (based on the previous week). The results are shown in Fig. 7.

We can see from the plot that the ROC curve and detection accuracy of our approximation technique (either $\mu = 0.015$ or $\mu = 0.045$) are extremely close to that of the centralized approach. It is surprising, that using only 10% to 20% of the data, our technique has a detection ability that is essentially as good as the fully centralized approach.

This figure also indicates that it is important to keep the principal components up to date because the performance drop-off is considerable for either the daily or weekly data update cases. We point out that in our technique, the recomputation of the principle components is done less frequently than in the original algorithm (we refer here to the version in which the PCA transform is updated every time interval). This is because in any time interval (e.g., 5 minutes in this example), if none of our monitors send anything to the coordinator, then the principle components are not recomputed. There may be additional ways to reduce this computation overhead such as checking the norm of the covariance matrix and only doing updates if the change to this norm "is large enough". We leave that for future work.

We also implemented our heterogeneous slack allocation and compared its performance to that of the homogeneous slack allocation. We found that the performance did not differ greatly (at most 3% in terms of communication costs) between the two. This indicates that the simpler solution may be good enough for the data type we consider. However the benefits of having the more general solution using heterogeneous slacks would need to be evaluated for each data type and application.

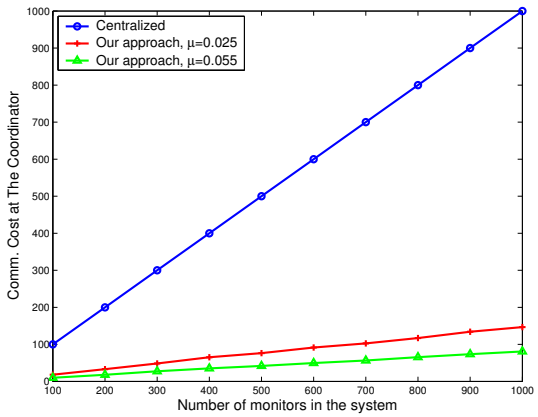


Fig. 8. System Scalability.

C. System Scalability

We now examine our system's scalability as the number of distributed monitors grows. Recall that one of the key reasons for controlling the communications cost is to avoid overwhelming the coordinator should it receive lots of data from many monitors. The communications cost metric we have been using until now (namely $num/n \cdot m$) is an average value for the cost *per monitor*. The communication cost coming into the coordinator is the sum of costs of all monitors, which is can be computed from num/m. This captures the average number of messages the coordinator receives in one time slot.

We plot the communications cost at the coordinator as a function of the number of monitors in Fig. 8. We varied the number of monitors from 100 to 1000, and used tolerable deviation of false alarm rate $\mu = 0.025$ and $\mu = 0.055$. For each system size n, we run 5 rounds of experiments, each of which runs on n randomly picked monitors. In the Figure, in our approach, as the system size increases: 1) the communication cost of each monitor roughly keeps constant (which is the slope of the line); and 2) the communication cost at coordinator increases linearly with system size with the slope roughly being 0.150 ($\mu = 0.025$) and 0.088 ($\mu = 0.055$), which are far less than 1.0, the slope of the centralized approach. This result indicates that the communication cost increases slowly as system size increases, and that our system thus scales gracefully.

VI. Conclusion

In this paper we extended the PCA-based anomaly detection method using ideas from "in-network" processing (to engage local monitors to filter based on global conditions) combined with ideas from stochastic matrix perturbation theory. Perturbation theory is used to derive bounds on the terms in the anomaly detector that are affected by error propagation when limited data is used. We designed an algorithm to select filtering parameters so that that monitors only send data to the central tracking site "when necessary". The necessity is determined from individual traffic behaviors, correlations across traffic streams and the global trigger tracking condition. We show that anomaly detection can still be done very accurately even when 80 or 90% of the original data is never sent to the coordinator. Thus the tradeoff between detection accuracy and communication savings is very lopsided - there is a huge reduction in communication overhead accompanied by a very small increase in errors. Moreover we illustrated that this data reduction leads to a system that scales gracefully as the number of monitors grows. In particular, we showed that the coordinator's input data rate grows more than an order of magnitude more slowly than a system that back-hauls all monitoring data for volume anomaly detection.

References

[1] Barford, P., Kline, J., Plonka, D., and Ron. A. A signal analysis of network traffic anomalies. In *IMW* (2002).

[2] Brand, M. Fast low-rank modifications of the thin singular value decomposition. In *Linear Algebra and Its Applications*, 415(1) (2006)

[3] Brutlag, J. Aberrant Behavior Detection in Timeseries for Network Monitoring. In *LISA* (2000).

[4] Cormode, G., and Garofalakis, M. Sketching streams through the net: Distributed approximate query tracking. In *VLDB* (2005).

[5] Dilman, M., and Raz, D. Efficient reactive monitoring. In *IEEE INFOCOM* (2001).

[6] Huang, L., Nguyen, X. L., Garofalakis, M., Jordan, M., Joseph, A.D., and Taft, N. In-network PCA and anomaly detection. In *NIPS* (2006).

[7] Huang, L., Nguyen, X. L., Garofalakis, M., Hellerstein, J.M., Jordan, M., Joseph, A.D., and Taft, N. Communication-efficient online detection of network-wide anomalies. UCB Technical Report, August 2006.

[8] Huebsch, R., and et al. Querying the internet with pier. In *VLDB* (2003).

[9] Jackson, J. E. and Mudholkar, G. S. Control procedures for residuals associated with principal component analysis. In *Technometrics*, 21(3) (1979).

[10] Jain, A., Chang, E. Y., and Wang, Y.-F. Adaptive stream resource management using kalman filters. In *ACM SIGMOD* (2004).

[11] Jensen, D. R. and Solomon, H. A Gaussian approximation for the distribution of definite quadratic forms. In *J. Amer. Stat. Assoc.*, 67:898-902 (1972).

[12] Keralapura, R., Cormode, G., and Ramamirtham, J. Communication-efficient distributed monitoring of thresholded counts. In *ACM SIGMOD* (2006).

[13] Lakhina, A., Crovella, M., and Diot, C. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM* (2004).

[14] Lakhina, A., Papagiannaki, K., Crovella, M., Diot, C., Kolaczyk, E. D. and Taft, N. Structural analysis of network traffic flows. In *ACM SIGMETRICS* (2004).

[15] Medina, A., Lakhina, A., Matta, I. and Byers, J. BRITE: an approach to universal topology generation. In *MASCOTS* (2001).

[16] Nucci, A., Sridharan, A. and Taft, N. The problem of synthetically generating IP traffic matrices: initial recommendations. In *ACM CCR* (2005)

[17] Olston, C., Jiang, J., and Widom, J. Adaptive filters for continuous queries over distributed data streams. In *ACM SIGMOD* (2003).

[18] Padmanabhan, V. N., Ramabhadran, S., and Padhye, J. Netprofiler: Profiling wide-area networks using peer cooperation. In *IPTPS* (2005).

[19] Sharfman, I., Schuster, A. and Keren, D. A geometric approach to monitoring threshold functions over distributed data streams In *ACM SIGMOD* (2006).

[20] Soule, A., Salamatian, K., and Taft, N. Combining filtering and statistical methods for anomaly detection. In *IMC* (2005).

[21] Spring, N., Wetherall, D., and Anderson, T. Scriptroute: A facility for distributed internet measurement. In *USITS* (2003).

[22] Stewart, G. W., and Sun, J.-G. *Matrix Perturbation Theory*. Academic Press, 1990.

[23] Xie, Y., Kim, H.-A., O'Hallaron, D. R., Reiter, M. K., and Zhang, H. Seurat: A pointillist approach to anomaly detection. In *RAID* (2004).

[24] Yegneswaran, V., Barford, P., and Jha, S. Global intrusion detection in the domino overlay system. In *NDSS* (2004).

[25] Zhang, Y., Ge, Z.-H., Greenberg, A., and Roughan, M. Network anomography. In *IMC* (2005).