Toward Sophisticated Detection With Distributed Triggers

Ling Huang* Minos Garofalakis† Joseph Hellerstein* Anthony Joseph* Nina Taft†
*UC Berkeley †Intel Research Berkeley
{hling, hellerstein, adj}@cs.berkeley.edu {minos.garofalakis, nina.taft}@intel.com

ABSTRACT

Recent research has proposed efficient protocols for distributed triggers, which can be used in monitoring infrastructures to maintain system-wide invariants and detect abnormal events with minimal communication overhead. To date, however, the constraints used to flag abnormality have been limited to simple aggregation functions like sums and counts. In this paper we show how distributed triggers can be extended so as to support sophisticated detection functions. We first consider an extension for constraints that involve quadratic aggregation functions. To do so, we select a particular application, that of anomaly detection, in which a centralized algorithm uses quadratic constraints to detect anomalies. We use this application to provide a detailed illustration of how to extend our triggers. This example also shows the utility of the distributed triggering system that essentially converts a centralized solution into a distributed one, and leads to a more communication efficient system. We show that this can be done with little to no sacrifice in detection accuracy. After the detailed illustration of trigger extensions, we discuss approaches to build more general extensions to support a broad range of complex constraints.

Keywords

Distributed Triggers, Anomaly Detection, PCA

1. INTRODUCTION

Distributed monitoring and anomaly detection systems have been proposed and deployed to aggregate status information and detect unusual events in large distributed systems such as server clusters and large networks [4, 13]. In these systems monitoring sensors are deployed throughout the network to collect system information from multiple vantage points. With the coordination of an operation center, monitors collaborate with each other to analyze and correlate distributed data for timely detection of system-wide abnormal events.

Distributed triggers have been proposed as a critical component in monitoring architectures [5]. One of the primary goals of a monitoring system is to ensure that the target system is well behaved. When the target system is distributed, we typically test for normal behavior of the global system by monitoring data at the local sites, shipping this data to a central place, and then checking expected behaviors against actual ones using the centralized data. The behaviors can be assessed using a well-defined set of logical predicates. After processing the data, unusual behavior can be revealed (i.e., "triggered") when some feature of the data exceeds a predefined "acceptable" threshold.

A key idea behind *distributed triggers* is to avoid centralizing all the data, and to do so by engaging the local monitors in part of the detection procedure. If we can reduce the amount of monitoring data that actually needs to get shipped to the central decision-making site (called a *coordinator*), then we can reduce the communication needs of such a system. There is ample motivation for reducing the communications needs of such systems. Enterprise networks typically do not over-provision the connections between office sites in different cities or countries. Sensor network applications clearly need to be mindful of power limitations on the sensor devices. Although ISPs today typically over-provision their backbone networks, emerging continuous monitoring applications may require much finer time and data granularities than what is typically collected today (e.g., SNMP polled on 5 minute time scale). To avoid transferring significant traffic volumes throughout a network, and also to avoid overwhelming the operation center, we need *communication efficient distributed triggers*. Naive solutions that continuously "push" the local data streams directly to a collection site simply will not scale to large distributed systems.

To support this functionality, a set of approaches have been proposed recently for communication-efficient tracking of distributed triggers [3, 8]. They aim to detect constraint violations via threshold functions defined on distributed information. However, existing approaches have significant limitations: they only support simple thresholds on distributed aggregate functions like `SUM` and `COUNT`, which are insufficient for sophisticated detection applications. In this paper, we present our initial efforts to support more sophisticated triggering functionality. We give some examples for detection schemes that may require checking non-linear threshold constraints, relative triggers, and so on. To illustrate these ideas more concretely, we use the example of a centralized PCA-based anomaly detection application [9]. This method detects anomalies in traffic volume levels by simultaneously examining the link load levels of all the links in a large network. It works by using a Principal Components Analysis

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

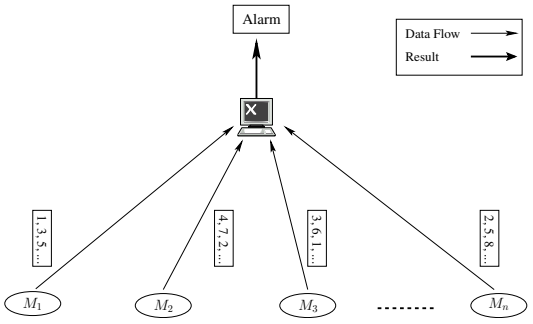


Figure 1: The system setup.

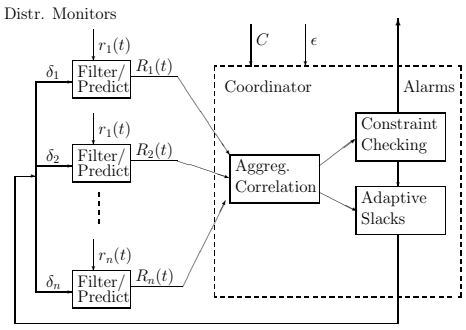


Figure 2: The distributed trigger tracking framework.

(PCA) technique to decompose network traffic into normal and residual components, and then detects anomalies by applying a threshold function on the residual components. One of our contributions is to illustrate how to redesign this application in a distributed fashion, with far less communication overhead, by using distributed triggers as an underlying paradigm. Abstractly, this requires distributed triggers that support complex threshold functions, and can be composed to lend support to sophisticated distributed detection applications. We describe, for example, a simple approximation to a non-linear threshold constraint that works well in our example application: our method achieves roughly the same level of accuracy as the centralized scheme, while communicating only around 20% of the underlying data.

Prior Work. The database community has extensively explored centralized triggering mechanisms [2, 14]. However, the goal of minimizing communication overhead in widely distributed Internet environments introduces new challenges. Keralapura *et al*. [8], formalized the thresholded counting problem and proposed solutions based on either static or adaptive algorithms, as well as a detailed optimality analysis of the solution. Our approach in [3] goes further by defining distributed triggers with general (zero, fixed, or varying) time windows, along with novel algorithms for these variants with firm detection guarantees. Both [3] and [8] solved the problem of detecting threshold violations with specified accuracy while minimizing communication overhead, as well as providing the flexibility for users to trade off communication overhead with detection accuracy. Recent progress in distributed monitoring, profiling and anomaly detection [12, 15, 16] aims to share information and foster collaboration between widely distributed monitoring boxes to offer improvements over isolated systems. These systems are examples of distributed monitoring systems for which a triggering tool such as ours would be useful. Lakhina *et al*. [9], carried out the pioneering work in detecting network-wide anomalies. Zhang *et al*. [17], extended it further and proposed a general "anomography" framework to infer anomalies at network level in both spatial and temporal domains.

2. DISTRIBUTED TRIGGERING SYSTEMS

As shown in Fig. 1, a typical distributed triggering system consists of a set of n widely distributed monitoring nodes M_1, M_2, ..., M_n and a coordinator node X. Each monitor continuously produces time series signals $r_i(t)$ on the variable(s) or condition(s) selected for monitoring. These time series signals are sent to coordinator X which acts as an aggregation and detection point. The purpose of the coordinator is to track conditions across its monitors and to fire a trigger whenever some limitation on the aggregate behavior of a subset of nodes is violated.

We assume all communication happens only between monitoring nodes and the coordinator, and no communication happens among monitoring nodes. To avoid sending all their data, the monitors filter their time series so as to only send update information to the coordinator "when necessary". Because the coordinator has imperfect knowledge of the monitored data, it can make mistakes such as *false alarms* and/or *missed detections*. Our design goal is ensuring the coordinator accurately fires the trigger, while simultaneously minimizing the communication between the monitors and coordinator. The heart of the problem is thus to determine how the monitors should do their filtering.

In [3], we provided a mathematical definition of the distributed triggering problem with three distinct types of constraint violations: *instantaneous*, *fixed-window*, and *varying-window* triggers. *Instantaneous* triggers fire when the value of the aggregate function violates the threshold value within a single time instance. *Fixed-window* and *varying-window* triggers aim to catch persistent threshold violations over a fixed or variable window of time, respectively. With these three types of violations, we allow the notion of a constraint violation to have an associated time period, and thus enable a broad range of triggering conditions. For the purposes of this paper, we will focus on instantaneous triggers only.

Our distributed triggering system has the notion of an *error tolerance*, denoted by ϵ. Let C denote the trigger threshold at the coordinator. Our goal is to track an instantaneous violation *approximately* to within the specified error tolerance ϵ around C. For example, if the aggregation function is a `SUM` function, then the trigger fires for any time instant t where $\sum_i^n r_i(t) > C + \epsilon$, for nodes $i = 1, \ldots, n$.

Our distributed triggering system is illustrated in Fig. 2. The global trigger threshold, C, and error tolerance, ϵ are user specified inputs. The time series data collected at monitor i is given by $r_i(t)$, while $R_i(t)$ denotes the filtered version of this data sent to the coordinator. The idea behind the filtering is to send a summary of $r_i(t)$ at some time t and then not to send anything at all until it is deemed necessary. From the point of view of the coordinator, $R_i(t)$ can be viewed as a prediction of $r_i(t)$ because when the coordinator is not receiving any data from monitor i, it assumes that its most recently received value of $R_i(t)$ accurately predicts $r_i(t)$ [6].

Each monitor M_i continuously tracks the (instantaneous) difference $d_i(t) = |r_i(t) - R_i(t)|$ between the true local signal and its (most recent) prediction. In order to upper bound this drift, monitor i uses a parameter δ_i and checks if $d_i(t) > \delta_i$.

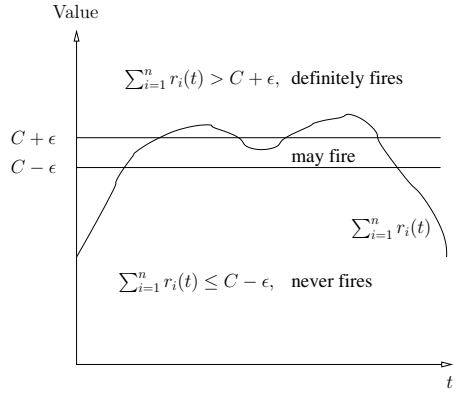


Figure 3: Instantaneous trigger tracking guarantees.

If this occurs at a time t, then M_i updates the coordinator by sending it the latest $r_i(t)$ value. Although in general, $R_i(t)$ can be any prediction function of $r_i(t)$, in our examples herein we simply use $R_i(t) = r_i(t)$. The parameter δ_i is called the *local monitor slack*, and intuitively this parameter indicates that no communication is needed as long as the prediction captures $r_i(t)$ to within a δ_i bound.

The coordinator has two jobs. The first is to continuously track the value of the aggregation function based on the predictions it has received. Using the `SUM` as an aggregation function, this means tracking $\sum R_i(t)$. A violation occurs whenever $\sum_{i=1}^{n} R_i(t) > C + \epsilon$. The coordinator's second job is to continuously estimate a parameter $\Delta(t)$, called the *total slack*, according to $\Delta(t) = \max(\epsilon, C + \epsilon - \sum_{i=1}^{n} R_i(t))$. Intuitively, $\Delta(t)$ captures how far away the system is from firing the trigger. The coordinator partitions this total slack into individual monitor slacks δ_i, such that $\sum_i \delta_i = \Delta$ and each δ_i is proportion to the variance of the local stream. These values are then sent to each of the monitors. When Δ is small, consequently the δ_i's will be small. This is intuitive because this means that when we are close to firing a trigger, the allowed drift between the coordinator's view and the monitor's view must remain small, and thus more updates will be sent to the coordinator so it can fire accurately[1]. Conversely, when the aggregation condition is far from the trigger threshold, little data needs to be sent to the coordinator. This scheme naturally adapts to the time series data.

A highly desirable feature of a distributed triggering system is to provide guarantees on its performance. The following theorem shows that our adaptive scheme indeed guarantees ϵ-approximate instantaneous trigger tracking.

Theorem 1 *Employing an adaptive global monitor slack equal to* $\Delta(t) = C + \epsilon - \sum_{i=1}^{n} R_i(t)$*, where* $R_i(t)$ *denotes the up-to-date prediction from monitor* m_i *(for all* i*) ensures that the coordinator check* $\sum_{i=1}^{n} R_i(t) > C$*: (1)* always fires *if* $\sum_i r_i(t) > C + \epsilon$*; and, (2)* never fires *if* $\sum_i r_i(t) < C - \epsilon$*.*

This theorem was proved independently both in [3] and [8]. Theorem 1 asserts a "band of uncertainty" (of size 2ϵ) around the trigger threshold C, where our tracking algorithm may or may not fire a trigger violation (see Fig. 3). The key observation is that both global and local slacks vary over time, and can be allocated in an *adaptive* manner that maximizes the effect of local filtering, and thus minimizes overall communication. Unlike earlier data-streaming work [6, 11], we are *not* interested in continuously guaranteeing that the coordinator's estimate of the aggregate function is within an ϵ-error. Our focus instead is highly accurate trigger firing: we care about accurate aggregate signal estimation *only if* its value is close to the trigger threshold C. Our adaptive slack allocation schemes exploit the trigger condition to yield significant communication reductions by allowing for much "looser" (and thus, more effective) filters at monitors when the signal is well below the C threshold.

In [3], we designed schemes for adaptively partitioning the global slack into individual monitor slacks and analytically showed that our schemes deterministically guarantee that the trigger fires with ϵ-accuracy. Our method enables users to tradeoff desired detection performance with communication overhead. Testing using real-life data streams from PlanetLab Intrusion Detection System showed our algorithms' significant communication-efficiency gains — a reduction in monitor data sent to the coordinator of more than 80% [3].

Our triggering protocols currently support simple linear functions (e.g., `SUM` and `COUNT`) and not advanced queries such as top-k, histogram, join, etc. In this paper we show how our distributed triggers can be extended so as to support sophisticated detection functions. We first consider an extension to support constraints that involve quadratic aggregation functions. To do so, we select a particular application, that of anomaly detection, in which a centralized algorithm uses quadratic constraints to detect anomalies. We use this application to provide a detailed illustration of how to extend our triggers. This example also shows the utility of the distributed triggering system that essentially converts a centralized solution into a distributed one, and also leads to a more communication efficient system. We show that this can be done with little to no sacrifice in detection accuracy. After the detailed illustration of trigger extensions, we discuss them more generally in Section 4.

3. EXTENDED TRIGGERS FOR NETWORK-WIDE TRAFFIC ANOMALY DETECTION

In this section, we first summarize a centralized algorithm [9] for doing volume anomaly detection. Network volume anomalies are unusual and significant changes in end-to-end traffic flows typically caused by worms, DoS attacks, device failures, misconfigurations, etc. We then show how this problem can be mapped onto a distributed triggering system and explain the functionality required at our monitors and coordinator to implement this approach.

3.1 Using PCA for Centralized Detection

Detecting anomalies is the first, critical step for network diagnostics, however they are usually hidden in large amounts of high-dimensional, noisy data. Volume anomalies usually propagate through the network and are observable on all links they traverse. Lakhina *et al.* [9] proposed a solution to uncovering such anomalies within a network by examining the traffic on all links inside a network simultaneously. Their approach assumes a protocol such as SNMP is available to collect link counts on every link and ship these statistics to a central network operations center (NOC). Their technique performs PCA on these link traffic measurements, and decom-

[1]Our experience shows that even though the communications traffic increases near the time of violation, the actual amount of traffic is far below a system that does not use filtering and simply pushes all the data to coordinator

poses the high-dimensional space occupied by a set of network traffic measurements into disjoint subspaces corresponding to normal and anomalous network conditions. By performing statistical analysis on traffic signals in anomalous subspaces, they can effectively detect, identify, and quantify network-wide traffic anomalies.

Their method is summarized as follows. Consider a network with n links each of which has a monitor, M_i, where $i = 1, ..., n$, that measures the traffic load every measurement interval. After T measurement intervals, the monitor effectively has a time series of link counts for its link. The times series data from all of the monitors is sent to the NOC where it is assembled in a matrix $\mathbf{Y}$, in which each column i denotes the timeseries measurements of the i-th link and each row t represents an instance of all the links at time t (where $t = 1, ..., T$). We use $\mathbf{y}$ to denote a vector of measurements of all the links from a single timestep, which is an arbitrary row of $\mathbf{Y}$, transposed to a column vector,

$$\mathbf{y} = \begin{bmatrix} r_1 & r_2 & \dots & r_n \end{bmatrix}^T$$

where $r_i = r_i(t)$, link i's value at time t, for $i = 1, \dots, n$.

PCA is a coordinate transformation method that maps a given set of data points onto principal components, which are ordered by the amount of data variance that they capture. Applying PCA to $\mathbf{Y}$ yields a set of n principal components, $\{\mathbf{v}_i\}_{i=1}^n$, which are computed as:

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

As studied in [10], the PCA technique reveals that OD flows of backbone networks have low intrinsic dimensionality. For the Abilene network with 41 links, the vast majority of the variance in each link timeseries can be captured by the first $k = 4$ principal components. This reveals that the underlying OD flows themselves effectively reside in an k-dimensional subspace of $\mathbb{R}^n$, referred to as the *normal* subspace $\mathcal{S}$. The remaining $(n-k)$ principal components constitute the *anomalous* subspace $\tilde{\mathcal{S}}$.

Detecting volume anomalies relies on the decomposition of link traffic $\mathbf{y}$ at any timestep into normal and anomalous components, $\mathbf{y} = \hat{\mathbf{y}} + \tilde{\mathbf{y}}$, such that: a) $\hat{\mathbf{y}}$ corresponds to modeled traffic (the projection of $\mathbf{y}$ onto $\mathcal{S}$); b) $\tilde{\mathbf{y}}$ corresponds to residual traffic (the projection of $\mathbf{y}$ onto $\tilde{\mathcal{S}}$). Mathematically, $\hat{\mathbf{y}}(t)$ and $\tilde{\mathbf{y}}(t)$ can be computed by

$$\hat{\mathbf{y}} = \mathbf{P}\mathbf{P}^T\mathbf{y} = \mathbf{C}\mathbf{y} \quad \textit{and} \quad \tilde{\mathbf{y}} = (\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{y} = \tilde{\mathbf{C}}\mathbf{y}$$

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

In general, a volume anomaly will tend to result in a large change to $\tilde{\mathbf{y}}$. A useful statistic for detecting abnormal changes in $\tilde{\mathbf{y}}$ is the squared prediction error (**SPE**): $\mathbf{SPE} \equiv \|\tilde{\mathbf{y}}\|^2 = \|\tilde{\mathbf{C}}\mathbf{y}\|^2$, which is a quadratic residual function. We may consider network traffic to be abnormal and fire an alarm if

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

where δ_α^2 denotes the threshold for the **SPE** at the $1 - \alpha$ confidence level. This trigger condition comes from the *Q-statistic*, (derived in [7]), that is a well known statistical test and is applied here to the residual vector. It can be computed by the principle eigenvalues.

3.2 Distributed Detection

Shifting from a centralized to a distributed anomaly detection algorithm raises some important issues. First, we want to understand which functionality can be pushed from the coordinator to the monitors, so as to engage them more actively in the detection process. In our system, this will involve "smart" filtering. The second issue is to ensure that even in this distributed system, and with a discrepancy between its view and the true network state, the coordinator can still fire the trigger on the global system aggregate accurately. There are at least two significant obstacles to extending the subspace method to a distributed setting:

1. With minimal communication overhead, maintain projection matrix $\tilde{\mathbf{C}}$ while matrix $\mathbf{Y}$ (formed by distributed link measurements) evolves over time.

2. With minimal communication overhead, track and fire triggers to indicate anomalies when $\|\tilde{\mathbf{C}}\mathbf{y}\|^2 > \delta_\alpha^2$

Maintaining the subspace projection matrix $\tilde{\mathbf{C}}$ in a distributed way is difficult, because computing $\tilde{\mathbf{C}} = \mathbf{I} - \mathbf{P}\mathbf{P}^T$ is equivalent to solving the symmetric eigenvalue problem for the covariance matrix $\mathbf{Y}^T\mathbf{Y}$, which involves quadratic terms of measurement data from all links. The stability of matrix $\mathbf{P}$ is a function of the stability of the network's traffic matrix, and impacts how often $\tilde{\mathbf{C}}$ needs to be updated. There is some indication that traffic matrices are stable for up to 5 day periods (weekdays) [10]. However, in general, it is assumed that the matrix $\mathbf{P}$ will need to be updated frequently (although the exact frequency is unclear). In this work, we assume a stable $\mathbf{P}$ and choose to focus on obstacle 2. We leave as future work the design of a method for maintaining matrix $\tilde{\mathbf{C}}$.

Given a relatively stable projection matrix $\tilde{\mathbf{C}}$, it is still not easy to compute the distributed function $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ and check whether it is above the threshold δ_α^2 in a communication-efficient way. This is because $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ is a quadratic function and involves the cross-product of measurements from different links (i.e., it has terms like $r_i \cdot r_j$ for $i \neq j$). It is unclear how local link measurement r_i impacts $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ without knowing the measurements from other links. Our way to tackle this issue is to use the first order approximation of the quadratic function. One can compute the partial derivative of $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ w.r.t. r_i, which is the marginal factor of r_i on $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$

$$g_i := \frac{\partial\|\tilde{\mathbf{C}}\mathbf{y}\|^2}{\partial r_i} = \frac{\partial\left(\sum_{j=1}^n \left(\mathbf{y^T}\tilde{\mathbf{c}}_\mathbf{j}\right)^2\right)}{\partial r_i} = 2\mathbf{y^T}\tilde{\mathbf{C}}\tilde{\mathbf{c}}_\mathbf{i} \tag{2}$$

where $\tilde{\mathbf{c}}_i$ is the i-th column of matrix $\tilde{\mathbf{C}}$. If we ignore second order terms, we can see that if r_i changes by 1 unit, then $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ would change by a factor of $g_i = 2\mathbf{y^T}\tilde{\mathbf{C}}\tilde{\mathbf{c}}_\mathbf{i}$ units. The coordinator can compute these derivatives g_i, because it has all information needed to do so. The coordinator sends each monitor i, its partial derivative g_i at the same time it sends the local slack δ_i. We can now describe the functionality at the monitors and coordinator as follows.

Each monitor M_i tracks the change of its $r_i(t)$ and sends the coordinator updates whenever $|r_i(t) - R_i(t)| > \frac{\delta_i}{|g_i|}$. Even though each monitor cannot see the data from other monitors, in this way, it can filter its traffic not only based upon how far the data is from its own most recent prediction (as in earlier triggers), but also according to the relative impact of a particular monitor's data on the aggregation function, relative

ϵ	Missed Detections		False Alarms		Comm. Overhead	
	week a	week b	week a	week b	week a	week b
0.00	0	0	0	0	0.13	0.30
0.05	0	1	1	0	0.12	0.24
0.10	0	1	0	0	0.10	0.21
0.15	0	1	0	0	0.10	0.19

Table 1: ***Detection error vs. communication overhead. Week a has 6 anomalies and week b has 15 anomalies.***

to other monitors.

The coordinator computes $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ and triggers an alarm indicating an anomaly if $\|\tilde{\mathbf{C}}\mathbf{y}\|^2 > \delta_\alpha^2$. Second, it continuously computes $\Delta = \max(\epsilon, \delta_\alpha^2 + \epsilon - \|\tilde{\mathbf{C}}\mathbf{y}\|^2)$, based on which it computes δ_i's. Third, it computes the partial derivatives g_i's according to Eqn.2, and disseminates the parameters (δ_i, g_i) the monitors. To maintain system stability, the coordinator uses low-pass filtering techniques (e.g., based on discretization intervals) to avoid disseminating new parameters for small, transient changes in Δ [3].

We justify using a first order approximation as follows: 1) $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ is a quadratic function of $r_i(t)$'s and has terms only up to the second order; 2) the approximation is only used to determine when to send $r_i(t)$ values to the coordinator, thus bounding the difference between $r_i(t)$ and $R_i(t)$. Once $r_i(t)$ is updated, the coordinator uses $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ for calculations; 3) when δ_i is small, which is the case as $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ approaches the threshold, $|r_i(t) - R_i(t)|$ is small and its high order is even smaller. Thus, the accuracy is sufficient for our detection purposes when using only the first order of $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ to control updates. Our experiments show that this approximation is accurate and does not introduce detection errors.

3.3 Evaluation

For a preliminary validation for our approach, we used two one-week long Abilene network traffic matrices, collected in 10 minute intervals on 41 individually monitored links. We set the threshold δ_α^2 to a $1 - \alpha = 99.5\%$ confidence level, and set $\epsilon = 0$. The results are shown in Figure 4. The solid curve is **SPE**, the timeseries of $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$, and the dashed line is the threshold δ_α^2. Note that our distributed algorithm (star points) detects all anomalies (6 in week a and 15 in week b) that are detected by the centralized algorithm (circle points). Examining the timeseries values of $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$, we find that the signal values of anomalies computed by our distributed algorithm are exactly the same as those computed by the centralized algorithm, when setting $\epsilon = 0$. These results demonstrate that our approximation up to the first order of the **SPE** function is accurate.

Table 1 shows the tradeoff between triggering accuracy ϵ, and missed detections, false alarms, and communication overhead (including messages from monitors to coordinator and from coordinator to monitors). When varying ϵ from 0.00 to 0.15, our distributed algorithm has low detection error (at most 1 missed detection/false alarm), and incurs modest communication overhead, ranging from 13% to 10% of original data for week a, and 30% to 19% of original data for week b. While using only 13% (or 30%) of original data, our distributed algorithm is as equally effective as the centralized algorithm. We hypothesize that this per-node communication overhead remains stable as the network size increases.

4. OTHER EXTENSIONS

We believe that more general constraints, that include varied aggregation functions, threshold functions, that use subsets of monitors, and that are defined over varying time period, can all be incorporated into a more general triggering protocol. We briefly introduce these ideas here.

4.1 Supporting Complex Constraints

In detection systems, the constraints are typically made up of an aggregation function, a threshold and a set of nodes whose data is used to determine whether or not the constraint has been violated. Our previous work used linear aggregation functions. In this work, we introduced methods for dealing with quadratic aggregation functions. We now discuss how a variety of complex constraints can be built by varying these elements of the constraint(s). Each of these variations can be expressed in the terms of our framework, and incorporated into a generalized triggering protocol.

In our anomaly detector application we used a Taylor series expansion (for the aggregation function on $f[R_1, \ldots, R_n]$), dropping higher order terms, to deal with a quadratic constraint. This approach can be applied more generally. For any such continuous function $f(t) = f[r_1(t), \ldots, r_2(t)]$, the Taylor Expansion is

$$f[R_1, \ldots, R_n] - f[r_1, \ldots, r_n] = \sum_{i=1}^{n} \frac{\partial f}{\partial r_i} \cdot (R_i - r_i) + \mathcal{O}\left[\sum_{i,j=1}^{n} (R_i - r_i) \cdot (R_j - r_j)\right]$$

Then, if $(R_i - r_i), i = 1, \ldots, n$, are small and independently, and we ignore all second and higher order terms, we can linearize this continuous function, and the distributed simple triggers can track this function based on its first order components. We define $g_i \equiv \frac{\partial f}{\partial r_i}$ as the marginal impact of local value $r_i(t)$ on the global function. This is incorporated into our system by the monitor who performs filtering using $|r_i(t) - R_i(t)| > \frac{\delta_i}{|g_i|}$. Note that the marginal factor g_i can be computed by M_i itself (simple constant) or computed by the coordinator (time-varying distributed values).

So far we have used a constant trigger threshold C. However, this threshold could, for example, vary in time. It could be determined or computed as a user specified input, or be a function of the data - computed using all the data (as in our anomaly detector), or using data from only a subset of the monitors. Other constraints can be built when the set of monitoring nodes are split into different subsets. Let A and B be two subsets of monitors of interest, with n_1 and n_2 monitors, respectively. To denote variables from monitors in the set, we use set labels in the superscript.

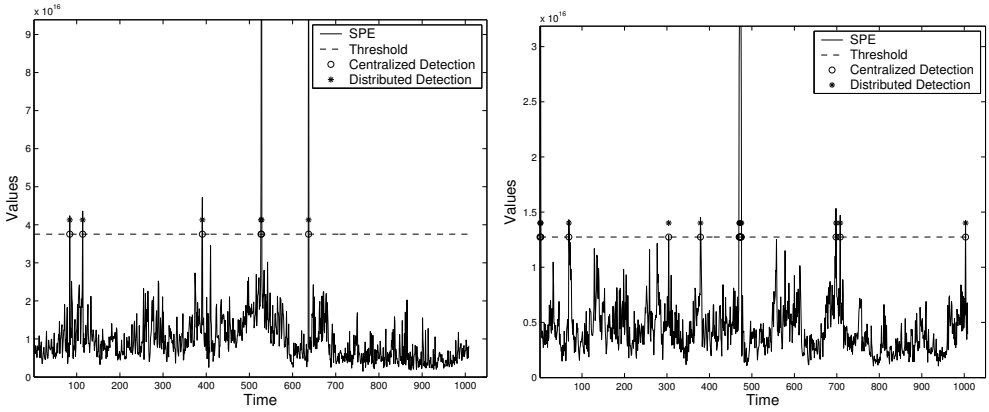
More complex threshold functions and the ideas of using subsets of monitoring data can be incorporated into the coordinator's protocol. The coordinator can track one or more global functions that are defined on subsets of distributed data streams. For example, let $f(t)$ be the function defined on set A, and $C(t)$ on set B. The coordinator computes

$$f(t) = f\left[R_1^A, \ldots, R_{n_1}^A\right], \quad C(t) = C\left[R_1^B, \ldots, R_{n_2}^B\right]$$

and triggers an alarm if $f(t) > C(t)$. Based on its view of global information, the coordinator computes a set of parameters and sends them to monitors when necessary as:

$$\max(\epsilon, C(t) + \epsilon - f(t)) = \delta_1^A + \cdots + \delta_{n_1}^A + \delta_1^B + \cdots + \delta_{n_2}^B$$

$$g_i^A = \frac{\partial f}{\partial r_i^A}, \quad g_j^B = \frac{\partial C}{\partial r_j^B}$$



(a) *Week a* (b) *Week b*

Figure 4: Distributed detection on the timeseries of SPE $= \|\tilde{\mathbf{C}}\mathbf{y}\|^2$ with $\epsilon = 0$.

Various optimization algorithms can be used to compute δ_i's that minimize communication, however our protocol remains applicable regardless of algorithm choice.

It is also possible to incorporate the notion of time, in case one seeks to detect constraints violations that occur over a period of time. To support such constraints, we can extend the form into varying-window triggers, which track the relationship between $f(t)$ and $C(t)$, and only trigger alarms when $f(t)$ exceeds $C(t)$ over time. These ideas are discussed in [3].

4.2 Advanced Queries For Load Balancing

We now demonstrate how a generalized triggering protocol can support advanced queries for hot spot detection in distributed systems. Consider: 1) **relative triggers** that alarm if the total workload of servers in set A is β times more than that of set B; 2) **any-set triggers** that alarm if the total workload of any $\alpha\%$ servers is more than C; 3) **composite triggers** that alarm if the total workload of any $\alpha\%$ servers is more than β portion of the total system workload.

Tracking relative triggers. We view relative triggers as normal ones with time-varying threshold $C(t)$ as follows: 1) the coordinator has threshold $C(t) = \beta \cdot \sum R_j^B(t)$, and 2) it triggers whenever $\sum R_i^A(t) > C(t)$. Monitors and the coordinator have to track both values of $\sum r_i^A(t)$ and $\sum r_j^B(t)$. One can easily extend the instantaneous trigger to detect unbalanced load and guarantee a "2ϵ-band" of detection accuracy.

Tracking any-set and composite triggers. One can prove that detecting whether the workload sum from any subset of k servers is above a threshold is equivalent to detecting whether the sum of the top-k workload is above the threshold. So by composing distributed top-k monitoring [1] with our triggering protocols, we can efficiently track both any-set and composite triggers with guaranteed accuracy. We leave as future work to extend and customize triggers for top-k monitoring.

5. CONCLUSIONS AND FUTURE WORK

We have presented our novel approach to extending simple threshold triggers for sophisticated anomaly detection problems. We designed a distributed protocol that can perform online detection of network-wide anomalies with modest communication overhead, and also discussed our general extensions to existing triggering protocols to support wide-range of detection tasks. Through a set of examples, we have shown that distributed triggers are an efficient and extensible vehicle for advanced detection algorithms. We plan to further extend this line of research, as well as engage in collaborations with domain experts on new application development.

6. REFERENCES

[1] Babcock, B. and Olston, C. Distributed Top-K Monitoring. In *ACM SIGMOD*, (2003).

[2] Hanson, E. N., Bodagala, S., and Chadaga., U. Trigger condition testing and view maintenance using optimized discrimination network. *IEEE TKDE*, 14(2) (2002).

[3] Huang, L., Garofalakis, M., Joseph, A. and Taft, N. Communication-efficient tracking of distributed triggers. Tech. rep., February 2006.

[4] Huebsch, R., Hellerstein, J., Lanham, N., Loo, B.-T., Shenker, S. and Stoica, I. Querying the internet with pier. In *VLDB* (2003).

[5] Jain, A., Hellerstein, J. M., Ratnasamy, S., and Wetherall, D. A wakeup call for internet monitoring systems: The case for distributed triggers. In *HotNets* (2004).

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

[7] Jackson, J. E. and Mudholkar, G. S. Control procedures for residuals associated with principal component analysis. In *Technometrics*, pages 341-349, 1979.

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

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

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

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

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

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

[14] Widom, J., and S.Ceri. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann, 1996.

[15] 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).

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

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