Toward Sophisticated Detection With Distributed Triggers

Ling Huang* Minos Garofalakis† Joseph M. Hellerstein* Anthony D. Joseph* Nina Taft†
*UC Berkeley †Intel Research

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, this work has been limited to simple thresholds on distributed aggregate functions like sums and counts. In this paper, we present our initial results that show how to use these simple threshold triggers to enable sophisticated anomaly detection in near-real time, with modest communication overheads. We design a distributed protocol to detect "unusual traffic patterns" buried in an Origin-Destination network flow matrix that: a) uses a Principal Components Analysis decomposition technique to detect anomalies via a threshold function on residual signals [10]; and b) efficiently tracks this threshold function in near-real time using a simple distributed protocol. In addition, we speculate that such simple thresholding can be a powerful tool for a variety of monitoring tasks beyond the one presented here, and we propose an agenda to explore additional sophisticated applications.

1 Introduction

Distributed monitoring and anomaly detection systems have been proposed and deployed to aggregate status information and detect unusual events for large distributed systems such as server clusters and large networks [2, 5, 14]. 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 [6]. A primary goal in a monitoring system is ensuring the target system is well behaved, which can be aided by maintaining a well-defined set of logical predicates or invariants over entire systems. To support this functionality, a set of approaches have been proposed recently for communication-efficient tracking of distributed triggers [4, 9]. They aim to detect constraint violations via threshold functions defined on distributed information. With such protocols, many system-wide conditions can be checked and invariants can be enforced cheaply. 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 advanced anomaly detection based on simple, efficient threshold triggers. We illustrate distributed triggers' potential as a vehicle for advanced detection algorithms, as well as discuss the unique features and constraints of embedding detection algorithms in a triggering framework.

Our extensions to simple distributed triggers [4] demonstrate their usefulness in implementing sophisticated real-time detection functions. We believe that simple, efficient, and extensible triggers are capable of a variety of monitoring tasks, and we show that these triggers can be extended and composed with existing query and detection techniques to enhance applications with sophisticated distributed detection capabilities. One of this paper's contributions is the design of an example application: a distributed protocol that detects network-wide anomalies in a dynamic Origin-Destination (OD) network traffic flow matrix by: a) using a Principal Components Analysis (PCA) technique to decompose network traffic into normal and residual components; b) applying a threshold function to detect anomalies on residual components [10]. Our results show that our triggers efficiently track this threshold function in near real time with modest communication overhead, demonstrating that our triggering protocol can improve the practicality of an existing centralized solution in a distributed way by reducing the communication overhead.

Prior Work. The database community has extensively explored centralized triggering mechanisms [3, 15]. However, the goal of minimizing communication overhead in widely distributed Internet environments introduces new challenges. Keralapura *et al*. [9], 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 [4] 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 [4] and [9] 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 [13, 16, 17] aims to share information and foster collaboration between widely distributed monitoring boxes to offer improvements over isolated systems. These systems are examples of dis-

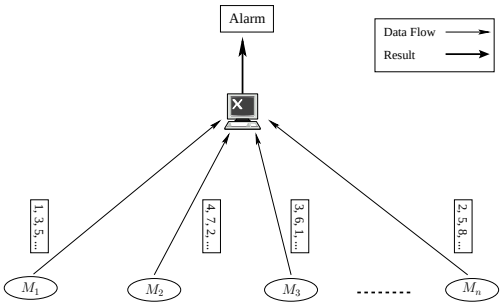

Figure 1: The system setup.

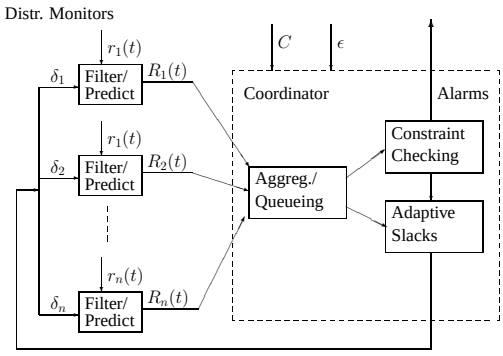

Figure 2: The distributed trigger tracking framework.

tributed monitoring systems for which a triggering tool such as ours would be useful. Lakhina *et al*. [10], carried out the pioneering work in detecting network-wide anomalies. Zhang *et al*. [18], extended it further and proposed a general "anomography" framework to infer anomalies at network level in both spatial and temporal domains.

Organization. We present our communication-efficient distributed triggers in Sec. 2; discuss our distributed triggering-based approach for network traffic anomaly detection in Sec. 3; present general extensions to the triggering protocol and other application areas in Sec. 4; finally, discuss future work and our conclusions in Sec. 5.

2 Tracking Distributed Triggers

As shown in Fig. 1, a typical distributed triggering system consists of a set of widely distributed monitoring nodes $M_1, M_2, \ldots, M_n$ 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. Monitors only send update information to the coordinator when necessary. Because the coordinator has imperfect knowledge of the monitored data (it receives filtered versions), it can make mistakes (e.g., *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.

In [4], we provided a mathematical definition of the distributed triggering problem with three distinct types of different constraint violation modes, and designed protocols that enable users to tradeoff desired detection performance with communication overhead. *Instantaneous* triggers (the focus of this paper) fire when an aggregate threshold value is violated within a single time instance, and *fixed-window* and *varying-window* triggers aim to catch persistent threshold violations over a fixed or varible window of time, respectively. Our adaptive protocol for instantaneous triggers exploits the specified trigger threshold to minimize communication while offering deterministic accuracy guarantees. Testing using real-life data streams from PlanetLab Intrusion Detection System monitors showed our algorithms' significant communication-efficiency gains — a reduction in monitor data sent to the coordinator of more than 80% [4].

2.1 Instantaneous Triggers

We define instantaneous violations as violations occurring in a single time instant t, where C denotes the distributed trigger threshold. Our goal is to track an instantaneous violation *approximately* to within a specified error tolerance ϵ around C, and our tracking algorithms exploit this error tolerance to minimize communication costs. More formally, using a `SUM` function, the trigger fires for any time instant t where $\sum_i^n r_i(t) > C + \epsilon$, for nodes $i = 1, \ldots, n$.

Fig. 2 shows a framework for distributed trigger tracking. Intuitively, our instantaneous triggers protocol works as follows. Each monitor provides the coordinator with a prediction function. The coordinator computes the difference (or "slack") between the aggregate signal from the prediction functions and the trigger threshold, divides up this difference, and sends the fractions to each monitor. Each monitor tracks the drift between its actual signal and prediction function, and only sends the coordinator an update when the drift exceeds its fraction of the slack.

More formally, let $R_i(t)$ denote the approximate representation of $r_i(t)$ that the coordinator uses; in general, $R_i(t)$ can be based on any type of *prediction model* for node M_i's behavior over time. Each monitor node M_i filters its updates to the coordinator based on *local monitor slack* parameters $\delta_i > 0$ which upper bounds the drift between the coordinator's view, $R_i(t)$, and the actual $r_i(t)$ signal. As long as the prediction accurately captures M_i's behavior (*i.e.*, within δ_i bounds), no communication is needed. Meanwhile, the coordinator determines

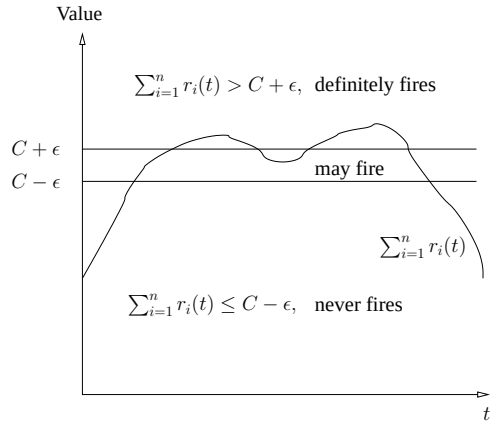

Figure 3: Instantaneous trigger tracking guarantees.

the amount of global monitor slack Δ and an allotment of Δ to individual local slacks δ_i such that $\Delta = \sum_i \delta_i$. Each δ_i value is sent to M_i which continuously tracks the (instantaneous) difference $d_i(t) = |r_i(t) - R_i(t)|$ between the true local signal and its (most recent) prediction. Whenever $d_i(t) > \delta_i$ occurs, M_i updates the coordinator with the latest $r_i(t)$ value, and a new prediction $R_i(t)$ that satisfies M_i's local filtering constraint.

The coordinator continuously tracks the aggregate predictions, and triggers a condition violation whenever $\sum_{i=1}^{n} R_i(t) > C$. The coordinator also continuously estimates the total amount of available slack as

$$\Delta(t) = \max(\epsilon, C + \epsilon - \sum_{i=1}^{n} R_i(t))$$

and distributes the slack to local monitor δ_i's (*e.g.*, using a marginal-gains strategy, as in [12]). 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* $\sum_i r_i(t) < C - \epsilon$*.*

In other words, 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 [7, 12], we are *not* interested in continuously maintaining a guaranteed ϵ-error aggregate at the coordinator. 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.

While our triggering protocols currently support simple linear functions (e.g., `SUM` and `COUNT`) and not advanced queries (e.g., top-k, histogram, join, etc.), we will show how, with non-trivial extensions, they are capable of supporting sophisticated detection functions. The problem of detecting unusual events (i.e., intrusions, anomalies, and hot spots) in distributed systems can be mapped to triggering problems, and their solution can be implemented in a distributed way using extensions to simple triggers.

3 Network-wide Traffic Anomaly Detection

In this section, we first review a centralized algorithm using a subspace method for online detection of network-wide anomalies by thresholding a quadratic residual function, and then show how we can efficiently track this function in a distributed way using extended `SUM` triggers.

3.1 Using PCA for Centralized Detection

Network volume anomalies are unusual and significant changes in Origin-Destination traffic flows typically caused by worms or DoS attacks, device failures, misconfigurations, etc. 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.* [10] use this property by applying the subspace method to diagnose traffic anomalies on backbone networks using only link traffic counts. Their technique performs PCA on link traffic measurements, and decomposes 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. Here we summarize their method and propose a solution for distributed and online detection of traffic anomalies.

Consider a network with n links in m successive time intervals of interest. There is a monitor M_i for each link, each of which produces a column of timeseries measurements. We let $\mathbf{Y}$ be the $m \times n$ measurement matrix, 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. 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:

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

As studied in [11], 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

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

where $\mathbf{P} = [\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}]$, 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 $\mathbf{SPE} > \delta_\alpha^2$, where δ_α^2 denotes the threshold for the SPE at the $1-\alpha$ confidence level. A statistical test for the residual vector, known as the $Q\text{-}statistic$ and derived in [8], can be computed by the principle eigenvalues.

The approach proposed in [10] is a centralized solution, which assumes measurement data from all links are shipped to the coordinator for analysis. It would incur excessive communication overhead if the network size is large, so a communication-efficient algorithm is necessary.

3.2 Distributed Detection

We use a novel approach to map centralized anomaly detection to distributed threshold triggering, and to address the following issues: 1) a distributed solution pushes functionality to monitors from the coordinator; 2) monitors do local processing (e.g., filtering or local decision making) to suppress unnecessary message updates; and 3) the coordinator makes global decisions and maintains detection accuracy by coordinating monitors.

There are at least two significant obstacles to extending the subspace method in a distributed way:

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. Fortunately, previous work has shown that (for OD flows) matrix $\mathbf{P}$ can be reasonably stable from week to week [11]. Since at a reasonable accuracy level, one need only compute the principal components occasionally rather than at each timestep, we only focus in this paper on obstacle 2. As future work, we plan to design communication-efficient protocols 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 $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$

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

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 1 unit, $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ would change by a factor of $2\mathbf{y^T}\tilde{\mathbf{C}}\tilde{\mathbf{c}}_\mathbf{i}$ units. The coordinator can help monitors track these derivatives, because it has all information to compute them and piggyback them when it feeds back slacks to monitors.

Based on this first order approximation, the quadratic residual function $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$ can be approximated by a linear function, and simple triggers can be used to track this function to detect traffic anomalies as follows.

Each monitor M_i tracks the change of its $r_i(t)$ and sends the coordinator updates whenever

$$|r_i(t) - R_i(t)| > \frac{\delta_i}{|2\mathbf{y^T}\tilde{\mathbf{C}}\tilde{\mathbf{c}}_\mathbf{i}|}$$

ϵ	Missed Detections	False Alarms	Comm. Overhead
0.00	0	0	0.23
0.05	1	0	0.21
0.10	1	0	0.19
0.15	1	0	0.18

Table 1: *Detection error vs. communication overhead.*

where $\Delta = \sum_i \delta_i$, and both Δ and $2\mathbf{y}^{\mathbf{T}}\tilde{\mathbf{C}}\tilde{\mathbf{c}}_{\mathbf{i}}$ are computed by the coordinator according to its view of global information of link matrix $\mathbf{Y}$.

The coordinator triggers an alarm indicating an anomaly if $\|\tilde{\mathbf{C}}\mathbf{y}\|^2 > \delta_\alpha^2$, and it continuously computes

$$\Delta = \max(\epsilon, \delta_\alpha^2 + \epsilon - \|\tilde{\mathbf{C}}\mathbf{y}\|^2)$$

based on which it computes δ_i's, and disseminates δ_i's and derivatives to monitors when necessary.

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 exact calculations without any approximation; 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 a one-week long Abilene network traffic matrix, 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 15 anomalies 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. When varying ϵ from 0.00 to 0.15, our distributed algorithm has low detection error (at most 1 missed detection), and incurs modest communication overhead, ranging from 23% to 18% of original signals. While using only 23% 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.

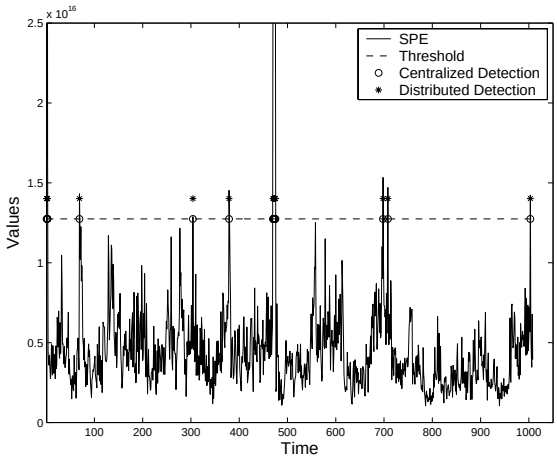

Figure 4: Distributed detection on the timeseries of $\|\tilde{\mathbf{C}}\mathbf{y}\|^2$.

4 Other Applications

In this section, we present general extensions to simple linear triggers and discuss other potential applications.

4.1 General Extensions

Simple triggers can be extended beyond linear functions and constant thresholds as follows. For a 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 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.

The trigger threshold C is not limited to a constant, but as discussed below, can detect whether a global (nonlinear) function $f(t)$ is above a time-varying critical threshold function $C(t)$, either predefined, or defined on subsets of distributed data streams.

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. To a first order approximation, the distributed protocols for a set of advanced detection problems can be expressed as follows when using instantaneous triggers.

Each monitor models its local data stream and tracks the deviation of its real values from the modeled values. Moni-

tor M_i sends the coordinator update information if its data stream has any surprising change, $|r_i(t) - R_i(t)| > \frac{\delta_i}{g_i}$. Marginal factor g_i can be computed by M_i itself (simple constant) or computed by the coordinator (time-varying distributed values).

The coordinator tracks one or more global functions which 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:

$$\max(\epsilon, C(t) + \epsilon - f(t)) = \delta_1^A + \cdots + \delta_{n_1}^A + \delta_1^B + \cdots + \delta_{n_2}^B$$
$$g_i^A = \frac{\partial f}{\partial r_i^A}, \quad g_j^B = \frac{\partial C}{\partial r_j^B}$$

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

When constraints are defined in terms of penalty, 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 and accrues sufficient persistent violation. Due to space limitations, we do not provide details here.

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 function to detect unbalanced load and guarantee a "2ϵ-band" of detection accuracy.

Tracking any-set and composite triggers. One can easily 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 how to extend triggers and customize them for top-k monitoring.

5 Future Work and Conclusions

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.

References

[1] BABCOCK, B. AND OLSTON, C. Distributed Top-K Monitoring. In *ACM SIGMOD*, (2003).

[2] CLARK, D., PARTRIDGE, C., RAMMING, J. C., AND WROCLAWSKI, J. T. A knowledge plane for the internet. In *ACM SIGCOMM* (2003).

[3] HANSON, E. N., BODAGALA, S., AND CHADAGA., U. Trigger condition testing and view maintenance using optimized discrimination network. *IEEE TKDE*, 14(2) (2002).

[4] HUANG, L., GAROFALAKIS, M., JOSEPH, A. AND TAFT, N. Communication-efficient tracking of distributed triggers. Tech. rep., February 2006.

[5] HUEBSCH, R., AND ET AL. Querying the internet with pier. In *VLDB* (2003).

[6] 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).

[7] JAIN, A., CHANG, E. Y., AND WANG, Y.-F. Adaptive stream resource management using kalman filters. In *ACM SIGMOD* (2004).

[8] JACKSON, J. E. AND MUDHOLKAR, G. S. Control procedures for residuals associated with principal component analysis. In *Technometrics*, pages 341-349, 1979.

[9] KERALAPURA, R., CORMODE, G. AND RAMAMIRTHAM, J. Communication-efficient distributed monitoring of thresholded counts To appear in *ACM SIGMOD* (2006).

[10] LAKHINA, A., CROVELLA, M. AND DIOT, C. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM*, (2004).

[11] LAKHINA, A., PAPAGIANNAKI, K., CROVELLA, M., DIOT, C., KOLACZYK, E. D. AND TAFT, N. Structural analysis of network traffic flows. In *ACM SIGMETRICS*, (2004).

[12] OLSTON, C., JIANG, J., AND WIDOM, J. Adaptive filters for continuous queries over distributed data streams. In *ACM SIGMOD* (2003).

[13] PADMANABHAN, V. N., RAMABHADRAN, S., AND PADHYE, J. Netprofiler: Profiling wide-area networks using peer cooperation. In *IPTPS* (2005).

[14] SPRING, N., WETHERALL, D., AND ANDERSON, T. Scriptroute: A facility for distributed internet measurement. In *USITS* (2003).

[15] WIDOM, J., AND S.CERI. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann, 1996.

[16] 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).

[17] YEGNESWARAN, V., BARFORD, P., AND JHA, S. Global intrusion detection in the domino overlay system. In *NDSS* (2004).

[18] ZHANG, Y., GE, Z.-H., GREENBERG, A., AND ROUGHAN, M. Network anomography. In *IMC* , (2005).