Rapid Detection of Maintenance Induced Changes in Service Performance

Ajay Mahimkar‡, Zihui Ge‡, Jia Wang‡, Jennifer Yates‡, Yin Zhang§,
Joanne Emmons‡, Brian Huntley†, Mark Stockert‡

AT&T Labs – Research ‡ AT&T, Inc. † The University of Texas at Austin §

ABSTRACT

Service quality in operational IP networks can be impacted due to planned or unplanned maintenance. During any maintenance activity, the responsibility of the operations team is to complete the work order and perform a check-up to ensure there are no unexpected service disruptions. Once the maintenance is complete, it is crucial to continuously monitor the network and look for any performance impacts. What operations lack today are effective tools to rapidly detect maintenance induced performance changes. The large scale and heterogeneity of network elements and performance metrics makes the problem extremely challenging.

In this paper, we present PRISM, a new tool for detecting maintenance induced performance changes in a timely fashion. PRISM uses association between maintenance and the network elements to identify performance metrics for time-series analysis. It uses a new **M**ultiscale **R**obust **L**ocal **S**ubspace algorithm (MRLS) to accurately identify changes in performance even when the baseline is contaminated. We systematically evaluate PRISM using data collected at four large operational networks: a tier-1 backbone, VoIP, IPTV and 3G cellular and show that it achieves good accuracy. We also demonstrate the effectiveness of PRISM in real operational environments through interesting case study findings.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management*

General Terms

Management, Performance, Reliability

Keywords

Service performance, Robust change detection

ACM CoNEXT 2011, December 6–9 2011, Tokyo, Japan.

1. INTRODUCTION

Maintenance is one of the important tasks in network and service management. New elements are introduced, old ones removed, software is upgraded and configurations and technologies are changed to introduce new network and service features. Even external activities such as highway road works can result in network maintenance being necessary. Each of these activities has the potential to introduce new performance issues into the network and thus must be closely monitored. Despite extensive testing of the new software or equipment before deployment, intrinsic errors and feature bugs might only manifest in operational settings because of the large scale, device type, vendor heterogeneity and complicated interactions across devices or software.

During a maintenance activity, the Operations team is responsible for completing the work order and performing check-ups to ensure that there are no unexpected service disruptions. Once the maintenance is complete, it is important to carefully monitor the network and look for any performance changes. The current operational practice in determining the impact of maintenance is to monitor a pre-defined list of performance metrics for unexpected degradations - metrics which have been identified as being potentially impacted if things go awry. This process is generally executed manually; which is cumbersome, error-prone and impossible to scale.

Our goal is to replace today's manual inspection of key performance metrics with an automated approach that identifies any statistically significant change in a broad wealth of service and network performance metrics. Our approach is to avoid incorporating detailed domain knowledge as to what may or may not have been impacted by any given maintenance activity, and instead detect issues observed in any of the vast wealth of service and network data, incorporating metrics ranging from user-perceived issues (*e.g.*, blocked voice calls), network performance (*e.g.*, packet queue drops), and network health (*e.g.*, device CPU utilization). Through this approach, we aim to rapidly detect issues which would have traditionally "flown under the Operation's radar", enabling considerably faster mitigation to service and network impacting issues, thereby improving overall customer service quality.

In our previous work [21], we proposed MERCURY to capture long term impact on performance likely induced by the maintenance activities. MERCURY explicitly looked for behavior changes that persisted over a long time (*e.g.*, level-shifts) as a result of maintenance activities and eliminated transient changes like spikes. In this paper, we present an alarming tool for quick detection of behavior changes immediately after maintenance activities are conducted. The types of behavior changes include a spike (*e.g.*, sudden increase in voice call drops), level-shift or even a slow ramp-up (*e.g.*, deteriorating condition). It is highly desirable to detect these behavior changes on a shorter time-scale (*i.e.*, on the order of minutes or hours) for timely mitigation. Furthermore, operating on a short time-scale imposes new challenges on detecting behavior changes in terms of handling diverse characteristics of different data sources (*e.g.*, seasonality, variability) because we lose the luxury of aggregating and smoothing out the data as we can do in MERCURY to pick up long-term persistent performance impact.

Challenges: There are several interesting challenges:

1. **Baseline construction.** Identifying the baseline to compare the performance after the maintenance is not easy - the baseline can either be a few minutes or hours before the maintenance, same time of day before the day of maintenance, or same day of week as the day of maintenance. The time-of-day or day-of-week (weekend versus weekday) effect may lead to incorrect inferences of changes in performance that in reality have no relation to the maintenance. It is thus important to carefully construct the baseline.
2. **Data diversity and baseline contamination.** The service and network measurements are highly diverse in nature and exhibit different characteristics such as seasonality (*e.g.*, server workloads), high variability (*e.g.*, router CPU utilization), randomness in the time-series (*e.g.*, layer-1 failures) and stationarity (*e.g.*, router memory utilization). Detecting real, significant changes in measurements that are inherently noisy and highly variable is extremely challenging. Using manual analysis, this can be basically impossible; even with automated techniques it can be challenging to detect the changes when the baseline is contaminated with impacts due to earlier maintenance and/or failures.
3. **Large number of performance event-series.** The performance impacts can be observed in any of the vast number of diverse network and service measurements collected. If the network operators are not looking at the right metric, then they may incorrectly declare no impact; only later to realize the issue via customer complaints, for example. There are on the order of thousands of time-series metrics that one can create from the available network data. Manually analyzing such a massive amount of data is simply infeasible.
4. **Spatial scope of maintenance impact.** Some maintenance activities have a local impact (*e.g.*, a line card upgrade can impact either itself or other line-cards on the same router), whereas others have a global impact (*e.g.*, a configuration change at a cellular base station in a region can impact service quality for all the users within the region). It is thus important to take into account the spatial scope of impact of maintenance when identifying the performance metrics of interest. A lack of consideration of the right scope can lead to missed detections.

Our Approach and Contributions: We propose a new tool PRISM[1] for rapid detection of performance impacts induced by the maintenance activities. For a given type of maintenance activity, it automatically selects the right spatial scope of impact and mines a wide range of performance metrics to discover the network and service performance changes. PRISM addresses the above challenges as follows:

1. PRISM addresses the time-of-day and day-of-week effects by accurately constructing the baseline using historical data (30 days before the day of maintenance in our prototype implementation). By comparing the performance time-series on the day of maintenance with the baseline, PRISM identifies statistically significant changes in performance that are induced by the maintenance.
2. PRISM uses a new **M**ultiscale **R**obust **L**ocal **S**ubspace (MRLS) algorithm for accurately detecting the performance changes induced by the maintenance. MRLS works across a diverse set of data and implicitly accounts for any seasonality, stationarity, or high variability. It constructs the baseline normal subspace using multiscale differencing and robust l_1 norm. This is effective even when the data is contaminated with changes from previous maintenance activities or failures.
3. PRISM uses automation to systematically mine a large number of diverse performance measurements collected at different layers, protocols and network devices. It parses the measurements logs, identifies the appropriate aggregation granularity depending on the spatial scope of impact and detects changes in time-series for each metric. Having such a capability enables analysis that go beyond what can be achieved manually.
4. To identify the spatial impact scope of the maintenance activity, PRISM uses the hierarchical structure of network components and topological information to construct an influence group (which are the network elements that can potentially be impacted). By monitoring the performance metrics at network elements within the influence group, PRISM increases the likelihood of capturing the performance impacts of maintenance.

We conduct extensive evaluation (Section 3) using data collected from four operational networks and services (tier-1 backbone, VoIP, IPTV, and 3G Cellular). Our results demonstrate the superior performance of MRLS compared to other subspace algorithms under varying degrees of baseline contamination and diverse data characteristics. Encouraged by the evaluation results, we have started applying PRISM to the operational networks on a regular basis. We discuss interesting case study findings in Section 4. PRISM has confirmed some of the earlier findings of operations and in some cases, revealed previously unknown behaviors.

[1] Analogous to dispersing white light into a spectrum of colors, PRISM takes a maintenance activity as input and identifies significant increases, decreases or no changes in performance.

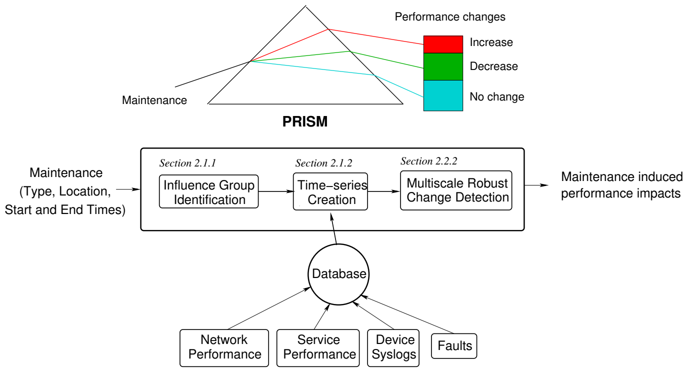

Figure 1: PRISM Overview and design.

2. PRISM DESIGN

In this section, we present the design of PRISM. After completing maintenance on a network device, the goal of the operations team is to carefully monitor the network and service to ensure there are no unexpected performance degradations. The impacts (if any) can typically be observed within a few minutes or hours after the maintenance when the network device is put back in service. PRISM serves as an automated tool to mine a wide variety of performance metrics and rapidly identify changes in a scalable fashion. Fig. 1 shows the system architecture of PRISM. For each maintenance event, PRISM identifies a group of network devices that could be impacted and then derives a list of performance metrics for time-series analysis (Section 2.1). The data to be fed into PRISM is ingested into a database with normalization (*e.g.*, all timestamps to GMT, and common router naming conventions) so as to facilitate easy analysis across different layers, devices and metrics. PRISM uses a novel multiscale robust local subspace algorithm (MRLS) to rapidly detect changes in performance that are induced by the maintenance even when the baseline is contaminated by other changes due to previously uncorrelated maintenance events or even failures (Section 2.2). PRISM is flexible to operate at multiple time scales to capture the impact.

2.1 Modeling Maintenance Impact

A maintenance activity is typically carried out during off-peak times. Traffic is moved away from the network element (NE) before starting the maintenance. Once the maintenance is complete, the NE is put back into service. The time duration between the start and end of a maintenance activity is called a *maintenance window*. The sequence of instructions to be executed during the maintenance is provided in a Method Of Procedure (MOP) document. The MOP also provides information about the type of network element undergoing maintenance (*e.g.*, a line card, an interface, or a router), the type of maintenance (*e.g.*, a software upgrade, configuration change, or a hardware replacement), which is useful to identify the impact scope of the maintenance. The impact scope is crucial to identify the NEs or services under influence (called an *influence group*). The performance metrics measured at the influence groups are monitored to identify impacts.

A maintenance window typically has multiple scheduled

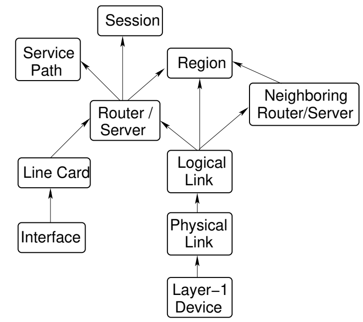

Figure 2: Hierarchy for constructing influence group.

activities across different network components. PRISM currently does not take into account the dependencies across multiple maintenance activities. It focuses on identifying the performance impact of each maintenance activity. For two or more correlated activities in quick succession, the Operations team can manually tie them to identify the performance impact after the last activity.

2.1.1 *Influence Group Identification*

Different types of maintenance activities have different impact scope. Some have a local impact (*e.g.*, a line card upgrade can impact performance of the same line card), whereas others have a global impact (*e.g.*, a BGP configuration change on a router can impact performance on a remote router which is the other end of the BGP session). Failure to capture the right impact scope can either lead to delayed detection after the damage has been done, or create a large number of false alarms. In PRISM, we use the *hierarchical structure* of network components and *topological information* to associate the maintenance with the influence group. Fig. 2 shows the hierarchy of network components: a physical link consists of multiple layer-1 devices; a logical link contains one or more physical links and connects two routers (or, servers); a line card consists of multiple interfaces; a router (or, a server) consists multiple line cards; a service path consists of multiple routers and logical links; and a session consists of two routers that can be multiple hops away.

For a given network component under maintenance, we identify the influence group as a collection of components at the same level in the hierarchy and any level higher up in the hierarchy. For example, if a line card is upgraded, the influence group would contain the same line card, the router to which it belongs, all other line cards on the same router, any protocol sessions on the router, and all services that have the router on the path. We however, do not include any network components lower in the hierarchy because it is less likely that any one of the lower level components would be individually impacted, instead of impacts across all of them. Intuitively, all lower level components are under the same influence and hence incorporating them into the influence group would add no more information than the component under maintenance. For example, in case of a line card upgrade, all interfaces on the line card can have similar impact and performance changes at the interfaces should also be observable in aggregate statistics at the line card. Whenever PRISM has access to service-level performance metrics such as call failures in Voice over IP, or data accessibility issues in the 3G cellular network, it incorporates end-to-end paths within the influence groups. For example, the influence group for a VoIP server upgrade would include all the trunk groups[2] traversing the server. PRISM uses the association between the maintenance type and network elements involved to identify the influence group. The hierarchy to construct the influence group shown in Fig. 2 is generic and can be easily applied or extended to different networks. The next step is to select the performance time-series measured at the network elements within the influence group.

2.1.2 Performance Time-series Creation

Depending on the maintenance, the operations teams monitor a pre-defined list of performance metrics ranging from device CPU and memory utilization, link load, packet errors and drops on the interfaces, end-to-end packet loss and delay, software errors, hardware faults and routing protocol flaps. The objective is to monitor the impact of maintenance on these metrics. Any change in performance immediately after the maintenance is a good indicator that it might be induced by maintenance. If the impacts are captured rapidly, then the damage can be prevented or minimized.

Instead of only looking at a pre-defined list of metrics, PRISM automatically mines all the performance metrics at the influence groups because sometimes the impacts might be hidden in previously unknown metrics. In this paper, we use measurements from SNMP MIBs, device syslogs, and service-level statistics. SNMP MIBs are recorded at regular intervals (*e.g.*, 5 minutes) and capture average statistics such as CPU and memory utilizations, packet counts, packet losses and errors. Device syslogs provide a rich source of information about the protocol and link state changes, software error conditions, hardware faults and environmental conditions. Service-level metrics provide information on the call failure percentages in VoIP, voice and data quality feedback from users in the 3G cellular network.

We construct a time-series for each performance metric by dividing the original series into n equal time-bins. Since our goal is to rapidly detect the changes in performance, we use small time-bins such as 5-minutes. For metrics like syslog messages, each time-bin contains the frequency of the message. Furthermore, depending on the granularity of the influence group member, we aggregate measurements from members lower in the hierarchy. For example, in order to detect the impacts of operating system upgrades, we would aggregate performance metrics from all line cards and interfaces on the upgraded router by computing the average per time-bin. Thus, we will have multiple performance time-series and our goal is to detect changes in these time-series that are induced by the maintenance activities.

2.2 Detecting Performance Changes

For each performance time-series, our goal is to determine if there are significant changes induced by the maintenance. The changes can either be spikes, level-shifts or slow ramp-ups. One approach is to compare time-series statistics like means, medians or entire distributions before and after the maintenance. Since we operate on finer time scales, we would like to eliminate changes caused by time-of-day effects. For example, link load has a high degree of seasonality and a CUSUM approach (used by MERCURY [21]) at 5-minute granularity would pick up change-points during the beginning and end of the peak times, resulting in a false correlation with the maintenance activity.

We approach the change detection problem by comparing the time-series within the segment of interest around the maintenance time with previous days before the maintenance. For example, compare time-series values between 8 AM and 4 PM with that of previous days. This approach is different from time-series analysis techniques like CUSUM, EWMA, ARIMA, and Holt-Winters where changes are detected over continuous time-series. Subspace algorithms are better suited and are shown to be more accurate [10, 23]. Subspace algorithms are based on Singular Value Decomposition (SVD) or Principal Component Analysis (PCA) of the time-series data. They construct the baseline by projecting the training data into a normal subspace and then looking for differences between the test data and the normal subspace. Krylov subspace learning [12] speeds up SVD by matrix compression and implicit inner product calculation.

It is well-known that SVD/PCA suffers from contamination of the normal subspace due to large outliers [27]. Such contamination is common in practice because the training data can contain big changes or outliers because of previously performed maintenance or failures. To address this shortcoming, we propose a new **M**ultiscale **R**obust **L**ocal **S**ubspace algorithm (MRLS) that can detect changes even when the baseline is contaminated with large outliers. Before we describe MRLS, we provide an overview of how subspace algorithms operate to discover the change-points.

2.2.1 Subspace Algorithms using SVD

For each performance metric, the goal is to test if time-series values before the maintenance are significantly different after the maintenance. We construct a matrix X for each performance time-series with N columns (each column corresponds to a day, and N^{th} column is the day of maintenance) and M rows (each row is a time-bin, *e.g.*. 5-minutes). $X(i,j)$ is the value of the performance metric on day j and time-bin i on day j. We need to identify changes on the N^{th} day, which is the day of maintenance.

Singular Value Decomposition (SVD). SVD can decompose any real-valued matrix X into three matrices such that

$$X = U\Sigma V^T \tag{1}$$

where the columns of U are the eigenvectors of XX^T, the columns of V are the eigenvectors of X^TX and the diagonal values in Σ are the singular values s_i. The singular values capture the energy of the time-series X and are typically arranged in decreasing order of energy ($s_i \geq s_{i+1}$).

The intuition behind applying SVD for change detection is that the normal subspace is captured by the first r singular values ($r \leq N$) with the most energy and the residual

[2] A trunk group is a collection of network elements and servers on which a call is routed.

subspace contains the outliers or changes. The residual signal is computed using $X_R = U\Sigma_R V^T$, where Σ_R contains the remaining $N-r$ singular values. $X_R(:,N)$ contains the residuals for the day of maintenance and follows a Gaussian distribution under the central limit theorem. The change-points during the day are those time-bins whose values are greater than $\mu+\tau\sigma$. μ and σ are the mean and standard deviation of the residual signal and τ is a threshold to test for significance. For standard Gaussian distribution, using value of τ as 2.33 achieves an accuracy of 99%.

Global versus local subspace. When we use all the time-bins within a day to construct the matrix X, we refer to the method as global subspace (GS). If instead, we focus on a few time-bins around the maintenance time, we call the method a local subspace (LS). In our implementation, we use a few hours before and after the maintenance time to construct the matrix for local subspace method. The matrix contains fewer rows than the one used for global subspace. The advantage of using a local subspace versus a global approach is we can filter any changes outside the time segment of interest, which otherwise would contaminate the global normal subspace. This makes sense in practice, because changes due to failures or erroneous conditions can occur at a different time than the maintenance. However, local subspace using the regular SVD cannot still mitigate the problems due to contamination within the local time segment.

2.2.2 *MRLS: Multiscale Robust Local Subspace*

Since the original SVD uses l_2-norm to compute the low-rank normal subspace, it suffers from inaccurate detection and high false positives when there are large outliers. Recently, there have been proposals to use the l_1-norm instead of l_2 because it is more robust to outliers.

Robust subspace computation using l_1-norm. [19, 34] show that for sparse residual matrix X_R and low-rank matrix X_N capturing the normal subspace, one can solve the following constrained minimization problem to exactly recover the normal subspace:

$$\min_{X_N,X_R} \|X_N\|_* + \lambda\|X_R\|_1, \quad \text{subject to } X = X_N + X_R \qquad (2)$$

where $\|\cdot\|_*$ denotes the nuclear norm of a matrix (*i.e.*, the sum of its singular values), $\|\cdot\|_1$ denotes the sum or l_1-norm of the absolute values of matrix entries, and λ is a regularization parameter. The optimization (2), referred to as Robust SVD, can be treated as a general convex optimization problem and can be solved using any off-the-shelf interior point solvers after being re-formulated as a semi-definite program.

We choose $\lambda = \frac{1}{\sqrt{\max(M,N)}}$, as described in [4]. M is the number of rows or time-bins and N is the number of columns or days for matrix X. Thus, the regularization parameter λ can be fixed and requires no training or tuning.

We develop an Alternating Direction Method (ADM) motivated by [19] to scale for large matrices. The key idea is to use augmented Lagrange multipliers for solving constrained optimization problems of the kind:

$$\min f(Y), \quad \text{subject to } h(Y) = 0 \qquad (3)$$

where $f: \Re^n \rightarrow \Re$ and $h: \Re^n \rightarrow \Re^m$. The augmented Lagrangian function is

$$L(Y,Z,\mu) = f(Y) + \langle Z, h(Y)\rangle + \frac{\mu}{2}\|h(Y)\|_F^2 \qquad (4)$$

where μ is a positive scalar, Z is the Lagrangian multipliers, $\langle A,B\rangle$ is the trace norm of $A^T B$ and $\|\cdot\|_F$ is the Frobenius norm.

For the optimization problem (2), we apply the augmented Lagrange multiplier method using

$$Y = (X_N, X_R), \;\; f(Y) = \|X_N\|_* + \lambda\|X_R\|_1, \;\; h(Y) = X - X_N - X_R \qquad (5)$$

Then, the Lagrangian function is:

$$L(X_N,X_R,Z,\mu) = \|X_N\|_* + \lambda\|X_R\|_1 + \langle Z, X - X_N - X_R\rangle + \frac{\mu}{2}\|X - X_N - X_R\|_F^2 \qquad (6)$$

Our alternating direction method then progresses in an iterative fashion. During each iteration, we alternate among the optimization of each one of X_N, X_R and Z while fixing the other variables. The procedure is guaranteed to converge quickly if we increase μ by a constant factor $\rho \geq 1$ during each iteration. We can further improve efficiency by replacing exact optimization with approximate optimization during each iteration.

Robust thresholding for detecting change-points. Once the residual signal X_R is obtained via robust computation using l_1-norm, the next step is to analyze the residues on the day of the maintenance *i.e.*, $X_R(:,N)$. The mean and standard deviation tests for Gaussian distribution are sensitive to outliers and we might miss some of the genuine significant changes. To overcome that, we propose to use a robust thresholding method by replacing the mean with median and standard deviation with median absolute deviation (MAD). This method is called the *Hampel identifier* [8] and is known to be effective even during the presence of outliers.

MAD for the residual signal on the day of maintenance is defined as the median of the absolute deviations from the maintenance day residual median:

$$\text{MAD} = \text{median}_i(|X_R(i,N) - \text{median}_j(X_R(j,N)|) \qquad (7)$$

A time-bin in the residual signal for the day of maintenance is considered to have a significant change if its value is greater than (median+τ MAD). For accuracy of 99% and distribution tail corresponding to 1%, $\tau = 2.33 * 1.4826$. We have a multiplicative factor of 1.4826 because $\sigma \approx 1.4826$ MAD.

Limitations of l_1-norm based robust subspace method. The robust method described above can deal well with spikes, however its fails to accurately construct the low-rank normal subspace in presence of high energy outliers like level-shifts or ramp-ups. We confirm this using experiments described in Section 3. The high-energy changes can occur either due to process changes, software and hardware upgrades. Thus, it is important to improve the robustness of the change detection algorithms to all sorts of high energy outliers.

Multiscale differencing. We apply multiscale differencing on the performance time-series and then compute the low-rank normal subspace using l_1-norm. Differencing has a nice property of converting big level-shifts or ramp-ups to spikes.

We use multiscale Haar wavelet transformation to achieve differencing at multiple time scales. The wavelet transform is applied to each column of the performance time-series X, thereby producing two matrices X_a^W and X_d^W consisting of the approximation (or, averaging) and detail (or, differencing) coefficients. The difference coefficients X_d^W are then input to the low-rank subspace computation using l_1-norm that can then handle the spikes in coefficients (if any) and accurately construct the normal subspace. The intuition behind applying the wavelet analysis is not de-noising, but de-correlating the data and turning high energy outliers like level-shifts, ramp-ups into spikes in wavelet coefficients. We perform a local subspace analysis by explicitly focusing on time-bins around the maintenance time. We call this approach **M**ultiscale **R**obust **L**ocal **S**ubspace (MRLS).

Filtering operationally insignificant changes. PRISM correlates the change-points on the day of the maintenance with the actual time of maintenance to ensure that there is atleast one change after the maintenance time. Through interactions with operations teams and case study analysis, we have found that if there is any change within the start and end of maintenance window, then in a majority of cases, it is an expected change. This is because of the nature of the maintenance activity (*e.g.*, router CPU often spikes within the maintenance window). Thus it is important to filter out these expected changes within the maintenance start and end times and only focus on changes after the completion of the maintenance.

3. MRLS EVALUATION

In this section, we present evaluation of MRLS using data collected from four large operational networks: tier-1 backbone, VoIP, IPTV and 3G cellular. First, using synthetic injection of changes into operational data, we compare MRLS with other subspace methods and show that it can accurately identify changes with a low false alarm ratio. Second, we demonstrate that MRLS is robust to contamination of the baseline with changes either due to other maintenance activities or failures. Third, we analyze the sensitivity to the number of eigenvectors selected for computing the normal subspace. Finally, we show that MRLS can scale to a large number of time-series data and rapidly detect changes, thereby demonstrating its feasibility in an online setting.

3.1 Data Sets

The service provider collects a plethora of data related to configuration, maintenance, workflow, faults and performance. In this section, we provide a brief overview of the network and data sets that are relevant to the rest of the paper. Fig.3 shows the architecture for the four networks.

Tier-1 Backbone Network. The tier-1 backbone network consists of devices from multiple vendors, and with different roles. We focus on six categories: access routers (AR), core routers (CR), aggregate routers (AGG), route reflectors (RR), edge routers (ER), and layer-2 aggregation devices (L2). We conduct our analysis using device syslogs, SNMP MIBs (device CPU and memory utilization levels, link loads, packet errors and losses), workflow logs (history of commands executed on the devices) and maintenance events.

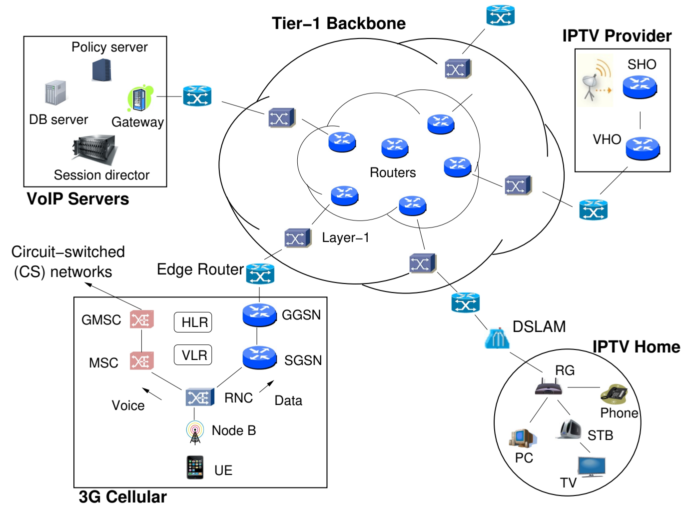

Figure 3: Architecture of four operational networks (tier-1 backbone, VoIP, IPTV and 3G cellular).

Voice over IP (VoIP) Service. We focus on four roles of VoIP servers: (i) gateway switches (GSX), (ii) policy servers (PSX) that handle call routing, (iii) session directors (SDA), and (iv) database servers (DB) that store call information. We analyze measurements collected at VoIP servers, which include CPU, memory utilizations, call failures and maintenance events. The call statistics are summarized for each trunk group traversing the server.

IPTV Service. In an IPTV system, IP packets are used to deliver live TV streams from the video servers to residential homes. The IPTV provider network consists of SHO (Super Head-end Office) which is the primary source of TV content, VHO (Video Hub Office) responsible for each metropolitan area, intermediate switches, and DSLAM (Digital Subscriber Line Access Multiplexer) that multiplexes signals to multiple homes. The home network consists of an RG (Residential Gateway), and a Set Top Box (STB) that serves as an encoder and decoder for TV signals. We focus on software crashes and upgrades on the STB, and syslogs collected at devices in the provider network.

3G Cellular Service. The 3G cellular system provides voice and data services and consists of a radio access network (RAN) that handles all radio-related capabilities, and a core network (CN) that is responsible for switching and routing of calls and data connections to external networks. The user equipment (UE) interfaces with the RAN. The key components of a RAN are a Node B that communicates with the UE and a Radio Network Controller (RNC) that controls radio resources within a domain, and manages the connections to the UE. The core network consists of SGSN (Serving GPRS [3] Support Node) for data routing, GGSN (gateway to external packet switched networks), MSC (Mobile Services Switching Center) to switch voice calls, GMSC (gateway to external circuit switched networks), HLR (home lo-

[3]GPRS stands for General Packet Radio Service.

cation register) which is a database of subscriber information and VLR (visitor location register) to support seamless roaming. In the paper, we focus on NetCool [25] alarms collected at the Node B and RNC within the RAN, and user feedback regarding service quality using a recently deployed capability called Mark the Spot (MTS) [24]. User feedback is extremely invaluable to the service provider to track user quality of experience [26, 33]. NetCool data provides information about faults, connectivity issues, protocol state conditions, resource availability, voice and data quality issues.

3.2 Comparison of Subspace Methods

We now compare different subspace methods using operational network data that exhibit different characteristics such as variability, seasonality, random with no structure and stationarity. The goal is quantify the performance across different types of changes (such as spikes, level-shifts, ramp-ups) and under varying degrees of baseline contamination. We consider the following subspace methods for comparison.

1. *Global Subspace* (GS): It applies regular SVD on a matrix with all time-bins on a day.
2. *Local Subspace* (LS): It applies regular SVD on a matrix with a few time-bins around the maintenance time.
3. *Multiscale Global Subspace* (MGS): It applies regular SVD on wavelet coefficients generated using a matrix with all time-bins on a day.
4. *Multiscale Local Subspace* (MLS): It applies regular SVD on wavelet coefficients generated using a matrix with a few time-bins around the maintenance time.
5. *Robust Global Subspace* (RGS): It applies robust SVD using l_1 norm on a matrix with all time-bins on a day.
6. *Robust Local Subspace* (RLS): It applies robust SVD using l_1 norm on a matrix with a few time-bins around the maintenance time.
7. *Multiscale Robust Global Subspace* (MRGS): It applies robust SVD using l_1 norm on wavelet coefficients generated using a matrix with all time-bins on a day.
8. *Multiscale Robust Local Subspace* (MRLS): It applies robust SVD using l_1 norm on wavelet coefficients generated using a matrix with a few time-bins around the maintenance time.

3.2.1 Methodology

Evaluating the accuracy of change detectors using real-world operational data is extremely challenging because of the lack of ground truth information about the changes. In our evaluation, we synthetically inject known change signatures into the time-series data and compute the accuracy for different detectors. We also test the sensitivity of the detectors to the change characteristics such as its duration, magnitude and degree of baseline contamination.

Time-series construction. For our synthetic injection experiments, we use operational data collected over 30 days. We categorize the data using four characteristics: *variability*, *seasonality*, *random* (no structure) and *stationarity*. We summarize the data we use in Table 1. For each data source,

Type	Performance Metric	Count	Operational Network
Variability	Router CPU utilization	7	Tier-1 backbone
Seasonality	Link load	4	Tier-1 backbone
	Server CPU utilization	3	VoIP
Random	Packet errors	1	Tier-1 backbone
	Call failures	1	VoIP
	BGP hold timer expiration	1	Tier-1 backbone
	Multicast flaps	1	IPTV
	Hardware faults	1	IPTV
	STB software crash	1	IPTV
	Coverage issues (MTS)	1	3G cellular
Stationarity	Router memory utilization	7	Tier-1 backbone

Table 1: Performance data collected over 30 days for comparison of subspace methods. We group them by their characteristics.

we construct a time-series containing events divided into 5-minute bins. For example, CPU utilization for a device over 30 days would be mapped to 288*30 sample points in the time-series (each day contains 288 5-minute bins).

Injection of changes on day of maintenance. For 30-days worth of data for each time-series, we assume that the last day is the day on which maintenance is being performed. We inject a change on the day of maintenance with the start time of the change being the time at which maintenance is complete. The change can either be a spike, a temporary level-shift (TL-shift) that lasts a few 5-minute bins, a temporary ramp-up (T-rampup) that ramps up and last for a few bins, a level-shift (L-shift) that lasts for the entire day and a ramp-up that last till the end of the day.

Injection of baseline contamination. We inject changes with different signatures on the days before the maintenance (*i.e.*, any day between day 1 & 29). This will contaminate the baseline and make it harder to accurately compute the normal subspace. This process will test the robustness of different subspace methods. The contamination change signatures can be spikes, temp level shifts, temp ramp-up, level-shifts for the entire day, or ramp-up till the end of the day.

Computing false positives and true positives. If for a change injected on the day of maintenance, if the subspace method correctly identifies the change, then we label it as a true positive of the method. False negatives are those changes that are missed by the subspace methods. If the method identifies the change, when the change was not injected, it is labeled as a false positive. True negatives capture instances that were not identified as changes when none were injected. For the rest of the paper, we only show true positives and false positives, because the other two can be inferred. We use a time margin of 25 minutes before and after the maintenance time within which if there exists a change-point, then it will be labeled as a change detected by the method. We construct samples by first randomly picking a maintenance time on the last day and then injecting a specific type of change with a specific magnitude. We repeat this procedure for each type of performance data as outlined in Table 1. For experiments with baseline contamination, we control three parameters: (i) number of changes injected on days before the maintenance (days are picked randomly), (ii) magnitude of the contamination, and (iii) signature of the change. We use the variance threshold of 0.9 to select the eigenvectors

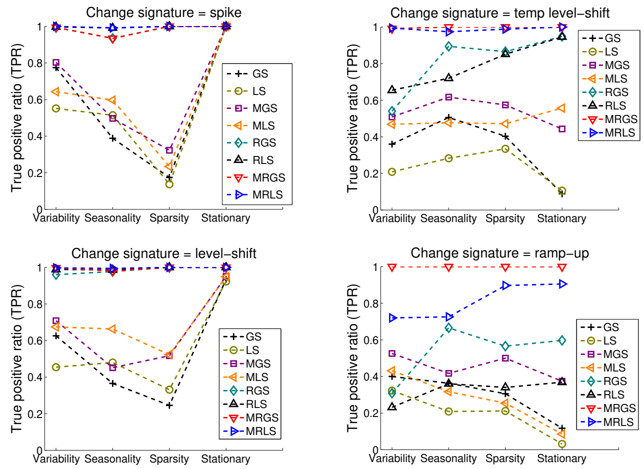

Figure 4: True positives across different data characteristics (X-axis) and changes injected on the day of maintenance.

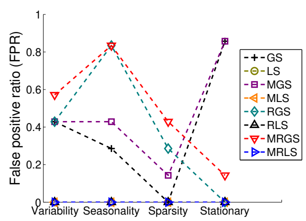

Figure 5: False positives across different data characteristics (X-axis) and no changes injected on the day of maintenance.

to construct the normal subspace. We later analyze the sensitivity of the subspace methods to the number of eigenvectors selected, which is captured by the variance threshold. A higher threshold means more eigenvectors are selected and this will account for higher variance in the normal subspace.

3.2.2 *Comparison across Change Signatures*

Fig. 4 shows the true positive ratio (TPR) for different data characteristics (X-axis) and different changes. We make the following observations: (i) Multiscale robust subspace methods (MRGS and MRLS) perform better than regular subspace and robust subspace using l_1 norm. (ii) Regular subspace methods perform badly in several cases because of their inability to accurately identify changes with small magnitude. (iii) Wavelet transformation helps improve the TPR for regular subspace methods as well; MGS/MLS perform better than their corresponding GS/LS. (iv) For all types of changes except ramp-up, the true positive ratio for MRGS and MRLS is similar. For ramp-up type of changes, the ramping up sometimes is slower and thus are missed by MRLS within a local time-window.

It may appear that MRGS is better compared to all methods, but along with a high true positive ratio, it also has a high false positive ratio (Fig. 5). MRLS achieves a high true positive and a low false positive, demonstrating its superior performance to other detectors. The local methods achieve a better FPR compared to the global methods.

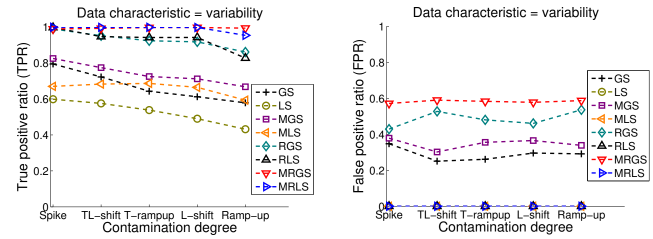

Figure 6: True positives and false positives for detectors applied to data with high variability and baseline contamination on the days before maintenance.

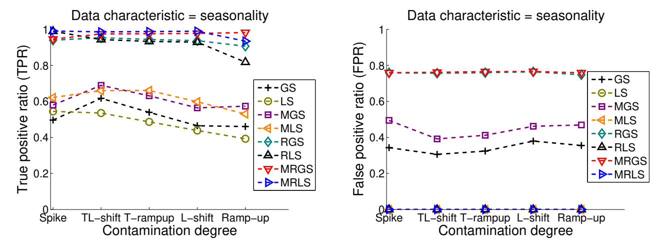

Figure 7: True positives and false positives for detectors applied to data with high seasonality and baseline contamination on the days before maintenance.

3.2.3 *Comparison under Baseline Contamination*

We now examine the impact of baseline contamination on the true positives and false positives for the detectors. Figures 6 and 7 show the TPR and FPR for data characteristics of variability, seasonality. Due to space restrictions, we do not show results for random and stationarity data characteristics. For both figures, X-axis represents the different signatures of the changes used to contaminate the baseline (days before maintenance). For each contaminated change signature, we vary the type of change on the day of maintenance and identify an aggregate true/false positive ratios.

Fig. 6 shows the true positive ratio (TPR) and false positive ratio (FPR) for the detectors when applied to data with high variability and varying degrees of baseline contamination. The TPR decreases as the contamination degree increases in terms of its energy from spike to level-shift to ramp-up. MRGS and MRLS achieve a much better TPR than other subspace methods indicating their robustness to baseline contamination. MRLS further achieves a low FPR indicating that performing multiscale robust subspace computation using a local time window around the maintenance time can eliminate other unrelated changes. For data with high seasonality, it becomes difficult for regular subspace methods to accurately identify the subspace as compared to data with high variability because of the underlying structure of the time-series. This is evident from the low TPR shown in Fig.7 as compared to Fig. 6. This problem exacerbates for data with no inherent structure or those have events recorded at random time intervals. For stationary data with almost negligible fluctuations, contaminated changes get amplified when transformed using wavelets and thus MGS has a very high false positive ratio.

In summary, our experiments across a wide variety of data, different change signatures and varying degrees of baseline

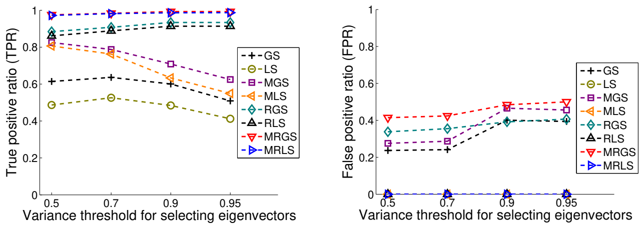

Figure 8: True positives and false positives when the variance threshold to select eigenvectors is varied.

contamination have shown that MRLS is not sensitive to underlying data properties as compared to others and achieves a high TPR at the same time maintaining low false alarms.

3.2.4 *Sensitivity to Selection of Eigenvectors*

In our sensitivity experiments, we vary the variance threshold to select the number of principal components or eigenvectors. Fig. 8 shows that regular subspace methods GS and LS are very sensitive to the selection of eigenvectors for constructing the normal subspace. This is consistent with the finding in [27]. A low variance threshold allows a small number of eigenvectors to construct the normal subspace and very small deviations in the time-series which are more likely to be false alarms propagate into the residual signal and get classified as changes. Under contamination, genuine changes might spill over to the normal subspace leading to missed detections for higher variance threshold. Our MRLS algorithm is robust to the variance thresholds. For the prototype system, we use variance threshold of 0.9.

3.2.5 *Computation Time*

We compare the computation time for different subspace methods for a single time-series containing 30 days worth of data with 5-minute time bins. As the variance threshold for selecting eigenvectors to construct the normal subspace increases, there is a small increase in computation time for robust methods. Global subspace methods take more time to complete than local because of the size of the input data. l_1-norm is known to be computationally expensive than regular l_2-norm and thus the robust methods take longer than regular subspace methods. Even though our MRLS algorithm takes longer than LS or MLS, it completes in less than 0.15 seconds for each time-series. With parallel processing, we can easily scale MRLS for rapid detection of maintenance induced changes to a large number of performance metrics.

4. OPERATIONAL EXPERIENCES

We now describe our case study experiences in applying PRISM on four large operational networks: tier-1 backbone, Voice over IP (VoIP), Internet Television (IPTV) and 3G cellular network. To evaluate PRISM before deploying at scale, we ran PRISM across historical data for these networks and services. The case studies discussed here demonstrate the effectiveness of PRISM in capturing the performance impacts of maintenance. In all the case study findings, PRISM was able to rapidly identify performance impacts after the maintenance - in some cases validating incidents that had been historically detected, while in other cases revealing issues that had flown under the operations radar (and in some cases, the issues were still ongoing).

We consider different types of maintenance activities ranging from operating system (OS) upgrades, software patches, configuration changes and circuit rolls. We construct the performance time-series using SNMP measurements (router CPU, memory utilization, link loads and packet loss) and router syslogs in tier-1 backbone, trunk group call statistics and server CPU/memory utilizations in VoIP, software crashes on STB in IPTV and NetCool alarms, performance counters and Mark The Spot (MTS) user feedback in 3G cellular network. Router syslogs and NetCool data contribute to a large number of performance time-series data. The large count for Set-Top-Box (STB) OS upgrades is due to the large number of STBs. We group the software crashes at the STBs by regions to give us one time-series per region.

For each maintenance activity, we collect thirty days worth of performance time-series data and for each time-series, use MRLS to identify if there are statistically significant changes on the day of the maintenance. Table 2 shows the summary of PRISM results. After sharing the results with the operations teams and conducting further investigations (time-series visualizations, browsing through workflow and configuration logs), we summarize our results into three categories as follows: (i) We confirmed some of the previously known impacts of maintenance. This boosted ours as well as operations confidence in PRISM's functionality. (ii) In a few cases, PRISM also revealed previously unknown impacts demonstrating its potential to be used in online settings. (iii) For some performance changes that correlate with the maintenance, we haven't yet been able to confirm a causal relation. Investigations are underway.

4.1 Impact of Circuit-roll in Tier-1 Backbone

In our first case study, we focus on identifying the performance impacts of circuit rolls in tier-1 backbone networks. A circuit roll is a process of moving a physical circuit (IP link) from one router port to another router port (potentially on a different router). These circuit rolls are typically executed before and after line card maintenance or for router replacement. One important goal is obviously to minimize the impact on traffic. To achieve this, the traffic is first moved away from the device to be replaced so that maintenance can be carried out with minimal performance impact. This can be achieved with minimal service impacts in the core by using the redundant paths in the network. However, the same cannot be achieved on customer-facing interfaces on access routers (AR). Typically, customers connect to an ISP via a single AR to the tier-1 network. Thus, while performing maintenance on the ARs, the circuit between the customer and AR must be seamlessly migrated with negligible impact.

We applied PRISM for circuit rolls across multiple devices in the backbone network using a wide array of performance measurements ranging from router CPU, memory utilization, packet errors and losses, and router syslogs. PRISM captured interesting changes in behavior that were previously not known to the network operators.

Network Type	Type of Maintenance	Type of Network element	Count	Influence Group	Performance Impact	Known/Unknown to Operations
Tier-1 Backbone	OS upgrade	Peering gateway	12	Same router	Upward level-shift in CPU	Known
	OS upgrade	Aggregate router	9	Same router	Upward level-shift in memory	Unknown
	OS upgrade	Edge router	6	Same router	Upward level-shift in CPU	Unknown
	Circuit roll	Access router	15	Same router	Upward level-shift in CPU	Unknown
	Circuit roll	Core router	1	All interfaces on router	Upward level-shift in packet errors & losses	Unknown
IPTV	OS upgrade	Set-Top-Box	556194	Region	Upward level-shift in software crash rates	Known
VoIP	Config. change	Application server	1	Trunk group	Upward spike in blocked calls	Known
	OS upgrade	Application server	7	Trunk group	Upward spike in blocked calls	Known
3G Cellular	Software patch	RNC	5	Same RNC	Upward level-shift in call drops	Known
	Software patch	RNC	10	Same RNC	Downward level-shift in call drops	Known
	OS upgrade	Switch	1	Neighboring RNC	Spike in blocked calls	Known
	OS upgrade	Edge router	7	Same router	Upward level-shift in CPU	Known
	Software patch	RNC	11	Same RNC	Upward level-shift in RAB failures	Known

Table 2: Maintenance induced performance impacts identified by PRISM.

1. **Circuit roll on AR induces significant changes in CPU utilizations.** We applied PRISM to monitor circuit rolls executed during the retirement of old access routers (ARs). Thousands of customers were being migrated from old routers onto new routers. PRISM revealed that on some small percentage of routers, immediately after some circuit rolls were completed, the old routers saw an increase in the router CPU values being reported. At first we thought that this might be due to de-aggregation of the static routes and announcements into iBGP. However, after careful examination of the router configuration snapshots on the days of the circuit rolls, we found diverging cases across routers. For some of the cases, we found that there was IP address duplication causing some of the packets to be sent to the old AR and thus causing CPU increases. However, there are still some cases for which we don't have an explanation. We are currently investigating this with the operations team and domain experts. However, the critical point here is that this increase in CPU load for certain migrations had not been detected until PRISM was introduced because it is simply impossible to manually monitor all time series as thousands of customers are migrated over an extended period of time. CPU increases were not expected, and thus no one was watching each and every customer migration for their CPU impact (instead relying on spot checks). Fortunately, the impact was minimal as these were occurring on routers from which the customers had been removed. However, it is still unexpected behavior that places some risk to the network and is indicative of routers behaving suspiciously, and must be explained and managed. This clearly demonstrates the value of PRISM and its ability to pick up unexpected changes in device performance.

2. **Circuit roll on a core router increases packet error rates.** In another case example, PRISM revealed that a circuit roll on a core router interface was time-correlated with an increase in packet errors and losses on a small set of other interfaces on the same router. The most likely explanation for this occurrence is that the fibers or line cards in question were likely impacted through accidental human involvement (probably bumped) during the maintenance. However, operations personnel monitoring the circuit roll would have missed this negative side effect as they would have been focused on monitoring the circuit being moved and could not scale to manually monitoring all other circuits on the router. Clearly, if the issues were significant enough an alarm would have been automatically raised; however, even a small but significant increase in errors during maintenance could be readily repaired if rapidly detected while the technician was still on site.

4.2 Impact of Server Software Changes in VoIP

In our second case study, we used PRISM to detect the performance impact of software changes on Voice over IP (VoIP) servers. The VoIP servers are responsible for managing the phone calls over the IP network. The correct design and implementation of the server software is essential for maintaining high quality of service. However, the server software is continually evolving either via new application installations to support new features, patches to fix bugs, or upgrades to improve performance. Extensive lab testing is performed on each change before it is deployed. However, unpredicted impacts might be observed in operational environments because of the large scale, complex interactions across protocols, and vendors, and overload conditions. Unexpected behaviors in the server software can lead to high call failures, causing significant customer distress. Thus, it is important to carefully monitor the server software behaviors and rapidly detect any performance impacts of changes.

We applied PRISM to track the impacts of server software upgrades on server CPU and memory utilization, and trunk group call statistics. Recall that a trunk group is used to route calls across different servers. PRISM identified the influence group for a software change on a server as a collection of all the trunk groups traversing the server.

1. **Increase in call failures due to a configuration change.** A configuration change had previously been applied to alter the flow of certain VoIP calls to bypass call control elements. Using PRISM, we detected that immediately after the configuration change, there was a significant increase in the fraction of call failures across multiple trunk groups. We verified through event logs that this was indeed a real, significant issue that was thoroughly investigated by Operations personnel at the time. The issue had arisen due to an unexpected timing issue that created a build up of stale sessions on application servers causing overload conditions and eventually resulting in call failures. To resolve the overload condition, a manual failover of the application servers was performed and

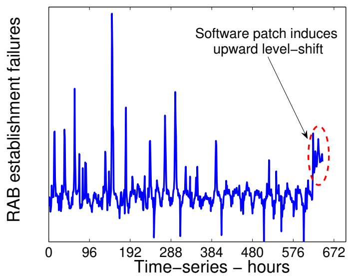

Figure 9: Time-series showing the increase in the RAB establishment failures induced by software patch on a RNC. The time-series has a high degree of seasonality with a few spikes on days before the patch is applied. The values on Y-axis are explicitly not shown due to proprietary reasons.

the configuration change was restored to route via the call control elements.

4.3 Impact of STB Software Upgrades in IPTV

Set-Top-Boxes (STBs) connect to TVs and are responsible for encoding and decoding TV signals. Their health is vital to guarantee good service to IPTV customers and their performance is continuously tracked to avoid service degradations. When the STB software is upgraded, the operations team carefully monitors for behaviors that may be indicative of new software bugs. Such conditions need to be detected rapidly to minimize customer impact. However, given the vast number of STBs deployed (in the case studied here, we had seven million being analyzed), it becomes incredibly challenging to monitor the impacts of upgrades. We thus applied PRISM to the STB logs to monitor the impact of an earlier software upgrade.

1. **OS upgrades on STB increases software crash rates.** The deployment of the new OS version was scheduled across a large number of STBs. PRISM automatically discovered that the OS upgrades had induced a significant increase in the software crash rates. This behavior was observed across multiple locations. We confirmed that Operations had previously been aware of this issue, and were actively engaged in resolving it. However, this confirmed that PRISM is effectively able to mine data associated with vast numbers of devices (in this case STBs) to identify patterns associated with software upgrades.

4.4 Impact of Software Changes in 3G

In our final case study, we report PRISM's findings associated with software changes to the Radio Network Controllers (RNCs) and switches in a 3G cellular network. The operations teams and radio engineers carefully monitor changes in data and voice service quality that may be induced by new software releases or patches. The large number of network elements and performance metrics makes it challenging to rapidly capture the impacts. The metrics we used in PRISM were NetCool alarms [25], performance measurements and Mark the Spot (MTS) logs [24].

1. **Changes in the call drop rate with the RNC software patches.** PRISM discovered using NetCool alarms that there was an increase in the call drops after the software patches were applied to the RNC. The change was also captured by PRISM in MTS feedback from users. We confirmed this finding with the operations team. Although Operations was made aware of the issue through traditional alarms, PRISM provides the added insight of rapidly associating the alarms with the maintenance activities. The Operations team had worked with the vendors and installed a new software patch to fix the problem. PRISM confirmed the decrease in the number of call drops with the new patch.

2. **Software installation on a switch impacts 3G voice calls.** Again validating a historical issue, PRISM revealed that a certain software upgrade on a switch unexpectedly resulted in an increase in blocked voice calls as observed via performance metrics reported on a neighboring RNC. However, the impact was not immediate - it was only observed as load increased later in the day. It was thus challenging for the Operations team at the time to relate the increased call blocked rate with the (much earlier) maintenance activities. However, PRISM immediately made the association and thus, if it had been deployed at the time, would have significantly reduced the incident duration and hence customer impact.

3. **RNC software patches increase the number of RAB establishment failures.** PRISM revealed that software patches on RNCs were time-correlated with significant increases in the number of RAB (Radio Access Bearer) establishment failures resulting in an impact on the data service quality. Fig. 9 shows the time-series plot of the impact. It illustrates that even when the time-series had high seasonality and few spikes on days before the patch was applied, PRISM could successfully detect the performance change (level-shift) on the day when the patch was applied. PRISM also identified that the behavior was observed only on a small fraction of the total number of RNCs in the cellular network. The Operations team put the software deployment on hold at other locations while investigating the issue with the RNC vendor.

We have thus demonstrated the effectiveness of PRISM across four different operational networks and services. Given this application of PRISM, the operations teams are planning on incorporating PRISM on an ongoing basis to rapidly capture the performance impacts of maintenance. PRISM is continuously run after the maintenance activity is complete and automatically picks up the service performance impact whenever it becomes evident as a statistically significant change in behavior.

5. RELATED WORK

There is a huge body of literature on network anomaly detection and diagnosis. Most prior work regarding in-network anomaly detection has focused on traffic or routing data. Principal Component Analysis (PCA) is one of the popular network-wide anomaly detectors [11, 16, 17]. Zhang *et al.* [35] proposed a single framework to capture a wide variety of detectors. Barford *et al.* [2] used wavelets to decompose the original signal into low-, mid-, and high-frequency components and then detect anomalies by close investigation of the high-frequency components. Zhang *et al.* [36] used compressive sensing to discover anomalies in traffic matrices. Ringberg *et al.* [27] showed that the performance of PCA is highly sensitive to the number of principal components chosen. Brauckhoff *et al.* [3] further showed that the sensitivities come from the lack of capturing temporal correlations and proposed to use Karhunen-Loeve Transform to improve robustness of PCA. PCA-GRID [6, 7] uses Projection Pursuit and Median Absolute Deviation (MAD) instead of standard deviation as a robust way to deal with outliers. Rubinstein *et al.* [28] showed that data poisoning (or baseline contamination) can dramatically reduce the accuracy of PCA and they combat poisoning by using a robust Laplace cut-off threshold called ANTIDOTE. ASTUTE [30] uses the equilibrium property and correlation across flows to discover a new class of anomalies. BasisDetect [9] uses basis pursuit instead of PCA to decompose the signal into normal subspace and residual. Soule *et al.* [31] used Kalman filtering and multiscale analysis to improve false positives. Note that the use of multiscale analysis in [31] is different from ours; they use it to reduce false positives by ensuring that an anomaly is captured at all timescales. Whereas, our goal of multiscale transformation is de-correlation of data to convert high-energy anomalies likes level-shifts into spikes.

Troubleshooting using statistical correlation across multiple data sources is complementary to PRISM. SCORE [15], Shrink [13], Sherlock [1] and Orion [5] focus on discovering service-level dependencies. NICE [22], WISE [32], Giza [20], NetMedic [14], URCA [29] and Minerals [18] use statistical mining to identify time-series dependencies.

6. CONCLUSION

We presented the design and implementation of PRISM, a new tool for detecting changes in performance that are induced by maintenance. PRISM incorporates the operational knowledge of the impact of maintenance using an influence group and uses it to drive the time-series analysis. It uses a novel multiscale robust local subspace algorithm (MRLS) to accurately identify changes in performance even when the baseline is severely contaminated. We validate MRLS using data collected from four operational networks (tier-1 backbone, VoIP, IPTV, 3G cellular) and show that it outperforms other subspace algorithms and accurately identifies changes across a diverse set of data sources. PRISM has confirmed some of the earlier findings of operations and in some cases also discovered previously unknown issues.

Acknowledgement

We thank Aman Shaikh, Vishal Misra (our shepherd) and the CoNEXT anonymous reviewers for their insightful feedback on the paper. We are grateful to the network engineers and operations team for their invaluable help on the data collection, the case-study analysis and application of PRISM. We thank Mukta Mahimkar for recommending the name PRISM.

7. REFERENCES

[1] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Sigcomm*, 2007.

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

[3] D. Brauckhoff, K. Salamatian, and M. May. Applying PCA for traffic anomaly detection: Problems and solutions. In *Infocom*, 2009.

[4] E. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Preprint*, 2009.

[5] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl. Automating network application dependency discovery: Experiences, limitations, and new solutions. In *OSDI*, 2008.

[6] C. Croux, P. Filzmoser, and M. R. Oliveira. Algorithms for projection-pursuit robust principal component analysis. In *Chemometrics and Intelligent Laboratory Systems*, 2007.

[7] C. Croux and A. Ruiz-Gazen. High breakdown estimators for principal components: the projection-pursuit approach revisited. *J. Multivariate Analysis*, 2005.

[8] L. Davies and U. Gather. The identification of multiple outliers. *Journal of the American Statistical Association*, 1993.

[9] B. Eriksson, P. Barford, R. Bowden, N. Duffield, J. Sommers, and M. Roughan. Basisdetect: a model-based network event detection framework. In *IMC*, 2010.

[10] H. Hassani, S. Heravi, and A. Zhigljavsky. Forecasting european industrial production with singular spectrum analysis. *International Journal of Forecasting*, 2009.

[11] Y. Huang, N. Feamster, A. Lakhina, and J. J. Xu. Diagnosing network disruptions with network-wide analysis. In *Sigmetrics*, 2007.

[12] T. Ide and K. Tsuda. Change-point detection using krylov subspace learning. In *SIAM International Conference on Data Mining*, 2007.

[13] S. Kandula, D. Katabi, and J.-P. Vasseur. Shrink: A tool for failure diagnosis in IP networks. In *MineNet*, 2005.

[14] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *Sigcomm*, 2009.

[15] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. IP fault localization via risk modeling. In *NSDI*, 2005.

[16] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Sigcomm*, 2004.

[17] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Sigcomm*, 2005.

[18] F. Le, S. Lee, T. Wong, H. S. Kim, and D. Newcomb. Minerals: using data mining to detect router misconfigurations. In *MineNet*, 2006.

[19] Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *UIUC Tech. Report UILU-ENG-09-2215*, 2009.

[20] A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao. Towards automated performance diagnosis in a large IPTV network. In *Sigcomm*, 2009.

[21] A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and J. Emmons. Detecting the performance impact of upgrades in large operational networks. In *ACM SIGCOMM*, 2010.

[22] A. Mahimkar, J. Yates, Y. Zhang, A. Shaikh, J. Wang, Z. Ge, and C. T. Ee. Troubleshooting chronic conditions in large IP networks. In *CoNEXT*, 2008.

[23] V. Moskvina and A. Zhigljavsky. Change-point detection algorithm based on the singular-spectrum analysis. *Communication in Statistics: Simulation and Computation*, 2003.

[24] Mark The Spot. `http://itunes.apple.com/us/app/at-t-mark-the-spot/id338307313?mt=8`.

[25] Ibm Tivoli Netcool. `http://www.-01.ibm.com/software/tivoli/products/netcool-omnibus/`.

[26] T. Qiu, J. Feng, Z. Ge, J. Wang, J. Xu, and J. Yates. Listen to me if you can: tracking user experience of mobile network on social media. In *IMC*, 2010.

[27] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. In *Sigmetrics*, 2007.

[28] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar. ANTIDOTE: understanding and defending against poisoning of anomaly detectors. In *IMC*, 2009.

[29] F. Silveira and C. Diot. URCA: Pulling out anomalies by their root causes. In *INFOCOM*, 2010.

[30] F. Silveira, C. Diot, N. Taft, and R. Govindan. ASTUTE: Detecting a different class of traffic anomalies. In *SIGCOMM*, 2010.

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

[32] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering what-if deployment and configuration questions with WISE. In *Sigcomm*, 2008.

[33] Using twitter to track dropped calls. `http://www.technologyreview.com/printer_friendly_article.aspx?id=26668`.

[34] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Neural Information Processing Systems (NIPS)*, 2009.

[35] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *IMC*, 2005.

[36] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and internet traffic matrices. In *Sigcomm*, 2009.