Troubleshooting Chronic Conditions in Large IP Networks

Ajay Mahimkar[§], Jennifer Yates[‡], Yin Zhang[§], Aman Shaikh[‡], Jia Wang[‡], Zihui Ge[‡], Cheng Tien Ee[‡]
The University of Texas at Austin[§] AT&T Labs – Research [‡]
{mahimkar,yzhang}@cs.utexas.edu {jyates,ashaikh,jiawang,gezihui,ctee}@research.att.com

ABSTRACT

Chronic network conditions are caused by performance impairing events that occur intermittently over an extended period of time. Such conditions can cause repeated performance degradation to customers, and sometimes can even turn into serious hard failures. It is therefore critical to troubleshoot and repair chronic network conditions in a timely fashion in order to ensure high reliability and performance in large IP networks. Today, troubleshooting chronic conditions is often performed manually, making it a tedious, time-consuming and error-prone process.

In this paper, we present NICE (Network-wide Information Correlation and Exploration), a novel infrastructure that enables the troubleshooting of chronic network conditions by detecting and analyzing statistical correlations across multiple data sources. NICE uses a novel circular permutation test to determine the statistical significance of correlation. It also allows flexible analysis at various spatial granularity (*e.g.*, link, router, network level, etc.). We validate NICE using real measurement data collected at a tier-1 ISP network. The results are quite positive. We then apply NICE to troubleshoot real network issues in the tier-1 ISP network. In all three case studies conducted so far, NICE successfully uncovers previously unknown chronic network conditions, resulting in improved network operations.

Categories and Subject Descriptors

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

General Terms

Design, Management, Performance, Reliability

Keywords

Network Troubleshooting, Chronic Condition, Correlation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ACM CoNEXT 2008, December 10-12, 2008, Madrid, SPAIN
Copyright 2008 ACM 978-1-60558-210-8/08/0012 ...$5.00.

1. INTRODUCTION

Today, IP networks carry traffic from a diverse set of applications. Many applications, such as Voice over IP (VoIP), Internet television (IPTV), video conferencing, streaming media, Internet games, and online trading have stringent requirements for network reliability and performance. For these applications, the best effort service is no longer an acceptable mode of operation; service provider networks must maintain ultra-high reliability and performance. Doing so requires accurate and timely troubleshooting of anomalous network conditions, which occur due to accidents, natural disasters, equipment failures, software bugs, mis-configurations, or even malicious attacks [14]. However, troubleshooting anomalous network conditions is extremely challenging in large IP networks due to their massive scale, complicated topology, high protocol complexity, and continuously evolving nature through either software or hardware upgrades, configuration changes or traffic engineering.

Network management systems in service provider networks have traditionally focused on troubleshooting large network events that persist over extended periods of time, such as link failures. However, this approach leaves other network events flying under the operations teams' radar, while potentially impacting customers' performance. In many situations, such events are short in duration – the symptoms have disappeared even before a network operator can react to them. If the event is truly a one-off event, then there may be little point in investigating it further. However, for chronic (*i.e.*, recurring) events, the symptoms keep re-appearing and can cause repeated performance degradation to customers. In some cases, the chronic condition can aggravate over time before it eventually turns into a serious hard failure. For example, repeated link flaps may be observed over time before the link completely fails. Even when the chronic event does not result in any hard failure, the performance degradation caused can add up to significant impact to customers over time. It is therefore crucial to eliminate such chronic conditions from the network. As with hard failures, uncovering the underlying root cause(s) of the chronic conditions is necessary before the conditions can be permanently eliminated from the network.

Troubleshooting chronic network conditions can involve a complex combination of (targeted) network measurement, data mining, lab reproduction and detailed software and/or hardware analysis. However, even when lab reproduction or vendor intervention is required, accurate diagnosis of chronic conditions is at the heart of the troubleshooting process. For example, gleaning as much information from available network data can rapidly guide lab reproductions in honing into the conditions, scenarios and potential triggers necessary to induce the events of interest. We therefore focus on mining measurement data to effectively diagnose chronic network events in this paper.

Traditionally, troubleshooting in operational IP networks has focused on manually identifying network events that co-occur with the symptoms, and then inferring the root cause(s) from these events. Manual troubleshooting makes the task cumbersome, tedious and error-prone, thereby severely limiting the events that can be even examined. Innovative, automated solutions are required to effectively manage and repair chronic conditions in a timely fashion. As IP networks continue to grow in size and complexity, and applications continue to mandate stricter performance requirements, we expect this to become an ever increasingly important issue.

Approach. In this paper, we propose **NICE** (**N**etwork-wide **I**nformation **C**orrelation **E**xploration), a novel infrastructure that enables the troubleshooting of chronic network conditions by detecting and analyzing significant statistical correlations across multiple data sources. Our key observation is that today's IP networks are often heavily instrumented to continuously generate diverse measurements ranging from network-wide performance to routing protocol events and device syslogs. As a result, the same chronic network events tend to manifest themselves as correlated signals in multiple data sources. Thus, by analyzing significant correlations that network operators are unaware of based on their domain knowledge, we are more likely to uncover chronic network conditions that previously fly under the operators' radar.

We quantify the correlation between two event-series using Pearson product-moment correlation coefficient [7], arguably the most commonly used correlation measure. Compared with simple co-occurrence based analysis (which is the state of the art for troubleshooting chronic conditions in network operations), correlation coefficient also takes into account the event frequency of each individual event. This is particularly important when one type of events occur frequently, resulting in high event co-occurrence count even when the two event-series are not strongly correlated.

Challenges. Several significant challenges must be addressed in order to apply statistical correlation to troubleshoot chronic conditions in large IP networks:

1. **Large number of network event-series.** Today's IP networks are instrumented to generate a large amount of diverse measurements. There are potentially on the order of tens of thousands of individual event-series that one can create from network data collected from different spatial locations. Digging through this mound of data to troubleshoot chronic conditions and identifying the root causes is analogous to finding needles in a haystack.
2. **Distributed event propagation.** Some network events have only local impact (*e.g.*, a router reload event on one router is known to cause protocol session timeouts on the one-hop neighboring router), whereas others have network-wide impact (*e.g.*, OSPF re-convergence events can cause CPU utilization to increase on routers across the network). It is important to take such impact scope into account when troubleshooting chronic network conditions. For example, if two event-series are strongly correlated but are not within each other's impact scope, then the correlation between them may not be of interest to network operators. Blindly correlating event-series without considering their impact scope can easily lead to an information "snow" of results, many of which would be false alarms.
3. **Auto-correlation within event-series.** In order to test whether the correlation coefficient between two event-series is significant, several classic statistical significance tests can be applied. Unfortunately, in our context, we often observe significant auto-correlation within each individual event-series. Existing significance tests for correlation coefficient either do not account for such auto-correlation at all or do not account for it sufficiently. As a result, they tend to overestimate the significance of correlation coefficients, resulting in too many false alarms.
4. **Inaccurate event timestamps.** Delay often exists between when an event occurs and when it shows up in measurement data. There can be several reasons for this. First, the measurement process may be far away from the location for the event, resulting in a propagation delay. Second, many measurement processes are periodic. For example, SNMP link loads are often polled only once every five minutes. As a result, a maximum delay of one measurement cycle may be added after an event occurs and before it gets recorded. Third, the event may not be immediately observable to the measurement process (*e.g.*, due to damping mechanisms in routing updates). Finally, the clocks of distributed measurement processes may not be synchronized and typically only have limited resolution. Inaccurate event timestamps may result in seemingly violation of causality (*i.e.*, the cause may be recorded after the effect). So, it is crucial to make correlation analysis robust to inaccurate event timestamps.

Contributions. We design and implement the NICE infrastructure (see Section 3), which we believe is the first *flexible* and *scalable* infrastructure for troubleshooting chronic network conditions using statistical correlation testing across multiple network data sources. NICE addresses the above challenges as follows:

1. Instead of blindly mining for correlations across all possible pairs of network event-series, NICE starts with the *symptom*[1] event-series and outputs the list of other network events that have statistically significant correlations with the symptom. This list represents the potential root causes and impacts of the symptom event.

[1] A *symptom* event is an event that is indicative of a chronic network condition and is directly observable to the operations team (*e.g.*, CPU overload or high packet loss rate).

2. NICE incorporates a spatial proximity model and the hierarchical structure of network components to capture the impact scope of network events. For example, to troubleshoot packet loss observed on a path, NICE identifies strong statistical correlations between loss and other events that occur on routers and links on the same path. By using the spatial scope, NICE can significantly increase the fraction of potentially interesting correlations in the final reports.
3. NICE develops a novel circular permutation test of statistical correlation significance that can deal with auto-correlation within each event-series. The statistical correlation test ensures that most events that co-occur by chance are eliminated.
4. NICE copes with imprecise event timestamps by adding "padding margins" when converting raw measurement data into event-series.

We then systematically evaluate NICE using real network data collected from a tier-1 ISP's network (see Section 4). Our results demonstrate that the correlations considered significant by NICE generally agree with the domain expertise. When there is a disagreement, it is typically caused by either our imperfect domain knowledge, measurement artifact, or genuine network chronic conditions. Encouraged by the validation results, we have used NICE to troubleshoot real network issues in collaboration with operators of the tier-1 ISP. Our deployment experience has been very positive. In all three case studies that we conducted (see Section 5), NICE uncovered previously unknown chronic network conditions that are performance impacting. Remedy actions have been taken as a result of the NICE reports. NICE is becoming a powerful troubleshooting tool within the tier-1 ISP.

2. NETWORK DATA

ISPs today collect a plethora of data related to fault and performance from the network. Below we provide a brief overview of the data collected by the tier-1 ISP. We mainly focus on data sources that are relevant to the rest of the paper.

2.1 Data Sources

Layer-1 Alarms. The ISP collects standard alarms and performance monitoring metrics reported by the layer-1 devices used to inter-connect routers in different locations. These alarms indicate link failures, performance impairments (*e.g.*, high bit error rates, loss of signal or frame), and protection switching events occurring at the physical layer.

Router Syslogs. Syslog messages provide information about states, state changes, and error conditions encountered by routers and other devices. The exact type and format of these messages depend on the manufacturer and model of the router. Examples include messages related to timer expirations for routing protocols, state changes for routing sessions, and errors in the internal functioning of routers.

SNMP. SNMP MIBs and traps provide a standardized way of collecting performance and fault data from network elements. The ISP pulls various MIB elements such as the number of bytes transmitted and received through interfaces, and CPU utilization at routers through periodic polling.

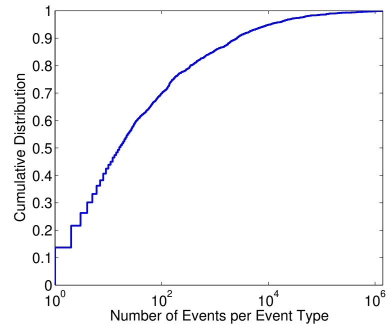

Figure 1: CDF of event counts per event type.

End-to-end Performance Measurements. The ISP network has an infrastructure of servers covering major PoPs (Point of Presence). Active data probes are sent in a periodic fashion between server pairs. These active probes provide measurements regarding delay and loss between PoPs.

Routing Data. The ISP uses OSPF as its intra-domain routing protocol, and collects OSPF LSAs (Link State Advertisements) using an OSPF Monitor [17]. We focus on the LSAs that indicate events caused by maintenance and failures in the network. We also use OSPF data to infer how traffic is routed across the backbone, and when and how routes are impacted (re-routed) by network events.

Router Command Logs. The router command logs provide a history of commands executed on routers through their command line interface (CLI).

Router Configuration. Router configuration information is crucial for mapping interfaces to routers, binding IP layer and layer-1/2 information of interfaces, and mapping IP links to their OSPF areas.

2.2 Data Characteristics

The measurement data collected by these systems are diverse. For example, some data sources consist of point events (*i.e.*, events with zero duration) such as layer-1 alarms, syslog messages, router commands, routing events. Some data sources, on the other hand, consist of range events. For example, CPU utilization pulled by SNMP is reported periodically over a fixed time interval. Similarly, loss and delay are reported over intervals used by the active probing system.

The data also vary in terms of number of "data points" over a time period. Fig. 1 shows the distribution of counts per event type over a one month time interval at the tier-1 ISP. We can observe significant diversity in the frequency of occurrence for different types of events. For example, around 15% event types have a single occurrence, whereas around 5% event types have counts greater than 10000 times. The mean and maximum number of occurrences are 8269.4 and 1759821 respectively.

Note also that a particular event may manifest itself in multiple different data sources. For example, a link going down can show up in router syslogs as layer-1/2 problems. This event can also trigger breakdown of a routing session

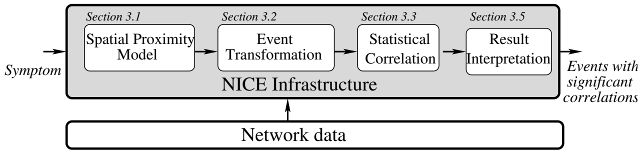

Figure 2: NICE infrastructure.

showing up in router syslogs and/or route monitors. Clearly, finding root causes amongst such large and diverse sets of data presents a huge challenge to the operators of ISPs.

3. NICE INFRASTRUCTURE

Troubleshooting chronic network conditions is often an ad hoc and cumbersome manual process, with experienced network operators fumbling through a large collection of heterogeneous data sources, looking for patterns of event co-occurrence, and conducting reasoning and diagnosis on a case-by-case basis. In this section, we describe our NICE infrastructure that can turn such ad hoc manual process into a simple and systematic task. Our design focuses on three main aspects for chronic troubleshooting: (i) a data model that unifies different types of event-series; (ii) a spatial model that captures the scope of impact for different events in a network; and (iii) an auto-correlation compensated correlation metric and evaluation process that automates the obscure co-occurrence based reasoning by network operators. Fig. 2 shows the NICE infrastructure that takes as input a symptom event for troubleshooting and outputs a list of other network events that have significant correlations with the symptom.

3.1 Formalization of Event Co-occurrence

When troubleshooting a chronic network condition, the network operators most often examine what other events occurred together, *i.e.*, co-occurred, with the symptom event. Some questions that arise in this context are: what constitutes an event? what qualifies as a co-occurrence? What spatial proximity of symptom event to consider? Since co-occurrence is also the basis for computing statistical correlation, below we address these questions and in the process formalize the notion of co-occurrence.

Basic Events. A network event is characterized by the *type* of the event, the *location* of the event, and the *time* of the event. The event type describes what happened: some examples include a period of router CPU overload, an OSPF adjacency reset, or an excessive loss of active probing packets from one router to another. The event location indicates where the event took place or was observed. This can be a specific interface on a router, a specific router, a link, a sequence of links (*i.e.*, a path), or a subset of routers (*e.g.*, an OSPF area). With respect to the event time, there are two types of events: a point event or a range event. Point events such as router syslog messages do not have a duration, and hence only a single time-stamp is associated with the event. Range events, on the other hand, are those that either take place over a period of time or are observed over a measurement interval too coarse to collapse into an exact time instance. For example, a spike in traffic volume is detected by

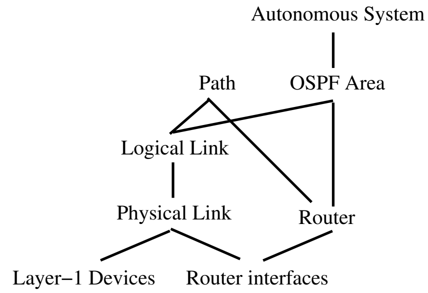

Figure 3: Location hierarchy for network events.

two consecutive polls of SNMP counters. So we define the event as a range event with the start time being the time of the first poll and the end time being the time of the second.

Event Composition. Basic events can be combined into composite events through logical operators such as intersection (AND), union (OR), and negation (NOT). In many cases, it is the composite events that contain the information for troubleshooting. For example, an OSPF adjacency down event followed (within a small time window) by an OSPF adjacency up constitutes an OSPF link-flap event. Logically, OSPF link-flap is the intersection of OSPF adjacency down and OSPF adjacency up events in the same time window. Our data model in Section 3.2 greatly simplifies such logical operations. As another example, the loss of probe packets from one router to another is expected to occur when congestion events (defined by high buffer overflow counts) take place on the data path between the two routers. Such path congestion event is a logical union of the congestion events at each of the links on the path. NICE supports event composition both over event type (as in the first example) and over event location (as in the second example).

Co-occurrence in Time. Strictly speaking, for two events to be co-occurring, they must happen at the same time. However, for troubleshooting chronic network conditions, we often care about causal relationships in which the impact of a cause event propagates through the network, usually delayed by some timers, finally manifesting as the symptom event. Hence, we define two events to be co-occurring in time when they occur within a time window of S seconds, where S is typically in the order of 10s of seconds. A large value of S means one can work with low precision of the event timestamp, and lack of perfect time synchronization amongst data sources. This is often the case in reality.

Co-occurrence in Space. While troubleshooting for a symptom event, network operators often focus on other events that occur at the same location or within some proximity of the symptom event, instead of looking for events occurring throughout the network. This is based on the experience or the domain knowledge related to the potential impact scope of the network events. Some events are known to have a local impact (*e.g.*, a physical link down event is known to cause protocol session to go down on the same link), whereas other events propagate from one location to another (*e.g.*, a router reload event on one router is known to cause protocol session timeouts on its neighboring routers).

To automate the troubleshooting process, we formalize the implicit spatial model used by network operators into a systematic model of the impact scope of network events. To specify the location of an event, we organize network components into a *location hierarchy*. As shown in Fig. 3, a physical link consists of interfaces on its two ends and all layer-1 devices in between; a logical link contains one or more physical links; a router comprises all its interfaces; an end-to-end path includes all logical links and routers on it; and an OSPF area includes all routers and links in its region. Intuitive as it may sound, the location hierarchy offers significant flexibility to network operators in specifying the scope of interest without drilling into the detailed topology or connectivity information of a particular event location.

We then use a *proximity model* to capture the scope of impact an event can have. The proximity model is specified in terms of the number of hops between the symptom event and potential cause events: for a given number of hops h, NICE considers only those events that occurred at most h hops away from the location of the symptom event for correlation. NICE allows the user to specify the value of h. For example, to understand packet loss on a path, it often suffices to correlate it with other events along the same path (*i.e.*, $h = 0$). Meanwhile, to troubleshoot a control plane anomaly on a router, the user typically needs to include router reload events on any router that is one hop away (*i.e.*, $h = 1$). Combined together, location hierarchy and proximity model provide powerful and flexible ways for operators to apply their domain knowledge in troubleshooting chronic events.

3.2 Event-series Transformation

Now that we have described what is meant by co-occurrence of events in time and space, theoretically it should be straightforward to correlate the symptom event-series with all other event-series. In practice, however, due to the diversity and heterogeneity of the data sources, it becomes arduous to create composite event-series out of basic ones and to determine the overlapping period of event occurrences. To solve this problem, we introduce an intermediate data representation in NICE, which greatly simplifies the task of event-series correlation while preserving the important information contained in the original event-series required for troubleshooting. The representation is also flexible enough to allow easy incorporation of new data into the NICE infrastructure.

For the intermediate data representation in NICE, we adopt a fixed-interval binary time-series representation, where value 1 represents the event occurrence within a time-window (for a point event) or overlap with the time-window (for a range event), while 0 indicates otherwise. NICE also supports numerical time-series representation. But our experience so far has found the binary representation to be adequate.

Transformation into a binary time-series involves three steps (shown in Fig. 4). Fig. 4(a) shows two original event-series A and B, where A is a point event-series and B is a range event-series. Below we describe the three steps:

1. **Conversion to range event-series (Fig. 4(b)):** To account for the lack of exact synchronization between different data sources and the delay involved in the propagation of an impact, we add a "padding margin" to each side of an event occurrence. This converts any point event-series into a range event-series and increases the interval of occurrence for a range event-series.
2. **Merging overlapping ranges (Fig. 4(c)):** With the addition of the padding margins, successive event occurrences can become overlapped when the end-time of an event becomes higher than the start-time of the subsequent event. In this case, we collapse multiple event occurrences into a single event occurrence.
3. **Conversion to fixed-interval binary series (Fig. 4(d)):** Different lengths of event intervals across various event-series result in complications for determining co-occurrences. To get around this problem, we convert all event-series into a fixed-interval binary series as follows:

$$X(i) = \begin{cases} 1, & \text{if } \exists\, a \in A \text{ covers } [i \cdot \delta, (i{+}1) \cdot \delta] \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where a is a range in a range event-series A, δ is the time-bin size, T is the total duration for the event-series, and i is an integer such that $0 \le i < \frac{T}{\delta}$.

Note that the time-bin size δ need to be chosen with care. At one extreme, choosing a very large δ will result in loss of precision, making everything appear as co-occurring in a time window. At the other extreme, choosing too small a δ will increase the number of points in the time-series, adding to the computation cost. The δ value should also be comparable to the padding margin so as to preserve the auto-correlation structure of the event-series, which will become clear in Section 3.3. Both δ and padding margin are configurable in NICE. We use a time-bin size (δ) of 20 seconds and a padding margin of 30 seconds throughout this paper. We have found these values to work well in practice.

3.3 A Novel Statistical Correlation Test

With the aforementioned data transformation, an obvious way of detecting co-occurring events is to simply identify the event-series in which value 1 is present in the same time-bin as the $1'$s in the symptom event-series. However, such a co-occurrence based approach suffers from two major problems: (i) some co-occurrences of events may be a mere coincidence or a one-time incident, in which case they are of no interest to network operators; (ii) event-series that have high probability of occurrence may overlap with symptom event by chance, in which case they are false alarms.

Having described the limitation of the co-occurrence based approach, we now look into applying statistical correlation to identify significant event co-occurrences. We first review Pearson's correlation coefficient and the classic correlation significance test. Then we explain why the classic significance test fails in our context. Finally, we develop a novel correlation significance test based on circular permutation.

Correlation Coefficient and Significance Test. Given two event-series x, y, the Pearson's coefficient of correlation is:

$$r = \frac{\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)}{(N-1)\sigma_x\sigma_y} \tag{2}$$

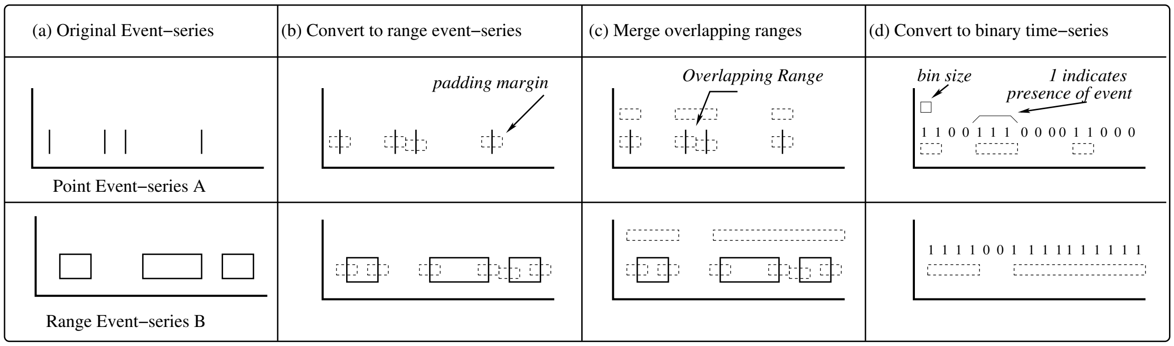

Figure 4: Event-series transformation to a binary time-series.

where x_i, y_i are individual samples in x and y, μ_x, μ_y are means for x and y, σ_x, σ_y are standard deviations for x and y, and N is the number of samples in each series (*i.e.*, sample size). The value of r should fall within $[-1,1]$, with 1 representing the two event-series being identical and -1 for the two being exactly opposite of each other.

Given the correlation coefficient, the classic approach to test for its significance is to apply Fisher's z-transform [7]:

$$z = \frac{1}{2} ln \left[\frac{1+r}{1-r} \right] \tag{3}$$

Under the null hypothesis that the two event-series are independent, z should be asymptotically Gaussian with mean $\mu_z = 0$ and standard deviation $\sigma_z = \frac{1}{\sqrt{N-3}}$. Thus, the correlation score defined below is expected to be standard Gaussian with a large value of N.

$$score \stackrel{\triangle}{=} \frac{z - \mu_z}{\sigma_z} = z \times \sqrt{N-3} \tag{4}$$

A large absolute value of the correlation score (*e.g.*, 1.96 at 95% confidence level) will reject the hypothesis that the two event-series of interest are independent.

Limitation of Existing Significance Tests. The above significance test assumes that each sample (*i.e.*, time-bin) in an event-series is independently and identically distributed. Unfortunately, this assumption does not hold in our context for two reasons. First, an event-series is inherently an auto-correlated series[2]. This is because most events are more likely to re-occur in a short time frame rather than after a long period. For example, a malfunctioning line card that produces high packet errors is more likely to generate high packet errors in the subsequent time bins. Second, the use of padding margin and discretization in the data transformation process can introduce auto-correlation. For example, a single CPU overload event in a five-minute measurement interval will be transformed into several consecutive $1'$s in the event-series when thirty-second time-bins are used during the conversion, resulting in significant auto-correlation at small lags. Without accounting for such auto-correlation, the classic test can easily overestimate the significance of correlation coefficients, yielding too many false alarms.

[2] Auto-correlation of an event-series captures the correlation between samples within the same series at different points in time.

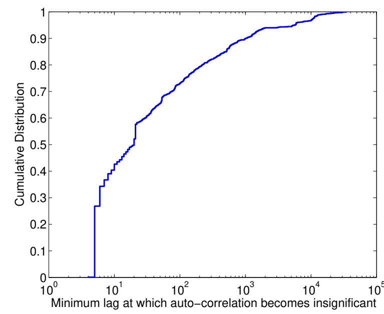

Figure 5: Cumulative distribution for the minimum lag after which auto-correlation becomes insignificant in an event-series. Each event-series contains 30 **days worth of data. The total number of event-series is** 2321**.**

A technique proposed by Dawdy and Matalas [3] can account for lag-1 auto-correlation by estimating the number of *independent* samples with the *effective sample size* instead of the original sample size N. Here lag-1 auto-correlation is the correlation between an event-series x and the shifted version of x by one time-bin. This approach works fine for lag-1 auto-correlation, but does not account for higher-lag auto-correlation. As shown in Fig. 5, network event-series often exhibit significant higher-lag auto-correlation. The figure shows the cumulative distribution for the minimum lag at which the auto-correlation becomes insignificant in an event-series for 2321 event-series extracted from the tier-1 ISP. As we can see, about 30% of event-series have significant auto-correlation at lag 100 or higher.

New Circular Permutation Based Significance Test. To capture high-lag auto-correlation, we develop a novel significance test based on circular permutation. Given two event-series x and y, we generate samples of the Pearson's correlation coefficient r by circularly shifting y to different lags with respect to x, and computing the correlation coefficient between x and the shifted versions of y. Specifically, for each lag $t \in [0, N]$, let $r(t)$ be the Pearson's correlation coefficient between x and the circularly shifted version of y at lag t:

$$r(t) = \frac{\sum_{i=1}^{N} (x_i - \mu_x)(y_{(i+t) \bmod N} - \mu_y)}{(N-1)\sigma_x \sigma_y} \tag{5}$$

Intuitively, by circularly shifting one event-series against the other, we destroy the cross-correlation between the two event-series while preserving the auto-correlation within each event-series. We can therefore use $\{r(t)\}$ ($t \in [0,N]$) to establish a baseline for the null hypothesis that the two event-series have no significant cross-correlation. Following this intuition, let the Fisher's z-transform of $r(t)$ be

$$z(t) = \frac{1}{2} ln \left[\frac{1+r(t)}{1-r(t)} \right] \tag{6}$$

We expect $\{z(t)\}$ to be asymptotically Gaussian for sufficiently large N. Let $\sigma_z^{\text{perm}} = \text{stddev}(\{z(t)\})$ be the standard deviation of $\{z(t)\}$. Instead of setting $\sigma_z = \frac{1}{\sqrt{N-3}}$ (as is done in the classic significance test), we use the sample-estimated σ_z^{perm} to compute the correlation score:

$$score_{\text{perm}} = \frac{z(0)}{\sigma_z^{\text{perm}}} = \frac{z(0)}{\text{stddev}(\{z(t)\})} \tag{7}$$

We consider a correlation score significant if it falls outside of the $[-2.5, 2.5]$ range. With z asymptotically Gaussian, this yields a low false positive ratio of around 1%.

We apply Fast Fourier Transform (FFT) to efficiently compute $r(t)$ at all possible lags $t \in [0,N]$. So the time complexity of computing the correlation score is $O(N \log(N))$. The CPU time in running a pair-wise correlation test on a Intel Pentium 2.4 GHz processor for event-series with thousands of sample points is a few milli-seconds. Even with a million sample points, it takes only around one second.

3.4 Extension for Multiple Symptom Events

Thus far, we have focused on troubleshooting a single symptom event-series. In operational practice, it is intuitive to look at multiple event-series together if they show the same symptom and are believed to share the same type of causes. For example, if one wants to understand the common causes for buffer overflow at edge routers, it makes sense to combine all edge router that show the symptom together in the analysis. Such aggregation can also be applied over the manufacturer of the equipment. For example, if one wants to gain knowledge about the CPU overload condition on routers from a particular vendor, it is desirable to examine those routers collectively and make the diagnosis.

NICE provides two methods to support multiple symptom event-series. (i) *event-series union*. In this method, NICE allows one to define a composite symptom event-series that is the union (*i.e.*, element-wise OR) of the individual symptom event-series, and test it against the union of the diagnostic event-series. The length of the event-series after taking the union is the same as each of the original event-series. While some correlation signal may get lost in the union process, we expect genuine and strong patterns of event correlation to remain, especially when the symptom events are rare. (ii) *event-series concatenation*. In this method, NICE allows multiple symptom event-series to be concatenated to form a single longer symptom event-series and so to all corresponding diagnostic event-series. With a modification of the significance test to limit the circular shifting to within their original event-series respectively, NICE computes a single correlation score that gives emphasis to common and strong correlations across different symptom event-series.

3.5 Result Interpretation

Finally, we describe the process by which NICE presents the results to operators and helps them prioritize the events.

Equivalence Class Grouping. To help network operators better interpret the correlation result, NICE groups similar event-series (that have significant correlation with the symptom event-series) into equivalence classes.

NICE quantifies the similarity of different event-series in the context of the symptom event-series based on the Jaccard similarity measure [7]. Specifically, let s be the symptom event-series, and E be the set of event-series that have strong statistical correlations with s. Let $\cap$ be the intersection of two event-series, which captures the co-occurrence of events in time. Let $\cup$ be the union (*i.e.*, element-wise OR) of two event-series. Given two event-series x, y ($\in E$), their similarity in the context of the symptom s is defined as the Jaccard similarity between $(x \cap s)$ and $(y \cap s)$:

$$sim(x,y) \triangleq sim_{\text{Jaccard}}(x \cap s, y \cap s) = \frac{|(x \cap s) \cap (y \cap s)|}{|(x \cap s) \cup (y \cap s)|} \tag{8}$$

To group event-series into equivalence classes, NICE first constructs a similarity graph in which each vertex represents an event-series that has significant correlation with the symptom event-series. An edge is added between two vertices if and only if the similarity between the two corresponding event-series (as defined above) is above a specified threshold τ. NICE then returns each connected component of the similarity graph as an equivalence class.

Prioritization. For each symptom, NICE outputs the list of events (that have strong statistical correlations) grouped as equivalence classes. The list is sorted by the fraction of symptoms events that co-occurred with the symptom series. This serves as the prioritized list for the operators. In addition, NICE also provides the operators with individual event counts, conditional probabilities and co-occurrence counts. This information helps in further analysis of the results.

4. NICE VALIDATION

In this section, we validate NICE using real data collected from the tier-1 ISP backbone. The goals are two-fold: (i) examine whether the statistical correlation results output by NICE make operational sense, and (ii) understand the applicability of the spatial proximity model used in NICE.

Specifically, we examined whether mathematically significant correlation (as reported by NICE) is also operationally significant (according to network domain knowledge) and vice versa. We pursued every case of disagreement to understand if it is a mistake made by NICE or not.

4.1 Methodology

To validate NICE, we identified a wide range of events for which either we or the network operators we collaborate with have sufficient domain knowledge. Table 1 summarizes the major categories of events that we considered and the data sources from which the event-series were extracted. These events span loss measurements, layer-1 outages, congestion, failures, route changes, CPU activity and operations

Category	Events	Data Sources
End-to-end performance	End-to-end loss	Active Probes
Traffic problems	Link congestion, queuing problems, packet errors	SNMP, Router Syslogs
OSPF events	Link up/down, router up/down, link metric changes	OSPF Monitor
Router CPU utilization	Spikes, thresholds	SNMP
Router internal problems	Line card crash, switching fabric problems	Router Syslogs
Routing session problems	Dead timer expiry, session downs	Router Syslogs
Router commands	Show commands to view routing state, router reload	Router Command Logs
Layer 1/2 problems	Link capacity changes, signal loss on interfaces, high bit error rate	Router Syslogs

Table 1: Major categories for event-series extracted from the data collected at the tier-1 ISP network.

activities. Each event-series contained six months worth of data. The actual number of events in the event-series ranged from 3 to 2,350,111, with mean and median values of 34,694 and 2,221, respectively. So the data had a lot of diversity with respect to the event count distribution.

We used the events of Table 1 to form a list of event pairs. Out of all possible event pairs, we extracted a subset of event pairs for which we could *a priori* determine the presence or absence of positive correlation that is operationally significant. These decisions were made based on either networking domain knowledge (*e.g.*, large routing events are likely to cause a short period of loss in the network) or anecdotal evidence of certain behavior observed by network operators (*e.g.*, certain router commands are known to be CPU intensive). The first and second columns of Table 2 show the list of event pairs for which we could determine the presence or absence of operationally significant correlation. The third and fourth columns show the number of event-series pairs that we expected to have operationally insignificant and significant correlations, respectively. Once we had all such event pairs, we ran NICE and compared its output with the estimate of operationally significant correlation.

4.2 Results

The fifth and sixth columns of Table 2 summarize the number of unexpected correlations (*i.e.*, correlations that are considered operationally insignificant by us but mathematically significant by NICE) and missed correlations (*i.e.*, correlations that are considered operationally significant by us but mathematically insignificant by NICE). We observe that for about 97% pairs, NICE's correlation output agrees with our estimate of operational significance.

For the remaining 3% pairs (for which NICE's correlation output disagreed with the operational significance), we drilled down further to understand the mismatch. The results are summarized in the seventh column of Table 2. The causes for the mismatches fell under three categories: (i) undesirable network condition, (ii) imperfect domain knowledge, and (iii) measurement artifacts.

Undesirable Network Condition. Modern service provider networks often use failure recovery mechanisms at layer 1 (*e.g.*, SONET ring protection switching) to rapidly recover from faults without inducing re-convergence events at layer 3 [20]. However, NICE identified strong correlations between layer-1 failure recovery and layer-3 re-convergence events on some links – completely contradicting expectations. After further drill down, it was determined that router bugs were causing this, and the issue mitigated.

Imperfect Domain Knowledge. We could explain 23 out of 24 unexpected correlations and 10 out of 29 missed correlations because of imperfect domain knowledge. As an example, one of the router commands is considered highly CPU intensive at least anecdotally. Therefore, we estimated that it would correlate strongly with a CPU utilization above high threshold values such as 80%. However, we did not find any correlation between execution of the command and CPU utilization even when the threshold was as low as 50%. It was only when we used a threshold of 40%, did we see correlations. What we learned out of this exercise was that the CPU-intensive command did cause CPU utilization to increase, but not as high as we had originally thought. This example brings out the fact that domain knowledge or the expected network behavior based on experience can be imprecise or even wrong at times because of scale, heterogeneity and complex interplay of hardware, software and operational practices in IP networks.

Measurement Artifacts. We found that 19 remaining missed correlations (out of 29) were explained by measurement artifacts. For example, we found low correlation between router reloads and CPU utilization, which was counter to our expectation. On closer inspection, we found that CPU utilization reports were unavailable while the router was rebooting.

4.3 Summary

Our validation results above demonstrate that the NICE's correlation output tends to agree well with our domain knowledge. That is, mathematically significant correlations (as reported by NICE) tend to be operationally significant (based on the domain knowledge) and vice versa. When NICE's correlation output disagrees with our domain knowledge, the disagreement can be explained by undesirable network condition that needs to be mitigated, or imperfect domain knowledge that needs to be revised, or artifacts of measurement process that needs to be improved.

5. OPERATIONAL EXPERIENCE

We have recently deployed NICE within the tier-I ISP to troubleshoot chronic network conditions. In this section, we describe three case studies where NICE was used to troubleshoot network issues. These studies demonstrate the efficacy and flexibility of NICE for handling diverse network problems and measurement data sets.

5.1 Overview

In each of the case studies discussed here, we selected a known network symptom, the spatial scope, and the time interval over which troubleshooting was to be performed. Our goal was to identify a list of other network events that

Event-series 1 Category	Event-series 2 Category	Results expected by operators		NICE correlation results		
		Insignificant Correlations	Significant Correlations	Unexpected Correlations	Missed Correlations	Results of drill-down
End-to-end loss	OSPF events	16	12	5	0	Imperfect domain knowledge
End-to-end loss	Layer 1/2 problems	0	3	0	0	
End-to-end loss	Router internal problems	0	2	0	0	
End-to-end loss	Traffic problems	0	11	0	0	
Router CPU	OSPF events	59	70	0	5	Measurement artifact
Router CPU	Router commands	28	28	0	10	Imperfect domain knowledge
OSPF events	Routing session problems	0	3	0	0	
Router reload	Router CPU utilization	0	14	0	14	Measurement artifact
Router reload	OSPF events	0	20	0	0	
Router reload	Routing session problems	1482	6	16	0	Imperfect domain knowledge
Router reload	Layer 1/2 problems	2	0	0	0	
Router reload	Router internal problems	2	0	0	0	
Layer-3 failures [a]	Layer-1 failure recovery	1	0	1	0	Undesirable network condition
OSPF down	OSPF events	2	10	2	0	Imperfect domain knowledge
OSPF down	Layer 1/2 problems	0	6	0	0	
OSPF down	Routing session problems	0	4	0	0	
OSPF down	Router internal problems	0	4	0	0	
Total		**1592**	**193**	**24**	**29**	

Table 2: Results for validation of NICE using six months worth of event-series data from the tier-1 ISP network.

[a] Analysis is performed for only one week.

Data source	Number of event types
Layer-1 Alarms	130
SNMP	4
Router Syslogs	937
Router Command Logs	839
OSPF Monitor	25
Total	**1935**

Table 3: Data sets used in case studies.

have strong statistical correlations with the known symptom event. This list incorporated both potential root causes and impacts of the symptom event. In each case, NICE identified several interesting correlations, some of which provided new insights while others revealed conditions and behaviors that were not previously understood.

The three case studies cover a wide range of network conditions, including (i) packet loss observed on the uplink of an access router, (ii) packet loss observed by active measurement between a pair of routers, and (iii) CPU spikes observed on routers across the ISP backbone.

Table 3 summarizes the data sets used in the case studies. Multiple event types are extracted from each data source. For example, we create 937 distinct event-series from the router syslogs, with each event-series corresponding to different error codes and messages within the syslogs. Similarly, we identify 839 different event-series from router command logs – each corresponding to a unique command entered by network operators. We consider a total of $1,935$ different event-series from our available data sources within our case studies here.

Table 4 summarizes the main results for each case study. For each symptom event and spatial scope of impact, it shows the trace duration, the number of pairs used for correlation testing, the number of events with strong statistical correlations, the number of equivalence classes and the percentage reduction. The events in the equivalence classes are those which NICE identified for the network operator to examine as part of troubleshooting the symptom. For all case studies, there is an approximately 90% reduction in the number of event types – a significant simplification in the analysis

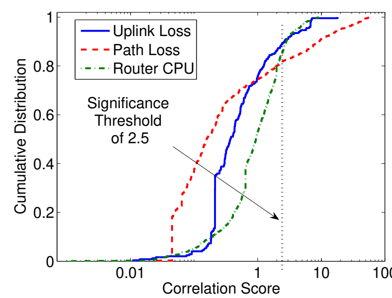

Figure 6: Correlation score distribution for case studies.

required by the operator. The correlation score distribution for each case study is shown in Fig. 6. In subsequent subsections, we explain each case study in more detail.

5.2 Case Study I: Router Performance Issues

In our first case study, we focus on troubleshooting a chronic packet loss condition on the uplink of an access router. *Access routers* are the routers to which ISP customers connect. These access routers consist of large numbers of interfaces (on the order of hundreds) directly connected to customer locations. Traffic is aggregated from these customers for transmission into the ISP network, via *uplinks* which connect the access router to the ISP backbone.

The router in question had been exhibiting intermittent packet loss on the uplink, and operators were challenged to determine the root cause so that it could be mitigated. Taking the packet losses on the uplink as the symptom event, we ran NICE using local (*i.e.* zero distance) spatial proximity at the router level for all other events in the scope, including those extracted from syslogs and SNMP measurements. The events included packet losses observed on each of the router interfaces.

Findings. Using NICE, we found that four customer-facing interfaces exhibited congestion-related packet loss that was highly correlated with the packet losses experienced on the relatively lightly loaded router uplinks. Congestion-related

Symptom	Spatial Scope	Trace Duration	Pairs to correlate (A)	Strong Correlations (B)	Equivalence Classes (C)	Reduction $(\frac{A-C}{A}) \times 100\%$
Uplink Packet loss	Single Router	4 days	245	26	17	93.06 %
End-to-end packet loss	Single Path	30 days	868	157	72	91.71 %
Router CPU	Composite	30 days	1286	179	128	90.05 %

Table 4: Results summary for all case studies.

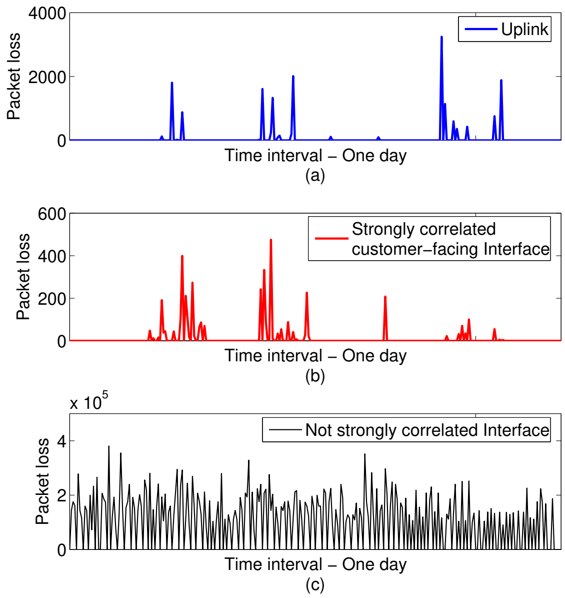

Figure 7: Time-series plots for a single day for the uplink packet loss case study: (a) Packet losses on the router uplinks. (b) Packet losses on one of the four interfaces connected to an enterprise customer (the other three had similar patterns). (c) Packet losses on a separate interface, which do not statistically correlate with uplink losses even though they have high co-occurrence.

loss on customer interfaces is far from uncommon – a result of the very bursty nature of individual customer's traffic and customer link bandwidths. Thus, although these four interfaces were not the only customer-facing interfaces demonstrating congestion-related loss, the correlations between these four interfaces and the uplink issues stood out well above those for any other interfaces. The four interfaces were also grouped into a single equivalence class. The time-series plots for a single day are shown in Fig. 7. Fig. 7(a) shows the packet loss observed on the router uplinks. As shown in Fig. 7(b), the time-series for one of the customer-interfaces correlates nicely with uplink losses. The other three interfaces in the equivalence class demonstrate very similar behavior. Interestingly, Fig 7(c) illustrates another customer interface, which in fact demonstrates much higher loss. However, even though the losses frequently co-occur with the uplink ones, NICE accurately identifies that the statistical correlation is insignificant – the interface is continually experiencing some level of loss as the customer appears to be overloading its interface with short term traffic bursts. The traffic overload related to this interface, however, is apparently not the root cause of the observed uplink issues.

Examination of router configuration files revealed that these four interfaces were connected to a single enterprise customer, and this customer was configured with packet load balancing. Short term traffic bursts flowing through the router to these customer interfaces appear to have been causing internal router limits to be momentarily reached, impacting traffic flowing out of the router (and hence other, unrelated customer's traffic). At the time of writing this paper, operators were in the process of re-homing this customer's interfaces to another access router.

This example demonstrates how NICE can be applied to rapidly isolate the root cause of a performance degradation, something which would have been either extremely painful or nearly impossible to achieve manually or just with a co-occurrence based approach.

5.3 Case Study II: End-to-end Packet Loss

Our second case study focuses on troubleshooting packet loss observed between a pair of routers across the ISP's backbone. Given the sensitivity of an increasing number of applications to packet loss, it is critical that network loss events are analyzed so that their root cause is identified, and efforts can be made to continue to drive loss out of the network. However, it is clearly impractical to examine each individual event by hand; automation is critical here.

The ISP has an automated tool called "Backbone Loss Analysis Tool", which, given a list of potential root causes of loss, assigns the most likely root cause to a given loss event. This tool is instrumental in characterizing at an aggregate level the impact of different failure modes and events in the network on loss, and has been used extensively by network engineers to make strategic decisions regarding the potential impact of new technologies considered for network deployment (*e.g.*, QoS).

However, the Backbone Loss Analysis Tool is only as good as the domain knowledge feeding it. To determine which of the massive number of events are indicative of potential causes of loss is simply impossible to achieve through manual inspection. As a result, only a relatively simple list of events is currently used for root cause analysis in the tool – routing, congestion, and active measurement errors. These events were selected based on domain knowledge.

To better diagnose the packet loss symptom reported by active probes between a pair of routers, we applied NICE to an entire range of router syslogs, command logs, layer-1 alarms, and routing data to automatically identify the event-series that are statistically correlated to these packet losses. We used local proximity at the path level as the spatial model, in which NICE calculated the routing path based on OSPF routing information collected from the network and performed event composition accordingly.

Findings. Out of a total of 868 candidate events, NICE identified 157 events that had statistically significant correlation with packet loss. These events were related to routing events, router software errors, hardware failures and internal router issues. In total, we were able to successfully identify network events that were indicative of root causes of packet loss for 98% of our sample loss events.

The set of network events reported by NICE is much more comprehensive than the relatively simple set of events currently used for root cause analysis in the Backbone Loss Analysis Tool. Close inspection of these events also yields several interesting findings that expand our domain knowledge. For instance, we find that active probe losses correlate with *composite member link returning to service* (reported by a particular syslog message) when a composite link is on the path. Note that a composite link is a logical IP link consisting of multiple physical links between a pair of routers. Although this event is a very small contributor to loss and the resulting lossy period only lasts for tens of milliseconds, it is a surprising behavior. This example demonstrates how NICE allows network operators to rapidly discover new signatures, gain new insights, and expand the domain knowledge to help better troubleshoot network problems.

5.4 Case Study III: Router CPU Utilization Fluctuations

Router central CPUs support routing and signaling processes, and provide the interface through which the outside world can configure, monitor and manage the routers. Thus, CPU overload conditions can result in potentially serious performance issues. For example, should routing protocols be unable to process routing messages, unnecessary (performance impacting) re-routes could occur. The ISP thus collects CPU utilization measurements for every router in the network at five minute intervals, via SNMP, allowing the operators to detect and troubleshoot anomalous CPU behaviors. In fact, monitoring CPU utilization serves as one of the vital "pulse-points" for gauging the overall health of the network.

The network operators have invested significant resources in troubleshooting anomalous CPU conditions to ensure that this limited resource is effectively managed on an ongoing basis. Operators have painstakingly dug through immense mounds of network data to uncover events which are either a cause of or are induced by CPU conditions, *i.e.*, events which are correlated with CPU anomalies. This process has involved months of manual analysis and is an ongoing process. The goal is to ensure that CPU load is kept to a minimum. Such ongoing monitoring and troubleshooting of CPU load has allowed the operators to work with router vendors to improve router implementation through code optimizations, bug-fixes, and appropriate prioritization of how CPU-intensive commands are handled. In a similar fashion, through CPU monitoring, operators have also been able to better understand how operations support systems impact CPU utilization on routers, resulting in refinements to these systems to minimize the impact.

Clearly, such a CPU troubleshooting analysis is an ideal NICE application. We thus decided to validate NICE against the work already executed by the operations ensuring that NICE could identify the relevant correlations. Operators provided us with the CPU anomalies of interest, which we used as the input chronic symptom event to NICE. We tested the CPU anomalies against all router syslog messages, routing events, router command logs and layer-1 alarms. We applied local proximity at the router level as the spatial model. Since the network operations team were interested in analyzing aggregate network-wide CPU conditions, we also adopted the extension in Section 3.4 to identify correlations to these CPU anomalies across the entire network.

Findings. NICE successfully identified all the earlier operations findings as well as uncovered some new ones. Consistent with the operations findings, NICE identified that control plane activity and certain router commands were dominant causes of router CPU spikes. Control plane activity usually incurs some processing by the CPU, so this was not surprising. Certain router commands such as those used to view routing protocol state were also known to cause CPU anomalies. However, the high impact of some other router commands (such as customer provisioning) on CPU utilization were more recent operations discoveries. All of these were also highlighted by NICE. In addition, NICE was also able to identify correlations between certain router commands and high CPU utilization that were not known before.

We also experimented with NICE's ability to analyze network behavior as it varies across different classes of network routers. Specifically, we compared CPU correlations for different classes of routers grouped by different vendors, router models and router roles (such as backbone routers, edge routers). Interestingly, CPU correlations were reasonably consistent within a class of routers, but varied across different classes. For example, while troubleshooting strong correlations between SNMP polling and CPU spikes observed on certain individual edge routers, NICE uncovered that the correlation was statistically significant for some classes of routers, but not for other classes. Operations personnel are currently working with router polling systems to refine their polling mechanisms to minimize router CPU impact.

We have thus successfully demonstrated that NICE can automatically identify correlations of interest from a large scale of data. NICE does this without immensely painful manual analysis, and uses statistics to ensure that the correlations identified are "real" and not simply some accidental co-occurrences.

6. RELATED WORK

Most of the effort to date has focused on diagnosing large and long-lasting events, such as hard link or interface failures. SCORE[10] models the cross-layer fault diagnosis problem using a bipartite graph. [11] use spatial correlation to detect and diagnose silent failures. [18] presents a nice survey of fault localization techniques. Sherlock [1] infers dependencies using conditional probabilities and a multi-level approach. NetDiagnoser [4] adapts the Boolean tomography technique to identify the location of failures. eXpose [8] uses mutual information and spectral graph partitioning to extract communication patterns from packet traces. A number of Bayesian network techniques are also proposed in [1, 9]. Commercial tools such as HP Openview [15], IBM Tivoli [19] focus on analyzing large events.

The state-of-art in diagnosing chronic condition is to look for co-occurrences across multiple data sources. The focus of this paper is to analyze statistical correlations (rather than individual co-occurrences) across a diverse set of network data sources, and identify performance impacting behaviors.

There has been increasing interest in identifying new patterns in time-series data using data mining and statistical learning tools. Approaches proposed for mining correlations include CORDS [6] using chi-squared analysis, SPIRIT [16] using Principal Component Analysis (PCA), [21] using Hidden Markov Models, and Minerals [13] using association rule mining.

Remarks. NICE differs from previous approaches through the use of (i) a novel auto-correlation incorporated correlation metric with circular permutation test for significance, (ii) a novel spatial proximity model that captures the impact scope for different network events, and (iii) a unified data representation, which together enable flexible and scalable troubleshooting of chronic network conditions. Also note that none of the previous approaches above takes into account auto-correlation within a time-series.

Anomaly detection [2, 5, 12, 22] is complementary to our work on correlation. Anomalies can be input as events of interest to NICE to identify other network events that are strongly correlated (*e.g.*, identify strong correlations between CPU anomalies and syslog messages on the same router, or identify strong correlations between link utilization anomalies and link packet loss anomalies).

7. CONCLUSION AND FUTURE WORK

We presented the design and implementation of NICE which provides a flexible and scalable infrastructure for troubleshooting chronic network conditions. NICE uses a novel statistical correlation test across multiple network data sources along with a spatial proximity model to significantly reduce the number of correlated events for troubleshooting the symptom event. We validated NICE using real network data collected from a large tier-1 ISP, and also demonstrated its efficacy using three case studies. Our results and operational experience indicate that NICE is becoming a powerful troubleshooting tool within the tier-1 ISP.

We plan to extend NICE in several directions. First, we plan to explore higher-order N-way correlations and multivariate techniques. We expect to be able to re-use much of the NICE library. Second, we would like to understand how anomaly detectors and our correlation infrastructure interact with each other and whether they can provide each other feedback to identify unknown network anomalies. Finally, we intend to design and prototype innovative applications to enable us to drill-down into network correlations.

Acknowledgement

We thank Joanne Emmons and Prasanna Ramachandran for their invaluable feedback during the case-study analysis and application of NICE. This work is supported in part by NSF Awards CNS-0546720, CNS-0627020 and CNS-0615104.

8. 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. R. Dawdy and N. C. Matalas. Statistical and probability analysis of hydrologic data, part III: Analysis of variance, covariance and time series. In V. T. Chow, editor, *Handbook of applied hydrology, a compendium of water-resource technology*, pages 8.68–8.90, 1964.

[4] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot. NetDiagnoser: troubleshooting network unreachabilities using end-to-end probes and routing data. In *CoNEXT*, 2007.

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

[6] I. F. Ilyas, V. Markl, P. J. Haas, P. Brown, and A. Aboulnaga. CORDS: Automatic discovery of correlations and soft functional dependencies. In *Sigmod*, 2004.

[7] S. K. Kachigan. *Statistical analysis: an interdisciplinary introduction to univariate and multivariate methods*. Radius Press, 1986.

[8] S. Kandula, R. Chandra, and D. Katabi. What's going on? learning communication rules in edge networks. In *Sigcomm*, 2008.

[9] S. Kandula, D. Katabi, and J.-P. Vasseur. Shrink: A Tool for Failure Diagnosis in IP Networks. In *MineNet*, 2005.

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

[11] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Detection and localization of network blackholes. In *Infocom*, 2007.

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

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

[14] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of failures in an IP backbone network. In *Infocom*, 2004.

[15] HP Openview. `http://www.openview.hp.com`.

[16] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, 2005.

[17] A. Shaikh and A. Greenberg. OSPF monitoring: Architecture, design, and deployment experience. In *NSDI*, 2004.

[18] M. Steinder and A. S. Sethi. A survey of fault localization techniques in computer networks. *Science of Computer Programming*, 2004.

[19] IBM Tivoli. `http://www-306.ibm.com/software/tivoli`.

[20] J.-P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[21] K. Yamanishi and Y. Maruyama. Dynamic syslog mining for network failure monitoring. In *KDD*, 2005.

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