Troubleshooting Chronic Conditions in Large IP Networks

(Paper ID: 1569138573 # Pages: 12)

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, zhang} @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 time period. 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.

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 happen due to accidents, natural disasters, equipment failures, software bugs, mis-configurations, or even malicious attacks [19]. 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., recurrent) 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 (Network-wide Information Correlation Exploration), 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 [31], 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 (for example, 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 (for example, 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.

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 pairs of network event-series, NICE starts with the symptom 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.

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 routers and other 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.

**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)\(^1\) using an OSPF Monitor [22]. We focused on the LSAs that indicate events caused by maintenance and failures in the network. We also used 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 10,000 times. The mean and maximum number of occurrences are 8,269.4 and 1,759,821 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 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 framework 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) an event 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 incorporated correlation metric and evaluation process that automates the obscure co-occurrence based reasoning by net-

---

\(^1\)LSAs are routing messages used by OSPF to disseminate link metric changes and network changes such as link up/down, and router up/down.
work 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 statistically significant correlation 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, excessive loss of active probing packets from one router to another, or the occurrence of a specific alert in the syslog of a router. 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 syslogs 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 two consecutive polls of SNMP counters, in which case, 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 and OSPF adjacency up events in the same time window.

**OSPF link-ap is the intersection of OSPF adjacency down operations.** As another example, the loss of probe packets adjacency up constitutes an OSPF link-ap event. Logically, our data model in Section 3.2 greatly simplifies such logical event followed (within a small time window) by an OSPF section (AND), union (OR), and negation (NOT). In many for troubleshooting. For example, an OSPF adjacency down composite events through logical operators such as inter-
such a hierarchical structure offers tremendous flexibility to
network operators in specifying the scope of interest without
drilling into the detailed topology or connectivity informa-
tion of a particular event location.

We then use a proximity model to capture the scope of
event impact. In most cases, the impact of a network event is
limited to the same location as the symptom event. The level
of the hierarchy is determined by the nature of the event.
For example, a user might be interested in identifying co-
occurrents between CPU spikes on a router with control
plane activities on the same router, in which case, we apply
the scope of impact for the CPU spike event to be of zero dis-
tance to the event location at the router-level. In some cases,
the distance can also be greater than zero. For example, to
troubleshoot a control plane anomaly on a router, we will in-
clude router reload events on any router that is one hop away.
With the combination of location hierarchy and spatial prox-
osti model, the NICE framework gives full flexibility for
operators to apply their domain knowledge, or to conduct a
brute-force search for impact-bearing co-occurring 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 straight-
forward to evaluate the symptom event-series against all other
event-series. In practice, however, due to the diversity and
heterogeneity of the data sources, it becomes quite 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 a unified intermediate layer
of data representation in NICE. This layer for event-series
representation greatly simplifies the task of event-series cor-
relation while preserving the important information contained
in the original event-series required for troubleshooting. The
representation is also flexible enough to allow easy incorpo-
ration of new data into the NICE framework.

For the intermediate data representation in NICE, we adopt
a fixed-interval binary time-series representation, in which
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 sup-
ports numerical time series representation when no event
composition or change of time-window size is needed. But
our experience so far has found the binary representation to
be adequate.

Transformation to a binary series is accomplished in three
steps. These steps are shown in Fig. 4. Fig. 4(a) shows
two original event-series, in which event-series $A$ is a point
event-series and $B$ is a range event-series. Below we de-
scribe the three steps:

1. **Conversion to range event-series (Fig. 4(b))**: To ac-
tcount for the lack of exact synchronization between dif-
f erent data sources and the delay involved in propagation
of 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 oc-
currence for a range event-series.

2. **Merging overlapping ranges (Fig. 4(c))**: The original
event-series can become ill-formed with the addition of
the padding margin. Successive event occurrences can
become overlapped when the end-time of an event be-
comes 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 series
results in complications for determining co-occurrences.
To get around this problem, we convert all event-series
into a fixed-interval binary series as follows:

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

(1)

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

Note that the time-bin size $\delta$ need be chosen with care.
At one extreme, choosing a very large $\delta$ will result in loss
of precision, making everything appear as co-occurring in
a time window. At the other extreme, choosing too small
a $\delta$ will increase the number of points in the time series,
adding to the computation cost. The $\delta$ 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 $\delta$ and padding margin are con-
figurable in NICE. We use a time-bin size ($\delta$) 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
c-occurrence based approach suffers from two major prob-
lems: (i) some co-occurrences of events may be a mere coin-
cidence 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 base
approach, we now look into statistical correlation for iden-
tifying significant event co-occurrences. We will first ex-
plor Pearson’s correlation coefficient and Fisher’s z signif-
icant test, then we will describe the reason why it fails for
our purposes, and finally describe our circular-permutation-
based correlation test.

**Correlation Coefficient and Significance Test.** To test if
two event-series are co-occurring statistically, we apply the
classic statistical correlation test as follows:

For two event-series $x, y$, the Pearson’s coefficient of cor-

Given the correlation coefficient, the classic approach to test for the correlation significance of the correlation is to apply Fisher’s $z$-transform [32]:

$$z = \frac{1}{2} \ln \left( 1 + r \right)$$  \hspace{1cm} (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 $\sigma_z = \frac{1}{\sqrt{N}}$. Thus, the correlation score defined below is expected to be standard Gaussian with a large value of $N$.

$$\text{score} = \frac{z - \mu_z}{\sigma_z} = z \times \sqrt{N - 3}$$  \hspace{1cm} (4)

A large absolute value of the correlation score (e.g., 1.96 at 95-percentile confidence level) will reject the hypothesis that the correlation between the two time series of interest are statistically insignificant.

**Limitation of Existing Significance Tests.** The above significance test uses an underlying assumption 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. This is because most events are more likely to re-occur in a short time frame rather than after a longer time. 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 CPU overload event in a five-minute measurement interval will be transformed into several consecutive 1’s in the event-series, contributing to its auto-correlation at small lags. It turns out that capturing such dependencies are very important for our application.

A technique proposed by Dawdy and Matalas [6] 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-order auto-correlation. As shown in Fig. 5, the event-series formed from networks often have a significant higher-lag auto-correlation. The graph shows 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 can be seen, about 30% of event-series have a significant auto-correlation at lag 100 or greater.

**New Circular Permutation Based Significance Test.** In order to deal with high-lag auto-correlations, we propose a novel way of computing correlation score and performing significance test which we term as a circular permutation test. The key idea is to preserve the inherent correlation structure of the event-series by circularly shifting one against the other. Given a pair of 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 $t$. We have:

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

The Fisher’s z-transform at lag $t$ is

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

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 test), we use the sample-estimated $\sigma^z_{\text{perm}}$ to compute the correlation score. The correlation score is then given by:

$$\text{score}_{\text{perm}} = \frac{z(0)}{\sigma^z_{\text{perm}}} = \frac{z(0)}{\text{stddev}(\{z_t\})}$$  \hspace{1cm} (7)

At a first glance, the circular permutation test appears to have $O(N^2)$ time complexity as it requires computing an $O(N)$-complexity correlation coefficient for $N$ different lags. However, it turns out that the structure of the circular shifting is highly similar to the Fourier Transform and hence a technique similar to the Fast Fourier Transform (FFT) can be applied to efficiently compute $r(t)$ at various lags $t$. The time complexity of computing the correlation score hence becomes $O(N \log(N))$. The CPU time in running a pairwise correlation test on a Intel Pentium 2.4 GHz processor for event-series with thousands of sample points is a few milliseconds and even with a million sample points in the event-series takes around one second.

For the significance test, we consider a correlation score significant if it falls outside of the $[-2.5, 2.5]$ range. Equivalently, if $z$ deviates from its mean (0) by more than 2.5 times the standard deviation ($\sigma_z$), then it is significant. With $z$ asymptotically Gaussian, this yields low rate of around 1% for false positives.

### 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 routers 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 supporting such practices. In the first method, NICE leverages on its capability of spatial event composition, in which one can define a composite symptom event-series that is the union of each individual ones, and test it against the union of the diagnostic event-series. Note that some correlation signal may get lost in the union process. However, genuine and strong patterns of event correlation should still manifest, especially when the symptom events are rare.

The second extension that NICE builds in supporting multiple symptom event-series is through event-series concatenation. NICE supports multiple symptom event-series to be concatenated into a longer single time-series and so to all corresponding diagnostic event-series. With a modification of the significance test that limits 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 locations.

### 3.5 Result Interpretation

In this subsection, we describe the process by which NICE presents the result events to operators and helps them prioritize the events.

**Equivalence Class Grouping.** We propose to group similar event-series (that were significantly correlated with symptom) into equivalence classes using a modified Jaccard similarity metric [30]. Let $s$ be the symptom event-series, and $E$ be the set of event-series that have strong statistical correlations with $s$. For each event-series pair $x, y \in E$, we compute the distance with respect to the symptom $s$ using:

$$d(x, y) = 1 - \frac{x \cap y \cap s}{(x \cup y) \cap s}$$  \hspace{1cm} (8)

The distance is smaller for highly similar event-series and larger for less similar series. The intuition behind intersection with the symptom series $s$ is this: since we are analyzing the correlation results for the symptom, we want to maintain the context of the symptom when computing similarity between $x$ and $y$. For example, consider OSPF adjacency timeouts between neighbor routers to be the symptom. Let layer-1 and layer-3 failures correlate strongly with the timeouts. If layer-1 recovers from the failure within a specific time, then the failure at layer-3 is not observed. This failure recovery mechanism is widely used by ISPs these days. So, layer-1 and layer-3 failures might have a high similarity distance when analyzed just amongst themselves. However, when we maintain the context of the OSPF timeouts, they will be grouped together, which makes sense, because a layer-1 failure that results in a layer-3 failure in turn would cause an OSPF timeout.

If the distance is less than a threshold $\tau$, the two event-series are grouped into an equivalence class. The distance between two equivalence classes is the minimum of distance between the event-series. For two classes $c_i, c_j$, distance is:

$$d(c_i, c_j) = \min_d d(x, y) \ \forall x \in c_i, \forall y \in c_j$$  \hspace{1cm} (9)

If the equivalence class distance is less than $\tau$, the two classes are grouped into a single 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 probabilities, 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 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 with respect to the event count distribution.

Out of all possible event pairs, we extracted a subset of event pairs for which we could a priori decide 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 decide operationally significant correlation. The third column shows the total number of such event-series pairs. The fourth column shows the number of event-series pairs with operationally significant positive correlations.

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 data 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 reconvergence events at layer 3 [28]. However, NICE identified strong correlations between layer 1 failure recovery and layer 3 reconvergence 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 ten out of 29 missed correlations because of imperfect domain knowledge. Consider this 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 utilization even when 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 wrong at times because of scale, heterogeneity and complex interplay of hardware, software and operational practices in IP networks.

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

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

---

3Analysis done for one week.
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 imperfect domain knowledge that needs to be revised, or artifacts of measurement process that needs to be improved, or some network behaviors that we were previously unaware of.

5. OPERATIONAL EXPERIENCE

We have recently deployed NICE within the tier-1 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 set of problems and data.

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 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 diverse conditions observed within the network which are:

1. Packet loss observed on uplink of an access router.
2. Packet loss observed by active measurement between a router pair.
3. CPU spikes observed on routers across the ISP backbone.

Table 3 summarizes the data sets used in the case studies. Multiple different event types are extracted from each data source. For example, we create 937 distinct time series from the router syslogs, with each time series corresponding to different error codes and messages within the syslogs. Similarly, we identify 839 different time series from router command logs – each one corresponding to unique commands entered by network operators. We consider a total of 1,935 different time 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 required by the operator. The correlation score distribution that were not previously understood.

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

<table>
<thead>
<tr>
<th>Event-series 1 Category</th>
<th>Event-series 2 Category</th>
<th>Correlation tests using NICE</th>
<th>Operationally significant positive correlations</th>
<th>Unexpected Correlations</th>
<th>Missed Correlations</th>
<th>Result of drill-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end loss</td>
<td>OSPF events</td>
<td>28</td>
<td>12</td>
<td>5</td>
<td>0</td>
<td>Imperfect domain knowledge</td>
</tr>
<tr>
<td>End-to-end loss</td>
<td>Layer-1/2 problems</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>End-to-end loss</td>
<td>Router internal problems</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>End-to-end loss</td>
<td>Traffic problems</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Router CPU</td>
<td>OSPF events</td>
<td>129</td>
<td>70</td>
<td>0</td>
<td>5</td>
<td>Measurement artifact</td>
</tr>
<tr>
<td>Router CPU</td>
<td>Router commands</td>
<td>36</td>
<td>28</td>
<td>0</td>
<td>10</td>
<td>Imperfect domain knowledge</td>
</tr>
<tr>
<td>OSPF events</td>
<td>Routing session problems</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Router reload</td>
<td>Router CPU utilization</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td>14</td>
<td>Measurement artifact</td>
</tr>
<tr>
<td>Router reload</td>
<td>OSPF events</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Router reload</td>
<td>Routing session problems</td>
<td>1488</td>
<td>6</td>
<td>16</td>
<td>0</td>
<td>Imperfect domain knowledge</td>
</tr>
<tr>
<td>Router reload</td>
<td>Layer-1/2 problems</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Router reload</td>
<td>Router internal problems</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Layer-3 failures</td>
<td>Layer-1 failure recovery</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Undesirable network condition</td>
</tr>
<tr>
<td>OSPF down</td>
<td>OSPF events</td>
<td>12</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>Imperfect domain knowledge</td>
</tr>
<tr>
<td>OSPF down</td>
<td>Layer-1/2 problems</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>OSPF down</td>
<td>Routing session problems</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>OSPF down</td>
<td>Router internal problems</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>1785</td>
<td>193</td>
<td>24</td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Data sets used in case studies.

<table>
<thead>
<tr>
<th>Data source</th>
<th>Number of event types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer-1 Alarms</td>
<td>130</td>
</tr>
<tr>
<td>SNMP</td>
<td>4</td>
</tr>
<tr>
<td>Router Syslogs</td>
<td>937</td>
</tr>
<tr>
<td>Router Command Logs</td>
<td>839</td>
</tr>
<tr>
<td>OSPF Monitor</td>
<td>255</td>
</tr>
<tr>
<td>Total</td>
<td>1935</td>
</tr>
</tbody>
</table>

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 reboots and CPU utilization, something that was counter to our estimate. On closer inspection, we found that the router did not report CPU utilization while it was rebooting.
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 symptom event, we ran NICE using local (i.e., zero distance) spatial proximity at the router-level hierarchy 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. We were able to identify four customer-facing interfaces which were demonstrating congestion-related packet loss that was highly correlated with the packet loss conditions experienced on the relatively lightly loaded router uplinks. Congestion-related 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 customer 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 terminating connectivity for the same 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, operations were in the process of re-homing this customer’s interfaces to another access router.

This example thus 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 near impossible to achieve manually or just with 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 each 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 Engineering to make strategic decisions regarding the potential impact of new technologies considered for network deployment (e.g., QoS).

However, this 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.

**Findings.** To 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 identify the event-series which are statistically correlated to these packet losses. We used local proximity at the path-level for spatial model, in which NICE calculated the routing path based on OSPF routing information collected from the network and performed event composition accordingly. Out of a total of 868 candidate messages, NICE identified 157 messages which were statistically correlated with packet loss. These messages were related to routing events, router software errors, hardware failures, and internal router issues. In total, we were able to successfully identify network messages which were indicative of root cause of packet loss for 98% of our sample loss events.

We would like to compare this list of events with the relatively simple set of events currently used for root cause analysis in the Backbone Loss Analysis Tool (namely, routing, congestion and active measurement errors). This new set of events and log messages exposed by NICE will allow us to further refine the root cause analysis performed by the Backbone Loss Analysis Tool, allowing deeper analysis of events and will also allow more events to be categorized.

### 5.4 Case Study III: Router CPU Utilization Fluctuations

Central router CPUs support routing and signaling processes within routers, and provide the interface with the outside world (through which the router is managed and configured, and from which alerts and measurements are identified and reported by the router). 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 result. The ISP thus collects CPU utilization measurements for every router in the network at five minute intervals, via SNMP. This allows the operators to detect and troubleshoot higher than expected CPU utilization events. Monitor CPU utilization this way serves as one of the vital “pulse-points” for gauging the overall health of the network.

We collaborated with operators to scalably analyze CPU utilization anomalies identified by them. These were then used as the input chronic symptom event to NICE. The other event-series included 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 operations wanted to analyze network-wide CPU conditions, we adopted the extension in Section 3.4 to identify correlations to these CPU anomalies across the network.

**Findings.** NICE identified that control plane activities and certain router commands were dominant causes for router CPU spikes. Control plane activities usually incur some processing by the CPU, so this was not surprising to the operators. 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 on CPU utilization was not previously well understood. Thus, NICE was able to expand the list of events which place high load on router central CPUs. The operators are using this expanded list to drill down to root cause for each high CPU event and to work with router vendors to ensure that high utilization commands are handled with appropriate prioritization on the router.

### 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[15] models the cross-layer fault diagnosis problem using a bipartite graph. Shrink [13] uses Bayesian approaches to address the problem. [16] uses spatial correlation to detect and diagnose silent failures. [25] presents a nice survey of fault localization techniques. Sherlock [1] infers dependencies using conditional probabilities and a multi-level approach. [5] uses Tree-Augmented Bayesian Networks for correlating system-level performance metrics with high-level performance states. NetDiagnoser [7] adapts the Boolean tomography technique to identify the location of failures. eXpose [12] 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, 5, 13, 24]. Previous work to analyze routing dynamics and to identify system failures include [3,
8. REFERENCES


