

Fault Localization via Risk Modeling

Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C. Snoeren, *Member, IEEE*

Abstract—Internet backbone networks are under constant flux in order to keep up with demand and to offer new features. The pace of change in features and technology often outstrips the pace of introduction of the associated fault monitoring capabilities that are built into today’s IP protocols and routers. Moreover, some of these new technologies cross networking layers, raising the potential for unanticipated interactions and service disruptions, which the built-in monitoring capabilities in each layer may not detect. In these instances, operators typically employ higher-layer monitoring techniques such as end-to-end liveness probing to detect lower- or cross-layer failures, but lack tools to precisely determine where a detected failure may have occurred. In this paper, we present a simple and effective mechanism to localize these failures. Our method applies spatial correlation to high-level failure notifications to identify the lower-layer root-cause. We show that our system works with accuracy exceeding 80% for many failure scenarios, while delivering extremely high precision (greater than 80%). We further report our operational experience using spatial correlation to isolate optical component and MPLS control-plane failures in a tier-I ISP.

I. INTRODUCTION

OPERATIONAL backbone networks are intrinsically exposed to a wide variety of faults and impairments. The networks are large, geographically distributed, and constantly evolving complex hardware and software artifacts. To appreciate the complexity better, consider a typical tier-I backbone network: It consists of about 1000 routers from different vendors, with different features, and acting in different roles in the network architecture, supported by access and core optical transport networks involving more than two orders of magnitude more network elements. The layered structure of backbone networks is intended to help contain complexity within simple well-defined abstractions; in practice, however, layering often gives rise to additional, complex failure modes involving interactions between the control planes of different layers.

A large fraction of any tier-I provider’s time is spent coping with operational failures. The essential problem of fault management is to detect, localize, mitigate and ultimately correct any condition that degrades network behavior. To assist fault management, many network elements (such as routers) are designed to continuously monitor certain behaviors (e.g., link connectivity) and raise an alarm in the event of a failure. In some cases, the network element can even perform automatic recovery actions such as re-routing packets through

other functional paths. In other cases, the alerts are transmitted to semi-automatic fault management systems [17], which assist a human operator in troubleshooting of the failure. The challenge in diagnosing the problem is that alarms only indicate that a network element observed a deviation from normal behavior (e.g., link failure or probe packet loss); the actual fault could lie anywhere in the network (e.g., a downstream router reboot or optical amplifier failure). In other words, the element that issues an alert is not necessarily the cause of the failure. Fault localization, therefore, is an important step that needs to be performed before failure diagnosis and repair can be performed.

Despite assistance from a variety of monitoring infrastructures, fault localization remains a significant challenge in complex backbone networks for a number of reasons. First, in rare but catastrophic failure cases, there may be no alarms generated by the network elements, leading to what are known as “silent failures”. In these cases, many customers can experience loss in connectivity for extended periods of time until the operator becomes aware of the problem, localizes it, and finally takes action to restore service. Even in the common case when network elements generate alarms, correlating these alarms across layers is non-trivial. This is due to the fact that, in many backbone networks, operations are naturally managed according to layers, with the alarms and other monitoring data housed in separate work groups across geographically diverse locations within the same domain, or even across domains (e.g., different companies managing different layers). For instance physical-layer faults may be handled by a dark-fiber provider, while IP-layer faults are managed by the ISP itself. Finally, even when the fault is entirely within a single layer, the operator may be inundated by cascading alarms from all over the network and face the daunting task of determining the causal relationship between them.

In this paper we propose an automated fault localization framework based on *risk modeling*, that is effective in automatically identifying possible network failure locations given a set of potentially related alarms. At a high-level, risk modeling involves creation of a dependency relationship between observable events and potential causes. For example, a single fiber cut can simultaneously affect multiple IP links that are carried over the fiber; therefore, the fiber represents a shared risk. In our framework, failure signatures collected from a fault detection system and the risk model are input to a localization algorithm which outputs a hypothesis corresponding to a set of likely faults in the network. In order to demonstrate the benefits of our localization framework in this paper, we describe two canonical fault localization scenarios where we successfully applied our framework—link and path failure localization. We have implemented and deployed localization systems for each in a tier-I ISP.

Portions of this manuscript appeared in ACM/USENIX NSDI 2004. This work was supported in part by the National Science Foundation through a NSF CAREER award (CNS-0347949).

R. R. Kompella and A. C. Snoeren are with the Department of Computer Science, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: ramana@cs.ucsd.edu; snoeren@cs.ucsd.edu).

J. Yates and A. Greenberg are with AT&T Labs – Research, Florham Park, NJ 07932 USA (email: jyates@research.att.com; albert@research.att.com).

LINK FAULT LOCALIZATION: In this scenario, we consider an IP network with point-to-point links (optical circuits) between different routers on top of an underlying optical topology. Monitoring alarms associated with the optical circuit failures are typically generated on an individual basis—for example, a router failure will appear as a failure of all links terminating at that router. Best current practice requires a manual correlation of the individual link failure notifications to determine that they are all a result of a common network element (e.g., router). In more complicated failure scenarios, however, it is substantially more challenging to group individual alarms into common groups, and often difficult to even identify in which layer the fault occurred (e.g., in the transport network interconnecting routers, or in the routers themselves).

PATH FAULT LOCALIZATION: For path failure localization, we consider an MPLS network, where MPLS labels are established between two edge routers along the shortest path identified by the underlying interior gateway protocol (IGP) such as OSPF or IS-IS. Such an architecture is currently used by many tier-I ISP networks. One commonly observed failure scenario occurs when OSPF re-routes due to a change in IGP link weight, but MPLS does not update its label-switched paths accordingly. Therefore, all the packets belonging to that particular MPLS tunnel attempt to traverse the old path only to be black-holed eventually in the network. While router work-arounds are being developed to deal with this problem, it is unlikely such problems are going to disappear completely. Another common reason for such black holes involves operator error. For example, a human accidentally has forgotten to turn on MPLS on one of the newly added interfaces causing the MPLS packets to not be routed further. In other failure instances, MPLS control plane is working properly (hence, no alarm), yet there is corruption in the forwarding plane due to poor implementations and/or configuration errors [9].

For both these scenarios, using real failure data and risk models that we constructed and systems deployed on a tier-I backbone network, we demonstrate the efficacy of our localization framework including the specific algorithms that we applied in both these applications. The remainder of this paper is organized as follows. We introduce the risk-modeling approach, localization algorithms and system architecture in Sections II, III, and IV respectively. We then present simulation results and experience with real data for both the systems in Sections V and VI.

II. SHARED RISK ANALYSIS

The key idea for localization of the two failure scenarios mentioned earlier is shared risk analysis. Roughly speaking, a physical object such as a fiber or an optical amplifier represents a *shared risk* for a group of logical entities at an upper layer such as the IP layer. That is, if the optical device fails or degrades, all of the IP components that rely upon that object fail or degrade. Similarly, a link between two routers can form a shared risk for multiple MPLS tunnels (or end-to-end paths) that pass through that link. In the literature, such associations are referred to as *Shared Risk Link Groups* or SRLGs [6]. This concept is well understood in the context of network planning

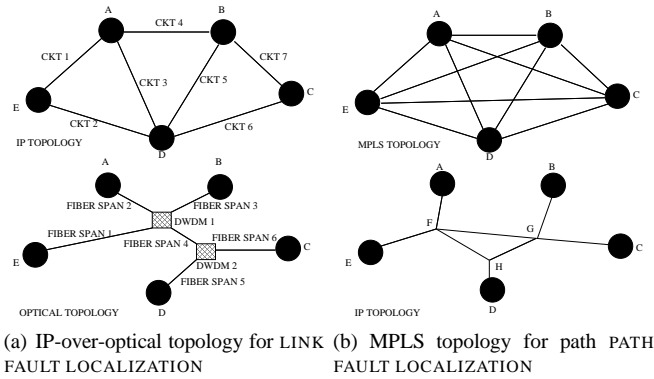


Fig. 1. Example of SRLGs for the two example failure scenarios considered in the paper.

where backup paths are chosen such that they do not have any SRLGs in common with the primary path, and sufficient capacity is planned to survive SRLG failures. The application of risk group models to real-time and offline fault analysis, however, has yet to be well explored.

Shared risks in LINK FAULT LOCALIZATION: We construct a model of risks that represent the set of IP links that are likely to be impacted by the failure of each component within the network. The basic IP network topology can be represented as a set of nodes interconnected via links. Inter-domain and intra-domain routing protocols such as OSPF and BGP operate with a basic abstraction of a point-to-point link between two routers. These IP links in turn are overlaid on top of an underlying optical network. Despite appearing logically disparate, multiple IP links can share components in the optical network, thus leading to shared risks in the network.

In many tier-I backbone networks, each inter-office IP link is carried on an optical circuit (typically using SONET). This optical circuit in turn consists of a series of one or more fibers, optical amplifiers, SONET rings, intelligent optical mesh networks and/or Dense Wavelength Division Multiplexing (DWDM) systems [27]. These systems consist of network elements that provide O-E-O (optical to electrical to optical) conversion and, in the case of SONET rings or mesh optical networks, protection/restoration to recover from optical layer failures. Multiple optical fibers are then carried in a single conduit, commonly known as a fiber span. Typically, each optical component may carry multiple IP links—the failure of a physical component results in the failure of all of these IP links. Therefore physical elements such as fibers, fiber spans, optical components constitute shared risks. In addition to the optical components, our SRLG model also includes routers (set of links originating from the router), modules (set of links common to a module within a router) and ports (the interfaces themselves). We can extend beyond physical components and include other logical software entities such as OSPF area, router software version, etc., in the list of SRLGs.

We illustrate this concept in Figure 1(a) with a simple network consisting of five nodes connected via six IP-over-optical links (circuits CKT 1 through 7). The logical topology is shown in the upper half while the bottom half of Figure 1(a) shows the optical layer topology over which the IP links are

routed. In the Figure 1(a), these shared risks are denoted as FIBER SPAN 1 to 6, DWDM 1 and 2. CKT3 and CKT5 are both routed over FIBER SPAN 4 and thus would both fail with the failure of FIBER SPAN 4. Similarly DWDM 1 is shared between CKT 1, 3, 4 and 5, while CKT 6 and CKT 7 share DWDM 2.

Shared risks in PATH FAULT LOCALIZATION: In this problem, MPLS tunnels share individual IP links and hence form the shared risks. Each individual link failure or degradation can simultaneously affect all the MPLS tunnels that ride through that particular link. In fact, extending it a bit further, if there are multiple VPNs that ride through a given MPLS tunnel that rides through a failed link, all these VPNs are affected by a failure on that link. However, for the purposes of this paper, we focus on the failure of MPLS tunnels, while noting that we can extend this arbitrarily into higher layers if need be.

For example, in Figure 1(b), edge nodes A through E are connected via intermediate nodes F through H. Clearly, from the figure, we can observe that the set of paths A-F-G-B, A-F-G-C, E-F-G-C, E-F-G-B (and the corresponding reverse paths) share a common link—the link from F to G. Hence this link {F,G} forms a shared risk for this group of paths.

Note that we do not include all the optical components and other SRLGs we model for LINK FAULT LOCALIZATION in this context. The reason is that our experience with real failures indicates that most MPLS black holes occur during IP topology changes and are not due to failures involving optical components (further discussed in Section VII).

III. FAULT LOCALIZATION VIA RISK MODELING

In the previous section, we have discussed two instances where a given shared risk failure affects all the entities that are dependent on it. In this section, we use these risk models to build a fault localization methodology that can aid network operators in troubleshooting failures in an automated fashion. Our approach has three main components—fault detection, risk modeling and fault localization. Failures detected at a higher layer (such as failed IP links or MPLS tunnels) are fed into a localization engine that spatially correlates these failures according to the risk model (based on the underlying topology) to identify a small set of likely locations of the failure. This localization step is the primary reconnaissance to a final—and often (necessarily) manual—step of actually diagnosing the root cause of the failure and fixing the problem.

A. Failure detection and risk model

One of the main inputs to a localization engine is the *failure signature*. The failure signature consists of a set of symptoms observed due to a given failure. The symptoms themselves are typically detected through explicit monitoring (either through probes or lower-layer alarms) at that layer. For example, in order to check IP link integrity, each router in the network injects periodic “HELLO” messages to the router at the other end, which then acknowledges the receipt of the message [23]. The set of IP link failures that are temporally correlated (occur almost simultaneously) constitutes the failure signature in this scenario. Similarly, in monitoring MPLS tunnels for black

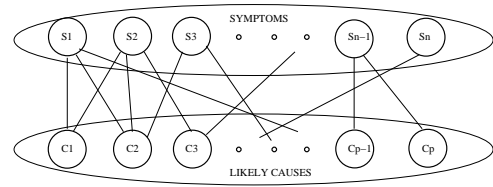


Fig. 2. Bipartite graph modeling of the failure localization problem as done in SCORE system.

holes, a monitoring server typically establishes connections (using, say, GRE tunnels [13]) with each edge router and injects periodic probes to every other destination edge router in the network and reports if any of the probes are lost. Therefore, the set of origin destination pairs (OD-pairs) that have lost connectivity (based on dropped probes) constitutes the failure signature in this scenario.

The other input required for fault localization is the risk model. We use a bipartite graph to represent the dependency between possible observable symptoms and corresponding likely causes as shown in Figure 2. An edge exists between a symptom and a likely cause if that symptom can be observed given a failure in that root cause. As shown in Figure 2, the top partition consists of the universal set of symptoms in the failure signature, and the bottom partition consists of the likely causes. We refer to the symptoms that have been reported by the failure detection system as *observations*. In LINK FAULT LOCALIZATION, the symptoms are IP link failures and the likely root causes are the SRLGs such as fiber spans, optical amplifiers, etc., while in PATH FAULT LOCALIZATION, observations are failed MPLS tunnels, while the underlying IP links make up the potential root causes.

B. Fault localization

We then apply *spatial correlation* for fault localization. The observations part of the failure signature are intersected according to the risk model to identify common shared risks. These shared risks form the most likely explanation for the failure signature and hence the localization engine outputs this set of shared risks as the *hypothesis*. For example, in our LINK FAULT LOCALIZATION example in Figure 1(a), if CKT1 and CKT4 both fail simultaneously, it is highly likely that the failure was in the optical component DWDM1 since this is the only shared risk among these circuits (or IP links). Similarly, in our PATH FAULT LOCALIZATION example in Figure 1(b), if the set of paths A-F-G-B, A-F-G-C, E-F-G-C, E-F-G-B all fail in a given time interval (temporally correlated failures), spatial correlation leads to the only link that is common to all these paths—the link from F to G. The localization engine would, therefore, return the singleton set {F-G} as the hypothesis in this failure scenario.

In many situations, however, it is not as simple to localize the failure as in the above toy examples. The localization engine does not have access to an oracle to determine whether a single failure, dual failure, or multiple failures occurred in the network; it has to determine the cardinality of the set of failures based on the failure signature alone. For any given failure signature, there could be many different

Algorithm 1 GREEDY(FailureSignature F)

```

1: E = {};
2: U = FailureSignature F;
3: H = {}; // Hypothesis set
4: while (U ≠ {}) do
5:   for (observation o ∈ U) do
6:     //All SRLGs for this observation o
7:     srlgVector = getAllSRLGs(o);
8:     //Update stats of SRLGs in srlgStatVector
9:     updateSrlgStats(srlgVector);
10:  end for
11:  srlgSet = identifyCandidates();
12:  //Move observations covered by SRLGs in srlgSet
13:  //from U to E
14:  moveEvents(srlgSet, E, U);
15:  addToHypothesis(H, srlgSet);
16: end while
17: return H;

```

likely explanations. Exploring all the possibilities is often not feasible. Such enumeration is also likely to contain many cases that are not typically observed in practice. We therefore attempt to identify hypotheses that can explain a given failure signature with the smallest set of failures. Our approach is in accordance with the principle of Occam’s razor, that suggests that the simplest explanation is most likely. In order to find the simplest explanation (the one that hypothesizes least number of failures), an obvious strategy is to find the minimum set cover for the bipartite graph. Each shared risk is associated with a set of observations according to the risk model. While the general set-cover problem is NP-hard, the greedy approximation finds a solution guaranteed to be an $O(\log n)$ approximation to the optimal.

We use the core algorithm GREEDY (shown in Algorithm 1) to iteratively pick SRLGs that are the best according to some metric; *updateSrlgStats()* computes the metric for all SRLGs and *identifyCandidates()* picks the best SRLGs in the current iteration. It then prunes the set of observations explained by these candidate SRLGs (in *moveEvents()*) from the failure signature and repeats the process until no more observations remain in the failure signature. In the classic set-cover problem, the metric used to determine which candidate SRLGs to pick in any iteration is the number of new observations covered by them. In other words, in each iteration, GREEDY just picks those SRLGs that have the most number of edges in the bipartite graph among those in the failure signature. However, we abstract the *identifyCandidates()* and *updateSrlgStats()* routines as black boxes deliberately, since the metrics can change according to the particular problem at hand.

C. Imperfections due to operational realities

The two main factors that affect these routines are noise and incompleteness in the failure signature and churn in the risk models. We explain how the algorithm can be adapted to factor in these imperfections in the next section.

1) *Imperfections in failure signature:* In most real networks, there is inherent noise due to various types of routing, congestion, maintenance and other related events. Due to this noise, there could be *spurious observations* in the failure signature that are not directly related to the particular real failures that we are localizing, thus complicating localization. Besides noise, inherent inefficiencies in the detection system or losses of the failure notifications due to unreliable transport mechanism could lead to an *incomplete failure signature*. The extent of both these imperfections vary based on the particular domain.

In the LINK FAULT LOCALIZATION, the detection mechanism uses both lower layer alarms (such as a Loss-of-signal alarm) and high-frequency router-to-router probes (such as OSPF Hello messages) to detect loss in connectivity. Therefore, detection of IP link layer faults is typically complete, in the sense that all IP links that fail due to a problem in the underlying optical network are detected with extremely high probability. However, a few fault messages can get dropped during transmission leading to relatively small levels of incompleteness in the failure signature.

On the other hand, for PATH FAULT LOCALIZATION, the detection mechanism is primarily based on end-to-end probes that operate at a much lower frequency. So for many reasonable failures, the failure signature is not complete. Of course, if the failure persists, eventually all low frequency probes that go through the failure are bound to detect the failure. But this means that we need to wait for a long time until the failure signature is complete. Furthermore, because the topology is large (order of a million paths), any failure can affect a significant number of paths leading to an overwhelming number of observations (large failure signature set). In order for the system to operate in real time, it is often required to down-sample the failure signature. Therefore, in this domain, the failure signature is expected to be largely incomplete. In addition to the incomplete failure signature, noise can cause some probes to be dropped in the network thus adding spurious observations to the failure signature.

2) *Imperfections in the risk model:* As is the case with the failure signature, there are likely to be imperfections in the risk model. These imperfections stem from inherent churn in the network and depend on the particular failure domain under consideration. In the LINK FAULT LOCALIZATION, the optical topology is more or less static¹. However, the risk models for the large majority of optical networks are maintained through human-entered databases that may drift away from reality leading to errors in the database. These errors can affect the localization results and hence must be factored into the localization algorithm.

On the other hand, the risk model in PATH FAULT LOCALIZATION for MPLS paths is primarily derived from the underlying IP topology. IP topology is inherently subject to churn due to various routing changes caused during failures, congestion and maintenance activities. Therefore, the risk model needs to be computed on-the-fly for this scenario from

¹Of course, some optical networks perform lower layer re-routes to mask higher layer failures, in which case, paths need to be dynamically computed. We ignore this detail for the purposes of this paper.

the topology snapshots during the failure interval. Because the topology is extremely large, it is often impractical to construct the entire risk model; the fault localization algorithm needs to be designed to operate with a partial risk model.

3) *Handling imperfections*: We use two different metrics to identify the candidate SRLGs in each iteration of the basic GREEDY algorithm. Let G_i correspond to the i th shared risk in the network and $|G_i|$ be the total number of observations that belong to the SRLG G_i . $|G_i \cap F|$ is the number of elements of G_i that also belong to F , the failure signature. We define *hit ratio* of the group G_i as $|G_i \cap F|/|G_i|$. In other words, the hit ratio of a group is the fraction of elements in the group that are part of the failure signature. The *coverage ratio* of a group G_i is defined as $|G_i \cap F|/|F|$, i.e., the fraction of the observation explained by a given risk group.

If we have access to the complete failure signature and an accurate risk model, we can exploit the fact that every failed shared risk would have all associated observations in the failure signature. In other words, the hit ratio for the failed shared risks should be 1, so an optimal algorithm would select SRLGs with the highest coverage among those with a hit ratio of 1. In the LINK FAULT LOCALIZATION, however, while we potentially have access to the entire failure signature and risk model, they are likely to contain a relatively small number of errors. In order to account for these errors, we define the SCORE algorithm [19] that considers SRLGs with a hit ratio greater than a particular error threshold, which is generally slightly less than 1. We explain later in Section IV-C how we determine the error threshold in practice.

On the other hand, if we do not have access to the entire failure signature or associated risk model, we cannot compute a meaningful hit ratio for each and every SRLG in the first place. Therefore, in such situations, the greedy heuristic picks those SRLGs that have the highest coverage-ratio in every iteration to output the hypothesis. In the resulting algorithm MAX-COVERAGE, the *identifyCandidates()* routine is similar to that of SCORE algorithm, except the condition on the hit ratio is eliminated. This algorithm, therefore, is applicable to PATH FAULT LOCALIZATION where we can compute the coverage ratios of all the SRLGs but not the hit ratios.

D. Metrics for comparison

In order to evaluate the performance of our algorithms, we need access to the ground-truth for the failures. We define two metrics for comparing our hypothesis with the ground-truth—accuracy and precision.

Accuracy is the fraction of elements in ground truth G also contained in the hypothesis H , or $|G \cap H|/|G|$. If G is a proper subset of H , then the accuracy is 1. This metric alone can not capture the efficacy of the localization algorithm, however. For example, if we design an algorithm that always outputs U where U is the universal set of elements, then $G \subseteq U$ by definition, thus always leading to an accuracy of 1. Such an algorithm obviously is not very useful in practice.

Therefore, we define another metric called *precision* that quantifies the size of the hypothesis in relation to the ground truth. It is defined as the fraction of elements in the hypothesis

that are also present in the ground-truth or $|G \cap H|/|H|$. In effect, precision captures the amount of truth in the hypothesis. For example, a precision of 0.9 would imply that the 90% of the elements in the hypothesis match the ground truth.

High accuracy in a localization system means few false negatives, and high precision implies few false positives. Typically, most algorithms tend to trade one metric for the other depending on how conservative or aggressive the algorithm is. A conservative algorithm tends to include all the possibilities in order to achieve better accuracy while losing precision, while an aggressive algorithm includes only the significant ones thus gaining precision while somewhat sacrificing accuracy. Our goal is to ensure that both these metrics are within reasonable bounds.

IV. SYSTEM ARCHITECTURE

So far, we have described the three required ingredients for fault localization—failure detection, risk model and the localization algorithm. In this section, we elaborate on how these components are constructed for both the LINK AND PATH FAULT LOCALIZATION problems.

A. Failure detection

The failure signature upon which spatial correlation is applied is obtained from various types of network monitoring data sources depending on the particular failure scenario. In any case, we assume a monitoring system continuously generates a raw event stream (as shown on the left in Figure 3) with associated timestamps. For detection of link faults, we use router syslogs generated by the router when it detects a link to be down and path failure notifications generated in response to the loss of end-to-end probes constantly polling connectivity between various origin-destination pairs.

Data sources that are based on discrete asynchronous events, such as router syslog messages need to be *clustered* to identify the failure signature. Note that a failure can cause symptoms that appear slightly off in time either due to time synchronization issues across various elements, or propagation delays in an event to be recorded. There are many different ways to cluster events. In LINK FAULT LOCALIZATION, we use a clustering algorithm based on gaps between failure events. We consider the largest chain of events that are spaced apart within a set threshold (called quiet period) as potentially correlated events. The intuition is that two events that occur within a time period less than a given threshold (e.g., 30 seconds) can be attributed to the same failure. In contrast, for PATH FAULT LOCALIZATION, due to the presence of an excessive number of events due to noise in the network, a clustering scheme such as above results in clustering together all events into one cluster. In order to deal with this, we divide time into 15-minute bins.

B. Risk model

In the middle portion of Figure 3, we show how the risk model is constructed for the two different failure scenarios. In LINK FAULT LOCALIZATION, the risk model is constructed from various disparate databases representing the

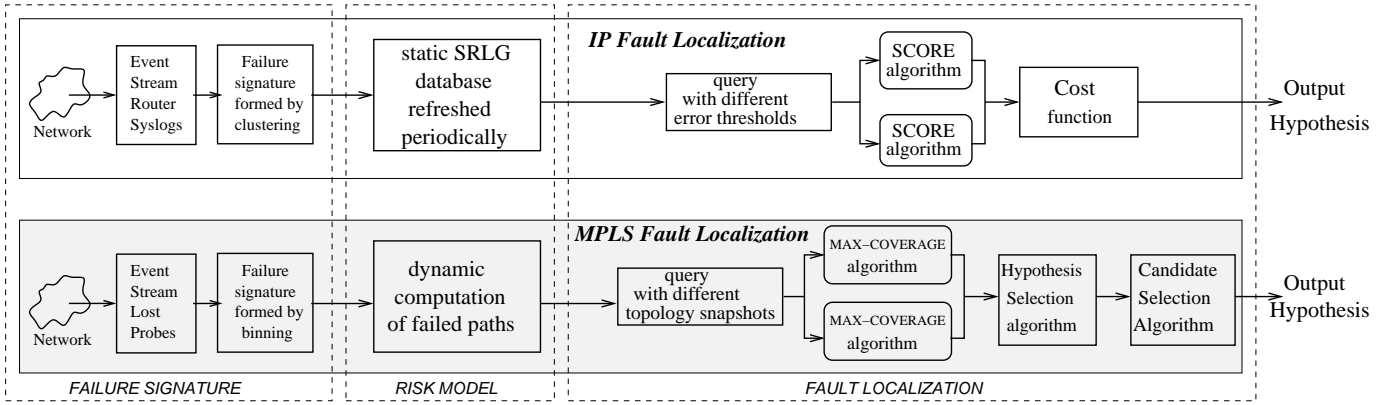


Fig. 3. System flow diagram for IP and MPLS fault localization problems.

different types of shared risks. For optical layer shared-risks, the SONET components that particular IP links traverse are extracted from databases populated from operational optical element management systems. Other risk groups such as OSPF area, router modules, etc., are populated by periodically polling configurations from the various network elements. The underlying databases track the network and therefore exhibit churn. We cope with database churn by regenerating risk groups multiple times during the course of the day.

For PATH FAULT LOCALIZATION, we construct the risk model dynamically from IP topology snapshots obtained through an OSPF monitor [30]. Because a significant number of failures in the MPLS domain are due to topology changes, we need to consider the topology snapshots both before and after the failure depending on the exact nature of the failure. Due to the dynamic nature of the risk model, we do not construct the entire topology for every snapshot, but instead obtain only the paths for the OD-pairs in the failure signature. Since there could be multiple paths between a given OD-pair due to equal-cost multi-path routing [15], we place the OD-pair in the risk groups corresponding to all the links that lie on at least one shortest path between the OD-pair.

C. Localization algorithm

Finally, once the failure signature is obtained and the risk model is constructed, we can perform fault localization as shown in the right portion of Figure 3. We proposed core algorithms for IP and MPLS fault localization in Section III-B; here, we discuss additional steps of processing required on top of the core algorithms for each of these systems.

1) LINK FAULT LOCALIZATION: Rather than fixing one particular error threshold for the system, we query the SCORE algorithm with multiple error thresholds (reducing from 1.0 to say 0.5) to obtain many different hypotheses as shown in Figure 3. The hypotheses obtained using different relaxations are then evaluated based on a cost function that depends on the error threshold and the size of the hypothesis. We use the ratio between the size of the hypothesis and the threshold; we seek to identify cases where a small relaxation in the threshold (an error threshold of 0.9, say) can reduce significantly the number of groups in the final hypothesis.

2) PATH FAULT LOCALIZATION: We use the MAX-COVERAGE algorithm that iteratively selects the links covering the most number of observations in the failure signature. There are two additional issues that arise in practice that need to be addressed. First, there can be potentially many different topology snapshots within a given failure interval and the question is which topology to use for localization. Second, additional post-processing is required to increase the precision of the system to handle noise.

To address these issues, we first generate multiple hypotheses for a given failure signature using all the available topology snapshots in the failure interval, and use a *hypothesis selection algorithm* for selecting the hypothesis using different topology snapshots, and *candidate selection algorithm* for selecting candidate links within the hypothesis. The hypothesis selection algorithm we implemented (called UNION) outputs the union of hypotheses generated with each of the available topology snapshots. By considering all possibilities there is no loss in accuracy compared to an oracle that somehow knows the right topology to pick. Recall that the localization algorithm adds suspect links to the hypothesis until the hypothesis completely explains all the failed probes, including those observations that arise from inherent noise in the network. Therefore, the candidate selection algorithm (called ABSOLUTE) removes candidate links from the hypothesis that explain fewer than a threshold number of observations.

D. Implementation

For both the systems, we implemented the main localization algorithms in C/C++ and the rest in Perl. We also implemented a Web-based user interface for both systems. The IP fault localization system contains slightly more than 1000 lines of C and about 2500 lines of Perl code, while the MPLS fault localization system consists of about 5000 lines of C++ and about 2000 lines of Perl code.

V. EVALUATION RESULTS FOR LINK FAULT LOCALIZATION

In this section and the next, we evaluate the performance of systems we deployed to address both the LINK AND PATH FAULT LOCALIZATION problems in a tier-1 ISP backbone, using simulations as well as real network failure data. We start by considering LINK FAULT LOCALIZATION.

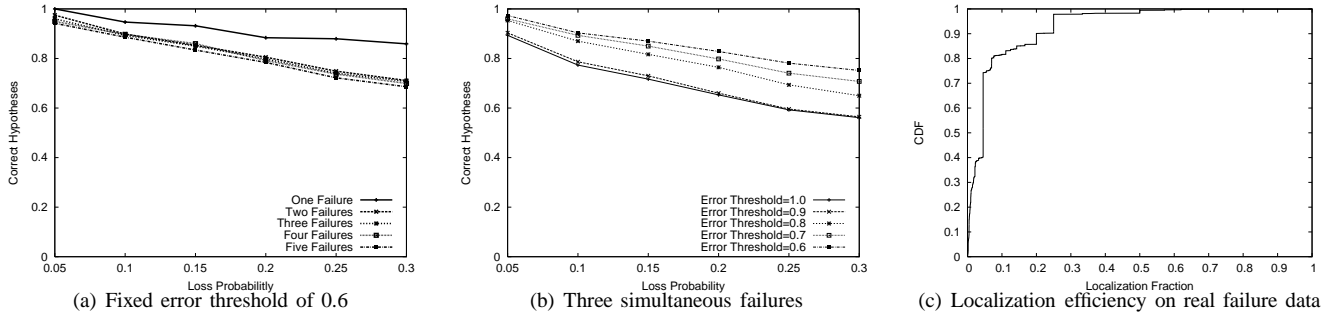


Fig. 4. The sub-plot (a) shows the accuracy of the SCORE algorithm with a fixed error threshold of 0.6 with a varying number of simultaneous failures as a function of observation loss. Sub-plot (b) illustrates the impact of different error thresholds. Sub-plot (c) shows the localization efficiency over 3000 real failure events.

A. Simulation results

The main goal of the initial experiments is to evaluate the accuracy of the SCORE algorithm for IP fault localization within a controlled environment by using emulated faults. We used an SRLG database constructed from the network topology and configuration data of a tier-1 service provider’s backbone. We injected varying numbers of simultaneous faults and studied the efficacy of the algorithm in the presence of database errors and lossy fault notifications.

1) *Algorithm accuracy*: We simulated multiple simultaneous failures by picking risk groups at random from the set of all network risk groups, and inputting the union of all IP links that are associated with these risk groups to the SCORE algorithm. We evaluated the accuracy of the algorithm in terms of the fraction of faults correctly localized by the algorithm.

As a baseline experiment (not shown), we measure the accuracy of SCORE as a function of the number of simultaneous faults for different types of SRLGs (ports, modules etc.). We observed that the accuracy of the algorithm on these data sets is greater than 95% for all types of risk groups for fewer than five simultaneous failures. For failure scenarios involving only a single fiber cut, router failure or module failure, which is the common case for hard failures, our simulation results indicate the accuracy is near 100%. These high accuracy numbers are expected since there are no imperfections; SCORE outputs a wrong hypothesis only when fault signatures from two different faults combine to produce another fault’s signature, which is typically rare.

2) *Imperfect fault notifications*: Next, we simulate imperfections due to operational realities, such as the loss of failure notifications. We consider three parameters: the error threshold used in the SCORE algorithm, the number of simultaneous failures, and the loss probability (which represents the percentage of IP link failure notifications lost for a given failure scenario).

Figures 4(a) and 4(b) demonstrate the accuracy of the algorithm under a range of loss probabilities for different numbers of simultaneous failures and error thresholds. Specifically, the figures plot the percentage of correct hypotheses as a function of the error probability. In Figure 4(a) the algorithm error threshold is fixed at 0.6 and the number of simultaneous failures is varied from 1 to 5. In Figure 4(b), the algorithm error threshold is varied from 0.6 to 1.0, while the number of

simultaneous failures is set to 3. As expected, increasing the loss probability reduces the accuracy of the algorithm. Under three simultaneous failure events and an error probability of 0.1, we can observe from Figure 4(b) that an algorithm error threshold between 0.7 and 0.8 restores the accuracy of the SCORE algorithm to around 90%. However, if we mandate perfect matching of failure observations to SRLGs (i.e., error threshold = 1.0), then our accuracy in isolating our fault drops to around 78%. This shows the necessity and effectiveness of the error thresholds introduced into the algorithm for fault localization in the face of noisy event observation data.

B. Experience with real failure data

The LINK FAULT LOCALIZATION system has been operating in a tier-1 backbone network in an off-line fashion to localize IP link failures reported in the network for more than a year. The implemented system operates on a range of fault and performance data, including IP fault notifications and optical layer performance measures. However, we limit our discussion here to our experience with link failure events reported in router syslogs.

1) *Manual analysis of real failures*: Determining whether or not the system correctly localized a given fault requires identification of the root cause of the fault via other means. In many cases, identifying this root cause involves sifting through large amounts of data and reports—a tedious process at best. We therefore selected a set of 18 faults for which we identified the root cause of the problem and compared with the hypothesis output by the system.

Overall, we were able to verify manually that SCORE successfully localized all of the 18 faults studied to the failed network elements (shown in Table I). However, when we used a threshold of 1.0 (i.e., mandated that an SRLG can be identified if and only if faults were observed on all corresponding IP links), we were typically unsuccessful—particularly for router failures, and for the protocol bug reported. In the majority of the router failures, even though these events corresponded to routers being rebooted, the remote ends of the links terminating on these routers did not always report associated link-level events. This may be due to a number of possible scenarios—the events may never have been logged in the syslogs, data may have been lost from the syslogs, the links may have been operationally shut down and, hence, did

Type of problem	Component Name	#SRLGS (Thld.=1.0)	Final Thld	#SRLGS (Thld.=Final)	#Correct localized	Comment
ROUTER						
Router	Router A	27	0.8	1	1	No event reported by some links
Router	Router B	20	0.9	3	3	No event reported by some links
Router	Router C	12	0.7	1	1	No event reported by some links
Router	Router D	1	1	1	1	-
Router	Router E	18	0.8	1	1	No event reported by some links
Router	Router F	1	1	1	1	-
Router	Router G	4	1	4	4	One router and three links failed
MODULE						
Module	Module A	1	1	1	1	-
Module	Module B	1	1	1	1	-
Module	Module C	1	1	1	1	-
SONET						
OA	Sonet A	8	0.9	2	1	No observation reported by one link and database problem
Failed Transceiver	Sonet B	1	1	1	1	-
Short term Flap	Sonet C	2	0.7	1	1	No observation reported by one link
OA	Sonet D	2	0.6	1	1	No observation reported by one link
FIBER						
Fiber Cut	Fiber A	3	0.5	1	1	Database problem
Fiber Span	Fiber Span A	1	1	1	1	-
PROTOCOL						
Protocol Bug	OSPF Area A	20	0.7	4	4	Incorrect SRLG modelling
Protocol Bug	OSPF Area A	4	1	4	4	OSPF Area A MPLS enabled interfaces

TABLE I
SUMMARY OF 18 ACTUAL, DIAGNOSED FAILURES IN A TIER-1 ISP.

not fail at this point in time, or the links were not impacted by the reboot. Independent of why the link notifications were not always observed, the router failures were all successfully localized when the threshold was marginally reduced. This highlights the importance of the threshold concept in the SCORE algorithm to localize faults in operational networks.

We studied four different SONET network element failures. The first—an optical amplifier failure—induced faults on 13 IP links. With a threshold of 1.0 our algorithm identified eight different SRLGs as being involved in the hypothesis. However, as the threshold was reduced to 0.9, the hypothesis size reduced to only two SRLGs, one of them being the actual failed optical amplifier. Further reductions in this threshold did not reduce the number of SRLGs in the hypothesis. Upon investigation, we found that our SONET database was missing one of the IP links in the failure signature. Thus, the SCORE algorithm was unable to attribute this particular IP link to the SONET SRLG, and instead incorrectly concluded that a router port was also involved (the second SRLG) to explain this individual link. The remaining 12 IP links however were successfully attributed to the failed optical amplifier. This example illustrates why lowering the threshold is required when there are errors in the database.

The other three SONET failures were all correctly isolated to the SRLG containing the failed network element; in two cases we again had to lower the threshold used within the algorithm to account for links for which we had no failure notification. In one of these cases, the missing link was

indeed a result of the interface having been operationally shut down shortly before the failure. Our topology snapshots are generated on a daily basis; therefore the topology change was not reflected in the risk model. These examples clearly indicate the need to deal with operational issues such as incorrect data and erroneous databases.

On another previously identified failure scenario impacted by a SRLG database error (fiber A in Table I), the system was unable to identify a single SRLG as being the culprit even as the threshold was lowered, because no SRLG in the database contained all of the circuits reporting the fault. So again, a database error was highlighted by the system's inability to correlate the failure to a single SRLG.

The final case that we evaluated was one in which a low-level protocol implementation problem (software bug) impacted a number of links within a common OSPF area. This scenario occurred over an extended period of time, during which three other independent failures were simultaneously observed in other areas. When a threshold of 1.0 was used in the SCORE algorithm, the event in question was identified as being the result of 20 independent SRLG failures—a large number even for the extended period of time. As the threshold was reduced to a final value of 0.7, the event was isolated to four individual SRLGs: three SRLGs in other OSPF areas (corresponding to the independent failures) and the OSPF area in question. Thus, the SCORE algorithm was correctly able to identify that the event corresponded to a common OSPF area. However, further investigation uncovered that the reason

why not all links in the OSPF area were impacted was that only those interfaces that were currently MPLS-enabled were affected. Thus, an additional SRLG was added to our SRLG database that incorporated the links in a given area that were MPLS-enabled; application of this enhanced SRLG database successfully localized all of the SRLGs impacted by the four simultaneous failures with a threshold of 1.0.

2) *Localization efficiency*: While the 18 faults we have studied demonstrate the ability of our system to correctly localize faults, it does not give an indication of how well our system could reduce the number of suspects. We evaluate the ability of our system to identify a small set of candidate faults using a metric we call *localization efficiency*. Localization efficiency is defined as the ratio of the number of suspect components after localization to the number before. In other words, it is the fraction of components that are identified by our localization algorithm that likely explains a particular fault (or observation) out of all the components that could cause a given fault (according to the SRLG model). Define $G_i = \{G_{i1}, G_{i2}, \dots, G_{ik}\}$ as the set of groups that a circuit c_i belongs to. O is an observation consisting of circuits c_1, c_2, \dots, c_i . The most probable hypothesis $H \subset \bigcup_{k=0}^i G_k$ is the hypothesis output by the algorithm. The localization coefficient is given by $|H|/|\bigcup_{k=0}^i G_k|$.

Figure 4(c) shows the cumulative distribution function of the localization efficiency in 3000 real faults experienced in a Tier-1 ISP. From the Figure 4(c), we can clearly observe that our system could localize faults to less than 5% for more than 40% of the failures and to less than 10% for more than 80% of the failures. This clearly demonstrates that the SCORE algorithm can efficiently ferret out likely causes from of a large set of possible causes for a given failure. Unfortunately, due to the extensive manual labor necessary to diagnose failures, we do not know the true root cause of all 3000 failures and cannot measure accuracy on this dataset.

VI. EVALUATION RESULTS FOR PATH FAULT LOCALIZATION

Similar to the previous section, we use both simulation and offline analysis of real failure data to evaluate the ability of a system based on MAX-COVERAGE to address the PATH FAULT LOCALIZATION.

A. Simulation results

We built a simulator that can inject artificial failures that mimic real-life failure scenarios, obtain observations corresponding to the failure, and then apply localization algorithm to evaluate the accuracy. As described earlier, the entire failure signature is generally not available for fault localization. Therefore, for these simulations, we vary the fraction of the failure signature and compare the accuracy and precision of the localization algorithm. The fraction of the signature α is directly related to the duration of the failure and the rate at which probes are issued. We simulate a fault detection system that issues periodic probes from every router at a rate of one per minute. Therefore, any persistent failure that lasts more

than a minute should be captured completely by the monitoring system.

For our simulations, we use the same tier-I network topology for which we present experiences with real failure data later in Section VI-B. We simulate three different scenarios: without any noise, with random noise, and with structured noise. The scenarios without any noise, while unrealistic, determine an upper bound on the accuracy of the algorithm. Random noise simulates failure scenarios where the failure signature is mixed with spurious probe losses in the network, often due to transient congestion. In our simulations, we added a random number of spurious observations with an average of 80 per failure. Structured noise, on the other hand, models scenarios where failures of short duration overlap with the main failure(s) and appear as noise. As an arbitrary starting point, we generated structured noise by failing 5 links at random for 5 seconds; the simulated “real” faults last for 60 seconds.

1) *Accuracy of the localization algorithm*: We measured the average accuracy as a function of the fraction of signature, α for varying number of simultaneous failures and for the three different types of failure scenarios. In simulations with no noise, we observed that the average accuracy is well above 90% even with five simultaneous failures and only 1% of the failure signature. Intuitively, this is because the groups of OD-pairs that form the failure signature for each link are large; hence, even a small fraction can create a sample of observations that can uniquely identify the injected failure.

When random noise is introduced into the failure signature (shown in Figure 5(a)), we observed that the accuracy is reduced. In particular, lower fractions of the failure signatures are much more susceptible to noise than the higher ones. For example, at $\alpha = 0.01$, the average accuracy is only 60%, while it reaches 90% at $\alpha = 0.16$. This is because at smaller fractions of the failure signature, there is a higher chance that the spurious observations can morph the failure signature of one shared risk into another. Since our algorithm tries to identify risk groups with highest coverage first, it is likely that the original failure signature and the noise will add up to a stronger candidate risk group different from the injected failure.

With structured noise (in Figure 5(b)), we observe a similar, although less pronounced, phenomenon. The accuracy dips a little compared to the case when there is no noise but is higher than with random noise. The reason is as follows. Since noise is more structured in this case, the resultant failure signature is a composition of α fraction of the original failure signature and $\alpha \times \beta$ fraction of the five noise links, where β is the ratio of the failure durations of the noise and the original failure ($\beta = 5/60$ in our simulations). Since even a small α is enough to achieve high accuracy even for five simultaneous failures with no noise, we achieve high accuracy for the structured noise case.

2) *Precision of the localization algorithm*: Along with the accuracy, we also evaluated the precision of the localization algorithm—the fraction of truth in the hypothesis—with varying signature fraction α . Without noise, the algorithm enjoys extremely high precision, especially when $\alpha > 0.16$. Precision

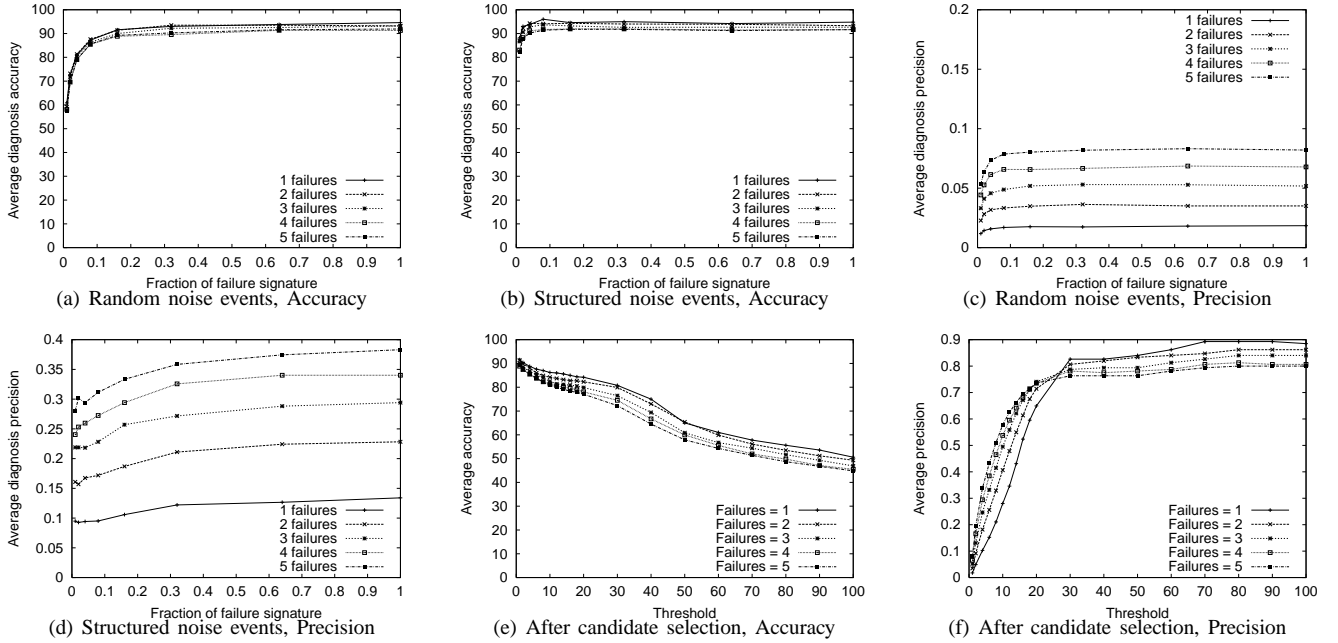


Fig. 5. Figure shows the accuracy and precision of MAX-COVERAGE for different numbers of simultaneous failures and when random and structured noise events are injected along with the actual failure. The y-axis is the average accuracy/precision measured over 500 random link failure scenarios, while the x-axis is the fraction of the failure signature. The last two graphs show accuracy and precision after applying the candidate selection algorithm to the output of MAX-COVERAGE for random noise events. For these two graphs, we fixed the signature fraction to 0.16.

drops with lower values of α since the failure signature is not strong enough to distinguish between multiple contending risk groups. We also observe that the precision, similar to accuracy, is higher for scenarios with one failure than those with five. Lower accuracy implies that part of the ground truth is not present in the hypothesis, which in turn means that the hypothesis might contain additional candidates not part of the ground truth (i.e., lower precision) to cover all observations, thus leading to lower precision.

In the presence of noise, we only considered the injected faults as part of the ground truth and not the noise itself. Thus the localization precision is expected to be much lower as the algorithm tries to cover all observations including those caused by noise, which in turn leads to a larger hypothesis. We can observe this trend for both the random (in Figure 5(c)) and structured noise (in Figure 5(d)) scenarios. For the structured noise, though, the precision is higher than that of random noise; fewer risk groups are required to cover the small fraction of structured noise introduced.

We also observe that the precision is higher for five failures than one in both noisy scenarios, while the opposite is true without noise. The reason for this is straightforward: Since the amount of added noise remains constant across the varying number of simultaneous failures, the number of spurious observations and, therefore, the additional risk groups required to cover them remains similar in all cases. The amount of truth, however, increases linearly with the number of simultaneous failures injected, thereby increasing the overall precision.

3) Candidate selection algorithm to improve precision:

Fortunately, we can improve the precision without significantly decreasing the accuracy by applying the candidate selection algorithm described in Section III-C.3. In Figures 5(e) and

5(f), we plot the accuracy and precision obtained after applying the ABSOLUTE candidate selection algorithm for different absolute thresholds. For this experiment, we fixed the fraction of the failure signature to 0.16, which is still very low.

Eliminating candidate links from the hypothesis that were less than the threshold improves precision significantly until a threshold of about 25, after which the decrease in accuracy out-weighs the additional benefit obtained by increasing the threshold. The optimum threshold will vary and depending on the specifics of the topology and fault detection system and should be derived empirically for a given deployment.

B. Experience with real data

In addition to the simulations, we also collected failure data from a section of a real MPLS-switched tier-I ISP backbone network. The system monitors MPLS tunnels that originate from a subset of edge routers in the backbone, traversing the backbone and finally terminating at other edge routers. Since the MPLS tunnels are established and maintained using the underlying IP topology (through OSPF), any IP-layer failure can impact the MPLS tunnels above the IP layer. The topology consists of a few hundred routers and the probes are transmitted at a periodic rate of one every minute.

The goal of simulation is to stress test our system; we consider every probe loss as part of the failure signature, including those due to noise. In the production version of the tool, we are mainly interested in characterizing large failures, thus noise can be reduced by considering only those OD-pairs with more than a threshold number of dropped probes.

We compared our hypothesis with ground truth extracted from three data sources: OSPF LSAs, syslogs and SNMP data. During many routing events in the network, the topology is

unstable for a short period and probes can get dropped. For such routing incidents, we compare the hypothesis generated by our algorithm with LSAs corresponding to the routing events. In the core backbone network, many IP links (known as composite links [1]) are in fact bundling of many member interfaces, load-balanced by the router. Member interface failures affect only the set of probes traversing that interface after the failure and before the router load-balances again between other members. Since the composite link is active, such failures do not cause OSPF LSAs but appear in router syslogs. In conditions of high link utilization, such as during failures or during maintenance, links can experience heavy packet loss, and therefore, can cause end-to-end probes to get dropped along these links; such congestion events are found in SNMP data.

Note that the ground truth obtained through these data sets is only approximate, as there can be instances when a link failure is reported in the ground truth (using LSAs, syslogs and SNMP data) but the event does not impact traffic forwarding. In these cases, the failure signature will not contain any OD-pairs that are impacted by the spurious LSA or syslog message. In such cases, the natural comparison with our hypothesis (namely, requiring that the ground truth be wholly contained in the hypothesis) is obviously unfair. As a relaxation from this strict accuracy metric (which we refer to as ALL), we define a more conservative accuracy metric called ATLEAST_ONE in which accuracy is defined to be 1 if at least one of the links in the ground truth is contained in the hypothesis and 0 otherwise.

1) *Candidate selection algorithm:* In Figures 6(a) and 6(b), we plot accuracy using both the ALL and ATLEAST_ONE metrics and precision of localization. For this experiment, we picked the hypothesis with best accuracy among those with different topology snapshots. On the x-axis, we vary the cardinality of the failure signature (number of observations) from 50 all the way up to 1000 observations in steps of 50. On the y-axis, the average accuracy/precision corresponding to all failure intervals that have at least x observations is shown. In effect, these figures show the trend in the accuracy/precision as the failures impact more and more OD-pairs.

Several conclusions can be drawn. First, the number of failure intervals reduces exponentially from about 600 bins with more than 50 observations to about 20 bins with more than 1000 observations (not shown). This is expected, since the number of large failures is typically much smaller than the number of small failures. Overall, we obtained accuracy and precision of about 80% when considering failures with more than 150 observations. Second, the accuracy and precision of localization increase as the failure size increases initially from 50 to 150 observations. However, it decreases slightly after that but is inconclusive as the number of failure intervals is too small to have statistical significance. Larger failure signatures can indicate one of three things, assuming noise in the network remains the same across all failures. First, the fraction of the failure signature captured could be higher, i.e., the failure lasted for a larger duration. Second, the failure might have impacted many OD-pairs in the network, thus the failure occurred on a popular link that lies on many paths. Finally, there could have been many simultaneous failures, the likelihood of which

is not insignificant due to router maintenance events. For the first two cases, it is not surprising that our fault localization algorithm performs well, as larger signature fraction means larger accuracy verified using simulations. For the final case, since we use the ATLEAST_ONE metric, there is a strong chance that at least one of the root causes is in our hypothesis. In fact, accuracy using ALL metric is about 40% less than the ATLEAST_ONE metric, both due to the approximate nature of our ground truth as well as the presence of many simultaneous failures in ground truth.

Third, an ABSOLUTE threshold of 30 that selects candidate links in the hypothesis that cover at least 30 observations seems to represent a good trade-off between accuracy and precision. Below this threshold, the precision is significantly lower while accuracy is only slightly higher. Increasing the candidate selection threshold beyond 30 leads to a marginal decrease in the average accuracy, while precision does not improve any further.

2) *Hypothesis selection algorithm:* In Figure 6(c), we plot the precision for the UNION hypothesis selection algorithm that combines multiple hypotheses obtained using different topology snapshots. The x-axis is the ABSOLUTE candidate selection threshold that we vary from 10 all the way up to 500. For each of these candidate selection thresholds, we identify all those failure intervals that had at least one candidate link remaining in the hypothesis after we apply the candidate selection thresholds and compute the average accuracy/precision for these failure intervals. The number of bins reduces with increasing candidate selection threshold (not shown in the Figure) due to the fact that we discard bins that do not have any candidates left in the hypothesis after we apply the threshold.

From Figure 6(c), we can observe that UNION performs similarly to an oracle that can clairvoyantly pick the best out of all the hypotheses generated using different topology snapshots. Because UNION includes the links from in all the hypotheses, it cannot cause a decrease in accuracy according to our definition. As shown by the upper two lines of Figure 6(c), however, precision reduces by a small amount overall. Because the network topology does not change during many of the 15-minute bins containing failures, however, Figure 6(c) under reports the impact of not knowing the correct topology. If we consider only the bins that had a change in topology (where UNION is not just a no-op) precision drops about 15%, as shown in bottom two lines of Figure 6(c).

3) *Real MPLS black holes:* We describe three silent failures we analyzed using our system. In the first incident, misbehavior of a new device that was connected to the periphery of the network caused many routes to go through the device which were then subsequently black-holed. This is a perfect example where we need to consider all the topology changes within a failure interval. In this case, our localization system outputted two candidate links as the hypothesis—the (properly functioning) link before and the (black hole) link after the re-routing of traffic. For this incident, the localization accuracy therefore is 100% while precision is only 50%.

In another failure scenario, the forwarding component of a line card failed to dequeue packets until the card was

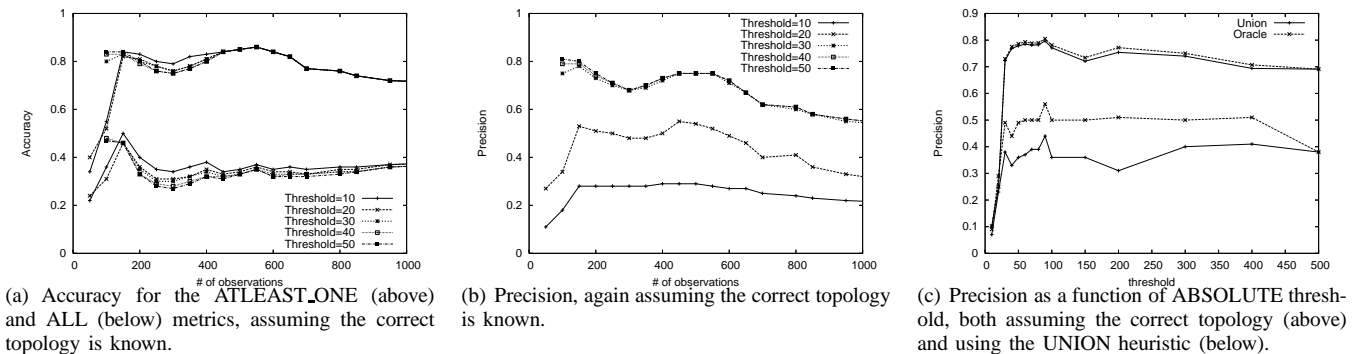


Fig. 6. Accuracy and precision for the MAX_COVERAGE algorithm on real failures from a Tier-1 ISP. The first two graphs assume the correct topology snapshot is known, while the third shows the decrease in precision due to the UNION algorithm that considers the superset of the hypotheses obtained using all the topology snapshots during the 15-minute bin.

reset. Our localization system output a hypothesis that had five candidate links, out of which, when we applied our ABSOLUTE threshold of 30 eliminated the four false positives out of the hypothesis and contained only the actual failed link. This hypothesis therefore has 100% accuracy and precision.

Another known black hole scenario happened due to a misconfiguration causing brief loss in connectivity to MPLS paths that traversed that link. Our localization algorithm output a hypothesis that contained four candidate links, two of which were eliminated after we applied our candidate selection algorithm. Out of the remaining two, one was the actual black hole while the other was a false positive. However, the false positive could not be easily distinguished from the actual black hole since both these links appeared on all the paths corresponding to the impacted OD-pairs.

VII. DISCUSSION

Not all failures require too much time to localize, even among the silent failures. Sometimes, just a simple visual inspection of the failure signature is enough to localize the failure. For example, when all the failed MPLS tunnels share one end-point (failure near the edge), it does not take much effort (a few seconds) to pin-point the location. On the other hand, if the actual fault lies in the core of the network, visual inspection alone is not sufficient to localize unless we join the signature with the risk model. In such cases typically, it can take 30 minutes to a few hours to localize the failure; an automated localization system such as ours reduces this time significantly.

Risk model: Given that the risk model is at the heart of our localization methodology, it is important to devise the right risk model for localization. Primarily, the risk model originates from operational domain knowledge through a careful analysis and understanding of the type of failures that one experiences. For example, for the MPLS fault localization, we observed from operational experience that the primary root-cause for most failures is a topology change. Therefore, our risk model consisted only of IP links. Had we observed that optical failures were causing black holes, we would have modeled optical layer equipment such as amplifiers, fibers etc., in the risk model. Besides, the risk model has to match the failure detection system. For example, in the PATH FAULT

LOCALIZATION, there is no need to model customer facing links in the topology, as the probes never traverse any of those links.

Constructing the right risk model is not easy, even if the category of risks to be modeled were known. For example, in an OSPF network, multiple paths can exist between a given source and a destination if the paths share the same cost (ECMP [15]). The router at the first fork of the paths splits traffic equally among the two paths based on a deterministic but unknown hash-function applied on the source-destination IP address of the packet. It is difficult to precisely know the exact path taken by a dropped probe among these multiple paths. Similarly, in *composite links* [1], many optical circuits are bundled together into one logical IP link and the router splits traffic according to a hash-function. Failures involving only a member circuit can result in only some probes that traverse the member circuit (out of all the member circuits) to fail, while others succeed. In both of these cases, the risk model needs to be constructed based on the instantaneous path traversed by the probe, which is difficult. Therefore, we were forced to consider a risk model that represented the union of all the paths, which is not entirely accurate.

It is often not enough to just model the risks once; determining the right risks to model is a continuous process in many cases. For example, in the IP fault localization scenario, we observed that modeling an OSPF area as a software risk shared by many IP links was not enough. One particular failure scenario involved only 70% of the OSPF area, which we detected using the error threshold. Upon further investigation, we found that we had to introduce a new risk model, an OSPF area with MPLS enabled in order to correctly capture the particular failure scenario. Of course, the error threshold was helpful in determining that none of the risk groups represented an exact fit with the failure signature. In general, therefore, it is a continuous process of learning what risk models to use based on the various failures we observe.

Risk models are almost always dynamic; the rate at which a given risk model exhibits churn varies depending on the problem. Because of this churn, there could be differences between the risk model and reality that affects localization. For example, if humans are managing the topology information from which the risk model is constructed, there is a strong

likelihood that the risk model is out-of-sync with reality due to human errors. Our IP fault localization system uses error thresholds to deal with these. Of course, exactly identifying the errors automatically is a challenge. On the other hand, automatic generation of the risk model by querying the network, in order to keep the risk model consistent with reality, can be burdensome on the network elements.

Challenges with real-time tools: If the topology is large ($O(n^2)$ paths) and dynamic in nature, it takes a lot of time to calculate the paths on the fly. One solution that is often effective in reducing the computation is sampling. We can down-sample observations to reduce the set of od-pairs for which paths need to be computed. This allows the algorithm to run real-time for operational needs. The other issue for real-time localization is clustering failed probes. If we cluster too closely, the failure signature might not be strong enough for diagnosis. On the other hand, if we cluster together events distant in time, then we would include failure signatures from multiple failures that complicates localization and delays hypothesis generation.

VIII. RELATED WORK

Monitoring and management is a challenging problem for any large network. It is not surprising, then, that a number of research prototypes [5], [8], [14], [20], [25], [26] and commercial products have been developed to diagnose problems in IP and telephone networks. Commercial network fault management systems such as NetFACT [16], OpenView [17], IMPACT [18], EXCperts [25], and SMARTS [31] provide powerful, generic frameworks for handling fault indicators, particularly diverse SNMP-based [4] measurements and rule-based correlation capabilities. However, their reliance on the inherent monitoring capabilities of the network restricts the types of failures these systems can localize.

Network engineers commonly employ the concept of SRLGs to provision disjoint paths in optical networks, as input into many traffic-engineering mechanisms, and in protocols such as Generalized Multi-Protocol Label Switching (GMPLS). Due to their importance, previous work has attempted to automatically infer SRLGs [29] in the optical domain. To the best of our knowledge, however, we are the first to use SRLGs in combination with higher-layer fault notifications to isolate failures in the optical hardware of a network backbone without the need for physical-layer monitoring.

The problem of fault isolation is not limited to networking, of course; similar problems exist in any complex system. Regardless of domain, fault detection systems have taken three basic approaches: rule- or model-based reasoning [2], [10], [17], codebook approaches [31], [37], or machine learning (such as Bayesian and belief networks [7], [36], [33]). The difficulty with probabilistic or machine-learning approaches is that they are not prescriptive: it is not clear what sets of scenarios they can handle besides the specific training data. Rule-based and codebook systems (otherwise known as “expert systems”) are often even more specific, only being able to diagnose events that are explicitly programmed. Model-based approaches are more general, but require detailed

information about the system under test. Dependency-based systems like ours, on the other hand, allow general inference without requiring undue specificity. Indeed, the specific use of dependency graphs for problem diagnosis has been explored before [11], but not in this particular domain.

More recently, Roughan *et al.* proposed a correlation-based approach to detect forwarding anomalies such as BGP-related failures [28]. Their approach was to detect events of potential interest by correlating multiple data sources, while our approach uses these events to diagnose the root cause(s), which may or may not be at the BGP layer. Our problem falls into a general class of inference problems that includes problems in other domains, such as traffic matrix estimation [39], tomography [3], [12], [22], [24], [34], [35], [38], etc. Hence, techniques applied in these domains could potentially be used to solve this problem. However, we have not explored these techniques yet and are part of our future work. Our detection system uses standard mechanisms in route and topology monitoring and packet probing to identify reliability metrics such as packet loss, delay etc. on a per-path basis (e.g., ZING [21], BADABING [32]).

IX. CONCLUSIONS

In this paper, we developed and evaluated a simple yet effective methodology for localization of faults in the network. Our approach based on risk models localizes faults even in the absence of any network-generated alarms, either because they were not available or because the failures were silent in nature, thus aiding network operators in troubleshooting failures even when conventional monitoring fails. While we discuss two specific scenarios in this paper, there may be many other scenarios where our methodology is directly applicable that are yet to be explored. Our extensive evaluation based on both controlled simulation and actual failure data obtained through real-world deployment in a tier-1 ISP, indicates spatial correlation can obtain high localization accuracy and precision in many failure scenarios.

REFERENCES

- [1] AVICI Systems Inc., <http://www.avici.com>.
- [2] S. Brughosi, G. Bruno, *et al.*, “An expert system for real-time fault diagnosis of the Italian telecommunications network,” in *3rd Symp. on Integrated Network Management*, 1993, pp. 617–628.
- [3] J. Cao, D. Davis, S. V. Wiel, and J. Yu, “Time-varying network tomography,” *J. Amer. Statist. Assoc.*, vol. 95, no. 452, pp. 1063–1075, 2000.
- [4] J. Case, M. Fedor, M. Schoffstall, and J. Davin, “A simple network management protocol (SNMP),” IETF, RFC 1157, May 1990.
- [5] C. S. Chao, D. L. Yang, and A. C. Liu, “An automated fault diagnosis system using hierarchical reasoning and alarm correlation,” in *Journal of Network and Systems Management*, vol. 9, no. 2, 2001, pp. 183–202.
- [6] S. Chaudhuri, G. Hjalmysson, and J. Yates, “Control of lightpaths in an optical network,” Jan. 2000, <http://www.research.att.com/areas/opticalnetworking/IPoverWDMpublications.html>.
- [7] M. Chen, A. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer, “A statistical learning approach to failure diagnosis,” in *Int. Conf. on Autonomic Computing*, New York, NY, May 2004.
- [8] R. H. Deng, A. A. Lazar, and W. Wang, “A probabilistic approach to fault diagnosis in linear lightwave networks,” in *Integrated Network Management III*, Apr. 1993, pp. 697–708.
- [9] L. Fang, A. Atlas, F. Chiussi, K. Kompella, and G. Swallow, “LDP failure detection and recovery,” *IEEE Communications*, vol. 42, no. 10, pp. 117–123, Oct. 2004.

- [10] G. Forman, M. Jain, M. Mansouri-Samani, J. Martinka, and A. C. Snoeren, "Automated whole-system diagnosis of distributed services using model-based reasoning," in *9th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, Oct. 1998.
- [11] B. Gruschke, "Integrated event management: Event correlation using dependency graphs," in *9th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, Oct. 1998.
- [12] A. Gunnar, M. Johansson, and T. Telkamp, "Traffic matrix estimation on a large ip backbone: A comparison on real data," in *ACM Internet Measurement Conference*, October 2004.
- [13] S. Hanks, T. Li, D. Farinacci, and P. Traina, "Generic routing encapsulation over ipv4 networks," IETF, RFC 1702, Oct. 1994.
- [14] P. Hong and P. Sen, "Incorporating non-deterministic reasoning in managing heterogeneous network," in *Integrated Network Management II*, Apr. 1991, pp. 481–492.
- [15] C. Hopps, "Analysis of an equal-cost multi-path algorithm," IETF, RFC 2992, Nov. 2000.
- [16] K. Houck, S. Calo, and A. Finkel, "Towards a practical alarm correlation system," in *4th IEEE/IFIP Symp. on Int. Net. Mgmt.*, 1995.
- [17] HP Technologies, Open View, <http://www.openview.hp.com>.
- [18] G. Jakobson and M. D. Weissman, "Alarm correlation," *IEEE Network*, vol. 7, no. 6, pp. 52–59, Nov. 1993.
- [19] R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "IP fault localization via risk modeling," in *Proc. Networked Systems Design and Implementation*, May 2005.
- [20] G. Liu, A. K. Mok, and E. J. Yang, "Composite events for network event correlation," in *Integrated Network Management VI*, Boston, MA, May 1999.
- [21] J. Mahdavi, V. Paxson, A. Adams, and M. Mathis, "Creating a scalable architecture for Internet measurement," in *INET'98*, July 1998.
- [22] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," in *ACM SIGCOMM*, Pittsburg, USA, August 2002.
- [23] J. Moy, "Ospf version 2," IETF, RFC 2328, Apr. 1998.
- [24] A. Nucci, R. Cruz, N. Taft, and C. Diot, "Design of IGP link weights for estimation of traffic matrices," in *IEEE Infocom*, Hong Kong, March 2004.
- [25] Y. A. Nygate, "Event correlation using rule and object based techniques," in *Integrated Network Management*, pp. 278–289.
- [26] P.Wu, R. Bhatnagar, L. Epshtein, M. Bhandaru, and Z. Shi, "Alarm correlation engine (ACE)," in *Network Operation and Management Symp.*, 1998, pp. 733–742.
- [27] R. Ramaswami and K. Sivarajan, *Optical Networks : A Practical Perspective*. Academic Press/Morgan Kaufmann, Feb. 1998.
- [28] M. Roughan, T. Griffin, Z. M. Mao, A. Greenberg, and B. Freeman, "Combining routing and traffic data for detection of ip forwarding anomalies," in *ACM SIGCOMM NeTs*, Aug. 2004.
- [29] P. Sebos, J. Yates, D. Rubenstein, and A. Greenberg, "Effectiveness of shared risk link group auto-discovery in optical networks," in *Optical Fiber Comm. Conf.*, Mar. 2002.
- [30] A. Shaikh and A. Greenberg, "OSPF monitoring: Architecture, design and deployment experience," in *NSDI*, Mar. 2004.
- [31] SMARTS Inc, "<http://www.smarts.com>."
- [32] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Improving accuracy in end-to-end packet loss measurement," in *ACM SIGCOMM*, Aug. 2005.
- [33] M. Steinder and A. Sethi, "End-to-end service failure diagnosis using belief networks," in *Network Operation and Management Symp.*, Florence, Italy, Apr. 2002.
- [34] C. Tebaldi and M. West, "Bayesian inference on network traffic using link count data," *J. American Statistical Assoc.*, vol. 93, no. 442, pp. 557–576, 1998.
- [35] Y. Vardi, "Network tomography: estimating source-destination traffic intensities from link data," *J. American Statistical Assoc.*, vol. 91, pp. 365–377, 1996.
- [36] H. Wietgreffe, K. Tochs, *et al.*, "Using neural networks for alarm correlation in cellular phone networks," in *Proc. International Workshop on Applications of Neural Networks in Telecommunications*, 1997.
- [37] S. A. Yemini, S. Klinger, E. Mozes, Y. Yemini, and D. Ohsie, "High speed and robust event correlation," in *IEEE Communications*, vol. 34, no. 5, 1996, pp. 82–90.
- [38] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," in *ACM SIGMETRICS 2003*, June 2003.
- [39] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An information-theoretic approach to traffic matrix estimation," in *ACM SIGCOMM*, August 2003.

PLACE
PHOTO
HERE

Ramana Rao Kompella is a fourth year Ph.D student at University of California, San Diego. His main research interests include fault-management in IP networks, scalable algorithms and architectures for high speed switches and routers, and scheduling in wireless networks. During his Ph.D, he interned at AT&T Labs – Research for two summers (2004 and 2005) to devise patent pending mechanisms to localize IP and MPLS layer failures in the network using novel spatial correlation approaches. Prior to joining UCSD, he worked as an ASIC Design Engineer at Chelsio Communications and as Design Engineer at SwitchOn Networks (acquired by PMC Sierra Inc in 2001) where he helped pioneer algorithms for packet classification and routing lookups. Together with several colleagues, he has six patents (two awarded and four pending), more than 15 research publications in the areas of packet classification, fault localization and wireless networks. He received his B.Tech degree from IIT Bombay in 1999 and M.S degree from Stanford University in 2001.

PLACE
PHOTO
HERE

Jennifer Yates is a Technical Specialist in IP Network Management and Performance group at AT&T Labs - Research. She has worked on issues relating to IP control of optical networks, IP and optical integration, and IP network management and performance. She is extremely active both within the Research community and within AT&T - the author of numerous papers, standards contributions, patent applications and an editor of Transactions on Networking. She received the PhD degree from the University of Melbourne in 1998.

PLACE
PHOTO
HERE

Albert Greenberg is an AT&T Fellow and the Director of Network Measurement and Engineering Research at AT&T Labs-Research. Alberts recent research includes: novel methods for packet and flow measurement and analysis, traffic matrix inference, anomaly detection, configuration management, IP/MPLS control plane monitoring, MPLS/GMPLS control and management, IP traffic and network engineering, IP fault management and troubleshooting, new route control architectures, database and systems applications, and network security. Alberts education includes a BA from Dartmouth College in Mathematics (1978), and MS and a PhD in Computer Science from the University of Washington (1981, 1983). In 2004, Albert was awarded AT&Ts Science and Technology Medal for the invention of the tomography method, which computes traffic matrices from link load information. Albert hails from New Orleans, where he was born in 1956, and where he returns as often as possible.

PLACE
PHOTO
HERE

Alex C. Snoeren is an Assistant Professor in the Computer Science and Engineering Department at the University of California, San Diego, where he is a member of the Systems and Networking Research Group. His research interests include operating systems, distributed computing, and mobile and wide-area networking. Professor Snoeren received a Ph.D. in Computer Science from the Massachusetts Institute of Technology (2003) and an M.S. in Computer Science (1997) and Bachelors of Science in Computer Science (1996) and Applied Mathematics (1997) from the Georgia Institute of Technology. He is a recipient of the National Science Foundation CAREER Award (2004), the MIT EECS George M. Sprowls Doctoral Dissertation Award (Honorable Mention, 2003), and the Best Student Paper award at the ACM SIGCOMM conference (2001).