Towards Root Cause Analysis of Internet Routing Dynamics

Matthew Caesar, L. Subramanian, Randy H. Katz

{mccaesar,lakme,randy}@cs.berkeley.edu

Abstract

The lack of a good understanding of the dynamics of interdomain routing has made efforts to address BGP's shortcomings a black art. To gain more insight into these dynamics, we need to answer two questions: *What is the cause of a routing change? Where does a routing change originate?* This paper proposes the design of a *BGP health inferencing system* that answers these questions by observing routing updates from multiple vantage points and inferring the type and location of an event that triggers a routing change. To build such a system, we solve two basic problems: (a) classify route updates into groups of correlated routing changes where all route updates in a group are triggered by the same event, (b) given the set of routing changes for an event, determine the location and the cause of the event.

By analyzing route updates from Routeviews and RIPE for over 18 months, we found that our approach can pinpoint the location where an update is triggered to a single inter-AS link in over 70% of observed updates. We found that the majority of updates are caused by a relatively small number of unstable links. In addition, 25% of prefixes are persistently unstable, causing 20% of all updates observed. Routes through the Internet core usually reconverge quickly after events, while an event taking place at the network edge is 9 times more likely to cause a long-term route change. We validated our approach by showing it can detect a variety of well-known events, namely: (a) session resets recorded in the NANOG mailing list; (b) routing problems within ISPs; (c) location of BGP Beacons. In addition, our inference methodology is able to detect several routing problems not publicly known. In summary, we believe that our health inference system is a first step towards forming a better understanding of inter-domain routing dynamics.

1. Introduction

"Is root cause analysis hopeless? What class of assertions can be made based on BGP updates alone?" – Timothy Griffin [6].

Internet routing is plagued with several problems today, including chronic instability and convergence problems and misconfigurations in routers [10, 2, 14]. Many of these problems arise due to the inherent complexity in the Border Gateway Protocol (BGP), the de facto interdomain routing protocol. BGP has evolved into a complex protocol with several policy knobs and features such that its dynamics have become hard to comprehend.

Without a clear understanding of these dynamics, efforts to address BGP's shortcomings have become a black art. First, we do not yet completely understand the impact of simple configuration changes on routing dynamics. Hence, we lack clear guidelines for configuring, diagnosing and debugging BGP [14, 5]. Second, while several modifications to BGP have been proposed to alleviate specific problems, these may introduce a newer set of problems currently unknown to the research community. For example, the route flap dampening mechanism was introduced to improve the stability of BGP but was later found to introduce routing convergence problems [16]. Finally, only recently have many problems relating to route oscillations, convergence and inter/intra domain protocol interactions been brought to light [7].

We believe that a first step towards improving our understanding is to develop a systematic methodology for analyzing routing changes and inferring *why* they happen and *where* they originate. Answers to these questions can provide useful insights into the sources of anomalous routing events and instabilities.

In this paper, we describe the design of a BGP health inferencing system for determining the root cause of routing changes. The health inferencing system collects and correlates route updates from multiple vantage points to determine the routing events that trigger each route update. We envision deploying our inference algorithms in data collection centers like Routeviews [25] and RIPE [26], which receive streams of route updates from multiple vantage points. More generally, we can use a BGP health monitor to continuously infer the state of the network. Such inferences may then be used: (a) *offline* for network performance monitoring and troubleshooting; or (b) *online* to improve path selection and damping of instability. In this paper we focus on developing accurate inference techniques, and not on how the inferences are used.

The key challenge to performing root cause analysis is to develop inference algorithms that can determine the cause and origin of routing events with a reasonable degree of accuracy. Inferences that can be made from a single vantage point may be very limited. However, route updates from multiple views can be used to improve the accuracy of our inferences. Additionally, our mechanisms are purely based on passive observations but could potentially be coupled with active probing techniques to yield more accurate results.

Our inference methodology uses three basic ideas to improve the accuracy of our results:

1. *Turbulent vs Quiescent periods:* The rate at which prefixes get updated signifies the type(s) of event that caused the stream of updates. In a Turbulent period, one or a few major routing events cause several routes to simultaneously get updated. We assume that many observations in such a period are correlated (i.e. arise from the same routing event). In a Quiescent period, when very few prefixes are updated, it is harder to determine which updates are caused by the same routing event. In this case, we analyze updates to each prefix in isolation.
2. *Multiple Vantage Points:* Observing the same event from several vantage points allows us to acquire additional information about the event. By comparing similarities and differences in observations across the views, and by measuring the magnitudes of the event at each view, we can distinguish the signature of the event from effects introduced by intermediate routers along the path.
3. *Matching causes with observations:* For every potential cause of a routing event, there exists different patterns of route updates that can be observed at a vantage point. Based on the pattern of observations, we classify the causes into *equivalence classes* where each class contains different causes that might trigger the same pattern of updates. While Griffin *et al.* [6] have shown that matching causes with observations is a hard problem, we find that defining simple patterns of updates (e.g. presence of route withdrawals) can help in improving the accuracy of the inferences.

Most ISPs do not wish to reveal the types or frequency of events taking place in their networks, making validation of our approach difficult. However, there are several well-known major events that are

public knowledge, such as the spread of Internet worms, or routing problems suffered by major ISPs [27]. In addition, we know the location where certain classes of updates are caused, for example updates pertaining to prefixes originated by the AS containing the vantage point, or updates generated by BGP Beacons [15]. We considered a large number of such updates, and found that inference was performed correctly in every case. Although we aren't able to directly validate all of our inferences using this approach, we are able to verify the correctness of a base set of rules that we used to acquire our results.

While the generic problem of determining the cause and location of an event is hard [6], we find that in practice, we are able to pinpoint the location where the update was triggered to a single inter-AS link for over 70% of the updates. During Turbulent periods the precision of our location inferences is within 2 AS's and the median precision during Quiescent periods is $3-4$ AS's. Additionally, we output a list of potential causes that might have caused an event, but may not always be able to identify the specific cause.

To demonstrate the utility of such a system, we apply our inference methodology to updates collected from Routeviews and RIPE over a period of 18 months. We make several observations from our analysis: (a) Our system can detect major routing anomalies, many of which were previously unknown. (b) We detected nearly 1,400 resets per month, and found certain inter-AS links to be perennially unstable. (c) Roughly 25% of prefixes continuously flap at least every 30 minutes, and these account for a large fraction (20%) of routing updates. (d) Routing events in the Internet core usually trigger short-term flaps, but an event taking place at the network edge is 9 times more likely to cause a long-term route change.

Section 2 describes the root cause inference problem, why it is challenging, and how our approach overcomes these challenges. Section 3 describes how we determine which observations are correlated. Sections 4 and 5 describe the inference algorithms we use during Turbulent and Quiescent periods of time. Section 6 describes our methodology and validation of our approach. We describe our results in Section 7, related work in Section 8, and conclude in Section 9.

2. Root Cause Inference Problem

Terminology: A *routing event* is an activity taking place at some location in the network that generates one or more route updates. A group of updates is *correlated* if they were all caused by the same event. Routing events can have different *causes*, e.g., failures, policy changes, and link repairs. Routing events have two properties: the cause of the event, and the location of the event. We refer to events affecting many prefixes as *major events*, and those affecting few prefixes as *minor events*. Our algorithm uses observations made at various routers called *views* or *vantage points*. Owners of the views volunteer to make their updates public by sending them to a *monitor* such as Routeviews [25].

The root cause inference problem can be stated as follows: *Given route updates observed at multiple vantage points, determine the suspect set=*$\{(C_1, L_1), (C_2, L_2) \dots\}$ *where* (C_i, L_i) *represent a potential cause and location that could have triggered a route update.* Corresponding to each cause C_i, L_i represents the list of potential locations that might have triggered a given update. Given that BGP route updates are only in the granularity of AS's, our location inferences are restricted to the level of AS's as opposed to router-level inferences.

We face the following set of challenges towards addressing the root cause inference problem:

1. *Overlapping observations:* A single routing event occurring at a particular location can trigger propagation of updates affecting a large number of other routers [6]. In a system as large as the Internet, many simultaneous events are taking place, introducing noise into the observations.
2. *Correlated vs simultaneous observations:* A single event can trigger multiple correlated observations, or multiple events can simultaneously occur (given the size of the Internet). Determining which observations are correlated and which are independent is hard, but necessary to determine the type of event.
3. *BGP policies:* BGP allows AS's to set their own policies for choosing the "best" route to a destination. For example, the LocalPref attribute can be used to set the degree of preference of a route, when comparing it with other routes to the same destination. The thumb-rules used for setting policies are not standardized and not disclosed. Also, humans are involved in setting policies introducing the possibility of mistakes. Additionally, AS's can control flap damping parameters and thereby introduce variability in the delay to receive up-to-date state information.
4. *Multiple peering links:* Many pairs of AS's have multiple peering links between them. Even if two AS paths intersect in the AS topology, they may not intersect in the underlying topology. This implies that if two views share a sub-path in the AS-topology, it is possible that an event in the intersection region of the AS paths is noticed by one view and not by the other. Providers may set the Multiexit Discriminator (MED) in updates as a hint to peers about the preferred peering link to use, when multiple peering links are available.
5. *I-BGP/E-BGP interactions:* BGP is composed of two protocols, I-BGP and E-BGP. I-BGP operates internally within an AS while E-BGP operates across AS's. Recent results have shown that interactions between I-BGP and interior routing can cause persistent E-BGP route oscillations [7]. By only observing E-BGP advertisements, we do not get any visibility into intra-AS route dynamics. Hence it is hard to determine whether a problem exists within an AS or on links between peered AS's.

2.1 Assumptions

We make the following assumptions while designing our inference mechanism:

1. Multiple independent minor events affecting a single prefix rarely happen close together in time. In particular, bursts of updates to a prefix by two independent minor events are typically separated by long silent periods. We verify this assumption by analyzing BGP update traffic over several months.
2. An AS that triggers any route change should be embedded in one or more route updates in the burst (either in the previous path, one of the intermediate paths, or final path)[1]. If an AS is wrongly specified or if the AS path is truncated due to implementation bugs, our inference mechanism will not work.
3. The effects of an event will propagate route updates to the vantage points within the order of a few minutes, unless flap damping is triggered.

One fundamental limitation of any inference algorithm is detecting minor events in the presence of major events. The few updates caused by a minor event get subsumed as noise when a major event triggers many route updates at the same time. Inferring minor events under these circumstances is a hard problem.

2.2 Our Inference Methodology

Figure 1 presents an overview of the flow of our inference algorithm. First, we use a Detection algorithm (Section 3) that determines whether a vantage point is in a Quiescent or Turbulent period based on number of updates it receives. It works by monitoring the number of prefixes that were recently updated. If this number exceeds a threshold, the algorithm decides the view is in a Turbulent period, otherwise it decides the view is in a Quiescent period. The assumptions we can make, and the mechanisms by which we perform inference differs based on which of these periods the views are in. Hence we developed two algorithms corresponding to each of these periods. *Tur-*

[1] We are not aware of any routing events where the AS that triggers an event is not present in any of the paths.

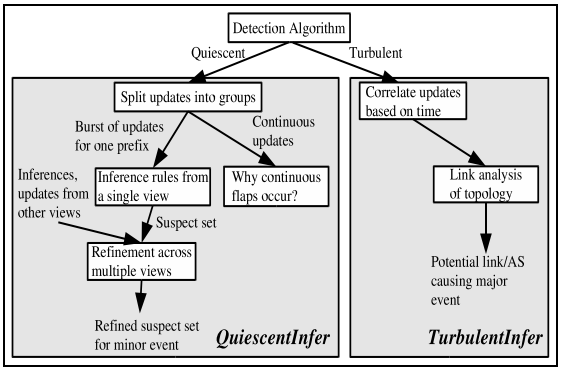

Figure 1: Algorithm flow.

bulentInfer (Section 4) performs inference during Turbulent periods. During Turbulent periods, we can typically perform more exact inference due to the large number of updates received, and hence it attempt to infer a single link or a single AS as the potential location of the major event. It does this by first observing, for each link, the number of prefixes that used the link that were recently updated. If this number is high for many links coming out of some AS, the place where they converge is the most likely location of the event. *QuiescentInfer* (Section 5) attempts to find the suspect set during Quiescent periods. It first correlates updates to a single prefix based on time, then from each view it determines a suspect set based on a set of inference rules. It then combines inferences from multiple vantage points to refine the suspect set.

3. Distinguishing correlated and independent observations

To address the root cause inference problem, we first need to identify groups of route updates in different vantage points which are triggered by the same routing event. We define each such group as a set of *correlated* observations. Determining which updates are correlated can help narrow down the suspect set, as it allows us to gain more information about the event. However, inappropriately assuming two observations are correlated can cause false negatives, which are particularly undesirable. The problem we focus on in this section is, given a set of updates, how does one partition them into groups such that the groups are disjoint, and each element in a group is caused by the same event.

We correlate updates across three dimensions: *time*, *views* and *prefixes*. The *time* dimension observes the set of updates that occur close together in time. The *views* dimension takes into account how the event affects different views. The *prefix* dimension correlates advertisements across multiple prefixes which are affected by the same event.

Correlation across Time: By observing the set of updates to a single prefix at a single vantage point, we classify prefixes as either *stable* or *continuously flapping*. A stable prefix receives updates in bursts and each burst is separated from the others by a long period of silence (*i.e.* a period of time with no updates) of size δ_{sil}. Based on an analysis of Routeviews and RIPE data, we observed that the silent periods for 99% of stable prefixes last longer than 1 hour. Hence $\delta_{sil} = 3700$ seconds is sufficient to separate bursts on stable prefixes. This is expected, since the state of a prefix can take up to one hour to converge after an event [16, 13]. For stable prefixes, we assume that each burst of updates at a single view is triggered by the same event. For a continuously flapping prefix, we observe a steady stream of route updates lasting for long periods of time. In practice, we find that only ∼75% of the prefixes in a routing table are stable. In this section, we will restrict our discussion to stable prefixes and discuss continuously flapping prefixes in Section 7.

Correlation across Views: If multiple views observe a burst of updates for the same prefix at roughly the same time, then we assume that these bursts of events are correlated. This assumption is motivated by the fact that the probability of *two minor events* occurring simultaneously and creating a burst of updates to the same prefix is small given that minor events are typically separated by long periods of silence. Additionally, for a stable prefix, the effect of a minor event is often visible across most of the views. We restrict this observation to only minor events since a major event affecting many prefixes typically subsumes a minor event. In the occurrence of a major event, we may not be able to determine the cause of minor events.

Correlation across Prefixes: The main idea behind correlating updates across prefixes involves distinguishing between two major classes of observations: *Quiescent* and *Turbulent*. In a Quiescent period, a few minor events occur causing only few prefixes to be updated. In such a period, we do not correlate updates across prefixes.[2] In a Turbulent period, a single view observes a large number of prefixes to be updated within a short period of time. Turbulent periods are typically caused by one or a few major events, such as a BGP session reset. Although some of the updates may be due to unrelated, independent events, the probability that a large number of independent events simultaneously occurred to cause this spike in activity is very small.

We now describe our algorithm to distinguish between Quiescent and Turbulent periods.

3.1 Turbulent vs Quiescent Periods

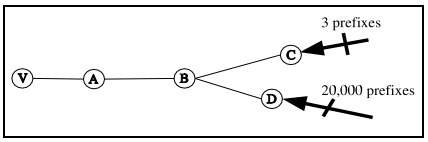

Figure 2: Example: Distinguishing Turbulent and Quiescent periods.

We motivate our detection algorithm using a simple example. Consider the topology shown in Figure 2 where V is a vantage point. V has routes for 20,000 prefixes traversing AS D, and has routes for 3 prefixes traversing AS C (AS C and D are intermediary AS's rather than origin AS's for the prefixes). We consider two example scenarios. First, suppose we observe updates at V to a lot of prefixes that traverse link (B, D). Then it is likely that some major event occurred in this region, generating a large number of correlated updates. Next, assume that we see updates to all three prefixes that use AS C. Although it is possible that these updates were all caused by the same event, they could have been caused by multiple independent events. and so we cannot say with certainty whether they are correlated. In general, if we observe a large number of prefixes with AS paths traversing some link to be updated within a short period of time, it is likely that a majority of those updates are correlated.

The Detection algorithm exploits this observation by maintaining a threshold α associated with each link in the underlying AS graph G. If the number $N(e)$ of distinct prefixes updated that use the link exceeds α, we assume a major event occurred and trigger TurbulentInfer. Otherwise, we run QuiescentInfer on the updates.

This approach for distinguishing between Turbulent and Quiescent periods is motivated by the following observation: the distribution of $N(e)$ is multi-modal in nature with a large gap between the first and second modes, as shown in Figure 3. The figure shows the cumulative distribution of $N(e)$ measured on several links over an 18 month period. The first mode appears to be due to natural background activity (Quiescent events), the second mode appears to be caused by reset

[2]In practice, a single minor event can create correlated updates across different prefixes. However, our inference methodology treats them as independent events. While this reduces the precision of our inference, we do not sacrifice correctness.

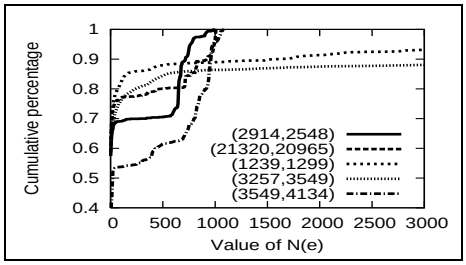

Figure 3: Frequency of occurrences various values of $N(e)$ that were observed for several inter-AS links. We use the gap between the first and second modes to distinguish Turbulent from Quiescent periods.

activity on this particular link (Turbulent events), and the other modes appear to correspond to resets taking place on other links (Turbulent events). All links that observed resets have a similarly large gap between the first and second modes, though the size and position of the second mode can vary with the frequency and magnitude of resets. The fact that there is such a large gap shows that we can use $N(e)$ to distinguish between Turbulent and Quiescent periods of time, by observing whether $N(e)$ is in the first mode or not. In general, we find that $\alpha = 400$ separates the first and second modes in the majority of links that observed resets. It may be possible to tune α by dynamically measuring this distribution, but we did not do this.

In practice, we estimate $N(e)$ as the number of updates observed on an inter-AS link e within a period of 15 minutes, the time required for BGP to converge in the presence of routing instabilities [10]. To ensure that the entire effects of a Turbulent event are recorded, we maintain a sliding window of size 15 minutes to maintain history about $N(e)$ for each link. In particular, we estimate $N(e)$ as the peak value measured over different 15 minutes intervals in the sliding window.

4. Inference during Turbulent periods

In this section, we present an algorithm for inferring the suspect set of AS's that might have triggered a Turbulent event. The main idea behind our approach involves searching for AS's common to a large fraction of the received updates. These AS's are suspect for having caused the Turbulent event.

Turbulent events can either be caused by failures on inter-AS links (E-BGP session failures) or failures inside an AS (I-BGP session failures). We consider two scenarios in our inference algorithm: (a) *single session* failure; (b) *simultaneous session* failures. While single session failures are the common case, simultaneous failures are also known to occur in practice. Two such cases are: (a) Internet worms like SQL Slammer and Code Red [27] cause simultaneous session resets due to increased levels of congestion at many locations in the network, and (b) one session reset can induce another session reset due to congestion arising from the shifted traffic.

We start by assuming a single session failure occurred, and show how to determine its location. We then eliminate this assumption by showing how to handle simultaneous session failures. Our approach can reduce the size of the suspect set to 2 AS's under the assumption that only one session failure happened, using data from only a single view. Since BGP updates are only at the granularity of AS's, it is often impossible to distinguish between internal AS failures and link failures, in which case two is the smallest number possible to achieve.

4.1 Single session failure

The approach works based on the simple observation that, if an inter-AS link is reset, a large number of paths traversing that link will be updated in a short period of time. An example is shown in Figure 4. Suppose at view V we observe updates to 1000 prefixes traversing link (A, C), and suppose very few other prefixes are updated. Then, it is likely that the event took place at (A, C) or downstream of C. On the other hand, suppose we instead observed updates to 4000 prefixes that traverse (A, B), and of these prefixes, 2000 traverse (B, E) and 2000 traverse (B, D). Then, the major event likely took place on the link (A, B). Suppose instead we observed roughly 2000 prefixes that traverse (A, B), and 2000 prefixes that traverse (B, E) become updated. Then, the event probably took place on link (B, E) or downstream of E. In general, if a large number of prefixes with AS paths containing some link are updated in a short period of time, a session failure is likely to have occurred on that link or some link downstream of it.

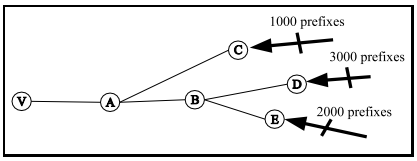

Figure 4: Example: Effects of a session failure on a single view.

Our approach to infer the location of a single session failure consists of three steps. First, for a single view, we search for large bursts of updates that affect many prefixes within a short period of time. This is done by the Detection algorithm described in Section 3.1. Next, we search for links and AS's that are shared across a large number of prefixes that are updated during the burst, and create a graph where they are marked as suspect. Finally, we traverse the resulting graph, identifying which of these links or AS's were most likely to have caused the observations.

4.1.1 Marking links

We use the graph G constructed by the Detection algorithm to identify links and AS's that are shared across a large number of updated prefixes. Recall that G consists of nodes corresponding to AS's, and links corresponding to peering relationships between AS's. We define two weights for each link $e \in G$: $T(e)$ is the total number of prefixes containing e in their AS path, and $N(e)$ is the total number of prefixes that were recently updated, and contained e in their AS path before they were updated. $N(e)$ corresponds to the likelihood that a major event occurred on e, and $T(e)$ corresponds to the degree of visibility the algorithm has into events occurring on link e.

We then use these weights to mark links likely to have caused the event. First, we eliminate all links with $T(e)$ less than α. We do this because for these links $N(e)$ would always be less than α, hence we would not be able to observe events on these links (we may be able to observe Turbulent events on these links from other views). We then mark all links with $N(e)$ greater than a threshold α as **M** (for "many"), and all other links as **F** (for "few"). Edges marked as M are suspect for having undergone a Turbulent event. To set α, we use the procedure discussed in Section 3.1.

4.1.2 Traversal

We then apply a set of rules to the resulting graph to determine the most likely location where the event could have occurred. That is, we start at the node representing the view, and use the rules to traverse the graph. Eventually we terminate at the link or AS most likely to have caused the event. The rules, as shown in Figure 5, are:

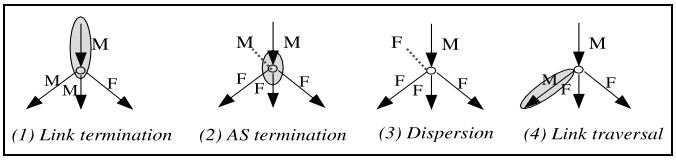

Figure 5: Rules to determine location of failure from observations at a single view.

1. *Link termination:* If there are several links marked M coming out of this node, either a single event occurred on the incoming link, or simultaneous events occurred on the outgoing links marked M. Since simultaneous events are rare, we halt and return the link between this node and its parent.
2. *AS termination:* If we arrive at a node marked M with an incoming link marked M, and no outgoing links marked as M, the event was not an E-BGP session failure. Rather, it is some event occurring inside an AS affecting many of its prefixes. Hence, the algorithm returns this AS as the location.
3. *Dispersion:* If we arrive at a node marked F with all outgoing links marked F, then the large $N(e)$ we saw on the incoming link was the aggregation of many individual Quiescent events, and was unlikely to be a failure. Hence, we assume that no major event occurred.
4. *Link traversal:* If there a single link marked M coming out of the node, then the failure could have occurred at the link coming into this node or downstream. However, if the node has several outgoing links marked F, it is unlikely that the failure occurred on the link coming into this node, since an event on that link would have most likely affected prefixes traversing several outgoing links. In this case, we traverse to the child, and try to apply these rules from that position.

There are two key issues that limit the visibility of events, which in turn limits the effectiveness of our approach. First, if there are several hops between the event and the view, then the effects of the event could have been reduced due to WRATE [3] [16] or flap damping [4]. Also, if the inter-AS link is actually comprised of several peering sessions, an event affecting one peering session will affect just a subset of the prefixes with paths traversing that link. Hence our results represent a lower bound on the number of such events.

4.2 Simultaneous session failures

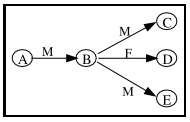

Figure 6: Detecting simultaneously occurring events.

The rule described in Figure 5a does not work if there are simultaneously occurring events, since observing several links marked M emerging from a node could be caused by multiple distinct failures. However, there are two techniques we can use to detect when simultaneous events are occurring. First, we can often detect when simultaneous events are occurring by associating a timer with each link, corresponding to when the first signs of the Turbulent event were detected on that link. If the times corresponding to two links marked M differ by a few minutes, then the two links underwent two overlapping disjoint session resets. For example in Figure 6, if we observe a burst of updates to prefixes traversing link (B, C) followed by a burst of updates traversing (B, E) that starts a few minutes later, then it is highly likely that two separate events occurred, one on link (B, C) and another on link (B, E).

We can also detect simultaneous events by using the state messages BGP peers use to establish and maintain the peering session. If we can observe these state messages, then we can tell with certainty when the link undergoes a session reset. Although state messages do not traverse more than one hop, we can observe state messages between the views and the monitor that logs the routing updates (e.g. RIPE or Routeviews). For example in Figure 6, suppose node A is a monitor, and B is a view. If we do not observe a reset on the link (A, B), but (B, C) and (B, E) are marked M, then it is highly likely that two separate events occurred on links (B, C) and (B, E) or downstream of these links. This lets us detect simultaneous events near the view.

We observed that simultaneous events are rare, but tend to occur simultaneously at many locations. Almost all simultaneous events were observed during periods when Internet worms were propagating. Although we cannot distinguish simultaneous events using observations from a single view, sometimes we can use multiple views to distinguish the two events. If views are properly placed, some views will only observe one of the two events, and we can hold out these observations from views that observed both to distinguish the other event.

[3]Withdrawal rate-limiting (WRATE) is a technique to rate-limit withdrawals of a prefix.

[4]Flap damping is a mechanism in routers that withdraws persistently unstable routes, thereby reducing routing instability. A prefix will be re-advertised if it remains stable for a long period of time [16].

5. Inference during Quiescent periods

In this section, we present an algorithm for inferring the *suspect set* during a Quiescent period. In Section 2, we specified three dimensions for inferring the suspect set: *time*, *views* and *prefixes*. Unlike the Turbulent period, we cannot reliably assume that updates across prefixes are correlated. Hence the inference algorithm for Quiescent periods uses only two of these dimensions: time and views.

Our inference algorithm for a Quiescent period consists of two key steps: (a) Inferences from a single view; (b) Refinement across views. From a single view, we classify updates for a single prefix into different categories based on the patterns of observations. For each pattern of observations, we apply a set of inference rules to associate suspect AS's with a *potential cause-list* as to why that AS might have triggered the set of updates. An AS that triggers an event should be associated with the same cause irrespective of the view we observe it from. We use this principle to narrow down the suspect set by performing an *intersection* of the suspect sets across views. Additionally, we apply two rules to refine the suspect set based on which views observe an event and which do not.

Given the complexity of BGP, it is hard to individually enumerate all potential causes of an event. In our methodology, we handle this case by classifying these causes into non-overlapping *equivalence classes* where each class of causes triggers a similar pattern of observations at a vantage point. From each view, it may be hard to distinguish between different causes within the same equivalence class.

5.1 Inference from a single view

Table 1: Enumeration of prefix state transitions (bursts).

Name	***Freq.***	***Description***
REROUTE	11.7%	Path change P_1 to P_2 where $P_1 \neq P_2$
PREFIXUP	2.1%	Transition from Withdrawn to Advertised state
PREFIXDOWN	3.43%	Transition from Advertised to Withdrawn state
FLAPUP	2.17%	PREFIXUP followed by PREFIXDOWN
PATHFLAP	73.39%	REROUTE, then returns to original state
ASMIGR	0.69%	First and last path have different last-hop AS
DUPCHANGE	6.51%	Change in # of prepended copies of some AS
LOOP	0.01%	A loop is inserted or deleted in the AS path

The basic step in our inference methodology is to identify different patterns of route updates that can be observed for a single prefix. Table 1 enumerates the different observable patterns based on how the state of a route changes during a burst of updates and their corresponding frequency of occurrence. Using these patterns, there are three main ways to distinguish the cause of a routing event:

1. A route can **improve** by becoming more preferred, or it can **worsen** by becoming less preferred with respect to other paths according to the BGP path selection algorithm [18]. For example, observing a REROUTE from path P_1 to path P_2 tells us that the event caused path P_2 to improve and/or path P_1 to worsen.

2. A route can undergo a **hard** event where some router along the previous state's path issued a withdrawal of a route or advertised a previously withdrawn route, or a **soft** event where a router changes preference between existing routes that are not withdrawn or newly advertised. For example, PREFIXDOWN and PREFIXUP observations are due to hard events because they contain a withdrawal or a newly advertised route, respectively. FLAPs and REROUTEs could occur either due to hard events or soft events, since although a withdrawal is not received by the view, some intermediate router could have generated a withdrawal triggering the observation.
3. BGP supports a mechanism to allow a downstream AS to tell an upstream AS how to to change routes. This is done by changing the **community** attribute in updates [9]. This attribute consists of a variable length string that can be used to convey additional information, and routers can customize reaction to this information according to policy. For example, a community attribute may indicate that the AS receiving the update should not advertise the route to its peers. Changing the attribute could then cause the AS to advertise the route, causing peer's routes to start traversing this AS. This type of change is unique in that it can trigger a route change several hops away from the AS that changed the attribute.

Table 2: Equivalence classes of events.

Class	*Description*	*Example causes*
Hard+Worsen (H_W)	Event worsens path properties, involves a Withdraw	link/router failure, hold down triggered, filtering rule added
Soft+Worsen (S_W)	Event worsens path properties, doesn't involve a Withdraw	MED increase, LocalPref increase, AS prepending increase, Community change
Hard+Improve (H_I)	Event improves path properties, involves a Withdraw	link/router repair, hold down expires, filtering rule deleted
Soft+Improve (S_I)	Event improves path properties, doesn't involve a Withdraw	MED decrease, LocalPref decrease, AS prepending decrease, Community change
Community (N)	Community attribute changed	Community change

If we can determine whether a route underwent a hard, soft, improve, or worsen event, we can gain information about the types of events that could have taken place. In particular, we can determine the types of events that could have occurred at AS's and links contained in the routing updates. We hence define the *equivalence classes* shown in Table 2 based on combinations of these event types. For example, S_I contains events that are soft (do not affect reachability) and also improve the path with respect to BGP's path selection algorithm. Events in an equivalence class are difficult to distinguish from other events in the same class, but are typically easy to distinguish from events in other classes. For completeness, we add two additional categories: a *multiple* category for elements that appear to be undergoing multiple events close together in time, and an *unknown* as a final catch-all category if we have no information about an event.

To simplify notation, we define I to be elements that could have undergone some event in $H_I + S_I$, W to similarly correspond to $H_W + S_W$, M to contain elements that could have undergone multiple events, and U to contain elements for which we have no information about the type of event it underwent.

5.1.1 Rules for singly-peered AS's

The idea behind our approach involves developing a set of rules that place the set of AS's contained in the burst into equivalence classes, based on the type of event that could have taken place at that AS that would have caused the observation. We motivate our approach with an example. Suppose view V's routing table contains an AS path

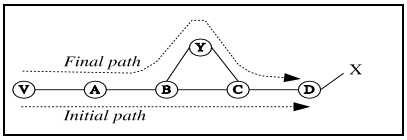

Figure 7: Example: Inference from single view.

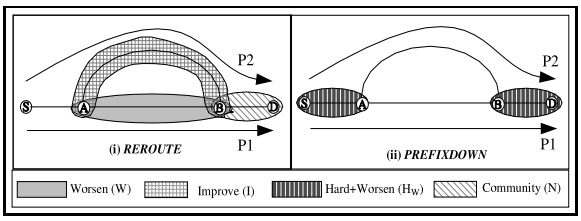

Figure 8: *Single view rules*: The event must have occurred in the shaded regions.

$[V, A, B, C, D]$ to a prefix X as shown in Figure 7, and assume for now that AS's are singly peered. Suppose after some time the path changes to $[V, A, B, Y, C, D]$ and remains stable for some time. There are several possible events that could explain this change: perhaps Y advertised a lower MED to B, or perhaps the link (B, C) failed, or perhaps D changed a community attribute in the message triggering a route change at B. However, certain events could not explain this change: any event happening at A, or a failure of link (B, Y), or Y advertising a higher MED to B could not have caused this observation. In general, there are three possible explanations: either (1) some event happened on the path $[B, C]$ to make it less desirable to B (worsened), (2) some event happened on the path $[B, Y, C]$ to make it more desirable to B (improved), or (3) some router on the path $[C, D]$ changed a community attribute. Therefore, we would place $\{B, Y, C\}$ into the S_I class, $\{B, C\}$ into the S_W class, and $\{C, D\}$ into the N class. We can then compute the *suspect location set* as $\{B, Y, C\} \cup \{B, C\} \cup \{C, D\} = \{B, Y, C, D\}$, and the *suspect cause set* as all causes corresponding to the Soft+Improve, Soft+Worsen, and Community equivalence classes.

Due to space limitations, we describe in detail only the two inference rules in Figure 8, the others are in Appendix II. For simplicity, we first describe the rules assuming there is a single peering link between adjacent AS's. We then describe how to eliminate this assumption in the following section.

1. **REROUTE:** If we observe a route change from path P_1 to path P_2, there are three possible explanations: some property of $W = P_1 - (P_1 \cap P_2)$ worsened, some property of $I = P_2 - (P_1 \cap P_2)$ improved, or router in the path downstream of AS B could have triggered a reroute by changing a community attribute.
2. **PREFIXDOWN:** A PREFIXDOWN indicates with certainty that some part of the path has worsened. Moreover, since we observe a withdraw, we also know this is a hard event. Suppose we observe an advertised prefix with path P_1 become withdrawn, with intermediate paths $\{P_2 \ldots P_n\}$. Then we know that before the event occurred, all the intermediate routes were working. Now, none of them are working since they have all been withdrawn. Hence, we know that the event must be located in the intersection of these paths. Hence, the event must have occurred in $W = \bigcap_{i=1}^{n} P_i$.

5.1.2 Rules for multiply-peered AS's

The greatest bottleneck in improving precision is due to the *multiple peering link* problem. This problem arises from the fact that a pair of AS's can be connected by multiple peering sessions. An example is shown in Figure 9. Suppose AS 1 observes an update to a prefix

contained in AS 5, but AS 2 does not. Suppose also that AS 2 has a valid advertised route for the prefix in its routing table throughout the event, and assume for simplicity that community attribute changes do not occur. If AS 3 and AS 4 are singly peered, then we know that the event causing the update must have occurred in AS 3, AS 1, or the peering link between them. On the other hand, if AS 3 and AS 4 are peered more than once, then we can no longer make the same claim. For example, a router failure in AS 4 could affect the route propagated to AS 1 and not affect the route propagated to AS 2.

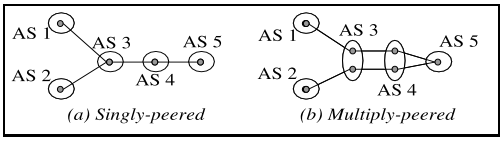

Figure 9: Knowing the details of peering relationships can help infer more precisely.

Knowing how many times a pair of AS's are peered (which can be inferred using [20]) can allow us to use the rules defined in the previous section as stated. In this work, we safely assume no such information is available. Hence, we modified the rules in the previous section to assume by default that two paths sharing a link are actually traversing two different peering sessions. For example, in the case of a REROUTE, we extend the sets I and W as far upstream (towards S) as the last place P_1 and P_2 must have traversed the same router before they reach AS A. Similarly, we extend these sets downstream (towards D) as far as the first place where the paths must have traversed the same router after they leave AS B. Finally, another difficulty is that flap damping may affect one of the peering links but not the other, which can cause different observations to be made at the two views. This is dealt with explicitly in the rules defined over multiple views, as discussed in the next section.

5.2 Refinement across views

In this section, we combine inferences across several views to narrow down the suspect set. The algorithm consists of three steps. (a) identify groups of related observations across views. (b) intersect suspect sets (c) additional inference rules across views. We explain each of these steps below.

Identifying related observations: We found that most prefixes are stable, and are updated in bursts separated by long silence periods. For stable prefixes, convergence time is known to be 2 to 3 minutes [12, 15], since flap damping will not be triggered. Because of this, all views observing an event affecting a stable prefix will observe its effects within a short period of time. Hence, for each burst, we collect all other bursts affecting the same prefix that occurred within a δ_{sc} second window at other views, and we group these together into a *superburst*. Accuracy of our approach was not particularly sensitive to the choice of δ_{sc}, as long as this value was chosen to be less than the minimum time separating bursts $\delta_{sil} = 3700$ seconds. On the other hand, if the prefix is unstable, then flap damping can cause bursts to be separated by long periods of time. However, we can always detect when flap damping could have occurred, since a dampened prefix is always withdrawn. We handle this by defining rules that ignore observations on prefixes that could have been dampened while the event is occurring.

Intersection of suspect sets: We first compute the contents of the equivalence classes H_I, H_W, S_I, S_W, N at each view in isolation, then the intersection of each class across views. We then determine the suspect set by computing the union of all remaining elements in the classes. Since we are unaware of the cause associated with elements in the M and U classes, we compute the suspect set for each equivalence class c as $S_c = \bigcap_{v \in V} \big(Sus(c,v) \cup Sus(U,v) \cup Sus(M,v)\big)$. This ensures that we don't eliminate the AS where the event occurred from S_c when the cause is unknown or when multiple events occurred.

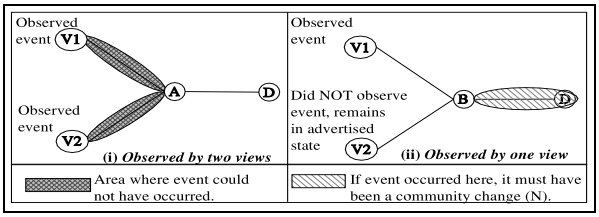

Figure 10: *Multiple views*: i) The event could not have occurred in the crosshatched region. ii) If the event occurred in the crossed region, it must have been due to a community change.

Inference rules across views: We apply the two rules shown in Figure 10 to S_c:

1. *V1 and V2 observe event:* Suppose the event is observed in two views $V1$ and $V2$. Consider the point at which their paths first traverse the same AS. We know that the event did not occur in the paths upstream (towards the views) of this point, as shown by the shaded area in Figure 10-i. If it did, it would have been observed in only one of the views. Hence, we remove from S all elements upstream of the first point at which the two paths enter the same AS.
2. *Only V1 observes event:* Suppose the event is observed in some view $V1$ but is not observed in another view $V2$ as shown in Figure 10-ii. Further, assume that the route to D at $V2$ remains in the advertised state throughout the event, so we know flap damping did not affect $V2$'s route. Then, the event did not occur downstream of the first point the two paths traverse the same router as shown by the shaded area in Figure 10-ii. If it did, it would have been observed by $V2$, since the route is not being dampened. Hence, we intersect S with elements downstream of the first point at which the two paths traverse the same router.

5.3 Simultaneous events

Table 3: Observations caused by simultaneous events.

View 1 obsv.	*View 2 obsv.*	*Freq.*	*Secondary effect or Independent events?*
PREFIXDOWN	FLAP	0.03%	Secondary or Independent
PREFIXDOWN	PREFIXUP	0%	Independent
PREFIXUP	FLAP	0.002%	Independent
REROUTE	FLAP	1.33%	Secondary or Independent

The algorithm given above is based on the assumption that only a single event affecting a particular prefix occurs at a time. Unfortunately, this is not always the case. First, an event can trigger a *secondary effect*, such as flap damping, non-determinism in path selection (for example, age-based tie-breaking [5]), or a congestion-related session reset arising from traffic shifted by the original route advertisement. Otherwise, two *independent events* can occur at the same time. However, in certain cases we know for certain that the burst observed in a particular view was the result of simultaneous events. In particular, certain combinations of observations across views can only be caused by simultaneously occurring events, as shown in Table 3. We cannot apply the rules discussed in the previous sections to bursts caused by simultaneous events, as these rules assume there is a signature of only a single event present in the burst. Hence, we ignore the bursts contained in Table 3 from consideration when acquiring our results. We found that only 1.4% of bursts are of this form, hence doing so did not significantly affect our results, yet reduced the probability of making inaccurate inferences.

6. Methodology and Validation

In this section we develop a methodology for validating the correctness of our approach. We use publicly available traces of BGP updates from Routeviews [25] and RIPE [26]. The contents include (a) whether the update was an announcement or withdrawal, (b) the IP address of the view forwarding the update, (c) the prefix being updated, and (d) the set of AS hops used to reach the destination (AS path) (e) a time-stamp when the update was logged. We discard the other fields [9] of the updates that are received by the monitor, since they are not used in our algorithm. One particular problem with the data collection approach of Routeviews and RIPE is that session resets may occur between the monitor and the view [23]. These events could cause our approach to erroneously infer that a Turbulent event occurred. However, as observed in previous works [19, 3], we can filter out the effects of these resets by discarding redundant updates that do not change the contents of the routing table.

The assumptions we can make during Turbulent and Quiescent periods are fundamentally different, and hence we need different approaches to validate during each of these periods. Now, we describe our methodology for validating our inferences during these periods.

Validation in Turbulent periods: We use three methods to verify the accuracy of inferences during Turbulent periods. First, as a sanity check, we show that our algorithm detects increased numbers of session resets when Internet worms are propagating. This is expected because it is well documented that worms cause large amounts of congestion, leading to session resets. This congestion can cause keep-alive messages exchanged between routers peered across links to be lost [23].

Next, we are able to cross-validate inferences made at each view in isolation, by showing that we can sometimes detect the same reset from multiple viewpoints. This provides additional confidence that our algorithm can correctly pinpoint the location of session resets.

Finally, we show that we can detect some known major events triggering session resets that are documented in the NANOG mailing list [31] and other sources. We also applied post-mortem analysis, by taking major events that we found and showing that we could often explain their cause by events discussed in various mailing lists and web sites[5].

Validation in Quiescent periods: Minor events that trigger updates during Quiescent periods are typically not documented. This makes validation of our inferences in Quiescent periods a hard task. However, for two specific types of BGP updates in Quiescent periods we are aware of the AS originating the event that triggered these updates. These include: (a) BGP Beacons, (b) updates received by a viewpoint at the originating AS. For these two classes of updates, we validate the correctness of our Quiescent inference algorithm by checking whether the AS originating an update is present in the suspect set generated by our algorithm.

BGP Beacons: BGP beacons are publicly documented prefixes that are advertised and withdrawn at regular intervals, and have been previously used to study BGP routing convergence [15]. It is highly likely that if any view observes an update to a beacon prefix with origin AS X, the update originated in AS X. We used both the RIPE beacons [29] and PSG beacons [15] to validate. The PSG beacons include 4 prefixes, and the RIPE beacons include 9 prefixes. Each prefix is cyclically advertised and withdrawn with a fixed period of two hours between events. These beacons are widely distributed geographically, and are originated from a variety of levels in the Internet hierarchy. Since the origin AS of each of the beacon prefixes is publicly documented, and the schedule of announcements and withdrawals is also public, we can use these events to determine the false negative rate of our approach.

[5]We have developed a web site (similar to that of http://www.cidr-report.org/) where we are running our algorithms in real time to search for events. It contains a more complete analysis of 18 months of updates. For additional validation, we plan to circulate this site to ISPs, and ask for feedback regarding accuracy.

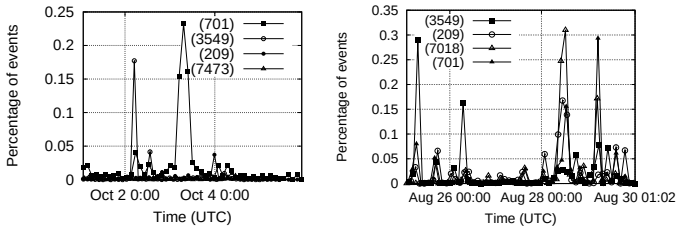

Figure 12: PDF of observed events occurring in 15 minute intervals during (a) UUNET's routing problems on 10/3/2002. (b) AT&T's routing problems on 8/28/2002.

Viewpoint at the origin: We use the simple observation that if a view in an AS observes an event on a prefix owned by the same AS, then the event must have occurred in that AS. Based on BGP routing tables we can determine the set of prefixes owned by each AS [6]. Given this AS-prefix mapping, if a viewpoint in some AS X observes a route update for a prefix owned by AS X, then we can be confident that this update was indeed triggered by AS X.

Our validation method works as follows: First, for each viewpoint V in AS X, we only consider the updates for prefixes owned by AS X. Next, we hold out updates observed at V and run our inference algorithm using updates solely from views in other AS's. Finally, we check whether AS X is indeed present in the suspect set corresponding to these updates. We measure the false negative rate as the number of times that our algorithm did not include AS X in the suspect set.

6.1 TurbulentInfer validation

Figure 11a shows the effect of the SQL slammer worm [27] on peering sessions. We observe that several peering sessions are reset during the SQL Slammer worm period. On the other hand, we observe very few occasions with multiple resets during a normal observation period. This validates our approach, since many session resets were known to occur during the spread of the SQL Slammer worm [23]. Figures 11b and 11c illustrate a similar result for two other popular worm events: the NIMDA worm on 9/18/01, and the Code Red worm on 7/19/01. We repeated our analysis for Code Red II and the Goner [27] worms and observed similar results.

Next, we cross-validate our Turbulent inferences at one viewpoint by attempting to detect the same events from other viewpoints. However, in a majority of cases, we find that a single session reset is visible only to a single view, and causes only this view to experience a Turbulent period. This phenomenon occurs because BGP routers rate limit the number of advertisements and withdrawals propagated within a unit time period. Hence, our ability to use multiple vantage points to cross-validate Turbulent inferences is limited. In practice, we found that 20% of session resets could be validated from multiple viewpoints. In almost all these cases, the event occurred on a link close to two or more viewpoints. In addition, we were sometimes able to observe a reset affecting paths traversing both directions on a link.

Table 4: Several known events used for validation.

Event	*Date*	*Where documented*
AT&T routing problems	8/28/2002	NANOG mailing list [31]
UUNET routing problems	10/3/2002	NANOG mailing list [31]
Peering link instability	7/21/2003	Sprint web site [32]
WorldCom peering problems	11/11/2002	web site [27]

Finally, we validated our technique by running our algorithm on traces collected during periods where we knew the exact location of

[6]There might be inconsistencies in this data set due to origin AS conflicts [24] caused by misconfigurations. We discard all inconsistent prefixes from this analysis.

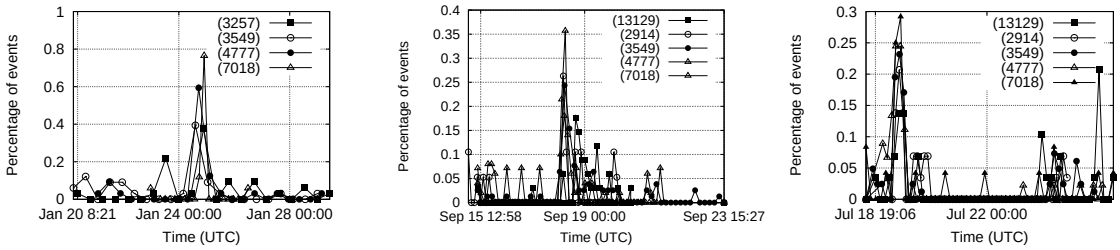

Figure 11: (a) Effect of the SQL slammer worm on peering sessions. Chart shows the PDF of the number of 15 minute intervals containing multiple resets. We filter out updates due to local resets. (b) Effect of NIMDA worm. (c) Effect of Code Red worm.

some major Internet event taking place [27], and measuring the ability of our scheme to pinpoint the location of the event. We considered 8 major events, 4 of which are listed in Table 4. In each case, we were able to exactly pinpoint the AS that caused the event and correctly distinguish between link and AS failures. For example, Figures 12a and 12b show the number of update bursts attributed to various AS's by the algorithm. Results from other AS's had trends similar to those shown here. We notice large spikes during these periods which pinpoint the AS undergoing the event.

6.2 QuiescentInfer validation

Table 5: Beacon analysis results

Suspect set size	1	2	3	4	all
Percent of bursts	61%	92%	93%	97%	100%
Inferences containing beacon AS	100%	100%	100%	100%	100%

BGP Beacons: We found our approach could classify beacon updates with very high precision. Since the events induced on beacon prefixes are origin advertisements and withdrawals, this result validates the PREFIXUP and PREFIXDOWN rules from Section 5.1, as well as the rules used to refine across views from Section 5.2. We used a data set comprising over 1 year of beacon updates, taken from July 1 2002 through August 31 2003. During this time we found 48,624 bursts corresponding to events on beacon prefixes. The origin of the BGP Beacon was present in the suspect set for *all* 48,624 bursts. In addition, we were able to narrow down suspect set to 1 in 61% of events, to 2 in 91% of events as shown in Table 5. While the average suspect set size output by QuiescentInfer is typically more than 3, we are able to achieve a very high precision in the case of BGP Beacons. This happens because these events tend to be visible from a number of vantage points, allowing us to narrow down the suspect set size by intersecting inferences made at different vantage points. These observations suggest that the inference rules have a very low false negative rate for origin related events.

Table 6: Viewpoint at the origin analysis results

Suspect set size	1	2	3	4
Avg. suspect set size	7%	20%	36%	54%
Incorrect inferences	0%	0%	0%	0%

Viewpoint at the origin: Unlike the case of BGP Beacons, a variety of events may trigger the set of updates that we notice at the viewpoint in the origin AS. For example, the event could be triggered by a local policy change within the AS, changes in peering link preference, origin advertisements/withdrawals, or intra-domain link weight changes. Given that we hold out the updates at this viewpoint, we provide no additional information to the Quiescent inference algorithm when we use the other views alone. If we had included the viewpoint in our analysis, the Quiescent inference algorithm would always be able to correctly pinpoint the origin AS as the only suspect. This approach validates the rules described in both Sections 5.1 and 5.2.

Our validation data set comprised 31 million updates observed from 23 views collected over a period of 10 days. For each of the 23 views, we held out data for that view, ran our algorithm on data from the other views, then checked to see if there was a conflict between the held out data and the inference. Table 6 summarizes the size of the suspect set and the number of incorrect inferences for all these events. We make two observations. First, as in the BGP Beacon analysis, the origin AS was present in the suspect set generated for all these events. Second, unlike the BGP Beacon case, the precision to which we can narrow down the suspect set is much smaller. This is because many of these events are observed in only a few views, thereby reducing visibility. For example, if only one view (other than the origin viewpoint) several hops away from the origin observes the event, the suspect set can be quite large. Additionally, the precision of the inference rules for certain patterns of updates (e.g. flaps and reroutes) is fundamentally limited. For example, if a viewpoint observes a flap from path $P \rightarrow Q \rightarrow R \rightarrow P$, where P, Q, R represent AS paths, any AS present in $P, Q,$ or R can be a potential suspect candidate for triggering the flap. We now analyze the precision of the Quiescent inference algorithm in greater detail.

6.2.1 Precision

In QuiescentInfer, we perform inference over a smaller number of updates than in TurbulentInfer. Hence, the precision to which we can localize the cause is inherently reduced. However, we found that it is still possible to reduce the suspect set to a small value, as shown in Figure 13. We computed the suspect set in three different ways: First, we consider a *naive* approach, considering each burst in isolation and computing the suspect set by taking the union of all AS's that appear (Figure 13a). Next, we applied our scheme to a data set containing updates from only a single view (Figure 13b). Finally, we used all the rules described in Section 5 including the refinement from several views (Figure 13c). We make several observations from our analysis. First, compared to the naive approach, we can achieve a drastic reduction in the suspect set size, by a factor of $3 - 4$ on average. Next, the suspect set size is dependent on the type of pattern observed at the views. For example, origin events (PREFIXUP, PREFIXDOWN) cause small suspect sets, as we observed earlier with BGP Beacons. The suspect set was larger for FLAP and REROUTE bursts, since they are often associated with a delayed convergence process that introduces many AS's as suspects. In addition, using multiple views can significantly reduce the suspect set size, by a factor of $1.5 - 2$ over inference at a single view in isolation. Overall, our approach can reduce the suspect set size to 4 on average. We consider this value to be small, since although we can reduce this size to 1 in certain cases, in general a suspect set of size 2 is optimal, since it hard to distinguish between a failure of an inter-AS link and a failure of a router inside an AS [4]. Also, for certain classes of events such as origin withdrawals, we can reduce the suspect set to 2 AS's on average.

7. Analysis and Observations

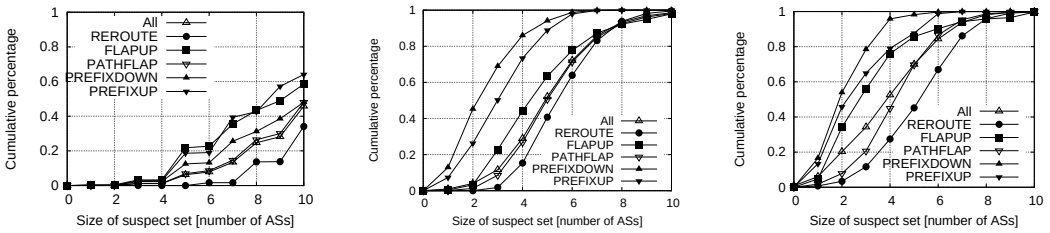

Figure 13: ***Precision:*** **We plot the CDF of the suspect location set size for three cases: (a) naive approach (b) our approach, using observations only at a single view (c) our approach using observations from all views collected over 10 days. For example, the FLAPUP curve passes through the point (4,0.45) in case (b). This means that in 45% of FLAPUP observations, we can reduce the suspect set to within 4 AS's.**

In this section, we demonstrate the abilities of our health inferencing system to detect anomalous routing events. Such a system can both be used by network operators to isolate and repair faults, and also to provide better insight into Internet routing dynamics. Based on our analysis of 18 months of updates from 23 views, we make several observations:

1. Our system can detect major routing anomalies, many of which were previously unknown (Section 7.1).
2. Session resets are a common occurrence, and a small number of inter-AS links are perennially unstable (Section *7.2.1*).
3. 25% of prefixes are continuously flapping and account for a large fraction of routing updates (Section *7.2.3*).
4. Events occurring in the Internet core are much more likely to result in flaps as compared to events occurring near the network edge (Section 7.3).

7.1 Previously unnoticed events

In this section, we provide some examples of previously unnoticed events that had significant effects on end-to-end routes. To our knowledge, many such events, though major, are not public knowledge and have previously gone unnoticed.

Peering link instability: On July 21 2003, the peering link between Sprint (AS 1239) and UUNET (AS 701) underwent a large number of session-reset like events, affecting the reachability of over 20,000 prefixes. The affected domains include cnet.com, excite.com, and weather.com. During this period of time, the AS paths traversed by these prefixes repeatedly cycled through several paths, occasionally interspersed with withdrawals. Sprint's web site [32] notes outages during this period of time but does not reveal the magnitude, cause, or location of the event.

Peering link instability II: On 12:08am, May 10 2002 (Friday), an event on the peering link between AS 3561 and AS 1239 caused over 9000 prefixes to be rerouted to alternate paths. This event triggered a period of instability lasting for 3 days, where these prefixes repeatedly flapped between using the link (3561,1239) and using alternate paths, generating over 135,000 updates. The problem was fixed 12 PM, May 13 2002 (Monday). This event affected 5 of the 50 most popular web sites [30], including Barnes&Nobles, Ebay, and Hewlett-Packard.

Reroute: On January 23 2003, at 9:52 am, some event on the peering link between AS 2914 and AS 3561 abruptly caused over 6000 prefixes to change to alternate paths. These prefixes abruptly returned to their original paths at 10:57 am. Prefixes owned by several major providers were affected by the event, including Bell Atlantic, Nortel, and Cable&Wireless.

Misconfiguration: On June 26 2003 at 7:14pm, AS 2500 began to advertise paths for over 500 prefixes it did not own. This event affected prefixes owned by several major providers, including AT&T WorldNet Services, NTT, and Cable&Wireless. The instability was short-lived, lasting about 15 minutes.

7.2 What are the major causes of routing updates?

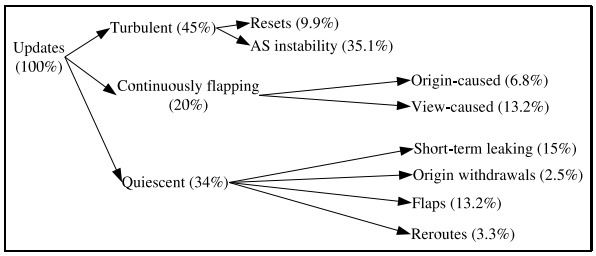

Figure 14: Breakdown of updates by cause.

Inferring the specific reason why an update was generated is a very hard problem. In this section we present a top down approach describing classes of observations for which we can narrow down the potential causes to a small list. Our key results are summarized in Figure 14.

7.2.1 Turbulent events

When an AS has a major routing problem, it is conventional wisdom that operators often contact each other either by phone or e-mails in operator mailing lists [31] to isolate/repair the problem. For example, there are over 100 messages discussing the UUNET routing problems of 10/3/2002 on the NANOG mailing list [31]. Using a health monitor to detect and display Turbulent events could be a very useful tool for these operators, allowing them to share information and troubleshoot faults causing Turbulent events. In addition, due to the large number of updates present, our algorithms provide a high level of precision when inferring Turbulent events. There are two major types of Turbulent events:

Resets: We detected 25,373 resets over the 18 month period. Approximately 10% of all updates are caused by resets, not including updates due to resets between the monitor and the views. We found that these events are very bursty: over 50% of Turbulent events we observed occured within 2 hours of another Turbulent event on the same link. For example, the link (1221,4637) underwent 737 resets, half of which took place in July-August 2003.

AS instability: We detected 14,014 turbulent events caused by AS's. Almost all of these events occurred in the AS containing the view. We found that AS 3257 and AS 852 were the most unstable, causing 31% of all observed events of this type.

7.2.2 Quiescent events

We found that most Quiescent updates are part of flaps, as shown in Table 1. In addition, advertisements/withdrawals of prefixes had the longest bursts and tended to affect a large number of views, and hence accounted for an even smaller fraction of events.

Although our approach has less precision inferring Quiescent events than inferring Turbulent events, in certain specific cases we can achieve

high precision. Next are two specific events which are important to study and for which we can achieve high precision.

Short-term leaking of subnets: It is common for a BGP router to have two or more prefixes of different lengths in its routing table corresponding to a particular IP address. Sometimes a router containing a particular prefix will receive an advertisement for a more specific prefix followed soon after by a withdrawal of that prefix [14]. This *leaking* of subnets should not occur in a properly configured BGP network, but can be caused by misconfiguration, migration/renumbering of part of an old network, brief changes to route filters, or load balancing [28]. About 15% of bursts affected prefixes already covered by a more persistently available supernet route. These routes are particularly unstable. For example, about 50% of these advertisements were withdrawn within 8 minutes of being advertised.

Origin withdrawals: Occasionally we observed a particular prefix being withdrawn from all views at the same time. Although an origin withdrawal can be a valid event, caused by a change in the origin AS's policy or by flap damping being triggered, it can also be caused by misconfigurations. These events are particularly undesirable, since the effects are typically associated with a long convergence process, and must in general propagate to every BGP router in the Internet. Hence it is important to pinpoint the AS's that generate a large number of origin withdrawals. We detected 794,748 such events over the 18 month period, which caused roughly 3% of all route updates. The majority of these events took place in a small number of AS's: 52% of the events took place in 5% of the 2257 AS's on which we observed these events.

7.2.3 Continuously flapping events

Here we discuss some preliminary results regarding events that cause prefixes to be continuously updated. We are currently investigating this category in more detail. A continuously flapping prefix is a prefix that is persistently updated over long periods of time, without experiencing a *silent period* during which the prefix is not updated. In general, the more often a prefix flaps, the more of a problem it becomes. We chose to define a prefix to be continually flapping if updates are received for a period longer than 65 minutes without a silent period longer than 60 minutes. We noticed an incredibly large portion of prefixes (25%) were continuously flapping. For example, prefix 63.162.136.0/23 was updated on average once every 30 seconds at view 208.51.113.254 for a period of over 80 days. Through our analysis, we observed two major classes of flapping prefixes.

Near-origin: Flaps where the prefix alternates between being in a withdrawn state and being in an advertised state. We found that this type could often be observed by more than one view, and that this type is usually caused close to the origin. One possible cause of this class of updates are traffic shaping equipment performing online load balancing. Roughly 33% of continuously flapping updates were near-origin.

Near-view: Flaps during which the AS path alternates between a certain set of paths without being withdrawn. We found that this type could rarely be observed in more than one view, but could often be observed by two views if both views were located within the same AS. Hence it is likely that this type is caused by instability in the AS containing the view. There are several possible causes for these events: inter/intra-domain BGP dynamics [7], policy conflicts [8], or equipment performing online traffic engineering [28]. Although we cannot distinguish between these causes, we can distinguish these Near-view flaps from those that occur at the origin.

7.3 Where do these events occur?

Turbulent events: We found that 10% of the links on which we could observe resets caused 86% of the observed resets. These links suffer over four resets per month on average. Also, we noticed that the most unstable link in one view was usually different from the most unstable link in another view. This shows that resets tend to affect certain views more than others, although we noticed resets on certain links can have very large and wide ranging impact. For example, resets of links near Tier-1 ISPs [21] have a much greater impact and hence tend to be observed from more than one view. We envision that our algorithm could be used to determine the peering sessions most susceptible to resets, which could then be made more robust. In addition, the magnitudes of the effects of resets appear to dissipate as the distance between the view and the reset link increases, due to WRATE [16] and a tendency of routes to fanout at each router.

We used the algorithm described in [21] to classify links as *peer-peer* and *provider-customer*. We observed a larger number of Turbulent events on peer-peer links than on provider-customer links. During the 18 month period, we observed 6358 Turbulent events on a total of 298 peer-peer links. Of these, on average 4099 prefixes were affected. On provider-customer links, we observed 1455 Turbulent events on a total of 134 links. Of these, on average 1801 prefixes were affected.

Quiescent events: We classified AS's by their level in the AS hierarchy [21]. We found that events affecting prefixes that were originated from higher levels in the AS hierarchy are more likely to be FLAPs. For example, 6.2% of FLAP events affected prefixes originated from core AS's, while only 1.3% of REROUTE events affected these prefixes. Since most views are either located in the core, or peered with a core AS, the path between the view and the origin is highly likely to transit only core AS's. From this, we surmise that FLAP events are more common in the core, while events that cause more permanent changes tend to occur near the edge. We confirmed this observation by considering events affecting the 50 most popular web sites as given by [30]. These prefixes are usually hosted by AS's in the core. As expected, we found that the prefixes containing these sites suffered from more flaps, and fewer Reroutes and Withdrawals, as compared to other prefixes.

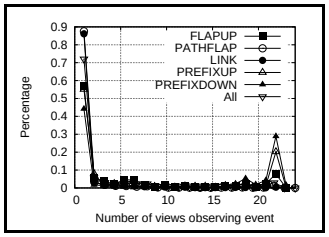

Figure 15: *Visibility:* PDF of number of views observing event.

Figure 15 measures visibility in terms of the number of views that observed an event. We can see that the distribution is multi-modal, with many events only observed in a single view and many observed in several views. This shows that a large number of updates received are not caused by the router originating the prefix, since they are not observed in all views, and hence are caused by some element between the router originating the prefix and the viewpoints observing the update.

8. Related work

There have been many recent advances in understanding why route changes take place in BGP. It is known that the majority of routing instability is due to pathological causes, such as poor implementation decisions, bugs, and misconfiguration of router software [11, 14]. Instability has been shown to be a major factor in causing outages and path oscillations, and in fact many outages can be predicted by observing BGP advertisements [4]. Stability can often be improved by propagating additional information about the change [16]. However, the most heavily trafficked Internet routes are the most stable, and most route updates arise from a small number of unpopular destinations [19]. We extend previous studies of BGP by showing a more accurate characterization of how often and where various events occur, the instability

they introduce, and their effects on Internet routing.

Another important body of literature studies inference techniques that determine the location and type of a routing event taking place from incomplete information. For example, passive observation of client-server traffic can be used to infer the location of lossy links with low error [17]. In the context of BGP, update messages can be used to discover and map certain administrative or topological features within an AS [1]. They can be used in conjunction with traceroutes to discover ISP topologies [20], and can be used in conjunction with UDP probes to find the location of failures within an ISP [4]. Due to the complex dynamics of inter-domain routing, many types of events can cause the same stream of updates, and many possible streams of updates can come from the same event [12, 6], making inference difficult. However, the update traffic observed at a router can be characterized as bursts of activity interspersed with silent periods [13], and this observation can be used to narrow down the suspect set [3]. We extend previous inference techniques by showing a complete algorithm to find the cause and location during Quiescent and Turbulent periods, and validating over a large number of updates.

9. Conclusions and future work

In this work we take some first steps towards developing a general solution for inferring the cause and location of origin of events triggering route changes in BGP. We developed techniques to distinguish correlated and independent observations, and how to find the set of locations where routing events could have occurred from the observations. We validated our technique by considering well-known major events, including routing anomalies in the backbones of major ISPs, and well-known minor events, such as BGP Beacons.

We found our approach could pinpoint the location where an update was triggered to a single inter-AS link in over 70% of observed updates. This allowed us to make several observations about inter-domain routing events in general: a small number of links cause the majority of updates, most routing events are observable from a small number of views, and flaps make up a larger percentage of events in the Internet core. In addition, we were able to discover several major events that were not public knowledge. These results show that it is possible to build an accurate health inference system, a key first step improving understanding of inter-domain routing dynamics.

For future work, we plan to investigate continuously flapping prefixes in more detail. We are also currently investigating whether applying statistical inference techniques to this problem will improve accuracy [17]. The location where views are placed can have a large effect on the degree of visibility of events, and which events can be observed, so we would like to develop an approach for determining which locations in the network are most critical for deploying vantage points. Finally, we plan to complete the design of the health monitor, by developing guidelines for determining which observations constitute unhealthy behavior. The system can then trigger alarms when these observations occur.

References

[1] D. Andersen, N. Feamster, "Topology inference from BGP routing dynamics," in *Proc. of IMW 2002*, November 2002.

[2] A. Basu, C. Ong, A. Rasala, F. Shepherd, G. Wilfong, "Route oscillations in I-BGP with route reflection" in *Proc. of SIGCOMM*, Pittsburgh, PA, August 2002.

[3] D. Chang, R. Govindan, J. Heidemann, "The Temporal and Topological Characteristics of BGP Path Changes," in *Proc. of ICNP*, Atlanta, GA, November 2003.

[4] N. Feamster, D. Andersen, H. Balakrishnan, M. Kaashoek, "Measuring the Effects of Internet Path Faults on Reactive Routing," in ACM SIGMETRICS, San Diego, CA, June 2003.

[5] N. Feamster, J. Borkenhagen, J. Rexford, "Guidelines for interdomain traffic engineering," in *ACM SIGCOMM Computer Communications Review*, Fall 2003.

[6] T. Griffin, "What is the sound of one route flapping?," presentation made at the *Network Modeling and Simulation Summer Workshop*, 2002.

[7] T. Griffin, G. Wilfong, "On the correctness of IBGP configuration" in *Proc. of SIGCOMM*, Pittsburgh, PA, August 2002.

[8] T. Griffin, G. Wilfong, "An analysis of BGP convergence properties," in *Proc. of SIGCOMM*, September 1999.

[9] S. Halabi, D. McPherson, *Internet Routing Architectures*, Cisco Press, second edition, 2001.

[10] C. Labovitz, A. Ahuja, F. Jahanian, "Experimental study of Internet stability and wide-area network failures," in *Proc. of Fault Tolerant Computing Symposium*, June 1999.

[11] C. Labovitz, G. Malan, F. Jahanian, "Internet Routing Instability," in *IEEE/ACM Transactions on Networking*, October 1998.

[12] C. Labovitz, R. Wattenhofer, S. Venkatachary, A. Ahuja, "The impact of Internet policy and topology on delayed routing convergence," in *Proc. of INFOCOM*, April 2001.

[13] O. Maennel, A. Feldmann, "Realistic BGP traffic for test labs," in *Proc. of SIGCOMM*, Pittsburgh, PA, August 2002.

[14] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *Proc. of SIGCOMM*, Pittsburgh, PA, August 2002.

[15] Z. Mao, R. Bush, T. Griffin, M. Roughan, "BGP beacons," in *Proc. Internet Measurement Conference*, October 2003.

[16] Z. Mao, R. Govindan, D. Varghese, R. Katz, "Route flap damping exacerbates Internet routing convergence," in *Proc. of SIGCOMM*, Pittsburgh, PA, August 2002.

[17] V. Padmanabhan, L. Qiu, H. Wang, "Server-based Inference of Internet Link Lossiness," in *IEEE Infocom 2003*, April 2003.

[18] Y. Rekhter, T. Li, "A Border Gateway Protocol 4 (BGP-4)", IETF, *RFC 1771*, March 1995. Section 9, pgs 33-46

[19] J. Rexford, J. Wang, Z. Xiao, Y. Zhang, "BGP routing stability of popular destinations," in *Proc. of IMW*, November 2002.

[20] N. Spring, R. Mahajan, D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. of SIGCOMM*, Pittsburgh, PA, August 2002.

[21] L. Subramanian, S. Agarwal, J. Rexford, R. Katz,"Characterizing the Internet Hierarchy from Multiple Vantage Points," in *IEEE Infocom 2002*, June 2002.

[22] K. Varadhan, R. Govindan, D. Estrin, "Persistent route oscillations in inter-domain routing," Tech. Rep. 96-631, USC/ISI, February 1996.

[23] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. Wu, L. Zhang, "Observation and analysis of BGP behavior under stress," in *Proc. of IMW*, November 2002.

[24] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Wu, L. Zhang, "An analysis of BGP multiple origin AS (MOAS) conflicts," in *Proc. of IMW*, November 2002.

[25] "Route Views Project," `http://www.routeviews.org`.

[26] "RIPE NCC RIS," `http://www.ripe.net/ripencc/pub-services/np/ris-index.html`.

[27] "Matrix NetSystems - In The Net - Event Advisories," `http://www.xaffire.com/press/ea/ea_archive/`.

[28] netvmg, Company, `http://www.netvmg.com`.

[29] "RIS Routing Beacons," `http://www.ripe.net/ris/beacon.html`

[30] "comScore Media Metrix, Top Internet Property Rankings," `http://www.comscore.com/press/release.asp?id=348`

[31] "NANOG Mailing List," `http://www.merit.edu/mail.archives/nanog/`

[32] "Sprintlink: Scheduled Maintenence and Outage" `http://www.sprintlink.net/maintview/`

[33] M. Caesar, L. Subramanian, R. Katz, "BGP health monitoring: realtime analysis," web site `http://www.cs.berkeley.edu/~mccaesar/hmon.html`