Towards Localizing Root Causes of BGP Dynamics

Matthew Caesar, Lakshminarayanan Subramanian, Randy H. Katz
{mccaesar,lakme,randy}@cs.berkeley.edu

Abstract

Today, we lack a clear understanding of the dynamics of the Border Gateway Protocol (BGP) and this has largely restricted our ability to address BGP's shortcomings. To gain more insight into BGP's dynamics, this paper proposes the design of a *BGP health inferencing system* that localizes the root causes of routing changes. Specifically, the inference system addresses two questions: *What is the cause of a routing change? Where does a routing change originate?* The inference system correlates routing updates across multiple vantage points to narrow down the *suspect set* of AS's that might have triggered routing changes. Our methodology is primarily targeted towards analyzing events affecting relatively stable prefixes (composing roughly 80% of the routing table), which are known to be the most popular destinations of Internet traffic. For 70% of observed updates to these prefixes, our approach can pinpoint the location of origin to a single inter-AS link. We analytically and empirically argue correctness of several key steps of our methodology and additionally show that our technique can correctly pinpoint the source of several well-known/ documented routing events.

1 Introduction

Internet routing suffers from several problems today, including chronic instability, convergence problems and misconfigurations in routers [9, 1, 13]. Many of these problems arise due to the complexity of the Border Gateway Protocol (BGP), the de facto inter-domain routing protocol. BGP has evolved into a complex protocol, with a number of configurable policies and features that make its dynamics 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 [13, 4]. 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 [15]. Finally, only recently have many problems relating to route oscillations, convergence and inter/intra domain protocol interactions been brought to light [8].

In this paper we take a first step towards improving our understanding, by developing a systematic methodology for analyzing routing changes. We apply this methodology towards the design of a BGP health inferencing system for analyzing route updates from multiple vantage points and localizing the root cause for each update. Localizing the cause involves narrowing down the location of potential AS's and the type of routing event at each location that might have triggered the update. For certain types of massive routing events (*e.g.,*, session resets), it may be possible to pinpoint the exact inter-AS link which might have underwent the reset due to the large number of constituent updates. However, for smaller events (*e.g.,*, route flaps), it may be fundamentally hard to precisely pinpoint the location of the routing event. While Griffin *et al.* [7] have shown root cause analysis to be a hard problem, we find that for a reasonable fraction of BGP updates we can narrow down the location of the event to within 2 AS's (a single inter-AS adjacency) for over 70% of updates.

Despite imperfect precision, we believe that a BGP health inferencing system can indeed be useful in improving the diagnosability of BGP dynamics. We envision deploying our inference algorithms in data collection centers like Routeviews [25] and RIPE [26], which continuously receive streams of route updates from multiple vantage points. Potential applications which can benefit from partial root cause information include: (a) Network operators can use this information for debugging and troubleshooting purposes; (b) One can set BGP policies based on historical statistics of link/node stability; (c) For events with a higher precision, BGP can use this information *online* to improve path selection and damping of instability; (d) Customers/users can leverage these inferences to choose of upstream providers. While our system currently uses only passive BGP update information, we believe that the precision can be enhanced by coupling it with active probing diagnostic tools like AS-traceroute [16].

Validating our inferences can be a very challenging task especially given that many ISPs do not wish to reveal the type or frequency of events within their network. We adopt a two-step methodology towards validation. First, we analytically and empirically show the correctness of certain key steps of our inference approach under a set of assumptions that are known to commonly hold. Second, our system is able to detect a variety of well-known events such as: (a) multiple BGP session resets during Internet worm attacks [23]. (b) publicly documented routing problems experienced by major ISPs [27]. In addition, we analyze the updates generated by BGP Beacons [14] and show that our system can pinpoint the source of these updates with high accuracy[1].

[1]Additionally, we analyze updates on a real-time basis and publish our results online under the hope that network operators can comment

Based on our analysis of route updates collected from Routeviews and RIPE over a period of 18 months, we make two key observations. First, our system can detect major routing anomalies, many of which were previously unknown. Second, we detected on average nearly 1,400 events of high magnitude (*i.e.,*, events that affect many prefixes) per month, and found certain inter-AS links to be perennially unstable.

1.1 Examples

Figure 1 describes four observations to improve intuition about how we determine root cause.

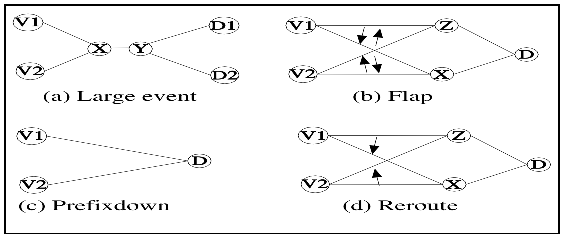


Figure 1: Example observations. We show a simplified version of the topology without AS numbers, without intermediate AS's, and with only a small subset of views.

(a) **Large event:** *If an event affects several prefixes simultaneously, then it is often easier to narrow down location.* For example, at 1am on April 26 2003, multiple views observed 2463 prefixes traversing link (X, Y) to switch to alternate paths, then return to their original state within a short period. From a single view, we can determine the event most likely occurred on the common sub-path across all these prefixes which includes the link (X, Y). Across views, the link (X, Y) appears as the only common candidate inter-AS link and hence, it is likely they were all caused by a single large event affecting AS X, AS Y, or the link between them.

(b) **Flap:** *It is more difficult to isolate cause for flaps, since many AS's are introduced as suspects.* For example, at 5pm on April 13 2003, all 23 views began to observe a flap of prefix 212.48.160/19 from path (V-Z-D) to path (V-X-D). Unlike the previous case, we can no longer pinpoint the AS causing the event, since any AS on either of the paths is suspect. However, we can eliminate certain AS's that don't appear in updates at other views, allowing us to narrow down the cause to a small set of AS's (AS X, Z, or D).

(c) **Prefixdown:** *Observing a withdrawal gives information about the type of event that occurred.* For example, at 6pm on April 3 2003, all 23 views lost a path to prefix 194.55.164/24 for a long period. Since all views lost paths within a short period, it is likely that a single event affected all of the 23 views. Being in a withdrawn state for a sustained period eliminates certain types of events from consideration: e.g. MED or Local-Pref changes, or BGP session resets.

(d) **Reroute:** *Certain state transitions give additional information about the location and type of event.* For example, at 12:30am on April 22 2003, all 23 views observed a route change to prefix 200.71.128.0/20. The paths of all prefixes before the route shared the path (X,D) and after the reroute they all shared (Z,D). The observations could be explained by: a failure at X, or D advertising a lower MED to Z for traffic engineering purposes, etc, but could not be explained by: a failure at Z, or D advertising a higher MED to Z, etc. In general we can say that either some event took place on the link (X,D) that made (X,D) less desirable to some BGP router, or some event took place on the link (Z,D) that made (Z,D) more desirable.

Roadmap: 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. Section 6 describes the 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* (in Section 3, we describe the notion of major and minor events more precisely). 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]. A *link* is an AS-adjacency, which may consist of several peering sessions.

The root cause inference problem can be stated as follows: *Given route updates observed at multiple vantage points, determine the suspect set*=$\{(C_1, L_1), (C_2, L_2) \ldots\}$ *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 at 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) *Correlated vs simultaneous observations:* Events may simultaneously occur [7] and determining the set of updates that are triggered by the same event is hard. (2) *BGP policies:* Policies used to express preferences between routes are not standardized and not disclosed. Also, humans are involved in setting policies introducing the possibility of mistakes. (3) *Multiple peering links:* Even if two AS paths intersect in the AS-topology, they may not intersect in the network-level 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. (4) *I-BGP/E-BGP interactions:* By only observing E-BGP advertisements, we do not get clear visibility

on the validity of our inferences. Our results are available at: `http://www.cs.berkeley.edu/~mccaesar/hmon.html` into intra-AS route dynamics or I-BGP/E-BGP interactions [8]. Hence it is hard to determine whether a problem exists within an AS or on links between peered AS's.

2.1 Assumptions

Our inference rules are based on two assumptions:

(1) *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).* We note this assumption might not hold in certain cases [2] where an AS prefers a provider or peer route to a customer route for a certain prefix, but such cases are known to be rare [22]. We further improve the likelihood that we observe this AS by correlating updates across prefixes and views when possible. However if such a case occurs, if an AS is wrongly specified, or if the AS path is truncated due to implementation bugs, our inference mechanism will not work.

(2) *Artificial damping (due to rate limiting and flap damping) of updates has little effect on updates during minor events.* To infer a minor event precisely, the majority of updates it causes must be observed. There are three reasons why these updates might not be observed. (i) The state of the route might change through several intermediate states while the route is held-down due to flap damping [3]. (ii) Withdrawals may also go unobserved if they are delayed due to WRATE [4]. (iii) The event may not affect any of the views due to its location. In addition, minor events may be unobservable in the presence of major events, since the few updates caused by a minor event get subsumed as noise when a major event triggers many route updates at the same time. However, if a minor event affects a prefix that is not continuously flapping, then updates to it are typically visible. To improve precision, we use multiple vantage points. In practice, we found that using at least one among $3-4$ vantage points typically observes a few updates triggered by a minor event.

2.2 Inference methodology

Figure 2 presents an overview of the flow of our inference algorithm. Our inference methodology uses three basic ideas to solve the root-cause inference problem:

Separating stable from continuously flapping prefixes: We separate prefixes which are relatively stable from those that get continuously updated. We observe that for stable prefixes, two

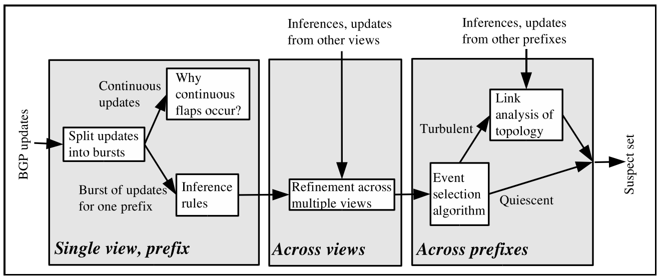


Figure 2: Algorithm flow.

properties commonly hold: (a) the routing events are typically visible and not affected by artificial damping; (b) two different events affecting the same prefix are separable in time. These properties make root-cause analysis for these prefixes feasible. Root-cause analysis is inherently hard for continuously flapping prefixes especially if many of the updates are dampened. While we have made some initial progress analyzing these prefixes (Section 7.2), determining the root-cause for these prefixes is largely an open research problem.

Equivalence classes: While the list of causes of routing events is innumerable, many of these causes can be classified into *equivalence classes*, where each class contains different causes that might trigger the same pattern of updates. Although events in different equivalence classes can often be distinguished merely based on the pattern of observations at a vantage point, causes within a single equivalence class may be indistinguishable. Hence we determine the root-cause only to the granularity of the equivalence class. While we have defined a specific set of equivalence classes in this paper (Section 4.1), there is room for defining a much richer set categorization of causes to obtain a finer distinction between different types of causes.

Correlation of routing updates: One of the key problems we address is how we correlate updates that are caused by the same event without incorrectly correlating observations from different events. We correlate updates across three dimensions: *time*, *prefixes* and *views* (as shown in Figure 2). From a single view, we cluster updates to each prefix based on how close they occur together in time so as to separate routing updates triggered by different routing events. We then cluster across prefixes by noting that the number of prefixes simultaneously updated signifies whether multiple prefixes are affected by the same event (example 1 of Section 1.1). We provide an algorithm for classifying time intervals into *Quiescent* and *Turbulent* periods based on the number of prefixes simultaneously updated, and show that many observations in a Turbulent period are triggered by the same routing event (Section 3). Finally, assuming that the clocks of the different views are loosely synchronized (we require times to be synchronized only on the O(minutes)), we correlate bursts of update activity for different prefixes to narrow down the suspect location of routing events.

3 Correlating route updates

An *observation* refers to a collection of one or more routing updates at a single vantage point. Given a set of observations at different vantage points, the first step towards addressing the

[2] Consider a network with AS's A, B, and C, where (A,B) and (B,C) are private peering links, and suppose A, B, and C have customer routes to reach some destination D. Suppose initially B prefers C to reach D. Suppose C's route fails, causing B to start routing through its customer, which it will advertise to A. If A prefers this path over its customer, then the view in AS A will observe a change from A's customer to B, but no updates will contain AS C. We are unaware of any example that does not require such a preference.

[3] 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 [15].

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

root cause inference problem is to determine the set of observations that are triggered by the same routing event. We define each such group as a set of *correlated* observations. The problem we focus on in this section is how to partition observations into groups such that the groups are disjoint, and each group of observations is caused by the same event.

3.1 Event visibility and separability

The ability to correlate observations triggered by a single event is dependent on two factors: (a) *event visibility*; (b) *event separability*. Event visibility refers to whether a routing event is visible to a vantage point, *i.e.,* whether the vantage point observes any updates triggered by the event. From the set of observations at a single vantage point, two events are said to be *separable* if the vantage point can distinguish updates caused by each event.

We characterize an event $E(X, t, P_S, d)$ using four parameters: X is the location of the event (at the granularity of an AS), t is the time of occurrence, P_S is the set of prefixes affected by the event and d is the time-duration of the event. We define the *magnitude* of an event to be the number of prefixes affected by E which is equal to $|P_S|$.

3.1.1 Event visibility

Whether an event $E(X, t, P_S, d)$ is visible at a vantage point A depends on whether: (a) P_S contains at least one prefix whose route from A traverses X; (b) intermediary routers dampen the updates triggered by E. The *observable prefix set* of an event at a view A refers to the subset of prefixes in P_S which satisfies the first criterion. Though an event may affect several prefixes, it may still not be visible at a vantage point if the observable prefix set is empty. Of this observable set, a vantage point may receive updates for only a fraction of the prefixes since others may potentially be dampened. We refer to this as the *observed magnitude* of an event at the vantage point.

Two forms of dampening can affect visibility of an event: flap dampening and rate-limiting. Based on data collected from Routeviews and RIPE, we notice two distinct route update patterns for different prefixes at different vantage points. *Stable* prefixes have an update process characterized by long silence periods interspersed with small bursts of updates. *Continuously flapping* prefixes are continuously updated over long periods. Although stable prefixes can potentially be affected by flap damping [15], in the absence of rate-limiting at least one update from the event will always reach the view (which might be a withdrawal triggered by hold-down). One update is sufficient for our algorithm to detect the event, though more updates can improve precision. In this paper, we focus primarily on stable prefixes and briefly discuss continuously flapping prefixes later in Section 7.2. Rate limiting can occur if the rate at which an event generates updates exceeds the rate the router forwards updates. For example, a session reset may cause many routes to change and then revert back to their previous states before the router's advertisement timer expires. Although stable prefixes can be affected by rate limiting, events that cause long-term changes (changes longer than the advertisement timer) can always be seen at vantage points.

Table 1: Effect of distance on observed magnitude.

Number of AS Hops from view	*Number of prefixes affected*
4	967
3	1311
2	920
1	1238

Despite dampening, we found that in practice an event with a high magnitude is visible at vantage points provided the vantage point has a large observable prefix set, as shown in Table 1 (we generated these results after narrowing down the location of the event from multiple vantage points using our algorithms described later). We notice that the observed magnitude of an event is fairly constant regardless of distance (AS hop count) from the view. In general, vantage points which have an observable prefix set of at least 1000 prefixes notice a significant number of them being simultaneously updated by the event. Also, an event can be suppressed if it simultaneously occurs with another event that affects the same prefix set. We found that such simultaneous events are rare. We describe how we can detect them in Appendix B.

In summary, we make the following observation:

Observation I: *An event that affects one or more stable prefixes is visible at a vantage point provided: (a) one or more affected prefixes are present in the observable prefix set of the vantage point; (b) the observations to at least one stable prefix are not dampened by other simultaneous events.*

3.1.2 Event separability

From the perspective of a single vantage point, events are separable across two dimensions: *prefixes* and *time*. Separability across prefixes involves distinguishing between events that affect disjoint sets of prefixes. To maximize accuracy, we can treat a single event which triggers updates to different prefixes as separate events triggering updates to each prefix. While this may affect precision in determining the source of the event, this does *not* affect correctness since we attempt to determine the root cause from updates to each prefix separately. However, inappropriately assuming two observations are correlated can cause incorrect inferences which is undesirable. Hence, we treat updates to each prefix separately and in the simplest case, analyze each prefix in isolation. This approximation is most useful when analyzing events which affect very few prefixes. For example, if an event affects only 2 prefixes, it is easier to analyze them separately rather than attempting to decipher which two prefixes are affected by the same event. Later in Section 3.2, we provide a criterion to determine when one can safely correlate observations across prefixes without sacrificing correctness.

Along the *time* dimension, two events affecting the same prefix are separable at a single vantage point if the updates generated by these events are separated by long silence periods. Given a threshold Δ, we can separate the arrival of route updates into bursts such that two bursts have a silence period of at least Δ between them. Let d refer to the duration of burst and T_m, the mean separation time between bursts. We make the following observation: *the probability that two bursts affecting the same prefix at a vantage point are caused by two separate events is dependent on the ratio d/T_m.* When T_m is larger than d, we can more easily separate events. For very low values of d/T_m, a single observed burst of updates is triggered by a single event with high probability.

Consider the simple scenario where events affecting a single prefix are independent and identically distributed, *i.e.,* arrival of events can be modeled by a Poisson process. In Appendix C, we show that the arrival process for many prefixes can be approximated as Poisson. The inter-arrival times of the events are hence drawn from an exponential distribution with a mean separation time of T_m. If d represents the duration of an event, the probability that one event occurs within a period d of another event is given by e^{-d/T_m}. For very low values of d/T_m, this probability is very small.

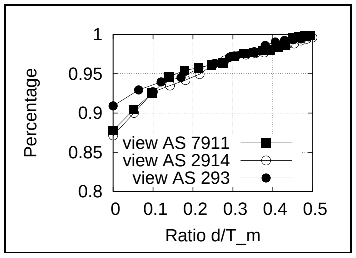


Figure 3: CDF of ratio of duration and separation time.

While the Poisson process assumption represents a crude approximation of the event arrival process, it helps in better analyzing the separability of observations into different events. In general, the event separability at a vantage point is dependent on the parameter d/T_m. In reality, while we cannot directly observe the event process, we estimate a upper bound on this value for stable prefixes as the length of the longest burst observed in a given time period divided by the smallest separation time observed between bursts. Figure 3 plots this distribution as measured for different stable prefixes across 3 different views over a period of 1 day. To estimate these bursts, we set a threshold of $\Delta = 1$ hour as the minimum length of a silence period[5]. For roughly 93% of the stable prefixes, the separation time is 10 times larger than the event duration. We classified the remaining prefixes under the continuously flapping category.

3.2 Quiescent vs. Turbulent periods

In this section, we provide a mechanism to distinguish between Turbulent (periods of high activity) and Quiescent periods (periods of low activity) and show that during Turbulent periods we can safely correlate observations across prefixes.

[5]Route flap dampening is not triggered beyond a time-period of 1 hour due to a single event and hence we chose this as a threshold value.

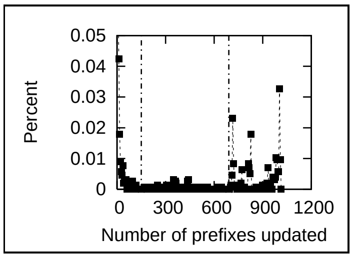


Figure 4: Frequency of intervals with various numbers of updated prefixes for a view in AS 1239. We use the gap (separation shown with vertical dashed lines) to distinguish Turbulent from Quiescent periods. Less than 0.2% of the mass is within the gap.

We define a probability distribution function $p(x)$ on a variable x to have a *gap property* if there exist two values α_1 and α_2 such that:

$$P(\alpha_1 \leq x \leq \alpha_2) = 0$$

In other words, x does not take any value in the range $[\alpha_1, \alpha_2]$. To generalize, a probability distribution nearly has a gap property if $P(\alpha_1 \leq x \leq \alpha_2)$ is negligible in comparison to $P(x < \alpha_1)$ and $P(x > \alpha_2)$.

We use $N(e)$ to denote the number of prefixes updated at a vantage point during a given time-period that traverse an inter-AS link e[6]. From our analysis, we noticed that the probability distribution of $N(e)$ measured over short time-intervals $(1 - 5$ minutes) nearly satisfies the gap property for a large fraction of the links where the gap typically starts at a value $\alpha_1(e) < 50$ and ends in a value $\alpha_2(e)$ much larger than $\alpha_1(e)$. $\alpha_2(e)$ typically varies between $100 - 1000$ depending on the link. Figure 4 shows the PDF of $N(e)$ for one such link with $\alpha_1(e) = 140$ and $\alpha_2(e) = 710$.

$N(e)$ represents the sum of observed magnitudes of various events that affect prefixes traversing the edge e. If we measure $N(e)$ across small time-intervals (the expected number of events within an interval is small), then the existence of a gap in the distribution in $N(e)$ signifies that: *certain events trigger an observed magnitude of at most $\alpha_1(e)$ and others trigger an observed magnitude significantly larger than $\alpha_1(e)$.*

This motivates the distinction between *minor* and *major* events where minor events cumulatively can account for an observed magnitude of at most $\alpha_1(e)$. In other words, if a vantage point observes more than $\alpha_2(e)$ prefixes to be updated along the edge e, then it can determine that at least one event has an observed magnitude of $\alpha_2(e) - \alpha_1(e)$ (since simultaneously occurring minor events can account for at most $\alpha_1(e)$ of the observed updates). This argument relies on the fact that a vantage point rarely observes simultaneous major events. This holds because of two properties (a) over small measurement intervals, we expect very few events to occur, and (b) $P(N(e) > \alpha_2(e))$ is

[6]In practice, we need to include the complete observed magnitude of every burst that occurred within a time-period in the computation of $N(e)$, *i.e.,*, for every burst, we consider all updates of a burst including the ones outside the interval.

small (roughly $3-4\%$).

In summary, a view that observes a gap property in the probability distribution of $N(e)$ between $\alpha_1(e)$ and $\alpha_2(e)$ undergoes a *Turbulent period* if it observes that the value of $N(e)$ is greater than $\alpha_2(e)$ and a *Quiescent period* if $N(e)$ is lesser than $\alpha_1(e)$. Additionally, we make the following observation:

Observation II: *Given the gap property between $\alpha_1(e)$ and $\alpha_2(e)$ on the probability distribution of $N(e)$ of an inter-AS link e, if a view observes more than $\alpha_2(e)$ prefixes involving edge e updated within an interval, then at least $\alpha_2(e)-\alpha_1(e)$ of these prefixes are updated by one major event.*

3.3 Correlation criteria

To summarize, we use the following principles to correlate observations across multiple vantage points.

Time dimension: At a given vantage point, a burst of updates for a single prefix is assumed to be caused by a single event provided the mean-separation time is at least ten times the mean-duration time of a burst[7].

Prefixes dimension: At a vantage point, if the number of prefixes updated in an interval is above a threshold ($\alpha_2(e)$) for some edge, then a majority of prefixes (at least $\alpha_2(e)-\alpha_1(e)$) are updated by a single major event. In Section 5 we use this principle to correlate observations across prefixes.

Views dimension: Let a view observe the start of a burst of updates for a prefix at time t and d represents the mean-duration of the burst for that prefix. In such a case, all bursts that begin for the same prefix across different views within the time $[t, t+d]$ are assumed to be caused by the same routing event provided none of the views are affected by a major event (*i.e.,*, a major event can overshadow the burst of updates to a prefix caused by a simultaneous minor event).

4 Single view, single prefix

In this section, we present an algorithm for inferring the *suspect set* based solely on updates made to a single prefix at a single view. Our inference algorithm is shown in Figure 6, and is motivated by Examples 3 and 4 in Section 1.1. We first partition the possible types of BGP events into *equivalence classes*, where each class contains events that trigger a similar pattern of observations at a vantage point (Section 4.1). We then define a set of rules to associate suspect AS's into these equivalence classes based on the contents of the updates (Section 4.2). This serves two purposes. First, we can gain some information about the event that took place. For example if an equivalence class contains no AS's, then that type of event did not occur. Second, since observations of an event must be consistent across views, we can eliminate AS's that appear in different classes in different views (Section 4.3). For example, if AS X appears in equivalence class A in one view, and class B in another view, the event must not have occurred at AS X.

[7] For more than 90% of the prefixes, the ratio of the times is at least 10. Here, we choose 10 as a simple threshold.

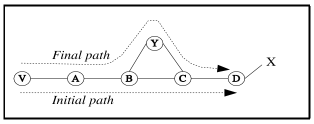


Figure 5: Example: Inference from single view.

Our approach formalizes this by 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 the example shown in Figure 5. Suppose view V's routing table contains an AS path $[V, A, B, C, D]$ to a prefix X, and assume for now that AS's are singly peered and community attribute changes do not occur (our approach does not make these assumptions). 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. 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. In general, there are two possible explanations: either (1) some event happened on the path $[B, C]$ to make it less desirable to B (worsened), or (2) some event happened on the path $[B, Y, C]$ to make it more desirable to B (improved). Hence, we would place $\{B, Y, C\}$ into an *Improve* class, and $\{B, C\}$ into a *Worsen* class. We can then compute the *suspect location set* as $\{B, Y, C\} \cup \{B, C\} = \{B, Y, C\}$, and the *suspect cause set* as all causes corresponding to the Improve and Worsen equivalence classes. Moreover, since an event must belong to a single class regardless of view or prefix, we can intersect these equivalence classes across observations made of the same event, to further improve precision. We show how to do this in later sections.

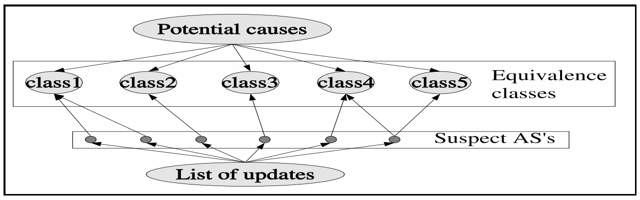


Figure 6: Single-view inference methodology.

4.1 Equivalence classes

We need to satisfy two criteria when defining equivalence classes. We choose equivalence classes to be *disjoint*, i.e. each event appears in at most one class, and *complete*, i.e. every possible event is contained at least one class. Disjointness eliminates overlap among classes, thereby maximizing the precision gain from intersecting across views and prefixes. Completeness ensures we can model every possible event. We define two pairs of equivalence classes:

Improve vs. worsen: 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, if we observe a path change from P_1 to P_2, we place P_1's AS's into the worsen class, and P_2's AS's into the improve class.

Hard vs. soft: 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, observing a convergence process followed by a withdrawal would imply a hard event occurred. A route change from P_1 to P_2 could occur either due to hard events or soft events, since although a withdrawal is not received by the view, some intermediate router on P_1 could have generated a withdrawal that triggered the change.

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
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 (N)	Community attribute changed	Community change

By definition, each pair of classes is disjoint. We hence define four disjoint equivalence classes corresponding to each possible combination of classes, as shown in the first four rows of Table 2. 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. Moreover, each pair of equivalence classes is complete: by definition, an event can only either improve or worsen a route, and it can only be either hard or soft. 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 $H_W + S_W$.

Observation III: *The resulting classes H_I, H_W, S_I, S_W are both complete and disjoint.*

In all equivalence classes discussed so far, the route change occurs in the same AS that undergoes the event. However, a change of the **community**[8] attribute is unique in that it can cause a route change to be triggered several hops away from the AS that undergoes the event. This is a clear signature that can be used to distinguish these events from other events. Although community attribute changes are contained in the H_I, H_W, S_I, S_W, defining a richer set of equivalence classes improves precision. Hence, we add another equivalence class to contain these events, and redefine the existing classes to exclude community attribute change events.

[8]The community attribute is a variable length string, and routers can customize reaction to this string according to policy.

Note that our approach is not sensitive to the specific mapping of causes to equivalence classes, since it is difficult to determine such a mapping completely. Instead, our approach reports location and the equivalence class associated with that location. Each class can then be associated with a list of causes, by using the mapping.

4.2 Rules for singly peered AS's

In this section, we describe the set of rules for mapping AS's into equivalence classes based on the pattern of route updates observed at a view. For simplicity, we first describe the rules assuming there is a single peering link between adjacent AS's, then describe how to eliminate this assumption in the following section.

Table 3: Enumeration of prefix state transitions (bursts).

Name	***Freq.***	***Regular expression***
REROUTE	12.6%	$A_1\{A\|W\}^*A_2$
PREFIXUP	2.3%	$W\{A\|W\}^*A$
PREFIXDOWN	3.7%	$A\{A\|W\}^*W$
FLAPUP	2.3%	$W\{A\|W\}^*A\{A\|W\}^*W$
PATHFLAP	50.2%	$A_1\{A\|W\}^*A_2\{A\|W\}^*A_1$
FLAPDOWN	28.8%	$A_1\{A\|W\}^*W\{A\|W\}^*A_1$

Patterns of observations: Table 3 enumerates six all the different patterns of route updates that can be observed for a single prefix. A pattern consists of a starting state, zero or more intermediate states, and a final state. For example, a REROUTE starts in an advertised state (A_1) and terminates in an advertised state with a different AS path (A_2), and may experience some intermediate announcements and withdrawals. Either the initial and final states differ (first three rows) or they are the same (last three rows). For each case, there are three possibilities corresponding to whether the prefix transitions to a withdrawn (W) or advertised (A) state.

Observation IV: *The 6 state transitions in Table 3 comprise the set of all possible state changes.*

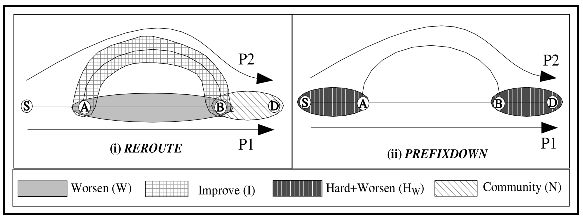


Figure 7: *Single view rules*: The event must have occurred in the shaded regions.

For each of the six patterns of observations, we use a set of inference rules to map the AS's into the five equivalence classes (H_I, H_W, S_I, S_W, N). Due to space limitations, we describe in detail only the two inference rules in Figure 7, the rest are in Appendix A. These rules are constructed under the assumption that AS's are singly-peered (we address multiply peered AS's in the next section).

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.

PREFIXDOWN: A PREFIXDOWN observation at a view refers to the case where the view with an initial state of path P_1 for a given prefix observes intermediary route updates with paths $(P_2, P_3 \ldots P_N)$ before the route becomes withdrawn for a sustained period of time. Given this pattern, we make two observations. First, the event that triggered this pattern is with certainty in the worsen and hard classes. We can learn that before the event occurred, at least one route to the destination was working, while after the event, none of the routes work. Second, if all AS's were singly peered, the location of the event must have occurred in the intersection of the paths explored. In other words, all AS's in $\bigcap_{i=1}^{n} P_i$ would be classified in the H_W equivalence class.

4.3 Refinements for multiply-peered AS's

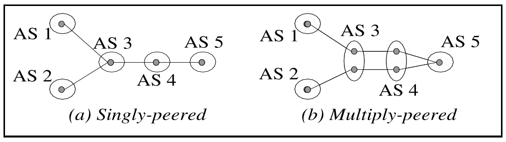


Figure 8: Knowing the details of peering relationships can help infer more precisely.

AS's can be connected by multiple peering sessions, which can complicate inference. An example is shown in Figure 8. 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 throughout the event, and assume for simplicity that community attribute changes do not occur. If AS 3 and AS 4 are singly peered, then the event causing the update must have occurred in AS 3, AS 1, or the peering link between them. However, 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.

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 all the way upstream to S, and all the way downstream to D. Knowing how many times a pair of AS's are peered could improve precision, as it is only necessary to extend these sets as far as the first place where the paths must have traversed the same router. Although active measurements can sometimes determine the number of peering links between AS's [20], we safely assume no such information is available when collecting results.

4.4 Refinement across views

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

Identifying related observations: Bursts of updates to the same prefix received nearly simultaneously at multiple views are likely to be correlated. We describe how to correlate observations across views in Section 3.

Intersection of suspect sets: Since an event must belong to a single equivalence class regardless of view, we can intersect equivalence classes across views to improve precision. 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 each view that observed the event. For example, if view 1 determines the suspect AS's are $H_I = A, B$ and $H_W = B, C$, and view 2 determines $H_I = A, C$ and $S_W = B, C$, then the event must be in the Hard+Improve class and must have occurred at A.

5 Refinement across prefixes

In the previous section, we showed how to find the suspect set for a single prefix at a single view. Here, we show how to narrow down the suspect set by correlating observations across prefixes when Turbulent periods are observed to determine the location and type of *major event* that triggered these updates.

Our algorithm during Turbulent periods consists of four basic steps. First, we analyze each prefix in isolation and classify AS's into equivalence classes using the algorithm in Section 4. Second, to detect that a turbulent period occurred, our algorithm uses the procedure described in Section 3 but, in practice, computes separate values $N(e)$ and $\alpha_2(e)$ (parameters defined in Section 3) for each equivalence class. Therefore, associated with each equivalence class X, we obtain a graph G_X where each edge e has a weight $N(e)$ which represents the number of prefixes for which the edge appeared in X. Third, we run our Turbulent inference algorithm (as described in Section 5.1) on the graph G_X separately for each equivalence class X to determine the potential location of each the major event that triggered the updates. Finally, for each equivalence class X, we intersect our inferences across different views that also observed a major event during the same time interval.

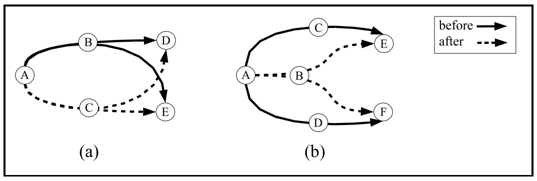


Figure 9: Example: Effects of a MED decrease on a single view.

Given the disjoint property of the equivalence classes, we can perform Turbulent inference within each class separately. We motivate this using a simple example from [21]. Suppose A

uses B to reach D and E, as shown in Figure 9(a). Suppose many prefixes simultaneously change to start using C. If we assume simultaneous events are rare, there are two possible causes: either an event took place on (A, B) that worsened the properties of the path, or an event took place on (A, C) that improved the properties of the path. A second example is shown in Figure 9(b). Suppose many prefixes simultaneously change to start using (A, B). Since it is unlikely two simultaneous major events occurred on both $[A, C, E]$ and $[A, D, F]$, it is likely that a single major event either (a) took place at (A, B) that improved the properties of the path, or (b) took place internally in A that worsened the paths of several prefixes using (A, C, E) and (A, D, F) (since A is the only common AS across the sub-paths). In both these examples, we can separate AS's into *improve* and *worsen* categories and perform Turbulent inference on each class independently.

5.1 Turbulent inference algorithm

In this section, we provide an inference algorithm assuming that a single major event occurred, and later in Appendix B, provide heuristics on how one can potentially differentiate simultaneous major events. While simultaneous major events are known to occur in practice[9], we assume that the probability of such an occurrence is small.

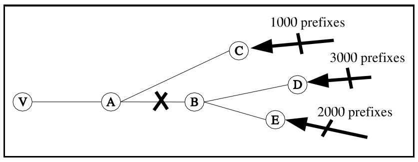


Figure 10: Example: Effects of a session failure on a single view.

The key step of our algorithm is to determine, from a single view, the potential set of locations that might have triggered a major event. We motivate this inference step using an example (Figure 10). Suppose at a view in AS 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 (V, A, C) or downstream of C. We have to include the link (V, A) as a suspect because AS V might be multiply peered with A where one of the links undergoes a session reset and the path to B traverses a different peering link. On the other hand, suppose we observed updates to 4000 prefixes that traverse (A, B), and of these prefixes, 2000 traverse (B, E) and 2000 traverse (B, D). Then, the event likely took place on the sub-path (V, A, B) but not beyond. In other words, *if many updated prefixes share a common path, and then fan out, the event most likely occurred in the common path*.

Our algorithm computes these common sub-paths for each view, and for each equivalence class and then intersects these sub-paths across views to narrow down the location further. This is done using a similar approach to that in Section 4.4. We now provide the details of our algorithm.

5.1.1 *Marking links*

We construct a graph G with 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 $\alpha_2(e)$. We do this because for these links $N(e)$ would always be less than $\alpha_2(e)$, 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 $\alpha_2(e)$ as **M** (for "many"), and all other links as **F** (for "few"). Similarly, we mark a node as **M** or **F** based on the number of prefixes terminating at that node that were updated. Edges marked as M are suspect for having undergone a Turbulent event. To set $\alpha_2(e)$, we use the procedure discussed in Section 3.2.

5.1.2 *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. We repeat this procedure for each equivalence class, and output the resulting set of links. The rules, as shown in Figure 11, are:

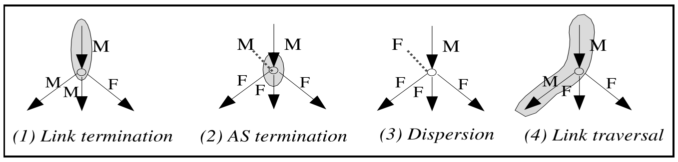


Figure 11: Rules to determine location of a Turbulent event from observations at a single view.

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.

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.

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

[9]Two such cases are: (a) Internet worms like SQL Slammer and Code Red [27] cause increased levels of congestion generating simultaneous session resets, and (b) one session reset triggers another second session reset due to shift in traffic.

link was the aggregation of many individual Quiescent events, and was unlikely to be a single major event. Hence, we assume that no major event occurred. We found this rule was rarely triggered, validating the observations in Section 3.2.

Link traversal: If there is a single link marked M coming out of the node, then the event could have occurred at the link coming into this node or downstream. In this case, we add both the incoming and outgoing edges into the suspect set. We then traverse to the child, and try to apply these rules from that position.

Observation V: *If only a single major event occurs, then the algorithm infers the correct link undergoing the event with high probability.*

The above rules may give incorrect inferences during the rare times when simultaneous events are taking place. We describe some heuristics to detect when simultaneous events are occurring in Appendix B.

6 Methodology and Validation

In this section we develop a methodology for validating correctness. 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 [18] of the updates that are received by the monitor, since they are not used in our algorithm. One particular problem with Routeviews and RIPE data 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, 2], 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: First, as a sanity check, we show that our algorithm detects increased numbers of major events 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 event from multiple viewpoints. This provides additional confidence that our algorithm can correctly pinpoint the location of major events. Finally, we show that we can detect some known major events triggering session resets documented in the NANOG mailing list [29] 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.

Validation in Quiescent periods: Minor events that trigger updates during Quiescent periods are typically not documented, making validation difficult. 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.

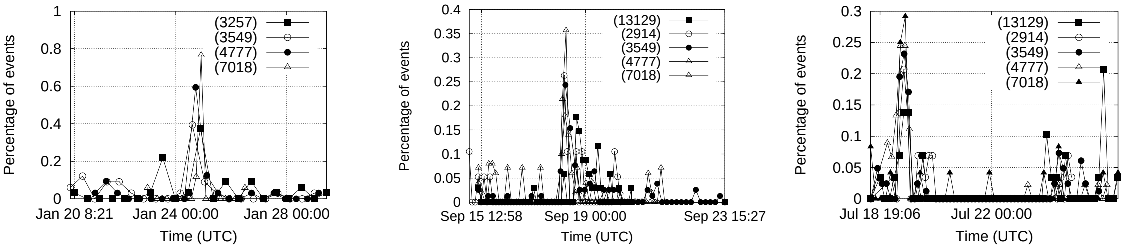
BGP Beacons: BGP beacons are publicly documented prefixes that are advertised and withdrawn at regular intervals, and have been previously used to study BGP routing convergence [14]. It is highly likely that if any view observes an update to a beacon prefix with origin AS X, the update originated in AS X. We used both the RIPE beacons [26] and PSG beacons [14] 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 validate.

Viewpoint at the origin: We use the simple observation that if a view in some 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[10]. 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 AS X was not included in the suspect set.

6.1 Validation during Turbulent periods

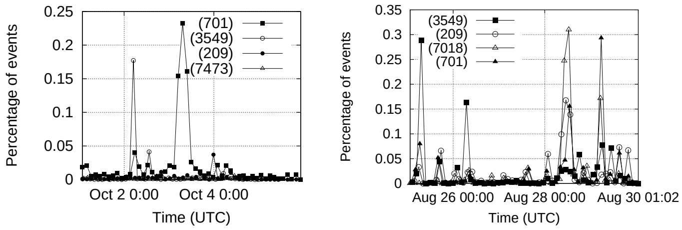
Figure 12a shows the effect of the SQL slammer worm [27] on peering sessions. We observe several reset-like events (events that caused many prefixes to flap) during the SQL Slammer worm period. On the other hand, we observe very few occasions with multiple reset-like events during a normal observation period. This validates our approach, since many session resets were known to occur during the spread of the SQL Slammer worm [23]. Figures 12b and 12c 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.

[10] 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.



(a) (b) (c)

Figure 12: (a) Effect of the SQL slammer worm on peering sessions. Chart shows the PDF of the number of 15 minute intervals containing multiple major events. We filter out updates due to local resets. (b) Effect of NIMDA worm. (c) Effect of Code Red worm.



(a) (b)

Figure 13: 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.

Next, we cross-validate our Turbulent inferences at one viewpoint by attempting to detect the same events from other viewpoints. In almost all cases that couldn't be verified from multiple vantage points, the event occurred in a region of the topology that was only visible from a single vantage point. In addition, we were sometimes able to observe a reset-like event affecting paths traversing both directions on a link.

Table 4: Several known events used for validation.

Event	***Date***	***Where documented***
AT&T routing problems	8/28/02	NANOG archives [29]
UUNET routing problems	10/3/02	NANOG archives [29]
Peering link instability	7/21/03	Sprint web site [30]
WorldCom peering problems	11/11/02	web site [27]

Finally, we validated our technique by running our algorithm on traces collected during periods where we knew the exact location of some major 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 13a and 13b 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 Validation during Quiescent periods

For each type of Quiescent validation, we ignore updates from views that are in the same AS as the destination prefix. We do this to avoid providing additional information to the Quiescent inference algorithm when we use the other views alone. If we had included these views in our analysis, the Quiescent inference algorithm would always be able to correctly pinpoint the origin AS as the only suspect.

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 4.1, as well as the rules used to refine across views from Section 4.4. 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 during Quiescent periods 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 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. Hence this approach validates the rules described in Sections 4.1 and 4.4. We used traces from 23 views collected over 10 days to validate. For each view, 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, or were flaps which tend to introduce many AS's as suspects. We describe precision further in the next section.

6.2.1 Precision

In Quiescent periods, we perform inference over fewer updates than during Turbulent periods. Hence, the precision to which we can localize the cause is inherently reduced. However, we found that it is still possible to achieve good precision, as shown in Figure 14. 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. Then, we applied our scheme to data collected at a single view (Figure 14b) and across 23 views (Figure 14c. We make several observations from our analysis. First, compared to the naive approach, we can reduce the suspect set drastically, size, by a factor of $3-4$ on average. Next, 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 [3]. Also, for certain classes of events such as origin withdrawals, we can reduce the suspect set even further.

7 Analysis and Observations

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.

Although we were unable to pinpoint the cause for any update, we were able to narrow down the cause into a several categories. Figure 15 gives the number of updates for each type of event. We found that we could pinpoint the location of virtually all major events we detected, and could narrow down the location of Quiescent events to within 2 AS's for 20% of updates. We found the majority of continuously flapping prefixes could be pinpointed to a single AS, as shown in Section 7.2. Overall, we could pinpoint the location to a single inter-AS link (pair of AS's) for 70% of updates.

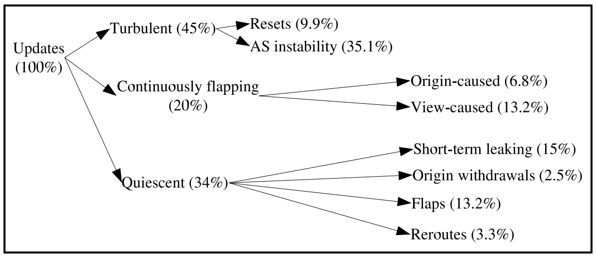


Figure 15: Breakdown of updates by cause.

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 AS 1239 and AS 701 underwent a large number of session-reset like events, affecting the reachability of over 20,000 prefixes. 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 [30] 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, 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. This event affected 5 of the 50 most popular web sites [28].

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.

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. The instability was short-lived, lasting about 15 minutes.

Overall, we found session reset-like events are a common occurrence. We measured on average 1,400 session reset-like events per month. We found a small number of inter-AS links are perennially unstable. For example, during January 2002, a link to a Chinese ISP underwent on average 2 session reset-like events per day for a period of two weeks. These events affected reachability to over 1000 prefixes in China.

7.2 Continuously flapping events

In this section we discuss some initial analysis of continuously flapping prefixes. A detailed analysis is ongoing work. In our

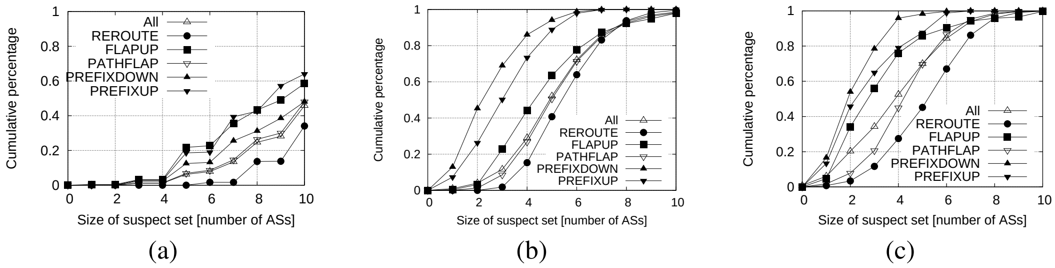


Figure 14: *Precision:* We plot the CDF of the suspect location set size for three cases: (a) naive approach (b) our approach, using observations only at a single view (c) our approach using observations from all views collected over 10 days. For example, the FLAPUP curve passes through the point (4,0.45) in case (b). This means that in 45% of FLAPUP observations, we can reduce the suspect set to within 4 AS's.

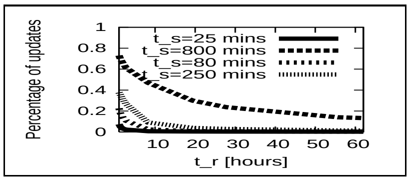


Figure 16: Percent of updates caused by flapping prefixes.

analysis we require a prefix to be stable for a period longer than $T_m = 1$ hour before performing inference on its updates. A surprisingly large portion of prefixes (25%) too unstable to perform inference. Although most traffic tends to visit stable prefixes [19], unstable prefixes are important to study as they place additional load on routers.

We define a continuously flapping prefix as a prefix that is persistently updated for a period longer than t_r, without experiencing a *silent period* longer than t_s. Figure 16 shows the relative percentage of updates due to continuously flapping prefixes from traces collected over a 3 month period. We can see if $t_s = \delta_{sil} = 1$ hour that these prefixes cause 20% of updates. We found that, counter-intuitively, continuously flapping prefixes cause fewer updates than stable prefixes. Although stable prefixes have longer silence periods, they receive update bursts that tend to contain many updates grouped together. Finally, there are a small set of prefixes that flap for a very long time. 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.

We restrict ourselves to performing a simple characterization of these events. 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 was caused close to the 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. Most continuous flaps we observed were near-view.

8 Related work

We classify related work into three categories:

Passive root cause analysis attempts to determine the location and type of a routing event based on BGP updates alone. Our work falls into this category and extends previous works by Di Fa Chang *et al.* [2] and Lad *et al.* [11] along three dimensions. First, we make a distinction between stable and continuously flapping prefixes and establish a criterion of when one can safely correlate updates across different stable prefixes without sacrificing correctness of root-cause inference. Our approach leverages some of the clustering strategies for determining groups of correlated updates [2, 11], but limits this clustering only to the cases where the safety criterion is met. For example, recent work by Teixeira *et al.* [21], shows specific examples where correlation across prefixes can potentially lead to incorrect inferences and we believe that our work is resilient to the concerns raised in this paper. In cases where we do not meet the safety criterion, we do not correlate across prefixes and analyze each in isolation thereby sacrificing precision for correctness. This said, our basic approach for analyzing the graph structure from a view and assigning weights to edges has similarities to Link-rank [11] and the clustering approaches in [2]. The second differentiating aspect of our work is the concept of *equivalence classes* to distinguish between different types of causes of an event. While there is definite scope for improvement, the concept of equivalence classes is fundamentally necessary since many types of causes may be indistinguishable merely based on observations. Finally, our inference methodology incorporates two additional constraints not addressed in previous works which make the root-cause analysis problem harder: (a) AS's have multiple peering links where this number is unknown; (b) BGP community attribute changes can cause upstream providers to change routes. While these constraints tend to increase the suspect set, our methodology relies on using multiple views to improve precision.

Active root cause analysis can be combined with passive techniques to improve precision. Recently Feldmann *et al.* [12] [6] [11] proposed a scheme to model BGP and locate instabilities using active and passive measurements. Traceroutes have been used to discover ISP topologies [20] to a reasonable

[11] This paper was recently accepted for publication and not yet publicly available.

degree of accuracy, and can be used in conjunction with BGP updates to produce more accurate IP to AS mappings [16]. BGP updates can be used in conjunction with UDP probes and traceroutes to find the location of failures within an ISP [3], or to produce more accurate IP to AS mappings. We believe that active techniques could improve precision of our approach.

Modeling BGP dynamics can help in inferring root cause, but is difficult to do completely. Many types of events can cause the same stream of updates, and many possible streams of updates can come from the same event [10, 7], making inference difficult. Many factors can influence the times when updates are received [14] However, the update traffic observed at a router can be characterized as bursts of activity interspersed with silent periods [12], and this observation can be used to localize the event location [2]. Another complicating factor is that it is difficult to predict the final result of a routing change [4]. However, the effect of external learned via BGP on the flow of traffic through an AS can be computed [5].

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 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: e.g., a small number of links cause the majority of updates. In addition, we were able to discover several major events that were not public knowledge.

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] 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.

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

[3] 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.

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

[5] N. Feamster, J. Winick, J. Rexford, "A model of BGP routing for network engineering," in ACM SIGMETRICS, New York, NY, June 2004.

[6] A. Feldmann, O. Maennel, Z. Mao, A. Berger, B. Maggs, "Locating internet routing instabilities," in *Proc. of SIGCOMM*, Portland, Oregon, August 2004.

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

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

[9] 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.

[10] 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.

[11] M. Lad, L. Zhang, D. Massey, "Link-Rank: A Graphical Tool for Capturing BGP Routing Dynamics," in *IEEE/IFIP NOMS*, Seoul, Korea, April 2004.

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

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

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

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

[16] Z. Mao, J. Rexford, J. Wang, R. Katz, "Towards an accurate AS-level traceroute tool," in *Proc. of SIGCOMM*, Karlsruhe, Germany, August 2003.

[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] R. Teixeira, J. Rexford, "A measurement framework for pinpointing routing changes", under submission.

[22] F. Wang, L. Gao, "On inferring and characterizing internet routing policies," in *IMC*, Miami, Florida, October 2003.

[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 RIS," `http://www.ripe.net/ris/`.

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

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

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

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

A Appendix: Quiescent algorithm details

In this section, we first describe in more detail the steps used to perform inference during Quiescent periods. We then describe the complete set of rules used to determine the suspect set from each received burst of updates.

A.1 Algorithm

This section describes the procedure used to perform Quiescent inference. The output of our algorithm is a set-tuple of the form
$\{(c_1, S_{c_1}), (c_2, S_{c_2}) \ldots (c_n, S_{c_n})\}$ where each equivalence class $c_i \in C_q$ represents a list of potential causes and S_{c_i} represents the corresponding location set of AS's that might have triggered the event. In other words, if we knew that the cause of an event triggering a set of updates is in c_i then the AS triggering in the event is present in the set S_{c_i}. In practice we do not know the cause, but expressing the output in this form allows us to determine where various types of events could have occurred. We can then compute the suspect list by unioning the locations and causes contained in the list. The pseudocode for the algorithm is shown in Algorithm 1.

Algorithm 1 QuiescentInfer ($Time = t$, $Prefix = P$)

```
1:  R_P(v,t) = correlated group of updates for a prefix P observed
    from view v in a neighborhood around time t
2:  C_q = Set of equivalence classes of causes = {
    H_W, S_W, H_I, S_I, N, M, U }
3:  V = Set of all views
4:  L = {}
5:  for each c ∈ C_q
6:     for each view v ∈ V
7:        Sus(c,v)= Suspect-location-set inferred using R_P(v,t).
8:     S_c = ∩_{v∈V} Sus(c,v).
9:     if S_c is not empty then L = L ∪ {(c, S_c)}.
10: return L
```

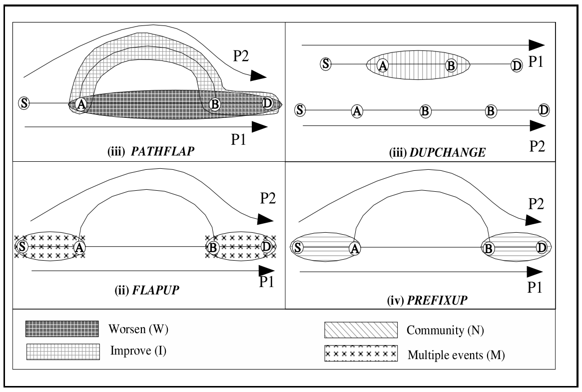


Figure 17: Additional single view rules.

A.2 Additional inference rules

In addition to the rules discussed in Section 4.2, we use the following rules to compute the suspect set.

- **PREFIXUP:** This rule is defined similarly to PREFIXDOWN, but in this case, we know that some element must have been worsened. So, if we observe an advertised path become withdrawn, with intermediate paths $\{P_1 ... P_n\}$ before finally converging a withdrawn state, then $I = \bigcap_{i=1}^{n} P_i$, $M = \emptyset$, $W = \emptyset$.

- **DUPCHANGE:** Suppose we observe a burst that contains paths P_1 and P_2 such that P_2 occurs later than P_1, and the number of duplicated hops in the AS path changed for some AS i. Assuming AS's are connected by single peering links, then the event occurred at either AS i or at the AS j immediately upstream from i. Hence we set $I = \{i, j\}$, $W = \emptyset$, $M = \emptyset$ if the number of copies decreases, $I = \emptyset$, $W = \{i, j\}$, $M = \emptyset$ otherwise. If there could be more than one peering link between i and j, this rule doesn't help and hence we attempt to apply some other matching rule to the burst.

- **PATHFLAP:** A PATHFLAP indicates one of two things: either the previous path temporarily worsened, or an intermediate path temporarily improved. Similar to REROUTE, it is possible that either $I = P_2 - (P_1 \cap P_2)$ improved or $W = P_1 - (P_1 \cap P_2)$ worsened. What is different is that due to delayed convergence, some element as far downstream as the origin could have caused the event to occur. For example, the origin AS could have temporarily withdrawn and readvertised the prefix. Hence, we extend the sets calculated in REROUTE downstream so as to include the path between AS B and AS D as shown in Figure 7-iii.

- **FLAPUP:** This burst consists of a PREFIXUP followed by a PREFIXDOWN. Hence if we observe a withdrawn route become advertised with intermediate paths $\{P_1 ... P_n\}$ then we set $M = \bigcap_{i=1}^{n} P_i$, $I = \emptyset$, $W = \emptyset$.

B Appendix: Heuristics for simultaneous events

B.1 Quiescent periods

Table 7: Observations caused by simultaneous events.

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

The algorithm given above assumes that only a single event affecting a particular prefix occurs at a time. 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 [4]), or a congestion-related session reset arising from traffic shifted by the original route advertisement. Or, two *independent events* can occur at the same time. However, in certain cases we can detect these simultaneous events. In particular, certain combinations of observations across views can only be caused by simultaneously occurring events, as shown in Table 7. 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 7 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.

B.2 Turbulent periods

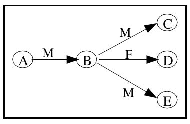


Figure 18: Detecting simultaneously occurring events.

The rule described in Figure 11a 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 events. However, we can use two techniques to sometimes detect when simultaneous events are occurring. First, we can associate 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 events. For example in Figure 18, 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 18, 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.

Table 8: Cross correlation between events.

Burst separation (N)	View AS 2914	View AS 3356
1	0.000332	0.000740
5	0.000635	0.000661
9	-0.000056	0.001742
13	-0.000049	0.000154
17	0.000110	0.000226

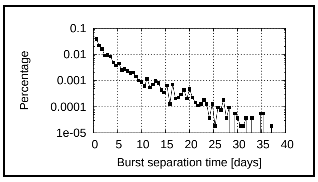


Figure 19: Example plot of burst inter-arrival times.

C Appendix: Event modeling

In this section, we show that the arrival process for events affecting a stable prefix can be approximately modeled as a Poisson process. While this approximation may not hold for certain specific prefixes like BGP beacons where the destination AS triggers updates at specified intervals, we notice this approximation to hold for a large fraction of prefixes, we sampled and analyzed. The intuition behind this modeling is that we expect the arrival time of an event affecting a stable prefix to be independent of previous events occurring later especially since the mean-separation time between bursts of updates is at least 1 hour (in many cases more than $5 - 6$ hours).

To show this hypothesis, we consider the arrival times of burst of updates to a prefix and consider each such burst to be triggered by a single event. We empirically show two properties about these inter-arrival times. First, we show that *inter-arrival times between bursts can be modeled using an exponential distribution.* In Figure 19, we illustrate a sample probability distribution of the inter-arrival times of one such prefix. Second, we show that *event arrival times are uncorrelated with each other*. For a given burst separation, N, we measure the cross-correlation between two events that are separated by $N - 1$ bursts and determine this value to be very close to zero for various values of N. Specifically, we show that for $N = 1$, the correlation is close to zero implying that the arrival times of adjacent bursts are uncorrelated. A sample set of cross correlation factors is shown in Table 8 for two separate views for different values of N. These two properties show that the underlying arrival process can be modeled as a Poisson process since we have established that the inter-arrival times are independent and exponentially distributed.