Back Out: End-to-end Inference of Common Points-of-Failure in the Internet (extended)

USC/ISI Technical Report ISI-TR-724

February 2018

John Heidemann
Yuri Pradkin
Aqib Nisar
University of Southern California / Information Sciences Institute

ABSTRACT

Internet reliability has many potential weaknesses: fiber rights-of-way at the physical layer, exchange-point congestion from DDOS at the network layer, settlement disputes between organizations at the financial layer, and government intervention the political layer. This paper shows that we can *discover common points-of-failure* at *any* of these layers by observing correlated failures. We use *end-to-end* observations from data-plane-level connectivity of edge hosts in the Internet. We identify *correlations in connectivity:* networks that usually fail and recover at the same time suggest common point-of-failure. We define two new algorithms to meet these goals. First, we define a computationally-efficient algorithm to create a *linear ordering* of blocks to make correlated failures apparent to a human analyst. Second, we develop an *event-based clustering* algorithm that directly networks with correlated failures, suggesting common points-of-failure. Our algorithms scale to real-world datasets of millions of networks and observations: linear ordering is $O(n \log n)$ time and event-based clustering parallelizes with Map/Reduce. We demonstrate them on three months of outages for 4 million /24 network prefixes, showing high recall (0.83 to 0.98) and precision (0.72 to 1.0) for blocks that respond. We also show that our algorithms generalize to identify correlations in anycast catchments and routing.

ISI-TR-724, Feb. 2018,

ACM Reference Format:
John Heidemann, Yuri Pradkin, and Aqib Nisar . 2018. Back Out: End-to-end Inference of Common Points-of-Failure in the Internet (extended) : USC/ISI Technical Report ISI-TR-724 February 2018 . Marina del Rey, California, USA, 17 pages.

1 INTRODUCTION

The Internet was designed to survive the face of component failure [13]. As a primary consideration in its design, this goal motivated routing algorithms that are *independent and distributed* across many organizations, with support for backup paths and load balancing (for example, see [9]), Because policies are distributed and often private, understanding the *actual* reliability the Internet achieves is difficult. Yet understanding reliability is important to many stakeholders, from companies that operate and depend on the Internet, to governments and policy makers who wish to foster development, oversight, and the role of regulation.

Network topology is an important component to understand Internet reliability, since loss of a link can disconnect or overload part of the network. Many groups have studied the Internet topology [1, 12, 21, 24, 27, 32, 43], exposing some of its rich complexity, particularly when long duration, careful observations are employed. *Completeness* is a challenge common to all evaluations of Internet topology. Because the Internet is distributed and private peerings are common, even the largest ISPs can directly observe only a small fraction of the Internet topology. BGP reveals a great deal about connectivity at one layer, but BGP-level AS-paths abstract away router-level connectivity, emphasizing business relationships [9, 22, 26, 32]. After nearly 20 years of router-level topology discovery with tools such as traceroute, and

extensive studies of AS-path topologies, it remains challenging to provide strong estimates of completeness in what is discovered [1]. A major source of incompleteness in the router-level topology is that active probing shows only the currently preferred path; backup paths are hidden until a partial failure makes them preferred [32].

Underneath the network-level topology of the Internet is an actual physical topology of optical fibers and other wired and wireless media. Although also difficult to study, recent work has assembled maps of the Internet's physical topology [16], revealing how physical bottlenecks can be common points-of-failure, even when paths appear independent at the network layer. Strikingly, these common vulnerabilities may not be obvious to ISPs, since they operate independently and such information is often considered proprietary. As one example, awareness of a critical tunnel between New York and New Jersey only emerged after a fire [5].

Business and politics loom over the network layer, and although not "technical", they have affected network traffic and reachability on multiple occasions. Business disputes over television programming [44] and traffic settlements [30] have resulted in de-peering and heavy congestion at exchanges. Such congestion can affect third parties, prompting multiple approaches to document these results [30, 46]. Governments have suspended Internet or mobile phone access out of concern for public safety [8], or to suppress dissent or news [14, 15, 20].

The goal of this paper is to show that *end-to-end techniques can systematically identify common points-of-failure across these many layers* of the Internet. By *end-to-end*, we mean techniques that are oblivious to the characteristics of a specific network layer. A complete network or physical map, and models of the business or political layers seem impossible, so we instead observe connectivity to hosts at the edge of the Internet across *all* layers. Our goal is to find *common points-of-failure*, single components than can affect specific parts of the Internet. Our insight is that *correlated failures* in end-to-end measurements can reveal these common points-of-failure—the *consistency* of failures over time allows us to distinguish likely underlying commonalities from the noise of random co-occurring failures. We call our approach *Back Out*, since infers backwards from outages.

Our algorithms consume data from end-to-end outage measurements, routing, or anycast catchment detection. For outages, we use three months of data from Trinocular [34]; covering about 4M /24 network prefixes (blocks), each with about 24k observations per block. We chose Trinocular because it is broad and relatively unbiased (it covers all networks that respond to enough pings) and its data is available at no cost to researchers [49]. In principle we could use other sources of outages such as iPlane [25], Pingin' [41], Disco [42], background radiation [15]; see §6. For routing and anycast (§5), we use data from RouteViews [50] and RIPE Atlas [39].

Our approach is *blind*, operating independent of domain knowledge specific to any layer. This ability to cut across all layers means it can identify common points-of-failure at *any* layer, even those unknown to domain experts. This ability is important because faults occur not just in the physical and network layers, where they have been subject to extensive technical study, but also in the more elusive business and political layers. Our approach does not directly identify root causes of common points-of-failure, but instead seeks to discover of previously unsuspected commonalities (see §4.5 and §5). Such discoveries may then focus future studies of the physical, network, and business layers that are actual root causes, either by re-focusing existing approaches or motivating new ones.

A limitation common to approaches that employ active measurement is that their coverage is incomplete. Our approach discovers common points-of-failure as a result of failures that occur, and not latent dependencies that have yet to manifest themselves. Prior studies of routing show that long observation with repeated measurements are important to reveal latent backup paths [32]; our approach shares this requirement. Measuring completeness is difficult given challenge getting ground truth. However, the 2014 Time Warner outage provides a large, natural experiment to test our ideas—we find very good recall (0.83 and 0.98) and precision (from 0.72 to 1.0) for the what is measurable, and recall is still moderate to good (0.62 to 0.92) even if we count blocks that never respond (§4.3),

The primary contribution of this paper is to develop new algorithms that reveal correlated failures from end-to-end outage measurements. First we show a new algorithm for efficient *linear ordering* to support visualization of outages (§2). This algorithm scales to work on millions of blocks over months of observations, running with $O(n \log n)$ time (for n blocks) in tens of minutes (§2). Second, in §3 we show a new algorithm that *clusters events*, using correlated failures to identify network blocks that share a common point-of-failure. This algorithm is $O(n^2)$ in runtime, where n is the number of blocks with concurrent failures. The algorithm parallelizes well with Map/Reduce, and we are able to process three months of outage data for 4M blocks in a few days on a moderate size (220 core) Hadoop cluster.

Our final contribution is identify several applications of these algorithms. First, our algorithms simplify analysis of huge network datasets (millions of networks by

thousands of observations) by identifying clusters with similar behavior. We show that they succeed at discovering missing information about dependencies (§4 and §5), through a study of the third quarter, 2014, and the August 2014 Time-Warner outage. Second, by identifying the scope of actual outages, it provides information to help identify the spatial scales that one must use to study network outages. Prior work has studied individual addresses [41], /24 network blocks [34], routable prefixes [25] or ASes [42], and country-level regions [15], and while larger regions may be needed to detect weaker signals, our work provides data to suggest reasonable minimum spatial sizes worth study.

We study our algorithms with publicly available datasets (§4.1). We will release our analysis software and make derived datasets available before this work is published. Our work has been reviewed by our Institutional Review Board (USC IIR00001648) and categorized as non-human subjects research.

2 EFFICIENT LINEAR ORDERING

We first consider how to place blocks in a *linear ordering* to support visualization. Our goal here is to take outage timeseries for a set of blocks and arrange them to emphasize similarities between blocks. The outcome of this algorithm is an image that summarizes the status of the network, allowing humans to identify trends visually. These visualizations support iteration with our event clustering algorithm (§3)—it discovers potential common points-of-failure, while visualization provides context for investigation into root causes.

2.1 The Problem and Prior Approaches

Two-dimensional visualizations showing network blocks against time have long been used to identify shared events [18, 28, 48]. The goal of optimizing the visualization to emphasize similarity was formally defined only recently [35]. Their problem takes as input b blocks, each with a timeseries $S_b(i)$ giving the status of block b at time i. Status are binary valued, with 1 and 0 indicating the network is or is not reachable, or the occurrence of a routing change.

The desired output is to assign blocks some linear ordering (position P_b for each block b), while maximizing the "similarity" of adjacent blocks. The definition of similarity is chosen to reveal correlations in the data. This goal is subjective (we know it when we see it), but for a given algorithm it can be quantified. Quan's algorithm defines similarity as the Hamming distance between each block's timeseries (that is, the sum of the exclusive-or of each observation). They evaluate clustering on outages and route changes, with clustering revealing events related to natural disasters and ISP changes.

block (b)	**status** ($S_b(\cdot)$)				**distance**
b1	1111	111**0**	1111	1111	2
b2	1111	1111	1111	111**0**	3
b3	1111	1**00**	1111	1111	1
b4	1111	1**00**	**0**111	1111	2
b5	1111	111**0**	1111	1111	

Figure 1: Input to linear ordering.

Figure 1 shows an example of four blocks, and in Figure 3 we show the the clustering we produce. Hamming distances between rows are given on the right, and we can quantify the quality of clustering as the sum of the Hamming distances, with total Hamming distance 8 in the input and only 4 in the output. The clustered output is an improvement, with a total distance of 4 bits of difference in the bottom version compared to 8 with arbitrary ordering. Subjectively, this clustering is good because it groups the outages in the middle of the period and places larger outages at the bottom.

Quan's algorithm clusters blocks by similarity using a simple greedy algorithm using Hamming distance. However, that algorithm is $O(db^2)$ for duration d and b blocks, with the quadratic term occurring because in each step of clustering, the best block is compared to all other unmatched blocks. While they report success for up to 2.5M blocks in less than one hour, these results are for short durations (48 hours). Our datasets instead exceed 4M blocks and have durations that stretch to months. Our implementation of this algorithm takes days to complete. Moreover, the greedy clustering algorithm does not parallelize easily, since selection of the most similar block at each step requires examining all other blocks *and* affects all subsequent choices.

2.2 Other Input: Routing and Anycast

We have applied our algorithms to routing updates and anycast catchment changes, in addition to outage data. All of these are large network datasets where commonality can be revealed through clustering; the challenge is to align other inputs to match binary outage timeseries.

For routing, we use routable prefixes instead of /24 blocks, and we map times of routing events (any BGP announcements and withdrawal messages, obtained from RouteViews [50] and BGPMon [53]) as an outage event (bit 0) in the observed timebin. We use timebins of 600 s to reflect route convergence times, and evaluate about 250k network prefixes.

To cluster changes in anycast catchments, we draw on DNS CHAOS queries [52] from RIPE Atlas [39]. Taken at 4-minute intervals, these observations identify the current Root DNS anycast site for each of about 9000 vantage points (VPs). We use this timeseries in our algorithms with VPs in place of blocks, and with timebins of 600 s to match more frequent reporting. Our clustering algorithms require that each observation be binary (on or off), not many-valued (as with many anycast sites). We map the initial anycast site for each VP to the value 1, then toggle this state in the timeseries any time the VP changes to a different anycast site. (In practice, most VPs only see two sites.) When studying denial-of-service (DoS) attacks we often have missing data; missed entries do not change timeseries state.

2.3 The Linear Ordering Algorithm

Our new algorithm for efficient linear ordering is possible because of two insights. First, we gain algorithmic efficiency by mapping clustering to sorting. Greedy clustering is $O(b^2)$ in blocks, since just like Bubble sort, each step requires a comparison against all remaining candidates. Instead, we turn to sorting to use the *relative* relationships discovered in each comparison to reduce the number of comparisons (to $O(b \log b)$).

The second insight is that mapping each block's timeseries to a multi-timescale representation allows the problem to be reduced to sorting. Before we define our representation explicitly, consider sorting a typical (single-timescale) timeseries. Each timeseries will be an array of many "up" bits followed by an 0-bit indicating an outage at a given time (its bit position). If we treat these timeseries as bitstrings and sort, we cluster by time of first outage, not by *overall* similarity across all times.

To capture behavior over *all* time, we generate many timeseries, each with a different timescale (the duration of each element). In the initial timeseries $S_b^{(0)}(i)$, at timescale $m = 0$, each element i represents one time bin with the shortest duration. At the next timescale ($m = 1$), we generate a timeseries $S_b^{(1)}(i)$ with half as many elements, each representing twice the duration. To do this, we combine adjacent elements with some aggregation function $f(s(i), s(i+1))$:

$$S_b^{(m+1)}(i/2) = f(S_b^{(m)}(i), S_b^{(m)}(i+1)),$$

Figure 2 shows an example of multi-timescale views of block b4 from Figure 1

We average elements (rounding down) as our aggregation function, $f_{largest}(a, b) = \lfloor (a+b)/2 \rfloor$. We call this function *Largest*, because short outages fade as they are overwhelmed by up periods. This function requires that we preserve the timeseries as real values (floating point, not just binary). Because $f_{largest}$ captures overall behavior, we use it as our combining function. In the top of Figure 2, the outage in this block is short enough to disappear at these timescales with $f_{largest}$.

agg. fn	**timescale** (m)	**status** ($S_b(\cdot)$)
largest	0	1111 **1100** **0**111 1111
	1	1 1 1 **0** **0*** 1 1 1
	2	1 **0*** 1 1
	3	1 1
	4	1
	all	1 11 1**0**11 111**00**111 11111**0000**1111111
first	0	1111 **1100** **0**111 1111
	1	1 1 1 **0** **0** 1 1 1
	2	1 **0** **0** 1
	3	**0** **0**
	4	**0**
	all	**0 00** 1**00**1 111**00**111 11111**0000**1111111

Figure 2: Multi-timescale views of block b4.

multi-timescale representation	**block**	**Hamm. dist.**
1 11 1111 1111111**0** 111111111111110	b2	2
1 11 1111 111**0**1111 1111111**0**11111111	b1	0
1 11 1111 111**0**1111 1111111**0**11111111	b5	1
1 11 1**0**11 111**0**1111 1111111**00**11111111	b3	1
1 11 1**0**11 111**00**111 1111111**000**1111111	b4	

Figure 3: Output from linear ordering.

We also examined an alternate aggregation function we call *First*. The First function aggregates values with logical-AND, $f_{first}(a, b) = a * b$. Logical-AND is simple, but it allows even a tiny outage to propagate across all timescales, since a zero value (an outage) in any time bin will dominate all up periods as they are ANDed together. As a result, f_{first} tends to cluster blocks by the first outage that occurs, not overall behavior. The bottom of Figure 2 shows the use of First, where the outage at times $i = [6, 8]$ propagates to mark all values in timescales $m = 3$ and 4 as down.

Finally, for sorting, we form the multi-timescale string by concatenating all timeseries from the largest-timescale to shortest-timescale. Thus:

$$S_b^{all} = S_b^{(M)} + S_b^{(M-1)} + \ldots + S_b^{(1)} + S_b^{(0)}$$

where + indicates concatenation of each timeseries.

Figure 3 sorts our sample blocks by their multi-timescale representation. In this example, sorting results in the same ordering as with Quan's algorithm (Figure 1). For large datasets this algorithm is much faster.

2.4 Linear Ordering Performance

Our goal in the new algorithm is to improve performance of linear ordering. Pragmatically, we are successful because the Quan algorithm could not run to completion on our full datasets of 4M blocks and about 12k observations (3 months at 11 minutes per observation) and it takes about a day to run even on highly downsampled datasets (1 in 200 sampling). The new algorithm can cluster the full dataset in less than one hour.

The core reason for this improvement is a shift from an algorithm that is $O(b^2)$ to one that is $O(b \log b)$ in runtime as a function of the number of blocks, as we shift from all-pairs greedy clustering to sorting on our multi-resolution bitstring. We add one caveat here: while the performance as a function of number of blocks is better, there is a cost in that the size of what is compared is larger. Each status bitstring is d observations long, but the multi-resolution version is actually $d \log d$ bits long, so technically the algorithm runtimes are $O(db^2)$ before and $O(db \log d \log b)$ with the new algorithm. However, there are far more blocks and so more timeseries than there observations in each one: b is around 4M for our dataset representing the full Internet, while there are only around $d = 12$k observations per quarter, so the d term is dominated by the b term.

The new algorithm also is faster than the old for several practical reasons. First, sort implementations are carefully optimized—we use either the implementation in GNU coreutils, or in Hadoop. Those implementations avoid unnecessary I/O and data copies, and can use parallelism. By contrast, our implementation of the prior algorithm was not carefully optimized and was implemented in Python. Second, when comparing timeseries, comparisons for sorting can terminate after the first differing element, while evaluations of Hamming distance require comparison of all elements.

3 EVENT CLUSTERING

While linear ordering supports visualization and enables human perception to come to bear, it is fundamentally limited—any block can have only two neighbors. Linear ordering of blocks can capture at most two relationships. In addition, linear ordering will cluster blocks on long outages over short, so smaller features can be overridden by long outages that propagate to larger timescales.

These limitations encourage our use of *event-based clustering*. Rather than focus on relationships between blocks, we look at the outages themselves. Our hypothesis is that blocks that consistently change state at the same time are reacting to some underlying, common factor.

3.1 Definitions and Overview

An *event* is when a block changes state, and the *input* to event clustering is the list of times and transition for each state transition for each /24 block. An event can be common to many blocks if they all transition in the same direction (up-to-down or down-to-up) at the same time. Times must be compared approximately to account for variation in when outage transitions are observed; Trinocular observations only guarantee precision within the measurement interval of 11 minutes.

Clusters are all blocks that show consistent responses for most events. We define *most* as 90% of events roughly correlated in time.

Our algorithm maps blocks into clusters based on events. From a graph theoretic viewpoint, blocks are vertices, we construct edges based on the consistency of events the blocks have in common, and clusters are the connected components that emerge.

Given outage data like that shown in Figure 7, we divide time into bins. In each time bin, we look for all blocks that have the same status transition (down-to-up or up-to-down). These *temporally-correlated events* are what we use to identify clusters, but before clustering we first list all *block pairs*, all possible combinations of blocks that share a temporally-correlated event.

We then combine all correlated-event block-pairs over all our observations. Block-pairs where more than a threshold of events are in common are temporally correlated suggest a some underlying common cause.

Block-pairs with only a few correlated events suggest random chance. When examining events to see if two candidate blocks are related, we compare how many events the two candidate blocks had together relative to all events seen by either block. Thus two blocks that have only four outage or recovery event in total and all are in common will have a 100% match, while if two were at the same time and two were at different times they would have a 50% match. We compare against *all* events seen by either block, so if b_1 and b_2 have two events in common, as do b_2 and b_3, then b_2's four total events mean that both the (b_1, b_2) and the (b_2, b_3) pairs have only a 50% match.

3.2 Event Clustering, Formally

Event clustering has several steps: input, cleaning, vertex creation, edge creation, and finally cluster formation.

Input: *Input* is a list of event times t (from observation duration $\mathbf{T} = [T_{start}, T_{end})$), each of which indicates that block b (drawn from all blocks, $\mathbf{B}$), transitions to new state s (drawn from $\mathbf{S} = \{0, 1\}$ indicating transition to out or up). We indicate the time of the jth transition to state s by block b as $E_b(s, j)$.

Before we receive outage data, we assume some pre-cleaning has been done to remove expected measurement artifacts [2]. Pre-cleaning confirms any outage that occurs is seen from at least three observation locations, avoiding mis-interpreting outages near a single observer as a global outage. It also removes blocks that show outages for very long period (more than two weeks), on the assumption that that block has either started filtering measurement traffic, or the block's operators have changed how it is used.

The *output* of event clustering is a set of clusters, each a list of blocks that are all strongly connected.

Data cleaning: First, we compute the marginal distributions of block status: how often is the block down, up, or unmeasureable. We define the marginal distribution for block b in state s as $M_b(s)$ and compute $\forall b \in \mathbf{B}, s \in \mathbf{S} : M_b(s) = \sum_{\forall j} E_b(s, j+1) - E_b(*, j)$.

We then compute the fraction of time each block is up: $U_b = M_b(1)/(M_b(0) + M_b(1))$. We define *strong* blocks B_{strong} as those that are mostly up: $B_{strong} = \{b \in \mathbf{B} \text{ s.t. } U_b > T_{strong}\}$. We currently use a threshold $T_{strong} = 0.8$. We ignore blocks that are down more than 20% of the time on the principle that a well maintained network should almost always be up. Looking for commonalities across diurnal or frequently down networks seems unlikely to be to be fruitful, and computation is quadratic with the number of concurrent events.

Vertex creation: We then map all events to time bins so we can find blocks that change status at roughly the same time. If we assume observation uncertainty E for outage beginning and ending, then time bins must be at least E seconds long.

Timebin duration is selected to avoid aliasing: if we map continuous times into discrete bins and an outage affects many blocks but at a time that is right at a bin transition, then half of the affected blocks appear in the prior bin and half in the next, diluting our conclusions. In addition, we do not require perfect consistency in when outages occur, and ideally we should run our algorithm with slightly different phases. In the end, we chose $D_{timebin} = 4096\,\text{s}$, slightly more than $6\times$ our $E = 660\,\text{s}$.

Another kind of measurement artifact occurs at the beginning and ending of our dataset. We discard the first and last events in each dataset for each block, since they represent beginning and ending measurement (a measurement artifact) and not an actual change at the target (a signal from the network). Most blocks are completely stable, emphasizing the need for extended observation to detect the rare outage events that inform about the network.

	cleaning				events
block	$M_b(0)$	$M_b(1)$	U_b	B_{strong}	N_b
b1	1	15	0.94	1	2
b2	1	15	0.94	1	1
b3	2	14	0.88	1	2
b4	2	14	0.88	1	2
b5	1	15	0.94	1	2

Figure 4: Data cleaning, marginal distributions, and number of events; from event clustering on our sample blocks.

Edge creation: Given the base timescale of each time bin, we now compute all *block-pairs*: the combinations of blocks that have the same state transition in that time bin. Each time bin i is a period P from $T_{start} + iD_{timebin}$ to $T_{start} + (i+1)D_{timebin}$. We find all blocks in time bin i with the same state transition to s: $B(s,i) = \{\forall b \in \mathbf{B} \text{ s.t. } E_b(s,j) \in P\}$ (for some transition j). We then list all pairs of blocks in time bin i with transition s: $P(s,i) = \{\forall b_1, b_2 \in B(s,i) \text{ s.t. } b_2 > b_1\}$.

We next evaluate how consistently blocks behave. We compute the number of events for each block and for each blockpair (without measurement start and end events): $N_b(s) = |E_b(s,*)| - 2$ and $N_{b_1,b_2}(s) = |P(s,*)|$, and combine both transitions $N_b = N_b(0) + N_b(1)$ and $N_{b_1,b_2} = N_{b_1,b_2}(0) + N_{b_1,b_2}(1)$. Commonality between two blocks is then $C_{b_1,b_2} = (N_{b_1,b_2} / \max(N_{b_1}, N_{b_2}))$. (Commonality is symmetric, so $C_{b_1,b_2} = C_{b_2,b_1}$).

Finally, we create edges ε from the commonality: $\varepsilon = \{(b_1, b_2) \text{ s.t. } C_{b_1,b_2} > T_{cluster}\}$. We set $T_{cluster} = 0.9$, a conservative choice. We studied variations from 0.29 to 0.9; we see similar results for all $T_{cluster} > 0.5$, but we see 3 to $4\times$ more edges when $T_{cluster} < 0.5$.

Cluster formation: Finally we can cluster blocks by this commonality. We use a simple breadth-first search (BFS) to create a cluster K of blocks that are reachable from some seed block b using the edges defined above. We label each cluster K^{id} by its lowest-numbered block. Thus the cluster-id for any block is the solution to the fixed point: $K_b^{id} = \min(b, b_2), \forall b_2 \text{ s.t. } (b, b_2) \in \varepsilon$. The blocks in a cluster with seed block b are all those with the same id $K_b^{blocks} = \{\forall b_1 \text{ s.t. } K_{b_1}^{id} = K_b^{id}\}$.

Clusters are usually complete or near-complete graphs, a result of our strict threshold ($T_{cluster}$). Near-complete clusters work well with BFS; a lower threshold may benefit from sophisticated clustering algorithms (perhaps OPTICS [3]) to avoid a single large, degenerate cluster.

3.3 An Example of Event Clustering

As an example, we consider event clustering on the blocks shown in Figure 1. Figure 4 shows the results of data

~~(b1,1,0)~~	(b1,0,3)	(b1,1,4)				~~(b1,u,8)~~
~~(b2,1,0)~~					(b2,0,7)	~~(b2,u,8)~~
~~(b3,1,0)~~	(b3,0,3)	(b3,1,4)				~~(b3,u,8)~~
~~(b4,1,0)~~	(b4,0,3)		(b4,1,5)			~~(b4,u,8)~~
~~(b5,1,0)~~	(b5,0,3)	(b5,1,4)				~~(b5,u,8)~~

Figure 5: Vertices (block, end state, timebin at timescale $m = 1$) created for event clustering on our sample blocks.

time bin	**events. . .**					
3	(b1,b3,0)	(b1,b4,0)	(b1,b5,0)	(b3,b4,0)	(b3,b5,0)	(b4,b5,0)
4	(b1,b3,1)		(b1,b5,1)		(b3,b5,1)	
N_{b_1,b_2}	2	1	2	1	2	1
C_{b_1,b_2}	1	0.5	1	0.5	1	0.5

Figure 6: Edges created from each time bin, and the number of mutual events for each block pair.

cleaning and the marginal distributions for these blocks. (None of these blocks are rejected.)

For this example we assume time bins for vertex creation are twice the base timescale, giving each block 8 time bins. We discard the first and last transitions (no measurement-to-up at timebin 0, and to-non-measured at timebin 8), leaving the vertices shown in Figure 5.

These edges induce verticies (Figure 6), with each row resulting from the indicated time bin. We compute events per block (right-most row of Figure 4) and evaluate the number of edges per block pair and block commonality (the bottom two rows in Figure 6). The result is to create these edges: (b1,b3), (b1,b5), and (b3, b5).

Finally we form clusters by walking this graph. We discover one cluster: b1, made up of blocks b1, b3, and b5. In this small example, we discover the three blocks that match perfectly at the timescale used for event clustering. We exclude block b4 from this cluster because it recovered later, and b2 because it has no outages in common with other blocks.

3.4 Event Clustering Performance

The most expensive step in event clustering is edge creation, with up to $O(n^2)$ edges, where n is the number of blocks in any time bin that have concurrent outages. In the worst case, this n could be the number of blocks b, but because outages are rare, for our data it is only a few percent of b. In practice, enumerating edges and counting matches typically consumes about half the runtime for our 3-month datasets.

Second, cluster formation requires repeatedly traversing the edges of the graph, and is $O(Db \log b)$, for b blocks and cluster diameter D. In the worst case, D can approach b, but in practice most clusters are densely connected and $D < 12$.

Fortunately, all of these algorithms parallelize well with Map/Reduce-style processing. We have implemented event clustering in Hadoop as job with ten stages of maps and reduces, followed by final non-parallel graph traversal stage. Our implementation is not optimal; we use simple database with primitives that are not very well optimized, and we could optimize away one reduce stage if we used custom code or had better implementations of join.

In practice, our algorithm runs in about two hours on three months of data for all 6415 blocks in 172/8, and in about five days on three months of Internet-wide data with about 4M blocks. These estimates are on a shared cluster of 220 cores of older (2011-era) x86 computers; they are rough the testbed is shared with intermittently competing workload. Much of the cluster idles during edge creation while the most active timbins generate edges. We suspect that optimization could reduce overall runtime.

4 CLUSTERING 2014Q3 OUTAGES

We study clustering of outages in third quarter, 2014.

4.1 Datasets and Events

Datasets: Table 1 lists the datasets we use to study our algorithms. We begin with two quarterly observations of Internet outages from Trinocular [34], each covering about 4M blocks; this data is available publicly [49]. Each quarterly dataset observes about hundreds of millions of state transitions together across all blocks. These datasets are composites of four geographically distributed observers; they include a state transition for each block any time one observer changes its conclusions, as well as a consensus that the block is up or down. Blocks are also sometimes marked as unmeasurable, either because too few observers are available to reach a strong conclusion (we require three, following prior work), or because observations suggest the block as "gone dark" by becoming non-responsive for more than 1 week [2]. The number of events per block is highly skewed, with many blocks showing only a handful of events, while other blocks change state daily [36].

In addition, we break out a subset of the 2014q3 dataset, examining just the 172.0.0.0/8 prefix in §4.3.

Events: Time Warner had a major outage affecting about 11 million customers on 2014-08-26t09:00 UTC for about two hours [6]. In later reports, Time Warner indicated that the problem was misconfiguration of their

period	start date (duration)	name	size (/24 blocks)	state changes
2014q3	2014-07-01 (92 days)	`internet_outage_adaptive_a17all-20140701` [49]	4,034,614	519,910,659
		172/8 subset of 2014q3	6,415	996,391
2015	2015-11-30 (1 day)	RIPE Atlas J-Root CHAOS [38]	9305 VPs	65,927

Table 1: Datasets used in this paper.

backbone network that occurred during planned maintenance [45, 47].

4.2 Linear Ordering for Visualization

Figure 7 shows example output of our linear ordering for visualization. Input to this plot is part of the 2014q3 dataset with 4M /24 blocks over three months. Here we geolocate all blocks using Maxmind Geolite City [29] and visualize only the U.S. blocks. We show the visualization at full resolution on our website [33]; here we have downsampled it by a factor of four for printing. At full resolution each row is a /24 block and each column a 1024 s period stretching over 90 days. We choose 1024 s (about 17 minutes) as relatively prime to the measurement frequency and therefore unlikely to cause aliasing. When required, we downsample blocks by discarding rows, and in the time dimension by averaging adjacent columns; after downsampling each pixel here represents about 68 minutes of time.

Linear ordering is dominated by the longest duration outage for each block (for example, see (c) in Figure 7). This outage is large enough that it propagates to even very large timescales. The jump from the left of the graph (d) to the right (c) occurs because outages in (d) last long enough they reach a larger timescale. Our visualization suggests studying these many long but uncorrelated outages to understand if they represent measurement error or some factor in Internet usage such as block reallocation or change in usage [37].

The more important operational feature is the long column at (b) that corresponds to the Time Warner outage. Because the outage was relatively short (about two hours) it is a secondary feature that assists in the ordering, but for this part of the ordering, only after the longer outages. For many blocks, the Time Warner outage is the only outage in these three months, so much later in the ordering (b) is the only visible feature. Another very small event is shown at (a).

These features illustrate the need for linear ordering: blocks of long outages shown at (c) and (d) are easily hidden if those blocks are dispersed across millions of others, yet clustered this way they stand out. While the Time-Warner outage did not last long enough to force all of its blocks to be adjacent, it stands out as a strong feature (b) as it frequently reappears. In both cases, our ordering algorithm improves visualization to assist a human analyst.

This example also shows the weakness of visualization alone—analysis cannot be automated, and a linear ordering can focus on the wrong feature. The Time Warner outage shows a dependency in the Internet infrastructure; our desire to automatically discover these dependencies motivates our development of event clustering.

4.3 Event Clustering in 172/8

The Time Warner outage serves as a natural experiment to evaluate event clustering's correctness and completeness. Here we examine all /24 prefixes in the 172.0.0.0/8 block in 2014q3. We examine this subset of data because it is small enough to visualize and validate manually. We chose this time period because it includes the Time Warner outage (with 2304 Time Warner /24 blocks in several non-adjacent groups), and this /8 block because, as original class-B address space, it has many /16 blocks allocated to different organizations.

4.3.1 Identifying Ground Truth. As ground truth, we take all measurable Time Warner blocks in 172/8, since that is the most that could be clustered. We will revisit this decision in §4.5. We then evaluate the precision ($tp/(tp+fp)$ for true and false positives) and recall ($tp/(tp+fn)$, considering false negatives) of event clustering at discovering this truth. This experiment is unusual, in that it is very rare for such a large ISP to have a complete outage, and difficult to replicate, because outages are rarely so well documented. The natural experiment in this one event provides some confidence that the much more common, smaller correlated failures that we see (for example, (a) in Figure 7) share a real underlying common point-of-failure.

We determine routed Time Warner prefixes and their origin Autonomous Systems (ASes) from 2014-10-01 RouteViews data [50]. We identify ASes using bulk whois provided by ARIN on 2015-04-15 [4]. We label as Time Warner all /24 blocks that are routed with "Time Warner" in *orgID* field for their origin AS.

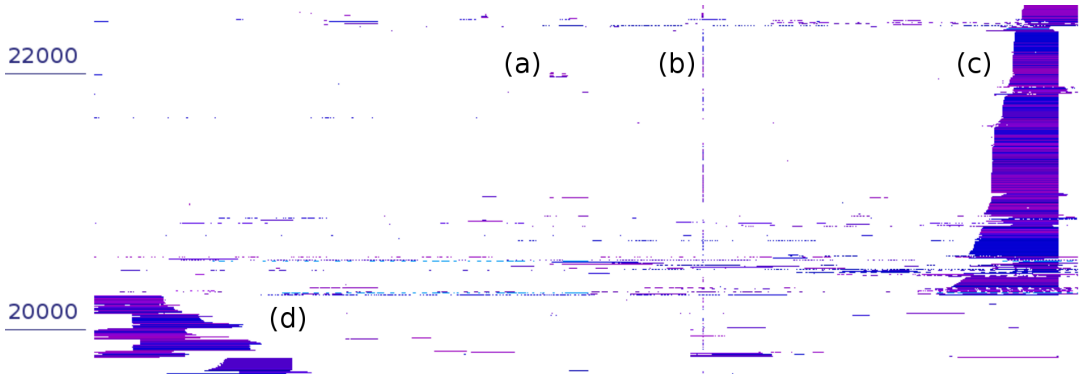


Figure 7: A subset of the U.S.-only blocks from 2014q3, showing at (b) the Time Warner outage on 2014-08-26.

2014q3:	**172/8**	**all**
measured /24 blocks	6415	4,037,157
after gone-dark cleaning	5818	4,034,613
never responded	869	292,402
mostly out	74	72,578
clean observations	4875	3,672,177
routed /24 TW blocks	1341	98,262
not measured	24	2,927
measured	1317	95,335
probed but never responding	295	4,241
removed (mostly out)	18	1,817
clean observations	1004	89,277

Table 2: Clustering input (in blocks) for 2014q3-172/8 and 2014q3-all.

The middle column of Table 2 characterizes the blocks in 172/8. Trinocular reports outages for only about 10% of the blocks; the remaining blocks cannot be measured because they did have enough ping-response addresses in the 36 months before this quarter. At this time, Time Warner operated about 2300 blocks in this range, and about half of these were measurable. We further clean these blocks after outage detection, discarding 296 blocks that never respond and 18 as rarely responsive (less than 25% of the time). Non-responsive block may be unused, used but not on the public Internet, or firewalled and non-responsive to ICMP echo requests. Rarely responsive blocks may be sparsely or intermittently used, or become firewalled early during observation.

In all, we observe 4875 blocks, of which 1004 are assigned to Time Warner.

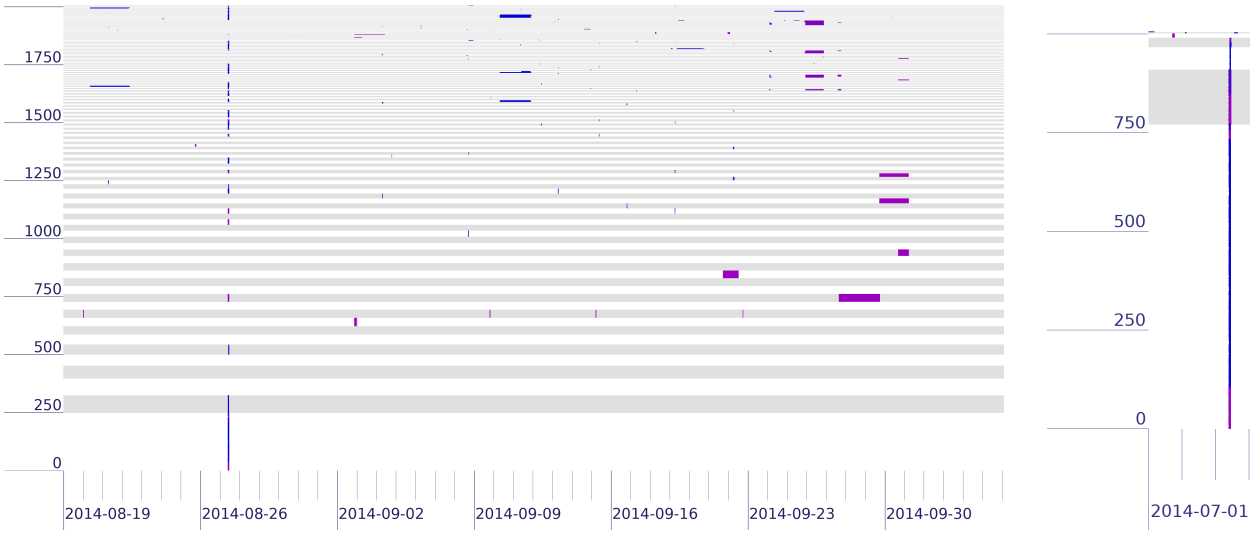
4.3.2 Clustering with Three Months: Discovery, Precision, and Recall. We first cluster with the full *three months* of data for 172/8, then examine what we find about Time Warner. We find 207 clusters, and Figure 8a shows outages from last 7 weeks of the three months. Of these, 90 clusters contain *only* TW blocks; Table 3 lists clusters by category.

Figure 8a shows the 207 clusters determined from 3 months. The Time-Warner outage is the vertical line on 2014-08-27, in the two largest clusters and in 88 smaller clusters. We next show that clustering *discovers* both events and common infrastructure with high *precision* and *recall*.

Discoving Large Events: These clusters *discover* the Time-Warner event. The two largest clusters in the 3-month dataset (the lowest two in Figure 8a), both show only the Time-Warner event. A researcher will discover large events by looking at the largest clusters in Figure 8a.

We see two large clusters (rather than one) because of inherent imprecision in measurements, coupled with time-binning in clustering. Trinocular may be as much as 660 s *late* in flagging an outage (although with multiple observers we often do better). With $D_{timebin}$ of 4096 s, 16% of the time (660/4096) the event will split across two timebins. We could reduce this chance by probing more frequently (increasing network traffic), or by lengthening the time bin (decreasing match precision). However, real outages are *not* always instantaneous because of distributed routing [23, 51]. Split events are still likely to be prominent enough for automatic detection and can be merged, if necessary, for further analysis.

Many of the remaining 88 clusters that contain Time Warner blocks are in different clusters because of *other* common outages. Figure 8a shows dozens of outages in the last 5 weeks of observation, and some of these affect certain TW blocks. These different outages reflect failures in different parts the TW infrastructure. In §4.3.3 we



(a) Clusters based on full *three months* of input. (Only the last 7 weeks are shown.) **(b) *Three day* input.**

Figure 8: Clusters (alternating backgrounds) for all measurable blocks in 172/8 based on different durations of input. Clusters decrease in size going up. Outages are shown by colored areas.

coverage	**2014q3-172/8**		**all 2014q3**	
input duration	**3 months**	**3 days**	**3 months**	**3 days**
unclustered blocks	2870	3868	2,875,305	3,514,745
all clusters	2005 / 207	1007 / 7	796,872 / 49,047	157,432 / 3,836
unclustered TW blocks	167	17	15,208	6,750
TW in TW-only clusters	837 / 90	57 / 1	20,930 / 2,742	967 / 129
TW:non-TW in mixed clusters	0:0 / 0	930:4 / 3	53,139:29,325 / 249	81,560:8,997 / 138
non-TW blocks in clusters	1168/ 121	16 / 3	693,478/ 45,971	65,908 / 3569

Table 3: Clustering (in blocks or blocks/clusters) for all blocks and clusters that contain Time-Warner-related /24 blocks, for 172/8 and full coverage, each with 3 months and 3 days of input.

show we find fewer, larger TW clusters when clustering on data around the TW event.

Precision and Recall: Clustering with three months provides very high *precision*: each of the clusters that relate to the TW event contains *only* TW blocks. If we regard non-TW blocks as false positives, and measurable TW blocks as ground truth, precision is *perfect* (1.0).

Recall measures how much of the TW infrastructure we discover. We consdier two definitions of ground truth: *all* TW blocks in 172/8, and all *measurable* blocks in 172/8, and assume an analyst identifies the 90 clusters as true positives (in §4.3.3 we justify this assumption). The first is pessimistic—*no* system with only active measurement can find the 1092 TW blocks that are not active, firewalled, or very rarely responsive. Even with this pessimistic definition, recall is 0.62 (837 found of all 1341 TW blocks). If we instead define our target as the 1004 measured and responding Time-Warner blocks that an ideal active system would find, our *recall is 0.83* (837 of 1004 blocks), fairly complete.

4.3.3 Clustering with Three Days: Blind Discovery. The top two events in the three-month clustering both point to the Time-Warner outage, and these events can lead to discovery of the event in other blocks. However, in *blind* discovery, one does not have prior knowledge of what organization to look for and where they already are. For blind discovery we aim for fewer, larger clusters.

We can *broaden* clustering by using a *shorter* observation period as input. The TW blocks were split into many clusters not because the TW outage was not prominent, but because outages at other times split the large TW event. The many resulting clusters are overly precise, revealing details about internals of the Time-Warner infrastructure, unfortunately losing the bigger picture.

Figure 8b visualizes the clusters that result from using only three days of data (starting 2014-08-25, two days

before the TW outage). We find far fewer blocks (from 2005 to 1007 blocks, Table 3, with different vertical scales in Figure 8) because there are only about 5 outage events in these three days. We find 7 clusters, with the 4 largest all TW-related (771, 139, 57, and 24 blocks), and three small clusters (11, 3 and 2 blocks, none TW).

Recall and Precision: We can quantify our ability to find more of the TW infrastructure with recall. With the shorter input dataset, *recall improves to 0.98* (987 of 1004 TW blocks). This improvement results because the shorter input window reduces the distractions at other times that would split individual TW blocks out of the large TW clusters. (And if we add unmeasurable TW blocks to ground truth, recall is still good: 0.74.)

Precision is 0.996 (987 of 991 blocks, taking only measurable TW blocks as ground truth), nearly perfect, but lower than the three-month data because of the 4 non-TW blocks in the three mixed clusters.

Iterative Discovery: Finally, the comparision between clustering over long and short inputs suggests that *iterative* discovery may be beneficial. One would begin with long-duration input to identify the time periods of big outages, then carry out focused clustering on these time periods to identify the largest number of blocks that match just that event. Iterative discovery could be automated, or, if done with human supervision, visualization (with linear ordering) can assist.

4.4 Clustering All Blocks of 2014q3

We next examine *all* blocks in 2014q3. The right column of Table 2 shows the input: more than 4M blocks, with almost 3.7M viable observations, 95k of which are are identified as Time Warner.

The clustering output is too big to visualize, with neary 800k blocks and 49k clusters for the full dataset, and 157k blocks in 3836 clusters when we focus on just the thee days around the Time-Warner outage (Table 3).

Recall and Precision: Continuing with recall and precision as defined before (§4.3.2).

Our results for recal from 172/8 hold for the complete dataset: recall is now 0.83 for 3 months, and 0.92 with 3 days (and 0.75 and 0.83 with our pessmistic definition), confirming that a focus on the TW outage avoids distraction. Precision is 0.72 and 0.90, both lower than in 172/8, but still quite high. Lower precision likely reflects that TW makes up a smaller fraction of the whole Internet than of 172/8, with more non-TW blocks to confuse.

Blind Discovery: Blind discovery is straightforward with full data: with three months, the largest cluster (34,691 blocks) is 97% Time Warner, as are the third and fourth largest (with 8330 and 8280 blocks), more than enough evidence to focus on the outage. Clustering on three days is even clearer, with the top four (sizes: 49k, 25k, 5.7k, and 4.6k) each 94% to 86% TW.

We are currently evaluating clustering for 2017q4, a period that includes a major outage for Comcast, a large U.S. ISP [11], and long-term outages in Puerto Rico resulting from Hurricane Maria. A full analysis of these events are outside the scope of this paper, but they will provide additional tests of blind discovery.

4.5 Revealing Unknown Dependencies

We next show that many of the non-Time-Warner blocks that appear in TW clusters are actually *previously unknown dependencies* of networks on Time Warner, not false positives. We investigate all four non-TW blocks in the 2014q3-172/8 clustering with 3-day input (§4.3.3)., and a random sample of 10 blocks from the four largest clusters from 2014q3-all clustering with 3-day input.

Of the 4 non-TW blocks in 172/8, two are mislabeled and two belong to hosting services. two (172.248.41/24 and 172.248.7/24) were not labeled in our dataset. However, all 995 of the 1024 /24 blocks in 172.248/14 that are labeled were listed as RoadRunner-West, a Time Warner service. (We confirm this use with traceroutes taken in Jan. 2018.) The other two blocks are assigned to Enzu.com, a hosting company; and Nobis Technology, a datacenter and hosting company that was purchased by LeaseWeb in 2016. We examined historical traceroutes from CAIDA [10] before, during, and after our 2014q3 dataset (although with DNS from2018), suggesting that 172.246.86/24 was operating Enzu's scalableds.com, and 172.247.0/24 was hosted by Above.net. We conjecture that these blocks may have been in Time Warner datacenters in 2014.

Of the 10 random non-TW blocks from 2014q3-all, *seven* were for Time Warner or subsidiaries (then unlabeled TW and RoadRunner, plus four for Bright House Networks), or partners (ECR Internet, a TW Business reseller), two were for other cable providers (GVT Communciations and Wayport), and one was for telmex.net.ar (a Mexican telecomm with operations in Argentina). We confirmed with public records that in 2014q3, Bright House was owned by Time Warner, and we discovered the partnership between ECR Internet and TW. We believe at least the seven TW-related blocks represent dependencies discovered through clustering.

Two of the four blocks in 2014q3-172/8 and seven of the ten in 2014q3-all are all *previously hidden dependences we discovered through event clustering.*

4.6 The Need for Long Observation

This quarter of data and the TW event demonstrate the importance of rare events and the need for long

observation. Our view of Time Warner was only possible because of this large event; to our knowledge, they have never had that level of outage before. If we had only considered 2014q4 data we would not have discovered the how many blocks and organizations depend on TW's backbone.

This need for long-duration observation confirmed by two related results. First, prior work reported that repeated probing for months is required for router-level topologies to discover backup paths, since they are only visible during failures [32]. Second, a similar trade-off has been reported in passive monitoring of network traffic to discover services [7]. While passive observation finds busy hosts in hours, it finds only 60% of ground truth after 18 days, and *half* of this discovery is due to external scanners walking the network. Good coverage in each these studies depends on rare external events (outages, backup paths, or external scans) that are outside the direct control of the observer.

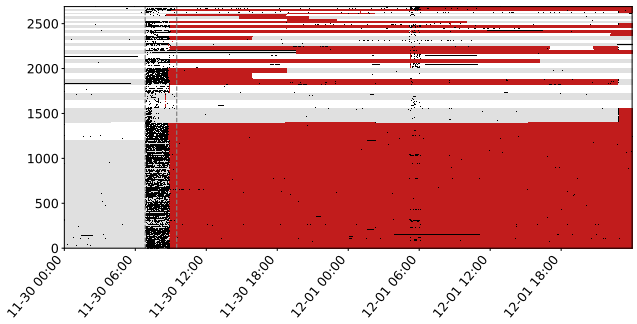


Figure 9: The largest 30 clusters with similar anycast catchment choices for J-Root DNS. Clusters are shown with alternating white and light gray backgrounds, with red showing a new anycast site and black missing data due to DoS.

5 CLUSTERING TO STUDY ROUTING AND ANYCAST

We next describe use of both linear ordering and event clustering to study anycast routing during Denial-of-Service (DoS) attacks. Our goal is to understand the relationship between routing changes and anycast catchments during the the Nov. 30, 2015 attack on the DNS Root server system [31, 40]. We use public information about routing (from RouteViews [50]) and RIPE Atlas [39] to infer routing changes and examine their effects. This problem is a good fit for our clustering algorithms these observations are numerous enough that they cannot be examined manually, but clustering can reveal underlying events (often routing) that result in anycast changes and often improved service.

We apply event clustering to 48 hours of observations 9305 RIPE Atlas Vantage Points (VPs or "probes"), starting at 2015-11-30t00:00 UTC. Each VP reports which anycast site it sees every 4 minutes. Here we report data for the the J-Root DNS service. Event clustering produces 237 clusters.

Figure 9 shows the largest 30 clusters. The clusters are shown in alternating background shades; the initial site for each VP is white, then it flips to red for each site change. Black areas indicate missing data due to the two DoS attacks, one starting at time 6:50, the second at 5:10 the next day.

We see that the largest two clusters (1200 and 300 VPs) switch sites mid-way through the attack and remain at the new site. Other clusters show different patterns of changes, and that many are temporary. Discovery of these clusters reduces how many routing events need to be studied to understand how operators and the Internet react to DoS attacks. A next step is analysis of BGP routing events for each cluster, a more tractable task with only 237 clusters rather than 9305 VPs.

6 RELATED WORK

Our work builds on prior work evaluating Internet robustness at routers, ASes, and business and politics.

We reviewed prior studies of Internet robustness in the introduction of this paper. There are many studies of Internet topology at the router level [1, 12, 21, 24, 27, 43], AS-level [9, 22, 26, 32], and the business level [30]. The strength of these studies is that by focusing on individual layers and mechanisms, they can identify specific root causes of problems. Prior work complements our end-to-end study, which can detect common failures across all layers, but is more limited in identifying root causes. A common problem across all of these approaches is the completeness of coverage; most events are rare, requiring long and careful observation [32].

Several groups measure network outages [15, 25, 34, 41, 42]. We use datasets from Trinocular because they have large scope, well defined precision, and are available to researchers at no cost [34, 49]. In principle our analysis could work over other sources of outage data [15, 25, 41, 42]. Some of these approaches place preconditions on what is measured: iPlane tracks only routable prefixes [25], and Pingin' follows weather events [41]. Use of background radiation is promising [15], but to our knowledge, datasets with labeled outages using this technique are not yet available. Recent work has examined outages at IXPs [19]; we instead study edge networks.

Beyond robustness, recent work has explored independent observation of network neutrality [54] and traffic policing [17]. As with our work, the goal of the network neutrality study is to allow a third party to detect a property of a multi-party Internet, but violations of network neutrality occur more frequently than outages. Traffic policing can be imposed at many places in the network, however the traffic policing study was carried out by Google, making use of its participation as one of the parties in video traffic. While the analysis in both of these papers differs from ours, our linear ordering may help visualization of their measurements.

7 CONCLUSION

This paper has shown two new algorithms to assist in identifying common points-of-failure in the Internet. Using end-to-end measurements of outages, we ordered address blocks to provide a visualization to support human pattern analysis. We then showed how correlations of failure events across blocks allow clustering of blocks that respond similarly, suggesting a common point-of-failure. We demonstrated these algorithms against data from 2014q3, showing that the outage experienced by Time Warner in that period allows us to not only discover all Time-Warner blocks that depend on their backbone, but also blocks of two CDNs that appear to share that dependency.

ACKNOWLEDGMENTS

We thank hosts of our pingers: Christos Papadopoulos (CSU); Midori Kato, Yohei Kuga, Rod Van Meter (Keio U. and WIDE); George Polyzos and George Xylomenos (Athens University of Economics and Business), Wim Biemolt and Gijs Rijnders (SURFNet).

We thank John Wroclawski for discussion and suggestions about the paper.

We thank Young Hyun (CAIDA) for providing historical traceroute data.

We thank Jay Bennett, John Healy, Brian Luu, Rasoul Safavian of the FCC for their input on Internet outage detection.

The in this paper is partially supported by the Department of Homeland Security (DHS) Science and Technology Directorate, HSARPA, Cyber Security Division, via the Air Force Research Laboratory, Information Directorate (agreement FA8750-17-2-0280), and contract number HHSP233201600010C. The U.S. Government is authorized to make reprints for governmental purposes notwithstanding any copyright. The views contained herein are those of the authors and do not necessarily represent those of DHS or the U.S. Government.

REFERENCES

[1] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. Anatomy of a large European IXP. In *Proceedings of the ACM SIGCOMM Conference*, pages 163–174, Helsinki, Finland, August 2012. ACM.

[2] Abdulla Alwabel, John Healy, John Heidemann, Brian Luu, Yuri Pradkin, and Rasoul Safavian. Evaluating externally visible outages. Technical Report ISI-TR-2015-701, USC/Information Sciences Institute, August 2015.

[3] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 49–60, Philadelphia, Pennsylvania, USA, June 1999. ACM.

[4] ARIN. ARIN Whois. http://www.arin.net/whois, April 2015.

[5] Associated Press. Fire in Baltimore snarls Internet traffic, too. *New York Times*, July 20 2001.

[6] Associated Press. Time Warner Cable says outages largely resolved. *New York Times*, Aug. 27 2014.

[7] Genevieve Bartlett, John Heidemann, and Christos Papadopoulos. Understanding passive and active service discovery. In *Proceedings of the ACM Internet Measurement Conference*, pages 57–70, San Diego, California, USA, October 2007. ACM.

[8] Eve Batey. BART defends decision to cut off cell service after civil rights, FCC concerns raised. *SF Appeal Online Newspaper*, Aug. 12 2011.

[9] Matthew Caesar and Jennifer Rexford. BGP routing policies in ISP networks. *IEEE Network Magazine*, 19(6):5–11, November 2005.

[10] CAIDA. CAIDA UCSD IPv4 prefix-probing dataset. http://www.caida.org/data/active/ipv4_prefix_probing_dataset.xml, 2013. Samples from July 2013 to April 2015 for 172.246.86/24 and 172.247.0/24. Provided by Young Hyun.

[11] Ashley Carman. Comcast's Xfinity internet service is reportedly down across the US. The Virge, November 2017.

[12] Kimberly Claffy, Young Hyun, Ken Keys, Marina Fomenkov, and Dmitri Krioukov. Internet mapping: from art to science. In *Proceedings of the IEEE Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, pages 205–211, Alexandria, VA, USA, March 2009. IEEE.

[13] David D. Clark. The design philosophy of the DARPA Internet protocols. In *Proceedings of the 1988 Symposium on Communications Architectures and Protocols*, pages 106–114. ACM, August 1988.

[14] James Cowie. Egypt leaves the Internet. Renesys Blog http://www.renesys.com/blog/2011/01/egypt-leaves-the-internet.shtml, January 2011.

[15] Alberto Dainotti, Claudio Squarcella, Emile Aben, Marco Chiesa, Kimberly C. Claffy, Michele Russo, and Antonio Pescapé. Analysis of country-wide Internet outages caused by censorship. In *Proceedings of the ACM Internet Measurement Conference*, pages 1–18, Berlin, Germany, November 2011. ACM.

[16] Ramakrishnan Durairajan, Paul Barford, Joel Sommers, and Walter Willinger. InterTubes: A study of the US long-haul fiber-optic infrastructure. In *Proceedings of the ACM SIGCOMM Conference*, pages 565–578, London, United Kingdom, August 2015. ACM.

[17] Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, and Ramesh Govindan. An Internet-wide analysis of traffic

policing. In *Proceedings of the ACM SIGCOMM Conference*, pages 468–482, Floranopolis, Brazil, 2016. ACM.

[18] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM Conference*, pages 350–361, Toronto, Ontario, Canada, August 2011. ACM.

[19] Vasileios Giotsas, Christoph Dietzel, Georgios Smaragdakis, Anja Feldmann, Arthur Berger, and Emile Aben. Detecting peering infrastructure outages in the wild. In *Proceedings of the ACM SIGCOMM Conference*, pages 446–459, Los Angeles, CA, USA, August 2017. ACM.

[20] James Glanz and John Markoff. Egypt's autocracy found Internet's 'off' switch. *New York Times*, page A1, Feb. 16 2011.

[21] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for Internet map discovery. In *Proceedings of the IEEE Infocom*, pages 1371–1380, Tel Aviv, Israel, March 2000. IEEE.

[22] Enrico Gregori, Alessandro Improta, Luciano Lenzini, Lorenzo Rossi, and Luca Sani. On the incompleteness of the AS-level graph: a novel methodology for BGP route collector placement. In *Proceedings of the ACM Internet Measurement Conference*, pages 253–264, Boston, Massachusetts, USA, November 2012. ACM.

[23] Craig Labovitz, Abha Ahuja, Abhijit Abose, and Farnam Jahanian. Delayed Internet routing convergence. In *Proceedings of the ACM SIGCOMM Conference*, pages 175–187, Stockholm, Sweeden, August 2000. ACM.

[24] Lun Li, David Alderson, Walter Willinger, and John Doyle. A first-principles approach to understanding the Internet's router-level topology. In *Proceedings of the ACM SIGCOMM Conference*, pages 3–14, Portland, Oregon, USA, August 2004. ACM.

[25] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An information plane for distributed services. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 367–380, Seattle, WA, USA, November 2006. USENIX.

[26] Priya Mahadevan, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Xenofontas Dimitropoulos, k. c. claffy, and Amin Vahdat. The Internet AS-level topology: Three data sources and one definitive metric. *ACM Computer Communication Review*, 36(1):17–26, January 2006.

[27] Pietro Marchetta, Pascal Mérindol, Benoit Donnet, Antonio Pescapè, and Jean-Jacques Pansiot. Topology discovery at the router level: A new hybrid tool targeting ISP networks. *IEEE Journal of Selected Areas in Communication*, 29(9):1776–1787, October 2011.

[28] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, and Christophe Diot. Characterization of failures in an IP backbone. In *Proceedings of the 23rd IEEE Infocom*, Hong Kong, China, March 2004. IEEE.

[29] Maxmind. Geolite city. Web page http://dev.maxmind.com/geoip/geolite, 2014.

[30] MLab. ISP interconnection and its impact on consumer Internet performance. Technical report, Measurement Lab Consortium, October 2014.

[31] Giovane C. M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Müller, Lan Wei, and Christian Hesselman. Anycast vs. DDoS: Evaluating the November 2015 root DNS event. In *Proceedings of the ACM Internet Measurement Conference*, November 2016.

[32] Ricardo V. Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. In search of the elusive ground truth: the Internet's AS-level connectivity structure. In *Proceedings of the ACM SIGMETRICS*, pages 217–228. ACM, June 2008.

[33] ANT Project. Ant internet outage browser. https://ant.isi.edu/outage/browse/, 2016.

[34] Lin Quan, John Heidemann, and Yuri Pradkin. Trinocular: Understanding Internet reliability through adaptive probing. In *Proceedings of the ACM SIGCOMM Conference*, pages 255–266, Hong Kong, China, August 2013. ACM.

[35] Lin Quan, John Heidemann, and Yuri Pradkin. Visualizing sparse Internet events: Network outages and route changes. *Computing*, 96(1):39–51, January 2014.

[36] Lin Quan, John Heidemann, and Yuri Pradkin. When the Internet sleeps: Correlating diurnal networks with external factors. In *Proceedings of the ACM Internet Measurement Conference*, pages 87–100, Vancouver, BC, Canada, November 2014. ACM.

[37] Philipp Richter, Georgios Smaragdakis, David Plonka, and Arthur Berger. Beyond counting: New perspectives on the active IPv4 address space. In *Proceedings of the ACM Internet Measurement Conference*, pages 135–149, Santa Monica, CA, USA, November 2016. ACM.

[38] RIPE NCC. RIPE Atlas J-Root server data. https://atlas.ripe.net/measurements/10316/.

[39] RIPE NCC. RIPE Atlas. web site https://atlas.ripe.net/, 2010.

[40] Root Server Operators. Events of 2015-11-30. Technical report, Root Server Operators, Dec. 4 2015.

[41] Aaron Schulman and Neil Spring. Pingin' in the rain. In *Proceedings of the ACM Internet Measurement Conference*, pages 19–25, Berlin, Germany, November 2011. ACM.

[42] Anant Shah, Romain Fontugne, Emile Aben, Cristel Pelsser, and Randy Bush. Disco: Fast, good, and cheap outage detection. In *Proceedings of the IEEE International Workshop on Traffic Monitoring and Analysis*, pages 1–9, Dublin, Ireland, June 2017. Springer.

[43] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. In *Proceedings of the ACM SIGCOMM Conference*, pages 133–145, Pittsburgh, Pennsylvania, USA, August 2002. ACM.

[44] Brain Stelter. Internet is a weapon in cable fight. *New York Times*, page B3, Oct. 20 2010.

[45] Brian Stelter. Time Warner Cable comes back from nationwide internet outage. CNN Media Website, August 2014.

[46] Srikanth Sundaresan, Danny Lee, Xiaohong Deng, Yun Feng, and Amogh Dhamdhere. Challenges in inferring Internet congestion using throughput measurements. In *Proceedings of the ACM Internet Measurement Conference*, pages 43–56, London, UK, November 2017. ACM.

[47] Time Warner Cable. This morning's outage. web http://www.twcableuntangled.com/2014/08/twc-identifies-cause-of-internet-outage/, August 2014.

[48] Daniel Turner, Kirill Levchenko, Alex C. Snoeren, and Stefan Savage. California fault lines: Understanding the causes and impact of network failures. In *Proceedings of the ACM SIGCOMM Conference*, pages 315–326, New Delhi, India, August 2010. ACM.

[49] USC/LANDER Project. Internet outage measurements. IMPACT ID USC-LANDER/internet_outage_adaptive_a17all-20140701 and https://ant.isi.edu/datasets/

internet_outages/, July 2014.
[50] Route Views. University of Oregon Route Views Project. web site http://www.routeviews.org, 2000.
[51] Feng Wang, Zhuoqing Morley Mao, Jia Wang, Lixin Gao, and Randy Bush. A measurement study on the impact of routing events on end-to-end Internet path performance. In *Proceedings of the ACM SIGCOMM Conference*, pages 375–386, Pisa, Italy, August 2006. ACM.
[52] S. Woolf and D. Conrad. Requirements for a mechanism identifying a name server instance. RFC 4892, Internet Request For Comments, June 2007.
[53] He Yan, Ricardo Oliveira, Kevin Burnett, Dave Matthews, Lixia Zhang, and Dan Massey. BGPmon: A real-time, scalable, extensible monitoring system. In *Proceedings of the IEEE Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, pages 212–223, Washington, DC, USA, March 2009. IEEE.
[54] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network neutrality inference. In *Proceedings of the ACM SIGCOMM Conference*, pages 63–74, Chicago, IL, USA, August 2014. ACM.

A CLUSTERING OF THE TIME WARNER OUTAGE

We next use observations from 2014q3 and the TimeWarner outage to understand what clustering can find.

We discuss outages in the 172/8 block of 2014q3 in §4.3.

Here we provide some additional information. Figure 10 shows the full three months of clusters.

B MORE ABOUT CLUSTERING ANYCAST CATCHMENTS

In §5 we discussed the use of clustering to evaluate anycast catchments. Figure 9 shows anycast catchments as seen by the clustering algorithm, with binary values, where white shows the original anycast site and it alternates with red for each subsequent site shift.

While that visualization is faithful to what clustering sees, several different catchments are shown as white. Figure 11 shows the two largest clusters where each site has a unique color.

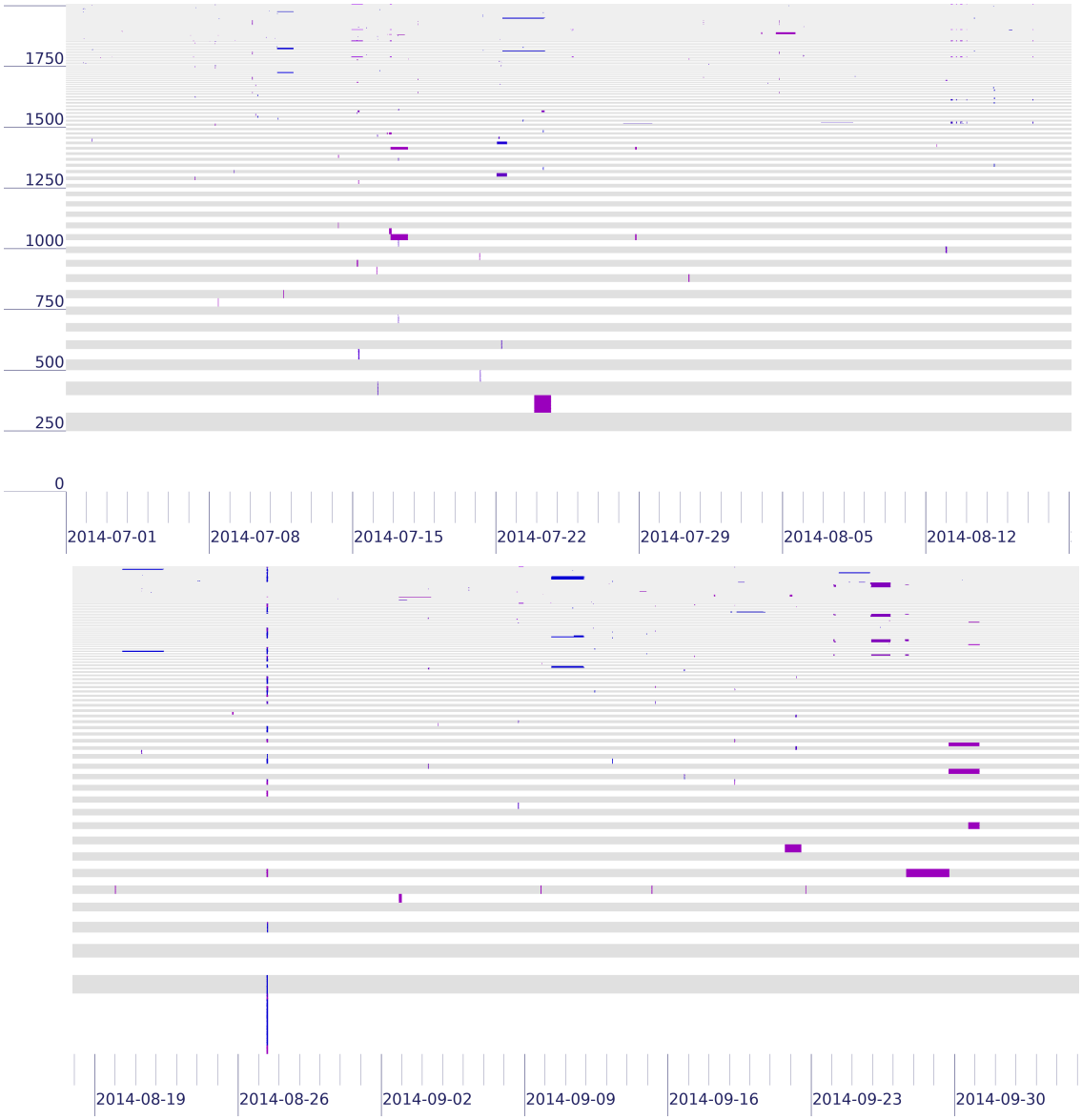
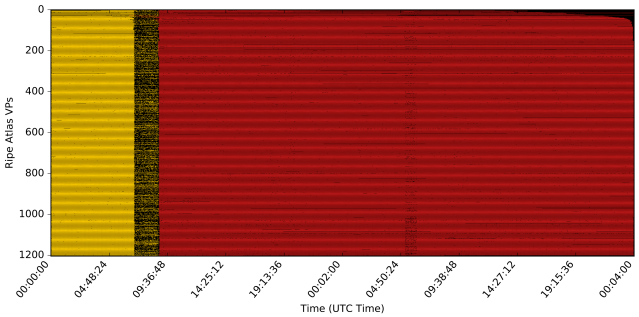
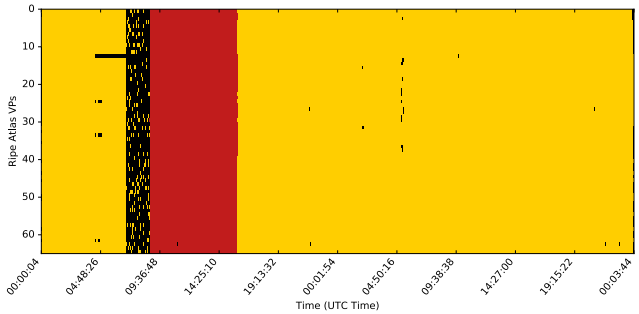


Figure 10: Clusters for all measurable blocks in 172/8 using all three months of 2014q3. Data is grouped by cluster and then linear ordering within each cluster. Clusters are shown by alternating white and light gray backgrounds. Outages are shown by colored areas.



(a) The largest cluster for J-Root.



(b) The second cluster for J-Root.

Figure 11: The largest two clusters in J-Root with similar anycast catchment choices. Colors indicate anycast site: yellow is AMS, red is Frankfurt. Black represents observations with no data.