BigBen: Telemetry Processing for Internet-Wide Event Monitoring

Meenakshi Syamkumar, Yugali Gullapalli, Wei Tang, Paul Barford†, Fellow, IEEE, and Joel Sommers

Abstract—This paper describes BigBen, a network telemetry processing system designed to enable accurate and timely reporting of Internet events (e.g., outages, attacks, and configuration changes). BigBen is distinct from other Internet-wide event detection systems in its use of passive measurements of Network Time Protocol (NTP) traffic. We describe the architecture of BigBen, and a cloud-based implementation developed to process large NTP data sets and provide accurate daily event reporting. We demonstrate BigBen on a 15.5TB corpus of NTP data. We show that BigBen identifies a wide range of Internet events characterized by their location, scope, and duration. We compare the events detected by BigBen vs. events detected by a large active probe-based detection system. We find only modest overlap between the two datasets and show how BigBen provides details on events that are not available from active measurements. Finally, we report on the perspective that BigBen provides on Internet events that were reported by third parties. In each case, BigBen confirms the event and provides details that were not available in prior reports, highlighting the utility of the passive, NTP-based approach.

Index Terms—Internet event monitoring, network time protocol, Internet outages, passive network monitoring.

I. INTRODUCTION

THE DYNAMIC nature of the Internet has been well documented over the years. Events such as network reconfigurations, flash crowds, outages, persistent congestion and attacks of various sorts are commonplace and have been shown to have diverse spatio-temporal characteristics. These events are of on-going interest from a research perspective, and have significant implications for day-to-day network management and operations.

The fact that events can occur at any time and in any place in the Internet complicates the task of detection. The service provider perspective on event detection focuses on a single infrastructure and assumes unfettered access to diverse measurement data such as active probe-based measurements, SNMP-based measurements, packet traces, flow-export data, and logs from devices deployed in the network. In contrast, the Internet-wide perspective on event detection assumes that data must be collected outside of (most) network perimeters, typically using active probe-based methods. Beyond gathering measurement data, challenges in event detection from both perspectives include (i) managing, fusing and processing potentially very large amounts of data, (ii) identifying events accurately and in a timely fashion and, (iii) organizing, refining and visualizing details of identified events. Prior studies on event detection have focused on a variety of these issues and in particular on measurements and identification methods as described in Section V.

In this paper, we describe a network telemetry processing and reporting system that we call BigBen, which is designed to provide an Internet-wide perspective on events. The high-level goal of BigBen is an efficient processing platform that supports timely and accurate event reporting based on measurements collected passively without injecting any new traffic into the network. The utility of event detection based on passive measurements is that it offers the opportunity to obviate standard challenges of active probe-based methods, which include management overhead, traffic overhead, potential for blocked probes and limited information provided by probes. However, this approach raises the question of what kinds of passive measurement data might be available that can provide useful insights into Internet-wide events.

To achieve our high-level goals we build on prior work that identifies the Network Time Protocol (NTP) as a compelling source of Internet-wide measurement data [66]. In particular, NTP is the only on-by-default protocol in the Internet, which means that measurements of client connections across the globe can be passively collected at NTP servers. Further, the timing-based nature of NTP data provides a means for extracting one way delays (OWDs) between clients and servers [37], which enables events to be identified e.g., by looking for changes in OWDs [70].

Our architecture for BigBen is designed for Internet-wide event detection based on NTP and includes (i) a distributed NTP traffic collection component, (ii) an Extract Transform Load (ETL) component, (iii) an event identification component, and (iv) a visualization and reporting component. The design and interfaces between each component are modular and validate the importance of scalability, data integrity, extensibility and manageability in the system. Details of the architecture for BigBen are provided in Section II.
We demonstrate the efficacy and utility of our design by developing a prototype implementation of BigBen. To support timely (e.g., daily or hourly) event reporting, a key requirement of the implementation is the ability to manage and process hundreds of GB to TB of NTP data per day [66]. Handling data at this scale is a significant engineering challenge and in this sense BigBen is related to network telemetry systems such as [40]. Our implementation utilizes a file-based approach for data management and Apache Spark for processing. Our implementation also includes an event detector based on Robust Principal Components Analysis, which was shown to be effective for detecting spikes in NTP data in [70]. However, recognizing the challenges in assessing event detection accuracy (due to the significant lack of Internet-wide ground truth), we emphasize that other detectors can easily be substituted into the system or run in parallel with the current detector. Details of BigBen’s implementation are provided in Section II.

We demonstrate BigBen through a study of Internet events based on data collected over a 5 month period from January through May, 2019. For this study, we collected data on a daily basis from 16 NTP servers in 7 locations in the U.S. resulting 15.5TB of total data. Our implementations of the ETL, event identification and reporting components of BigBen were deployed in CloudLab [14]. Our system ran on a single (multi-core) node and was able to process data contributed by all 16 servers over a 24-hour (about 100 GB) in about 1 hour and 15 minutes. This easily accomplished our design goal of being able to process gigabytes of raw NTP data and extract Internet events on a daily basis. Moreover, the system can easily be extended to accommodate contributions from additional servers.

Next, we drill down on events that are identified in the data. We categorize events based on the fidelity of the signals in the NTP data: For IPv4, we find that about 100K events per day are detected and that these events are indicated in about 60K prefixes. For IPv6, we find that about 120 events per day are detected and that these events are indicated in about 90 prefixes. We find that IPv4 events range in duration from 32 seconds to 27 hours, with an average event duration of 2.7 hours. We find that IPv6 events range in duration from 64 seconds to 24 hours, with an average event duration of 3.5 hours. To the best of our knowledge, this is the first report of network events in IPv6 address space. It is important to note that the fidelity of these measurements on the order of seconds is much more precise than what is possible from active probe-based systems (e.g., [41], [65]) that periodically scan the entire Internet.

To provide perspective on the events identified by BigBen, we compare with reports of events identified by ISI’s Trinocular system [41], [65] that took place during the same time as our data collection. While Trinocular covers substantially more /24 prefixes (4.2M vs. 488K), 76K prefixes are visible in the NTP data that are not visible in the ISI data. Overall, Trinocular reports about 120K events per day vs. the 100K events per day reported by BigBen when aggregated over all signal levels. We find relatively low event match rates between the two data sets, which we attribute to the relatively low overlap on /24 prefixes (only about 40K). We argue that these results highlight the utility and complementary nature of BigBen vs. active probe-based systems for Internet-wide event detection.

Finally, we investigate how BigBen views events that have been reported by third parties during our data collection period. We posit that these reports provide a measure of validation for the implemented event detector. In each case, we find that BigBen detects an event that corresponds with what was reported. We highlight how the events manifest in our data, and show that BigBen often provides an expanded perspective over what was reported.

In summary, the two main contributions of this paper are as follows. First, we describe the architecture and implementation of BigBen, a system for Internet-wide event detection based on passive collection of NTP measurements. Second, we demonstrate the capabilities of BigBen on a large corpus of NTP measurements collected over a 5 month period. Our results show that BigBen is capable of detecting and reporting diverse characteristics of events across both IPv4 and IPv6 address space. In our continuing work, we plan to keep a log of outage events that will be made available to the community and to refine the event reporting capabilities toward the goal of making BigBen a resource for broadly understanding Internet behavior on a daily basis.

II. ARCHITECTURE AND IMPLEMENTATION

In this section we describe the design objectives and architecture of BigBen. We also describe the implementation that we developed to demonstrate the efficacy and utility of the system.

A. Overview and Objectives

The starting point for our work is the observation that NTP is a unique and compelling source of Internet-wide measurement data that can be readily applied to the problem of event detection [37], [70]. In particular, an Internet-wide event detection system based on NTP data offers the following advantages over active probe-based detection systems:

- Easier to manage since distributed probe sources are not required,
- Not subject to traffic blocking,
- Doesn’t introduce additional traffic into the network,
- Fine-grained, one way delay measurements.

Another attractive aspect of NTP is that the scope and geographic diversity of clients that synchronize with any given server can be very broad [66]. This offers the opportunity to detect Internet-wide events with data contributions from a relatively small number of NTP servers. Thus, the basic model for our system is a set of distributed NTP servers that contribute data to a centralized processing system that manages

1Historically, reporting of events by ISPs, affected entities or third parties is rare, which is important motivation for our work.

2All software developed in this work plus a sample NTP data set and the 5 month event data set will be made public when this paper is published.
the data, applies detection algorithms and generates reports on

189 events.

The primary objective for our system is to support accurate
190 and timely reporting of Internet events. The secondary objec-
191 tive for our system is modular design. Accuracy is defined in
192 the standard way in terms of low false identification rates (our
193 system configuration is conservative and aims toward erring
194 on the side of false negatives). Accuracy depends intrinsically
195 on the data and algorithms applied to it. While algorithms
196 can be assessed using synthetic data, it is difficult to measure
197 accuracy in practice due to the lack of outage reporting by
198 service providers or other entities. Partnerships with service
199 providers are an appealing possibility. We posit that a useful
200 way to gain perspective on accuracy is by comparing events
201 detected by entirely different systems. This is exactly what
202 we do in our evaluation of BigBen in Section IV. We posit
203 that hourly event reporting is appropriate for this use case, but
204 for the purpose of results described in Section IV, we use
daily reporting. An important implication for timely reporting
206 is the ability to scalably and efficiently process data from partic-
207 ipating NTP servers. Modularity is the secondary objective
208 of the design, enabling algorithms within components to be
209 updated or replaced. In the implementation described below,
210 any component can be replaced and any step within a compo-
211 nent can be reconfigured or replaced with minimal impact on
212 other steps or components.

The design of BigBen includes four major components as
shown in Figure 1. The measurement component is deployed
on participating NTP servers, and the other components
are deployed on a cloud-based processing infrastructure. We
describe each of the components in detail below.

B. Measurement Collection Component

The objective of this component is to collect and transfer
details of interactions between NTP clients and participating

225 servers. Standard NTP implementations do not inherently pro-
226 vide a logging function that collects detailed data on every
227 connection with a client, so a purpose built capability is
228 required. The primary requirements are that the software that
229 implements this component is easy to install and operate, that
230 it operate securely and robustly, and that it have minimal
231 processing/storage/network demands.

Data collection is organized into discrete time epochs to
236 ensure timeliness of reporting. As a side-effect, discrete time
237 epochs also help to ensure that the sizes of trace files remain
238 modest for transfer and processing. At the beginning of each
time epoch, a program is initiated to reset trace collection to
239 an e file, and to compress and transfer the trace into a cloud
240 repository for further processing, as we discuss below. Old
241 traces are removed from the server to conserve disk space.

This component is implemented in about 150 lines of
244 Python code. Once initiated, this component runs permanently
245 on the NTP server, unless it is terminated explicitly by the
NTP operator or due to undesirable conditions such as network
outages, server hardware failures, and so on. Each instance
requires several configuration parameters, including the time
period over which to create new packet trace files, and authen-
tication credentials for data transfer using scp. The basis for
measurements is the standard tcpdump utility, which is used
to collect packet traces on server port 123 (NTP) and write
the trace data to a libpcap-format file.

The computation and storage demands of the BigBen com-
249 ponent that runs on NTP servers is almost entirely due the
250 packet capture process. Prior work has shown that tcpdump’s
load on a server is proportional to the traffic sent/received by
the system [68] (in our case NTP exchanges) and has a negli-
gible impact even under high traffic loads. While we did not
make specific measurements of this component, we worked
closely with all NTP operators in our deployments and none
were concerned about increased load caused by BigBen.
We asked NTP operators who were contributing data to set the time period interval to 1 hour. We note that the server contributing data for our tests with the heaviest load generates about 1.2GB compressed trace per hour on average. We found 1 hour to be a reasonable configuration of the time period because it ensures that the remote transfer step does not overload either the network or server disk usage.

C. Cloudlab Infrastructure

The remaining data processing elements of BigBen are run on cloud-based infrastructure. This enables high bandwidth network access, and significant processing and storage capability. Our implementation of BigBen was run in Cloudlab [14]. We employ a single node with the following configuration: (i) CPU: Two Intel E5-2660 v3 10-core CPUs at 2.60 GHz, and (ii) RAM: 160GB ECC memory. Cloudlab instances typically offer two configurable 1.2 TB HDDs. Since our data storage requirement far exceeds the default available disk storage space, we employ an external 28 TB NFS disk storage.

D. Extract-Transform-Load (ETL) Component

ETL is a standard component in many commercial big data processing environments. The basic notion is to collect and combine data from multiple sources and to place it in a target data store. In our context, the objective of the ETL component is to receive and organize the raw NTP data from participating servers, and to prepare it for further processing. The steps in BigBen’s ETL component include: (i) check the receipt of the compressed trace once every time epoch and to send email notifications if trace files are missing; (ii) organize the compressed trace files by month for each NTP server; (iii) extract relevant NTP information from the compressed libpcap traces; (iv) assure data integrity by removing empty or corrupted records; and (v) transform the extracted NTP information from the files to a CSV format for further processing.

This component makes our data processing pipeline more robust by identifying and reporting interruptions in the receipt of the compressed traces to the NTP server operators. In practice this feature proved very useful in interactions with the operators and re-initiation the measurement collection component (see Section II-B) on multiple occasions.

The high-level steps of this component are implemented in about 180 lines of Bash code. This script is scheduled to run as a cron job once during every time epoch (1 hour - coordinated with measurement collection). This component takes less than a minute for all 16 servers (see Section III). The transformation step which converts tcpdump data to CSV format is implemented in about 700 lines of C code. For every packet in the tcpdump data, CSV schema includes the following fields: (1) packet number, (2) source IP address, (3) destination IP address, (4) latency (NTP origin timestamp - NTP receive timestamp), (5) NTP polling exponent, (6) packet timestamp, (7) root delay, (8) round-trip time, and (9) NTP reference IP address.

E. Event Identification Component

The objective of this component is to generate a set of events that constitute a significant deviation from normal behavior for a given network prefix. This definition aligns directly with a standard definition of anomaly detection. In our context of NTP-based monitoring, it relates to the idea of identifying a significant change in one way delays (OWD)3 between one or more clients in a given network prefix and a participating NTP server. While OWDs computed from NTP packets can be subject to error (e.g., due to path asymmetry [74]), our choice of RPCA to detect large changes in OWDs was made to accommodate variability that might be caused by such errors.

The high level steps involved in the event identification component are as follows. First, we process the CSV data to identify OWDs from clients that are in tight synchronization with the NTP server. Next, we cluster clients into IPv4 /24 or IPv6 /96 prefixes, which were selected as conventions for detection and reporting in this paper. We select a /24 prefix size for IPv4 as these aggregates generally reflect a fairly localized geography [39]; the IPv6 prefix size is chosen as a similar aggregate. We then employ an event detector based on Robust Principle Components Analysis (RPCA) to identify delay spikes experienced by clients within a cluster. We apply a classifier based on Z-scores to label events from high to low confidence. Finally, we consolidate events to provide a single, global view of events. We describe the details of these steps below.

1) Extracting OWDS: A filtering step is required to extract OWDs from trace data. In particular, we want to use OWDs for event detection from clients that are in tight synchronization (i.e., the NTP exchange has been observed to be stable over a period of time) with their server since these delays are more likely to have smaller error bounds [37].

A key consideration for this step is the tradeoff between the volume of data that must be processed (e.g., about 72 GB/24 hrs. for one particular server) and the fidelity of OWDs. While the measurement and the ETL components execute hourly, we chose to run this component every 24hr to ensure that there are ample data points for the latency filtering algorithm described below.

We implemented OWD filtering described in [37] in Apache Spark [1] to parallelize CSV trace processing from a single NTP server. The filter is written in pyspark. It reads the CSV data as Resilient Distributed Dataset (RDD) chunks in a stand-alone mode, single node Spark cluster.4 We process only the client-to-server packets since they provide OWD measurement values in both directions (client-to-server (c2s) and server-to-client (s2c)).

Next, we identify any trend in changes to the NTP polling value for each client and classify the clients into constant, increasing, decreasing, and variable polling types. For each classification type, we implement the specific filtering algorithm described in [37] to identify tightly synchronized (TS) clients. These OWD values associated with these clients are

3OWDs are computed from packets sent between clients and NTP servers that are in tight synchronization as described in [66].

4This is not an inherent limitation, but simply reflects our CloudLab setup.
then written into RDD-specific chunk files. Next, we cluster clients into /24 and /96 prefixes for IPv4 and IPv6 respectively. We note that the prefix length is configurable; we also experimented with /48 and /56 aggregates for IPv6. The pyspark implementation and the clustering implementation is executed by a wrapper Bash script (about 290 lines of code), which executes filtering and clustering for each NTP server sequentially.

2) **RPCA Event Detection:** We use an RPCA-based method to identify events in the form of OWD spikes experienced by a cluster of clients. RPCA was shown to be an effective technique for processing NTP data to detect events in [70]. The modular design of BigBen enables other detectors to be substituted or used in parallel. Our implementation of the RPCA algorithm is shown in Algorithm 1. For each network prefix cluster, we generate a $t \times n$ matrix similar to [70] (steps 1 to 21), where $t$ is the time bin used to group every client in a row and $n$ is the number of clients in a prefix cluster. We employ the Robust Covariance Estimator (RCE) [34] for the clusters, by default (steps 22 to 32). We found that nearly 50% of the clusters have a covariance matrix with non-negative definite, hence RCE fails on such clusters. For the clusters on which RCE fails, we employ the Elliptical PCA (EPCA) [49] method to detect outliers. We compared RCE and EPCA results to confirm that RPCA scores are similar for both of these methods.

Our RPCA-based detector is implemented in about 1000 lines of Python code. We use Pebble [24] for multiprocess generation of the $t \times n$ clusters, invocation of RPCA (we leverage PcaNA R package [25]), and generation of events by aggregating contiguous timestamp bins with RPCA outliers—false flag values (steps 33 and 34). If an event is detected in a single timestamp bin we classify it as single spike. If an event is detected in multiple timestamp bins, it is classified as an event.

3) **Z-Score Event Classification:** To aid in assuring internal consistency and to provide a measure of confidence in detected events and single spikes we employ a Z-score computation. For every $t \times n$ cluster, we generate Z-scores on OWDs to identify outliers defined as $2\sigma$ deviation from $\mu$. We then classify events as follows (steps 35 and 36 in Algorithm 1): (1) class A: 75% to 100% correlation between RPCA outliers and Z-score outliers (these are the highest confidence events), (2) class B: 50% to 75% correlation, (3) class C: 25% to 50% correlation. Due to minimal correlation with Z-score outliers, we discard the remaining events.

The event identification component generates two primary output files. The first is the event details output file with entries consisting of prefix cluster, number of clients, event start time, event end time, and event class details for every event. The second is the matrix details output file consisting of all the $t \times n$ matrices for the prefix clusters, which forms the input to BigBen’s visualization component (see Section II-F3).

4) **Event Consolidation:** The final step in the event detector consolidates the events detected in individual NTP servers trace datasets to provide a single global view of events. The consolidation process begins by taking the events detected using one server as the base set. We then employ the following steps in order to merge the remaining server events:

(i) obtain the list of matching /24 or /96 prefix clusters and the corresponding events (ii) employ Python’s intervaltree package [23] to determine direct temporal overlap between previously observed events on the same prefix, and (iii) merge the events which have overlap by considering the earliest event start time, latest event end time, and best possible class between class A, class B, and class C (see Section II-E3). At
the conclusion of this step the raw /24 and /96 event data and the consolidated raw event data are output to files.

F. Visualization and Reporting Component

A significant amount of raw event data can be generated by BigBen. This calls for tools for post processing that can be used for organizing, analyzing and understanding Internet-wide events on a daily or hourly basis. The system has capabilities to generate daily reports and graph visualization of OWD clusters.

1) Daily Report Generator: The daily report generator provides simple statistics on events detected over the previous 24 hours. This includes information about BigBen itself including the total amount of data contributed by participating servers, total size of the daily CSV files, total number of clients, etc. Daily reports also include the total number of /24 and /96 prefixes in which events were detected, the top 10 largest events in terms of larger network aggregates, the top 10 longest duration events and other basic information. The daily reports are useful for monitoring BigBen itself and for developing a general understanding of network event behavior.

2) Geographic Footprint Visualizer: The geographic footprint visualizer component enables visualization of the geolocation of IPv4 clusters (network prefixes) for which events are detected. Given a network prefix, this component geolocates each IP address using MaxMind’s IP Geolocation service [18]. Then, it plots the geolocated points on a map and creates a convex hull of the geolocated points. A similar approach is reported in [71]. For /24 prefixes, typically this component generates a visualization of a circle around the single geographic location to which the IP addresses are geolocated. The backend processing for this component is implemented in about 500 lines of Python code and the frontend visualizer is built using ESRI ArcGIS [15].

3) Cluster OWD Graph Visualizer: The cluster OWD graph visualizer component produces one-way delay time-series graphs for an IPv4 or IPv6 cluster with detected events. This component is implemented in about 150 lines of Python code. To handle the varying number of clients per prefix, this component uses seaborn [26] to generate a lineplot for each client and automatically assign color mapping to the clients using the hue parameter. Before plotting the timeseries, we replace the NA values for each client with the minimum latency for the client, similar to the RPCA processing (see Section II-E2).

III. NTP DATASETS

In this section, we provide an overview of the NTP trace datasets collected from 16 NTP servers distributed across 7 geographic sites. The geographic sites include: (i) Chicago, IL (C), (ii) Edison, NJ (E), (iii) Jackson, WI (J), (iv) Philadelphia, PA (P), (v) Salt Lake City, UT (S), (iv) Urbana-Champaign, IL (U), and (vii) Madison, WI (M).

Table I summarizes the basic statistics from each NTP server for the months of January to May 2019. A total of 15.5TB raw NTP trace dataset was collected during this time period. As shown in the table, the amount of data collected from the servers varies significantly, with server S1 generating 3.2GB per hour on an average, which is much larger when compared to the datasets from the rest of the NTP servers. The variability across servers is a direct reflection of how clients have been configured – a process which depends on system administrator and sometimes equipment manufacturer preferences [60]. In the U.S., NIST along with universities have offered stable, high-quality NTP services to the public for decades. However, in 2003, the NTP Pool project began offering a dynamic service as a means for balancing load across servers [29]. The big data processing requirement (100GB per day on an average) was a key consideration in BigBen's system design and implementation. Furthermore, we note that the majority of the NTP servers support both IPv4 and IPv6 clients; both protocols are also supported in BigBen’s event detector component (see Section II-E). To the best of our knowledge, our event detection system is the first to provide complete support for detecting events in the IPv6 address space.

Data provided during the study was fairly consistent. However, there were days in which we did not receive data from individual providers. Reasons for gaps in the dataset include: downtime of NTP server, outage of Cloudlab network, and NFS network downtime. Overall, this represented a small fraction of the total period of collection.

The coverage and accuracy of detected events relates directly to servers providing data to BigBen. As will be shown in Section IV, our data set includes clients from only a subset of the v4 address space. However, we argue that the volume and coverage is sufficient for demonstrating the capabilities of BigBen and to show how NTP-based monitoring provides a useful, complementary perspective on events. We plan to expand the number of NTP servers monitored by BigBen in future work.
IV. RESULTS

In this section, we demonstrate BigBen’s ability to detect events. We begin by reporting the basic characteristics of the events detected in our NTP data set. Then, we compare the events detected by our methodology with the events detected through active probing from ISI’s Trinocular project [41]. Finally, we report on the perspective that BigBen provides on events that were reported by third parties, highlighting the detailed results from one specific event.

A. Basic Event Characteristics

Figure 2-(left) shows the total number of IPv4 prefix clusters observed in BigBen dataset, and total number of IPv4 prefixes in which events were detected from January 2019 to May 2019. It also shows the number of IPv4 prefixes on which single spikes were detected. The scope of the covered IPv4 prefixes ranges from 2 clients to 244 clients. Figure 2-(right) shows similar results for IPv6 /96 prefixes, whose scope of coverage ranges from 2 clients to 84 clients; we note that results for /48 and /56 IPv6 aggregates have similar qualitative characteristics. These figures show that the number of prefixes on which BigBen detects events is quite consistent on a daily basis and has a slightly increasing trend for IPv4 over the period of study, while the number of prefixes affected for IPv6 shows no clear trend. The basic characteristics of observed prefixes observed and events detected is relatively equivalent from both the c2s (client to server) and s2c (server to client) perspective.

The duration of the detected c2s IPv4 events ranges from 32 seconds to 27 hours, with an average event duration of 2.7 hours. For IPv4 s2c events, the duration range remains the same, with an average event duration of 2.6 hours. For IPv6 c2s events, event durations range from 64 seconds to 24 hours, with an average event duration of 3.5 hours. For IPv6 s2c events, the average event duration is 1.7 hours. IPv4 and IPv6 single spikes (both s2c and c2s) have a very short lived durations, ranging from 1 second to 22 minutes (due to clients with high NTP polling values). Our observations indicate that there is a wide range of durations during which there are periods of elevated latencies that may be due to persistent congestion, outages, or because of ISP traffic management policies.

In Section IV-D we examine specific events and their causes.

Figure 3 shows details of the timeseries of IPv4 events (left) and IPv6 events (right) detected by BigBen in our data set. The figure shows some consistency in terms of number of

---

**Fig. 2.** Number of IPv4 /24 (left) and IPv6 /96 (right) prefixes observed and number of events detected within those prefixes over 5 months.

**Fig. 3.** Number of events detected for IPv4 /24 (left) and IPv6 /96 (right) prefixes broken down by event classes as described in Section II-E3.
events identified by BigBen on a daily basis. We posit that many of these reflect standard Internet dynamics such as route changes, congestion or address block reconfiguration. We also observe that there are fewer high-confidence (class A) than lower-confidence events (classes B and C) over time. However, the qualitatively tight correspondence between high and low confidence events suggests that high confidence events can be a focus for identifying the onset events and lower confidence events can aid in understanding event scope and impact. We Interestingly, the spikes observed in the timeseries (e.g., May 23) do not correspond to reported events from third parties; we are investigating these events in our ongoing work. When considering similar timeseries of single spike events (not shown due to space constraints), we (a) observe a similar pattern of fewer high-confidence events (class A) than lower-confidence events, as well as (b) a very high count of single spikes (as expected) due to the permissive nature of spike identification.

B. Event Consolidation and Aggregation Across Prefixes

The consolidation process is applied individually to events and single spikes (c.f. II-E2). On a daily average, we are able to consolidate 103 IPv4 events, 9 IPv6 events, 82 IPv4 single spikes, 5 IPv6 single spikes detected on c2s OWDs, and 114 IPv4 events, 2 IPv6 events, 102 IPv4 single spikes, 5 IPv6 single spikes detected on s2c OWDs. We expected the count of events observed across servers to be low, as we initially consider clusters of small-sized prefixes (/24 or /96), showing that additional NTP servers will further diversify and enhance coverage.

The event aggregation post-processes the consolidated events (see Section II-E4) detected on /24 or /96 prefixes into larger network aggregates (i.e., shorter prefixes). To perform this aggregation, we consider prefix lists from CAIDA [13] and Team Cymru [27] and look up each /24 or /96 prefix to find the corresponding larger IP prefix aggregate, typically included in BGP announcements. For every higher level IP prefix where we observe at least 2 constituent /24 or /96 prefixes, we aggregate the events using the same steps as the consolidation algorithm (see Section II-E4). For this aggregation process, we only consider the BigBen events and omit BigBen single spikes. Note also that the confidence level assigned to an event is the highest observed across the events that get aggregated together. We take the view that multiple indications of the same event increase confidence in it, thus supporting use of the highest confidence level observed. Figure 4 shows the count of aggregated IPv4 events (left) and IPv6 events (right).

C. Comparison With ISI/Trinocular

Table II shows the ranking for top 5 IPv4 ASes in terms of identified events, including AS number, name, count of BigBen events observed throughout the 5 month dataset. Interestingly, we observe a large number of events for Amazon, Inc. We show that the same Amazon AS suffers collateral damage during an outage event reported by a third party (see Section IV-D). BigBen also identifies a sizable number of events in tier-1 service providers (e.g., Verizon, CenturyLink).

Table III shows a similar ranking for top 5 IPv6 ASes.

![Fig. 4. Number of IPv4 (left) and IPv6 (right) events detected by announced network prefix based on prefix lists provided by CAIDA and Team Cymru during the period of study.](image-url)
Fig. 5. Number of ISI/Trinocular event matches (left) and corresponding classes for matching BigBen events (right).

TABLE III

<table>
<thead>
<tr>
<th>Rank</th>
<th>ASN</th>
<th>ISP</th>
<th># of events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32748</td>
<td>SteadFast</td>
<td>8,631</td>
</tr>
<tr>
<td>2</td>
<td>14061</td>
<td>DigitalOcean</td>
<td>2,871</td>
</tr>
<tr>
<td>3</td>
<td>26347</td>
<td>New Dream Network</td>
<td>1,792</td>
</tr>
<tr>
<td>4</td>
<td>7922</td>
<td>Comcast Cable</td>
<td>1,435</td>
</tr>
<tr>
<td>5</td>
<td>54825</td>
<td>Packet Host</td>
<td>1,390</td>
</tr>
</tbody>
</table>

Table III shows the top 5 ASes ranked based on the number of IPv6 events identified from January-May 2019.

IPv4 /24’s (minus those that have requested no probing). For this comparison, we consider the ISI dataset from January to March 2019. During this period, the ISI dataset represented a total of 4,226,299 IPv4 prefixes. BigBen’s dataset over the same period included clients from 488,269 IPv4 prefixes. Interestingly, 76,545 IPv4 prefixes are observed in our data that are not observed by ISI. This highlights one of the complementary aspects of our NTP-based approach and the benefit of using passively collected NTP data which can give insight into events taking place in networks that may block active measurements. A combination of both approaches is likely to be most beneficial for gaining a comprehensive view of network events. During the comparison period, ISI reports an average of 121,695 IPv4 events on a per-day basis, while BigBen reports an average of 99,878 c2s events and 109,761 s2c events on a per-day basis.

1) Event Matching: We match events detected by BigBen with ISI-reported events based on /24 prefixes using a similar approach to our event consolidation (see Section II-E4). Apart from looking for a direct temporal overlap between BigBen events and ISI events we also consider events that match within a time window, which we set to be 1 hour. We include both events and single spikes for this comparison. Figure 5-(left) shows the number of BigBen events and single spikes that have direct matches, pre-matches, and post-matches with ISI events on a per day basis. It also shows the corresponding number of matching ISI events. We attribute the relatively low event match counts to the fact that we observe a similarly low match in /24 prefixes with events between the two datasets—only 40,097 matches (for the entire 3 month dataset). Figure 5-(right) shows the classes to which the matching BigBen events and single spikes belong to. As expected, we see higher number of matching events fall under class A (BigBen’s highest confidence level class).

D. Events Reported by Third Parties

We compare BigBen detected events to a set of events reported by third party sources including CAIDA, the Outages mailing list, and ThousandEyes, Inc. We consider these third-party reported events a form of ground truth and useful for demonstrating the utility of NTP-based event detection. It is worth mentioning that when we generated a list of the largest events (terms of total number of /24 prefixes affected) detected by BigBen in over the period of study, none overlapped with the third-party reported events.

Table IV shows a summary of reported events found during our data collection period. For every event, we report the event start and end time observed by the third party and the corresponding start and end times observed by BigBen. We also list the Autonomous Systems (AS) impacted during the event and the count of /24 prefixes on which BigBen observes the event. We also list the geographical region of impact of the event, obtained by visualizing the event using our Geographic footprint visualizer (see Section II-F2). These results highlight BigBen’s capabilities to detect events across geographical regions throughout the world. We show visualizations from one of the third party reported events below and omit the rest due space limitations.

1) May 13, 2019 China Telecom Event: ThousandEyes reported [17] that on May 13, 2019, China Telecom suffered a significant network outage lasting about 5 hours. BigBen’s reporting showed significant OWD elevations on multiple prefixes during this time. Figure 6 shows the BigBen’s Cluster OWD graph visualization (c2s OWDs) for prefix cluster 117.136.4.0/24 affecting 59 clients. This prefix is allocated to AS56042 (China Mobile Communications Corp.). In the figure, we observe steady state behavior indicated by horizontal lines or lines with minor variation as well as delay...
TABLE IV

<table>
<thead>
<tr>
<th>External event start time</th>
<th>External event end time</th>
<th>BigBen detected events start time</th>
<th>BigBen detected events end time</th>
<th>BigBen - AS(ies) affected</th>
<th>BigBen - # of /24 prefixes affected</th>
<th>BigBen - countries affected</th>
<th>External event source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 20, 4:28 PM</td>
<td>Jan 20, 7:08 PM</td>
<td>Jan 20, 3:30 PM</td>
<td>Jan 20, 8:04 PM</td>
<td>AS18809, AS20473</td>
<td>87</td>
<td>Panama, USA</td>
<td>CAIDA [7]</td>
</tr>
<tr>
<td>Jan 25, 4:28 PM</td>
<td>Jan 25, 6:48 PM</td>
<td>Jan 25, 3:28 PM</td>
<td>Jan 25, 7:47 PM</td>
<td>AS7018</td>
<td>437</td>
<td>USA</td>
<td>Outages mailing list [22], Verizon network status [28]</td>
</tr>
<tr>
<td>Feb 21, 7:49 PM</td>
<td>Feb 21, 11:09 PM</td>
<td>Feb 21, 8:06 PM</td>
<td>Feb 21, 8:07 PM</td>
<td>AS327712</td>
<td>1</td>
<td>Algeria</td>
<td>CAIDA [4], NetBlocks [21]</td>
</tr>
<tr>
<td>Feb 26, 2:00 PM</td>
<td>Feb 27, 3:15 AM</td>
<td>Feb 26, 1:49 PM</td>
<td>Feb 27, 2:36 AM</td>
<td>AS6400, AS51964</td>
<td>15</td>
<td>Dominican Republic, Germany</td>
<td>CAIDA [5]</td>
</tr>
<tr>
<td>Mar 7, 12:50 PM</td>
<td>Mar 13, 8:30 PM</td>
<td>Mar 08, 10:29 AM</td>
<td>Mar 13, 6:41 PM</td>
<td>AS8048, AS18286, AS51964, AS11878, AS38895</td>
<td>5</td>
<td>Venezuela, Germany, USA</td>
<td>CAIDA [8], Miami Herald [19], Havana Times [16]</td>
</tr>
<tr>
<td>Apr 28, 10:00 AM</td>
<td>Apr 29, 6:00 AM</td>
<td>Apr 28, 11:14 AM</td>
<td>Apr 28, 11:20 AM</td>
<td>AS36907</td>
<td>1</td>
<td>Angola</td>
<td>CAIDA [2]</td>
</tr>
<tr>
<td>April 29, 12:30 PM</td>
<td>May 1, 4:00 AM</td>
<td>Apr 29, 3:29 PM</td>
<td>May 1, 1:43 AM</td>
<td>AS3491</td>
<td>4</td>
<td>UK, USA Greece</td>
<td>CAIDA [3]</td>
</tr>
<tr>
<td>May 13, 7:30 PM</td>
<td>May 14, 12:30 AM</td>
<td>May 13, 6:32 PM</td>
<td>May 14, 1:26 AM</td>
<td>AS16591, AS56042, AS16509, AS38895</td>
<td>7176</td>
<td>USA, China</td>
<td>CAIDA [9], ThousandEyes [17], Internet Disruption Report [20]</td>
</tr>
<tr>
<td>May 13, 11:30 PM</td>
<td>May 14, 9:00 AM</td>
<td>May 13, 10:33 PM</td>
<td>May 14, 9:54 AM</td>
<td>AS63008, AS209</td>
<td>1174</td>
<td>USA</td>
<td>CAIDA [10]</td>
</tr>
<tr>
<td>May 18, 12:30 AM</td>
<td>May 18, 11:43 PM</td>
<td>May 18, 12:00 AM</td>
<td>May 18, 11:27 PM</td>
<td>AS237, AS7377, AS3651</td>
<td>15</td>
<td>USA</td>
<td>CAIDA [11]</td>
</tr>
<tr>
<td>May 19, 8:30 AM</td>
<td>May 19, 9:00 PM</td>
<td>May 19, 10:12 AM</td>
<td>May 19, 8:36 PM</td>
<td>AS63008, AS11878</td>
<td>4</td>
<td>USA</td>
<td>CAIDA [12]</td>
</tr>
</tbody>
</table>

![Fig. 6. May 13, 2019 China Telecom outage OWD graph for clusters 117.136.4.0/24.](image)

Fig. 6. May 13, 2019 China Telecom outage OWD graph for clusters 117.136.4.0/24.

spikes that reach nearly 900 milliseconds, indicating a 4x increase in OWD. These observations are consistent with those reported by ThousandEyes. In BigBen’s geographical impact plot, we observe effects across various parts of the United States (see Figure 7), as well as in China. While ThousandEyes reported that the U.S. West Coast was affected, BigBen’s report shows that the impact was more widespread across the U.S. Consistent with ThousandEyes reporting, BigBen’s report also observes Amazon AS16509 and AS38895 suffering collateral damage due to this outage event.

2) Feb 26/27, 2019 Dominican Republic Event: Figure 8 shows steady state behavior in c2s OWD values indicated by horizontal lines and the spikes observed in these values during the event for two prefixes 190.80.164.0/24 (left)—74 clients and 117.136.91.0/24 (right)—34 clients. Both of these prefixes...
are part of AS6400 (CODETEL). Similar to the China Telecom event, these graphs show OWD value spikes up to nearly 1000 milliseconds. We observe similar spikes on s2c OWD values. BigBen’s observations are consistent with ASes on which CAIDA [5] reports the event. In BigBen’s geographical impact plot (not shown due to space constraints), we observe that the event had impact in the Dominican Republic as well as Germany. In CAIDA’s report, only the Dominican Republic was identified as having been affected, which suggests that BigBen’s observation of the event was somewhat broader.

V. RELATED WORK

There have been many prior studies of detecting, locating, and diagnosing network outages. Similar to our work, a number of these previous studies have relied on streams of passively collected data to infer failures and outages. For example, some works have used feeds of routing updates from BGP or from other routing protocols, e.g., [46], [47], [53], [61]. Others have analyzed router BGP configurations to detect or better understand the causes of certain types of failures [38], [51]. Router logs (syslogs) have also been a key data source for analyzing the nature of network anomalies and faults [48], [62]. The potential synergy of combining routing updates with syslog data has been examined by Turner et al., [72], and Roughan et al. used the same data sources to detect forwarding anomalies which may be indicative of misconfigurations or impending failures [67]. Additional work in the area of network fault detection has evaluated communications among network operators to analyze network failures [30]. Turner et al. used syslog data, email announcements, and router configurations in an effort to understand causes behind network failures [73]. Network service providers are in a privileged position to collect a great deal of passive data for understanding the nature and causes of network faults; Markopolou et al. reported on findings from the Sprint IP backbone [53]. Other studies over the years have used passively collected data from honeypots to identify both malicious and benign Internet events [33], [57]. Studies have also used ever-present network protocol traffic [78] to identify network prefixes where faults may be taking place through detection of changes in delays or the absence of ordinarily-present traffic.

Our work is most similar to prior studies on NTP traffic and the opportunity to utilize this traffic to detect network events [37], [66], [70]. Indeed, we build directly on that body of work. However, our contributions are in the design and implementation of a scalable NTP data processing system that can be used on a daily basis for Internet-wide event monitoring and easily modified to utilize a variety of detection methods.

Another category of studies related to ours are those that have focused on network-wide monitoring and event detection within large data center infrastructures. Data center monitoring challenges are similar to those that we faced in development of BigBen including managing vary large data sets and identifying events accurately and in a timely fashion. Systems for data center monitoring and event detection are often based on passive measurements collected on end hosts or switches. For example, Moshref et al. describe Trumpet, which is a system designed to provide high precision data center-wide monitoring on end hosts [55]. Similarly, techniques for software defined measurement on programmable switches have been proposed in [43], [54], [76]. The primary difference between these studies and ours is our focus on Internet-wide monitoring, which includes the challenge of gathering data from diverse networks.

Many studies, in contrast to those mentioned so far, have used active measurement to identify and localize network outages and impairments. RFC 2678 identifies metrics for network connectivity and reachability, which relate directly to the notion of network faults and disconnection [52]. Many prior works have used the basic notion of RFC 2678—that a packet sent to a network address receives a response within some finite amount of time—as the basis for identifying network faults. For example, the Trinocular project uses periodic pings at a low rate to the full IPv4 address space, and detect disruptions and outages through a probabilistic framework [63], [64], [65]. As noted above, we use some of the data from this project as a point of comparison. Along the same lines, Padmanabhan et al. evaluated the response times to pings across the IPv4 address space and found that a small fraction of responses (5%) take at least 5 seconds to...
VI. CONCLUSION

Fiber cuts, malicious attacks, configuration changes, censorship and persistent congestion are examples of the events that routinely degrade or disrupt service in the Internet. Identifying and characterizing the details of such events is critical in network management and operations and can lead to new methods and configurations that improve Internet service. This paper describes BigBen, a cloud-based network telemetry processing system designed to support accurate and timely detection of events throughout the Internet. BigBen’s design includes (i) distributed collection of passive measurements of NTP traffic, (ii) an Extract Transform Load component that organizes data into a normalized format, (iii) an event identification component, and (iv) a visualization and reporting component. We develop a cloud-based implementation of BigBen that is able to process tarabytes of data on a daily basis. We demonstrate this implementation on a 15.5TB corpus of NTP data collected over a period of five months. The results show a wide range of events characterized by their scope and duration. We compare the events detected by BigBen vs. events detected by a large active probe-based detection system. We find only modest overlap and show how BigBen provides details on events that are not available from active measurements. We also show how BigBen identifies outage events that have been more broadly reported by third parties. Next steps for BigBen include expanding partnerships with NTP servers to broaden reach and depth of event identification, expanding visualizations and reporting, and to develop methods for forensic analysis and classification of events.

ACKNOWLEDGMENT

The authors wish to thank the NTP server operators who installed BigBen and provided data for this work. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES


Meenakshi Syamkumar received the Ph.D. degree in computer sciences from the University of Wisconsin–Madison in 2019. Her Doctoral research focused on methods for detection and analysis of Internet events. She is a Teaching Faculty with the Department of Computer Sciences, University of Wisconsin–Madison, where she leads the teaching efforts for the largest introductory Data Science course offered by the CS Department. Her current research interests are in network measurements and CS education. She is also passionate about improving diversity in computing.

Wei Tang received the master’s degree in computer science from the University of Wisconsin–Madison in 2018. She currently works with Amazon Web Services and is focused on serverless data integration. Her interests are distributed computing and big data.

Yugali Gullapalli received the master’s degree in computer sciences from the University of Wisconsin–Madison in May 2019. She is currently working as a Technical Program Manager with Microsoft, Redmond, WA, USA. During her graduate studies, her research activities focused on improving security of shapefiles of network map and Internet topology. She presented her work at the Annual ACM Sig Spatial Conference in 2019. She is a member of ACM-W.

Paul Barford (Fellow, IEEE) received the Ph.D. degree in computer science from Boston University in 2000. He is a Professor of Computer Science with the University of Wisconsin–Madison, where he is the Founder and the Director of Wisconsin Advanced Internet Laboratory. His research interests are in computer networking with a focus on measurement and analysis of Internet performance, robustness, security, and topological structure. His most recent interest is in fiber-optic-based techniques for ground motion sensing. He is the Founder to two successful startup companies, and a Fellow of ACM.

Joel Sommers received the Ph.D. degree from the University of Wisconsin–Madison in 2007. He is a Professor of Computer Science with Colgate University, Hamilton, NY, USA. His research interests are in network measurement and traffic analysis and in designing systems for performing repeatable and realistic network experiments.