Crowdsourcing Service-Level Network Event Monitoring

David R. Choffnes†, Fabián E. Bustamante†, Zihui Ge‡
†Northwestern University, ‡AT&T Labs - Research
{drchoffnes,fabianb}@eecs.northwestern.edu, gezihui@research.att.com

ABSTRACT

The user experience for networked applications is becoming a key benchmark for customers and network providers. Perceived user experience is largely determined by the frequency, duration and severity of network events that impact a service. While today's networks implement sophisticated infrastructure that issues alarms for most failures, there remains a class of silent outages (e.g., caused by configuration errors) that are not detected. Further, existing alarms provide little information to help operators understand the impact of network events on services. Attempts to address this through infrastructure that monitors end-to-end performance for customers have been hampered by the cost of deployment and by the volume of data generated by these solutions.

We present an alternative approach that pushes monitoring to applications on end systems and uses their collective view to detect network events and their impact on services - an approach we call Crowdsourcing Event Monitoring (CEM). This paper presents a general framework for CEM systems and demonstrates its effectiveness for a P2P application using a large dataset gathered from BitTorrent users and confirmed network events from two ISPs. We discuss how we designed and deployed a prototype CEM implementation as an extension to BitTorrent. This system performs online service-level network event detection through passive monitoring and correlation of performance in end-users' applications.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network monitoring
C.2.4 **Distributed Systems** Distributed Applications

General Terms

Measurement, Performance, Reliability

Keywords

Service-Level Network Events, Crowdsourcing, Anomaly Detection, P2P

1. INTRODUCTION

The Internet is increasingly used as a platform for diverse distributed services including online multiplayer gaming, content distribution and IPTV. Given the popularity and potential for revenue from these services, their *user experience* has become an important benchmark for service providers, network providers and end users [17].

Perceived user experience is in large part determined by the frequency, duration and severity of network events (e.g., outages, route changes or misconfigurations) that impact a service. There is thus a clear need to detect, isolate and determine the root causes of these service-level network events so that operators can address them in a timely manner, minimizing their impact on revenue and reputation. In this work, we develop a practical monitoring approach enables online detection (within seconds or minutes) of network events impacting the user experience for services at the network edge.

While today's networks generally implement sophisticated infrastructure that detects and issues alarms when core network elements fail, there remains a class of events that often go undetected – the so-called silent failures. Configuration errors (e.g, incorrect ACL settings), routing anomalies (e.g., routing loops), and router bugs (simply because routers are incapable of detecting their own internal failures) are common causes for silent failures that can impact performance for services. Beyond these issues, in-network alarms fail to provide information to help operators understand the impact of network events, nor do they assist in detecting events caused by external ISPs. Despite efforts to use infrastructure to monitor end-to-end performance [16, 17, 36], the cost of deployment, the number of services to monitor and the volume of data generated by these solutions hampers their network visibility and limits their scalability and effectiveness.

This paper presents an alternative approach to detecting, isolating and reporting service-level network events – we call this approach *CEM*, for *Crowdsourcing Event Monitoring*. The key idea behind CEM is to push service-level event monitoring to the end systems where the services are used. Building on end systems has a number of clear advantages. First, the approach provides flexibility in the types of monitoring software that can be installed inside or alongside services, facilitating immediate and incremental deployments. Second, by leveraging the unique perspective of participating end systems, it offers the potential for broad network visibility into an increasingly opaque Internet. Finally, its collaborative model enables a highly robust and more scalable system by drawing from every node's resources and avoiding any centralized components.

Detecting events from the network edge also poses a number of interesting challenges. First, any practical approach must address scalability constraints imposed by managing information from potentially millions of end systems. Second, to assist operators in addressing problems promptly, events should be detected

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.

quickly (i.e., within minutes), isolated to specific networks (e.g., BGP prefixes) and this information should be available to ISPs for troubleshooting and problem mitigation. Last, the approach must facilitate a broad (Internet-scale) deployment of edge-system monitors that ensures user privacy and provides trustworthy event detection. We show how our approach addresses these challenges and present the *Network Early Warning System (NEWS)*, a proof-of-concept implementation for online event detection in BitTorrent.

In Sec. 2, we describe the challenges faced by end-system monitoring in more detail and discuss potential solutions. Sec. 3 addresses the general problem of how to detect network performance events from the edge. Specifically, we develop a framework for our CEM approach in which each end system performs a significant portion of event detection locally, then uses a distributed approach to corroborate their findings.

Demonstrating the effectiveness of any edge-based approach is challenging due to the lack of representative testbeds and the sheer scale and diversity of networks worldwide. In Sec. 4, we address this issue using a large dataset of diagnostic information from edge systems, gathered from users running the Ono plugin [8] for the Vuze BitTorrent client. We use our findings to motivate the design and implementation of the *Network Early Warning System*.

We present results of our evaluation in Sec. 5. In addition to comparing NEWS-detected events with confirmed ones, we demonstrate that our crowdsourcing approach detects network events worldwide, including events spanning multiple networks. Our approach is robust to various parameter settings and incurs reasonably low overhead.

NEWS has already been installed 45,000 times, demonstrating not only the feasibility of our approach for a real application, but also that there are appropriate incentives for widespread adoption (Sec. 6). We are currently investigating other potential hosting applications and services for our CEM approach. Last, to assist with quickly resolving problems causing detected network events, we have implemented *NEWSight*[1] – a system that accesses live event information gathered by NEWS and publishes its results. We are currently beta-testing NEWSight interface with ISPs.

2. CROWDSOURCING MONITORING

Monitoring *service-level events* – issues that impact end-to-end performance and the user experience – is important for users, service providers and network operators. While most networks are instrumented with systems that detect and raise alarms for failures in network elements, their visibility is restricted to a single network and aggregate flows that make it difficult to extract user-perceived performance. Infrastructure-based distributed monitoring can detect events across multiple networks, but this approach is limited by both the fraction of the Internet that remain invisible to traditional measurement techniques and the large number of Internet locations that need to be monitored [5]. Motivated by these limitations, we propose online detection of service-level events through monitoring software that runs inside or alongside applications on the end systems where they are used.

There are a number of important issues that must be addressed in this new context. An edge-based monitoring system must be able to detect sufficiently fine-grained events that impact service performance, while scaling effectively to large numbers of users. Also, any viable deployment model must protect privacy, provide trustworthy results and ensure widespread adoption.

Scalability. As one moves toward the edge of the network, the number of network elements – and thus the opportunities for failures – rapidly increases. With more than 1 billion Internet users worldwide, an edge monitoring system that includes even a small fraction of the population must support millions of hosts. As such, collecting and processing raw performance data using a centralized infrastructure is neither scalable nor practical.

We propose a decentralized approach to event detection in which each host uses its own passively gathered performance information to detect local problems as potential network events. By processing performance data at each monitoring host, CEM facilitates an immediately deployable, scalable monitoring system.

Granularity. Any online network monitoring system should quickly identify network events and determine the affected network region. The time to detect a problem is largely dependent on how frequently a system can sample performance information. For instance, in an end-system monitoring approach like Hubble [16], the number of networks to monitor and the overhead for active measurements limits its resolution to 15 minutes. By passively gathering and processing this information *locally* at each end system, CEM can enable event detection with fine granularity (on the order of seconds) and relatively low CPU and memory overhead. To isolate the scope of network events, CEM correlates multiple locally detected events from the same network region. These regions can be drawn from publicly available BGP prefixes and AS numbers, or richer information such as AS relationships and topologies for cross-network problems.

Privacy. Any implementation of an edge-based network monitoring service is subject to privacy concerns. In previous work that used control-layer information (e.g., BGP updates), network probes (e.g., traceroutes) or aggregate flows to identify network events, privacy is ensured because no personally identifiable information (PII) is exchanged. However, in an edge-based approach that relies on corroboration among multiple vantage points to confirm and isolate events, users must share information about their network views. We demonstrate how edge-based monitoring can remain effective without publishing any PII.

Trust. Most existing network event detection approaches are implemented as closed systems, where third parties are unable or highly unlikely to affect the accuracy or validity of detected problems. In the context of edge-based detection, an open, decentralized approach is vulnerable to attack. For example, one ISP may wish to "poison" the system by introducing false reports of events detected by users in a competitor's ISP. We propose several ways to harden an implementation against such attacks.

Deployment model. Any network event detection approach is limited by the coverage of its deployment. As an application-layer approach CEM is essentially free to deploy and there are practically no limitations as to where participating hosts can be located. The main challenge for CEM is thus gaining widespread adoption. There are a number of ways in which this can be addressed, such as incorporating the software into an OS, providing it as a background service, and/or distributing it as part of networked applications.

In deployments where users must install new software, an appropriate incentive model is essential. Existing approaches to network monitoring have used incentives such as micropayments [27], altruism [31] and mutual benefit [8]. Based on the success of Ono [8], we propose using a mutual benefit model where providers and customers both gain from participation. In this instance, customers (i.e., those running the monitoring software) benefit from immediate notification and logging of network performance problems while network providers receive a more detailed view of their network for improving the quality of their service. This

[1] http://aqualab.cs.northwestern.edu/projects/news/newsight.html

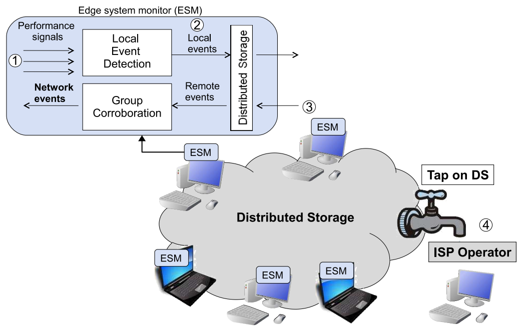


Figure 1: Schematic view of our edge detection approach.

has been sufficient for a prototype implementation of CEM already installed over 45,000 times.

3. CEM FRAMEWORK

The previous paragraphs discussed many of the issues faced by any edge-based monitoring system. In this section we describe how we address these challenges in CEM. Fig. 1 depicts the CEM architecture as a collection of cooperating *edge system monitors* (ESMs). We assume that each ESM has access to one or more sources of performance information that it can use to identify a problem (e.g., transfer rates, latency jitter and dropped packets). Further, each ESM can connect to a distributed storage system to share information about detected events.

As previously mentioned, it is infeasible for edge systems to publish detailed performance data for scalability and privacy reasons. To address this issue, our approach detects events affecting each ESM using only locally available performance data, gathered mostly through passive monitoring (step (1) of the figure). We describe the CEM approach to local detection in Sec. 3.1.

Local event detection presents new design challenges for determining the scope and severity of events. CEM addresses this through a decentralized approach to disseminating information about detected events and the network(s) they impact. In particular, each edge system publishes its locally detected events to distributed storage (step 2), allowing any other participating host to examine these aggregate events. When multiple hosts detect the same problem at the same time in the same network, our approach determines the *relative likelihood* of the detected problem being caused by the monitored network (as opposed to coincidence, for example). We discusses how CEM determines the likelihood that a set of these locally detected problems corresponds to a *widespread* event in Sec. 3.2.

In our architecture, the scope of detected events can be determined by ESMs or via third-party analysis. Each participating host can use the distributed store to capture events corresponding to its network (step 3), then determine whether these local events indicate a network event. Additionally, a third-party system (e.g., run by an ISP) could use the distributed store to perform this analysis (step 4). Thus network customers can monitor the level of service they receive and operators can be informed about events as they occur, expediting root-cause analysis and resolution.

3.1 Local Detection

The first step in CEM is to analyze local performance information to determine whether the monitored host is experiencing a problem. In general, network event detection consists of monitoring a number of *signals* and using a detection algorithm to determine when there may be a problem. In this section, we discuss the types of available performance signals and techniques for detecting local performance events in CEM.

3.1.1 Performance Signals

By pushing detection to end systems located at the edge of the network, CEM can use a wide variety of service-level information to diagnose local performance problems (Table 2). Examples of these performance signals available to any monitored application include flow and path-quality information such as throughput, loss and latencies. Our approach can also incorporate service-specific information to distinguish normal performance changes from potential network events. For instance, P2P file-sharing systems can provide information about whether a transfer has completed and a VoIP application can indicate when there is silence. Our approach can also use system-level information for local event detection. For example, the operating system can provide information about throughput consumed by *all* running applications, allowing CEM to account for the performance impact of concurrent applications. Because these types of information can be gathered passively, they can be sampled frequently so that events are detected as soon as they occur.

Finally, *to assist with diagnosing* network problems, our approach can incorporate limited active measurements such as traceroutes, pings and available bandwidth probes.

3.1.2 Event Detection

CEM uses signals described in the previous section to detect local performance events. The goal of local detection is to provide sufficient information for determining the scope of a problem, i.e., whether the problem is local (isolated to a single ESM) or network-related. To this end, the output of local detection is a summary of each event describing its type (e.g., throughput drop, lost video frame), the time of detection, where in the network it was discovered and how it was detected.

The types of events that can be detected and the appropriate technique to detect them are dependent on the service being monitored. For instance, when monitoring end-to-end throughput for a host (e.g., for video streaming), we show that moving averages can identify drops in transfer rates potentially caused by a network issue like congestion. In the domain of IPTV [23], video quality (among other factors) may indicate network problems. Alternatively, a VoIP application may experience sudden jitter that impacts call quality. Our approach is agnostic to how these events are detected, so long as they correspond to service-level problems.

Correlating local events. Performance changes for monitored services do not necessarily indicate *widespread* problems. In a P2P application like BitTorrent, for example, download rates often drop to zero abruptly. While this may appear at first to be a network problem, it can be explained by the fact that downloading stops when the transfer is complete. Additionally, information gathered at the operating system level can assist in evaluating whether changes in performance are caused by interactions among concurrent applications (e.g., VoIP and P2P file sharing) instead of the network.

As we remove these confounding factors from our analysis, we improve our confidence that a detected problem is independent of the monitored service. Similarly, concurrent events occurring in *multiple* performance signals for a service (e.g., download and upload rates), further increases our confidence that the event is independent of the service.

Publishing local events. After detecting a local event, CEM determines whether other hosts in the same network are seeing the same problem – this requires hosts to share local event detection results (but no local performance data). To ensure scalability, distributed storage (e.g., a DHT) is an appropriate medium for sharing these events.

3.2 Group Detection

Locally detected events may indicate a network problem, but each local view alone is insufficient to determine if this is the case. We now formulate a technique for using multiple hosts' perspectives to confidently identify when a network problem is the likely source.

3.2.1 Corroboration or Coincidence?

To identify events impacting a particular network, CEM first gathers a list of events reported by monitors in that network. This can be done periodically or on demand (e.g., in response to events detected by an ESM). If multiple events occur concurrently in the same network, our approach must determine if these events are likely to be due to the network.

There are a number of reasons why multiple hosts can detect events concurrently in the same network. For example, problems can be isolated to one or more related physical networks due to a router malfunction or congestion. The problem can also be caused by the service driving network activity, e.g., performance from a Web server or from a swarm of P2P users. Finally, simultaneous events can occur simply by chance, e.g., with multiple users experiencing interference on separate wireless routers.

Below, we discuss how CEM accounts for service-specific dependencies and correlated events that occur by coincidence. After accounting for service dependencies, our approach tests the null hypothesis that each host experiences events *independently* and not due to network problems. By comparing this value to the observed rate of local events occurring concurrently for hosts in a network, CEM can determine the *relative likelihood* of the detected problem being caused by the network instead of by chance. We quantify this with a *likelihood ratio*, which has been used, e.g., in the field of medicine for diagnostic testing to determine the probability that a condition (e.g., a disease) is present.

Accounting for dependencies. The first step in the likelihood analysis is to determine the probability that each of the N participating hosts detects local service problems independently. Thus, for each host h we produce a series $S_h = \{s_{h,i}, s_{h,i+1}, ..., s_{h,j}\}$ for the time period $T = [i, j]$, such that at time t, $s_{h,t} = 1$ if a local event was detected and $s_{h,t} = 0$ otherwise. During the time period T, we use the observed detection rate to estimate the probability of host h detecting a local event in any given bucket as:

$$L_h = \frac{1}{j-i}\sum_{t=i}^{j} s_{h,t}$$

To control for service-specific dependencies, any set of hosts whose performance is mutually dependent during a time interval $(i-1, i]$ are treated as the same host during that interval for the purpose of the analysis. Thus, such hosts do not corroborate each other's events. For example, in the case of a P2P file-sharing application, performance problems seen by peers that are downloading the same file and connected to each other are *not* treated as independent events. Besides such explicit dependencies, our approach can incorporate automatically generated ones from a tool like Orion [6].

After this step, our approach quantifies the probability of n (out of N) independent hosts detecting an event at the same time *by coincidence*, i.e., the joint probability that for a given time t,

$$\sum_h s_{h,t} \geq n.$$

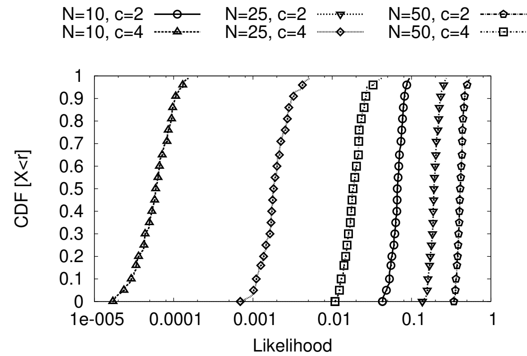


Figure 2: How increasing the number of hosts corroborating an event decreases the likelihood of it occurring by chance.

In general, this is calculated as the union probability of any one of N participating hosts seeing an event:

$$P(\bigcup_{h=1}^{N} L_h) = \sum_{h=1}^{N} P(L_h) - \sum_{j>h=1}^{N} P(L_h \cap L_j) + ... \\ + (-1)^{n-1} P(L_1 \cap ... \cap L_N) \quad (1)$$

We are testing the hypothesis that the events are independent, so we can simplify the union probability:

$$P(\bigcup_{h=1}^{N} L_h) = \sum_{h=1}^{N} P(L_h) - \sum_{j>h=1}^{N} P(L_h)P(L_j) + ... \\ + (-1)^{n-1} P(L_1)...P(L_N) \quad (2)$$

This equation gives the union probability for *any one* host seeing an event, i.e., without corroboration. Generally, this is much larger than the probability that at least n hosts $(1 < n \leq N)$ in the network will see concurrent events. To calculate this, we peel off the first $n-1$ terms of Eq. 2. For example, the probability that at least two hosts will see concurrent events is:

$$P(\bigcup_{j>h=1}^{N} L_h \cup L_j) = \sum_{j>h=1}^{N} P(L_h)P(L_j) \\ - \sum_{k>j>h=1}^{N} P(L_h)P(L_j)P(L_k) \\ + ... + (-1)^{n-1} P(L_1)...P(L_N) \quad (3)$$

Effect of corroboration. Intuitively, our confidence in a detected event being due to the network increases with (i) the number of hosts detecting the event and (ii) the number of independent performance signals indicating the event. We now quantify the impact of these factors through a simulation of a region of interest (e.g., a BGP prefix) with N hosts. Each of these hosts provides multiple performance signals as described in Sec. 3.1.1. The probability of host h witnessing an event in a signal, L_h, is chosen uniformly at random in the range $0.005 \leq L_h \leq 0.05$. We then determine the probability of c hosts $(1 < c \leq 5)$ seeing an event by coincidence for networks with $N = 10, 25, 50$ hosts, and we compare this value with the probability of any *one* host seeing an event. For each setting, we run 100 randomly generated networks.

Fig. 2 uses a CDF to show the effect of varying the size of the network on the probability of seeing correlated events by coincidence. In general, the figure confirms the intuition that relatively large numbers of monitored hosts are unlikely to see network events at the same time simply by coincidence. More concretely, for $N = 50$, four hosts are an order of magnitude less likely to see simultaneous events than two hosts. We observed a similar effect when varying the number of signals detecting local events – the more signals experiencing performance events concurrently, the less likely that the events are occurring by chance. When $N = 25$, e.g., it is three orders of magnitude less likely that five peers experience synchronized events in three performance signals than in one signal.

Relative likelihood. As we discussed at the beginning of this section, we would like to determine the relative likelihood that concurrent local events are due to the network and not happening by coincidence. To quantify this, we propose using a *likelihood ratio*, i.e., the ratio of the observed probability of concurrent events to the probability of concurrent events happening independently.

To derive this ratio, our approach first takes events seen by n peers in a network at time t, and finds the union probability P_u that the n (out of N) peers will see a performance problem at time t by coincidence. Next, CEM determines the empirical probability (P_e) that n peers see the same type of event (i.e, by counting the number of time steps where n peers see an event concurrently and dividing by the total number of time steps in the observation interval, I). The likelihood ratio is computed as $\mathrm{LR} = \mathrm{P_e}/\mathrm{P_u}$, where $\mathrm{LR} > 1$ indicates that detected events are occurring more often than by coincidence for a given network and detection settings. We consider these to be events indicative of a network problem.

3.2.2 Problem Isolation

When many hosts in a network detect an event at the same time, it is usually a problem best addressed by the responsible network operators. In such cases, our approach should be able to identify the network(s) affected by the event so as to provide the necessary information for operators to determine the root cause and fix the problem.

In our approach, the scope of a problem is explicitly determined by the grouping of hosts for the likelihood analysis. As such, the approach supports localization of problems using structural information about the organization of networks and their geographic locations. For instance, it can use events detected by hosts in the same routable BGP prefix or ASN, and use geographic information to localize events to cities and countries. Further, CEM can use an AS-level Internet graph to localize network issues to upstream providers or a router-level graph to isolate problematic routers and links.

4. IMPLEMENTING CEM

The previous section described our CEM approach for detecting events from edge systems. Designing, deploying and evaluating CEM poses interesting challenges given the absence of a platform for experimentation at the appropriate scale.

A promising way to address this is by leveraging the network view of peers in large-scale P2P systems. P2P systems use decentralization to enable a range of scalable, reliable services and are so prevalent that reports indicate they generate up to 70% of Internet traffic [30]. By avoiding the need to deploy additional infrastructure and offering hosts that are already cooperating, these systems are an appealing vehicle for monitoring – one that grows naturally with the network [9, 36].

Based on these advantages, we choose to design and evaluate a prototype implementation of CEM in a large P2P system. To guide the design of our prototype and evaluate its effectiveness at scale, we take advantage of a large edge-system dataset comprising traces of BitTorrent performance from millions of IP addresses. The following paragraphs describe this unique dataset, a collection of confirmed network problems we rely on for evaluation, and a particular case study we use in our presentation. We close the section describing the *Network Early Warning System (NEWS)*, our prototype edge-based event detection system that uses BitTorrent as a host application. NEWS is currently deployed as a plugin for the Vuze BitTorrent client [34], to facilitate adoption and to piggyback on the application's large user base.

Category	Number (Pct of total)
Number of users	1,000,000
Countries	212
IP addresses	4,300,000
Prefixes	72,100
Autonomous systems (ASes)	8,700
IPs behind middleboxes	$\approx$ 82.6%

Table 1: Summary of our P2P vantage points.

Building on P2P systems to provide network monitoring is not without limitations. For one, each monitor contributes its view only while the P2P system is active, which is subject to user behavior beyond our control. Second, the end system may run other applications that interfere with P2P applications and event detection. Finally, some event detection techniques require access to information not accessible to a P2P application, e.g., system calls.

4.1 Datasets

The following paragraphs present the data collected from BitTorrent and a set of confirmed network problems from two ISPs.

4.1.1 BitTorrent traces

The BitTorrent traces we use are gathered from users of the Ono plugin for the Vuze BitTorrent client.[2] Ono implements a biased peer selection service aimed at reducing the amount of costly cross-ISP traffic generated by BitTorrent without sacrificing system performance [8]. Beyond assisting in peer selection, the software allows subscribing volunteers to participate in a monitoring service for the Internet. With more than 1,000,000 users today, distributed in 212 countries, this system is the largest known end-system monitoring service. The following paragraphs describe the data collected; summary information about Ono users is in Table 1.

Trace details. Our dataset consists of transfer rate for each connection and cumulative transfer rates (over all connections), all sampled once every 30 seconds. Besides this, the dataset includes protocol-specific information such as whether each peer is "leeching" (both downloading and uploading) or "seeding" (only uploading), the total number of leechers and seeds, and information about the availability of data for each download. The complete list of collected signals is in Table 2. Note that this collection is for our design and evaluation only and it is **not** required for NEWS event detection.

Edge coverage. Any dataset is subject to limits in the coverage of its measurement hosts. The dataset we use currently contains connection information from users to more than 390,000,000 peer IPs; collectively, its users monitor more than 17 million paths per day. Ono's user base has grown to over 72,100 prefixes (covering nearly every country) in less than three years. Collectively, these users have established connections to peers in over 222,000 routable prefixes and 21,800 ASNs.

Besides covering many paths, the dataset reaches true edge systems located in portions of the Internet not accessible to existing distributed research and monitoring platforms. For example, over 80% of the user IPs correspond to middleboxes (i.e., they are assigned private IP addresses). Further, we find that the P2P traffic in these traces covers *paths* invisible to public views. Specifically,

[2]Users are informed of the diagnostic information gathered by the plugin and are given the chance to opt out. In any case, no personally identifiable information is ever published.

we map three months of BitTorrent traffic data to AS pairs, then use 13 months of publicly available BGP data to map the traffic to AS-level paths. We find that *a large majority* of BitTorrent traffic in this dataset cannot be mapped to a public BGP path [7]. These results are consistent with findings by Chen et al. [5] demonstrating that public views miss large portions of the Internet topology covered by P2P users.

4.1.2 *Confirmed network problems*

Evaluating the effectiveness of a network event detection approach requires a set of events that *should* be detected, i.e., a set of ground-truth events. Among the different strategies adopted by previous studies, manual labeling – where an expert identifies events – is the most common [28].

As one example, we use publicly available event reports from the British Telecom (BT Yahoo) ISP[3] in the UK. This site identifies the times, locations and nature of network problems. During the month of April, 2009 there were 68 reported problems, including Internet and POTS events.

In addition, we use network problems reported from a large North American ISP. For nondisclosure reasons, we cannot report absolute numbers for these events.

Despite its many advantages, the set of labeled problems for a network is restricted to events that can be detected by the in-network monitoring infrastructure, using currently deployed techniques, or generated by user complaints. Further, human experts can introduce errors and disagreement, e.g., in reporting the time and duration of an event. As a result, we can determine when confirmed events are detected by our approach, but cannot draw strong conclusions about false positive and negative rates.

4.2 Case study

To assist with the presentation of NEWS, we pick one of the events from the previous section. Specifically, we demonstrate how NEWS detects a reported problem in BT Yahoo: On April 27, 2009 at 3:54 PM GMT, the network status page stated, "*We are aware of a network problem which may be affecting access to the internet in certain areas...*" The problem was marked as resolved at 8:50 PM.

Fig. 3 presents a scatter plot timeline of upload rates for peers located in the same routable prefix in BT Yahoo (81.128.0.0/12) during this event, which is depicted as a shaded region. Each point in the graph represents an upload-rate sample for a single peer; different point shapes/colors represent signals for different peers. The figure shows that multiple peers experience reduced performance between 10:54 and 16:54, while another set of peers see a significant drop in transfer rates at 14:54. These are consistent with the reported event, when accounting for delays between the actual duration of an event and the time assigned to it by a technician. Further, we see that there were two distinguishable network problems corresponding to the single generic report.

4.3 Network Monitoring from BitTorrent

We now discuss key design aspects of NEWS, a prototype edge-system monitor for BitTorrent. Throughout this discussion we use the confirmed BT Yahoo event in Fig. 3 to explain our design choices. With respect to the design challenges listed in Sec. 2, we address scalability and granularity through our local event detection and group corroboration approach; the remaining issues of privacy, trust and adoption are covered in the subsequent sections. We provide low-level implementation details in Sec. 6.

[3] http://help.btinternet.com/yahoo/help/servicestatus/

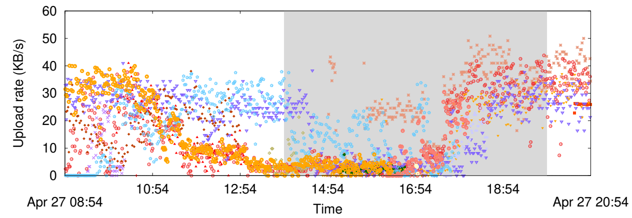


Figure 3: Upload rates for peers in a routable prefix owned by British Telecom during a confirmed disruption (shaded region). Best viewed in color.

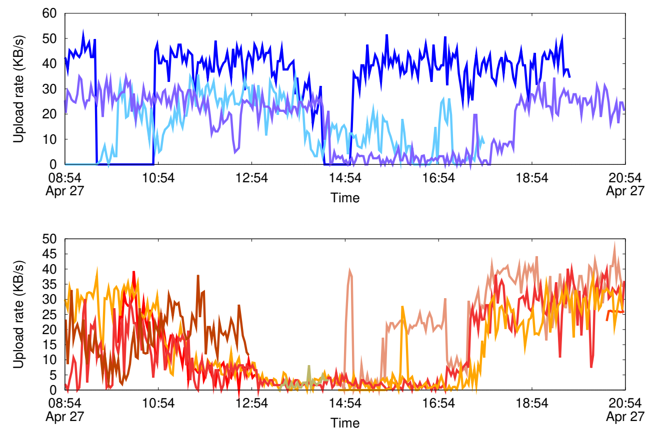


Figure 4: Moving averages facilitate identification of separate network events affecting transfer rates for two groups of peers during the same period shown in Fig. 3. Best viewed in color.

4.3.1 *Local Event Detection*

Any CEM system must define what constitutes a service-level event that could be due to a network problem. In NEWS, we define these to be unexpected drops in end-to-end throughput for BitTorrent, which corresponds to steep drops in the time series formed by BitTorrent throughput samples.

Event detection in BitTorrent. NEWS employs the simple, but effective, moving average technique for detecting edges in BitTorrent throughput signals. Given a set of observations $V = \{v_1, v_2, ..., v_n\}$, where v_i is the sample at time i, the technique determines the mean, μ_i, and the standard deviation, σ_i of signal values during the window $[i - w, i]$. The moving average parameters are the observation window size for the signal (w) and the threshold deviation from the mean ($t \cdot \sigma$) for identifying an edge. Given a new observation value v_{i+1} at time $i + 1$, if $|v_{i+1} - \mu_i| > t \cdot \sigma_i$, then an edge is detected.

To demonstrate visually how moving averages facilitate edge detection, Fig. 4 plots the 10-minute averages of upload rates for two groups of affected peers extracted from Fig. 3. Using these averages, it becomes clear that there is a correlated drop in performance among a group of three peers at 14:54 (top graph), while the bottom graph shows a series of performance drops, the first near 10:54 and the last around 13:00. Both groups of peers recover around 17:30.

The window size and deviation threshold determine how the moving average detects events. Tuning the window size (w) is analogous to changing how much of the past the system remembers when detecting events. Assuming that the variance in the signal is constant during an observation window, increasing the number of

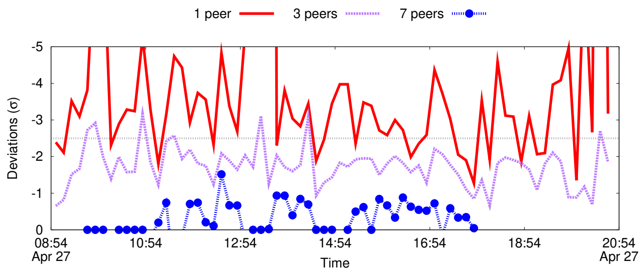


Figure 5: Timeline of the maximum performance drops for at least n peers (moving average window size of 10, $n = 1, 3, 7$). Deviations for any one peer are highly variable; those for seven peers rarely capture any performance drops. The peaks in deviations for three peers correspond to confirmed events.

samples improves our estimate of σ and thus detection accuracy. In general, however, σ varies over time, so increasing the window size reduces responsiveness to changes in σ.

The detection threshold $(t \cdot \sigma)$ determines how far a value can deviate from the moving average before being considered an edge in the signal. While using σ naturally ties the threshold to the variance in the signal, it is difficult *a priori* to select a suitable value for t. If our approach to local detection is viable, however, there should be some threshold $(t \cdot \sigma)$ for identifying peers' local events that correspond to network ones. To demonstrate this is the case, Fig. 5 shows how deviations behave over time for peers experiencing the network problems illustrated in Fig. 4, using a window size of 10. Specifically, each curve shows the maximum drop in performance (most negative deviation) seen by at least n peers in the network at each time interval. Because these deviations vary considerably among peers, we normalize them using the standard deviation for the window (σ).

The top curve, where $n = 1$, shows that the maximum deviations from any one peer produces a noisy signal with a wide range of values, and features of this signal do not necessarily correspond to known network problems. The bottom curve, where $n = 7$, shows that it is rarely the case that seven peers see performance drops simultaneously, so features in this signal are not useful for detecting events during this period. Last, the middle curve, $n = 3$, produces a signal with a small number of peaks, where those above 2.5σ correspond to real network problems. This suggests that there are moving-average settings that can detect confirmed problems in this network. In Sec. 4.3.2, we show how NEWS can extract network events from a variety of settings, using the analysis from Sec. 3.2.

Confounding factors. A drop in a BitTorrent host's throughput signal is not necessarily due to network events (Sec. 3.2). Thus, when monitoring BitTorrent it is essential to use service-specific information to distinguish expected behavior from network events.

Table 2 lists the information available when monitoring BitTorrent. NEWS uses several of these signals to eliminate well known confounding factors. For instance, NEWS tracks the transfer states of torrents and accounts for the impact of download completion. To eliminate performance problems due to the application (as opposed to the network), such as missing torrent data or high-bandwidth peers leaving a swarm, all peers connected to the same torrent are treated as the same peer. As another example, NEWS accounts for correlations between the number of peers connected to a user and the average transfer rate for each peer.

NEWS also requires multiple performance signals to see concurrent events before publishing an event. As we discussed in Sec. 3.2, improving our confidence that the event is independent of the application also improves our confidence that it is caused by the network.

Signals General to P2P Systems	
Overall upload rate	Overall download rate
Per-connection upload rate	Per-connection download rate
Connected hosts	RTT latencies
Signals Specific to BitTorrent	
Availability	Connected seeders/leechers
Number available leechers	Number available seeds
Number active downloads	Number active uploads

Table 2: Signals available when monitoring from BitTorrent.

When detecting an event, NEWS must not only determine that there is a problem with *a* network, but specifically identify *the host's network* as the one experiencing the problem. If a host's connections were biased toward a single AS, for example, it would be unclear if detected problems were specific to the host's AS or the biased one. To explore this issue, we determine the number of routable prefixes visited per hour by each peer's connections during a 6-day period, then find the average of these hourly totals for each peer. We find that the vast majority of our vantage points (99%) connect to peers in four or more prefixes during an average hour-long period; the median number is 137. This range indicates that it is unlikely that problems in remote networks would be falsely interpreted as problems in the host's network.

4.3.2 Group Corroboration

As discussed in Sec. 3.2, after detecting local events, CEM determines the likelihood that the events are due to a network problem. Thus, once a local event has been detected, NEWS publishes local event summaries to distributed storage so that participating hosts can access detected events quickly.

We now apply this likelihood analysis to the events in BT Yahoo as described in Sec. 4.2. Recall that we would like to detect synchronized drops in performance that are unlikely to have occurred by chance. To that end, we determine the likelihood ratio, $\mathrm{LR} = \mathrm{P_e}/\mathrm{P_u}$, (Sec. 3.2.1). For this analysis, we use one month of data to determine P_e and P_u; as we show in the following paragraphs this is sufficient to detect confirmed network events.

Figure 6 depicts values for LR over time for BT Yahoo using different local event detection settings. In both figures, a horizontal line indicates $\mathrm{LR} = 1$, which is the minimum threshold for determining that events are occurring more often than by chance. Each figure shows the LR values for up to three local signals (e.g., upload and download rates) that see concurrent performance problems for each peer. As previously mentioned, the more signals seeing a problem, the more confidence we can attribute to the problem not being the application.

In Fig. 6 (top), we use a detection threshold of 1.5σ and window size of 10. Using such a low threshold not surprisingly leads to many cases where multiple peers see synchronized problems (nonzero LR values), but they are not considered network problems because $LR < 1$. Importantly, there are few values above $\mathrm{LR} = 1$, and the largest corresponds to a performance drop potentially due to congestion control, since it occurs when peers have simultaneously saturated their allocated bandwidth after the confirmed network problem is fixed.

Fig. 6 (bottom) uses a detection threshold of 2.2σ and window size of 20. As expected, the larger threshold and window size detect fewer events. In this case, *all values that appear above* $\mathrm{LR} = 1$ *correspond to the known network problems*, and they are all more than twice as likely to be due to the network than coincidence.

These examples demonstrate that our approach is able to reli-

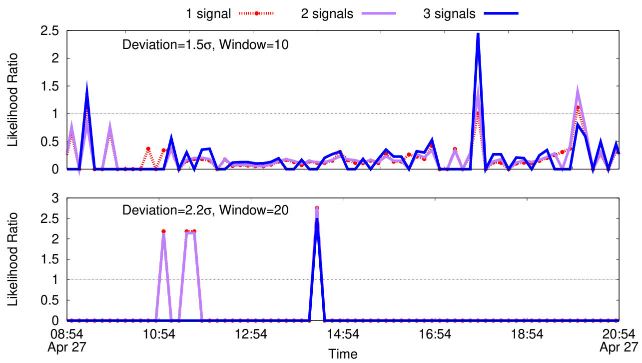


Figure 6: Timeline showing the likelihood ratio for different moving average settings. In each case, there are few events with $LR > 1$, and nearly all correspond to confirmed events.

ably detect different problems with different parameter settings. They also suggest that the approach generally should use *multiple* settings to capture events that occur with different severity and over different time scales. Because CEM uses passive monitoring approach, an implementation can use several detection settings in parallel with minimal additional overhead, then use the likelihood ratio threshold as a single parameter to select cases where each setting identifies likely network problems. As we discuss in Sec. 6, we use this strategy in our current implementation.

4.3.3 Privacy and Trust

Any implementation of a network monitoring service is subject to important considerations such as privacy and trust. To ensure user privacy, NEWS does not publish any personally identifiable user information (e.g., IPs or download activity). Rather, it reports only detected events and assigns per-session, randomly generated IDs to distinguish events from different users.

While this approach to ensuring privacy is appealing for its simplicity, it opens the system to attack by malicious parties. For example, one ISP may wish to "poison" the system by introducing false event reports for a competitor's ISP. There are several ways to harden an implementation against such attacks. First, we include each host's L_h in the event reports, and recall that larger L_h leads to a smaller contribution to the likelihood (Eq. (3)). This mitigates the effect of an attacker generating a large volume of false event reports using NEWS. While an attacker could forge L_h, any participating host could detect that it is inconsistent with the number of reports placed in the distributed store. In addition, simple rate-limiting can be applied to a centralized attacker and a Sybil-like attack can be mitigated with secure distributed storage [4]. Such an approach eliminates *anonymity* by assigning identities to users; however, the privacy of the details of their network activity is maintained.

4.3.4 Participation Incentives

In general, our approach does not require incentives for adoption, e.g., if applications are deployed with instrumentation by default. For our prototype system in BitTorrent, however, the deployment model relies on users installing third-party software.

Based on the success of Ono [8], we propose using a similar *mutual benefit* incentive model. The incentive for users to install Ono is based on users' selfish behavior – the software offers potentially better download performance while at the same time reducing cross-ISP traffic. To encourage NEWS adoption, we rely on this selfish behavior by offering users the ability to ensure they receive the network performance they pay for. Similar incentives have been successfully used by the Grenouille project[4] (20,000 users) and various network neutrality projects (e.g., Glasnost [11], installed more than 350,000 times).

Specifically, NEWS users contribute their network view (at essentially no cost) in exchange for early warnings about network problems that impact performance. As these problems may indicate changes in ISP policies, violations of SLAs or ISP interference, such warnings provide a mechanism for users to ensure that the Internet service they pay for is properly provided. This has been sufficient incentive for NEWS, which has already been installed over 45,000 times.

5. EVALUATION

We use one month of data gathered from BitTorrent users to answer key questions about the CEM approach as implemented in NEWS. We first demonstrate its effectiveness using confirmed events from two large ISPs. We show that using a popular P2P service as a host application can offer sufficient coverage for edge-system event detection and present a summary of results from our detection algorithm on networks worldwide. Last, we evaluate the robustness of NEWS to parameter settings.

NEWS is designed to detect any event impacting the performance of BitTorrent hosts at the edge of the network. On the other hand, confirmed events from ISPs are typically restricted to significant outages. Thus, one cannot draw strong conclusions about false positives/negatives and the results presented here are necessarily limited to the kinds of events detectable by BitTorrent users.

5.1 Effectiveness

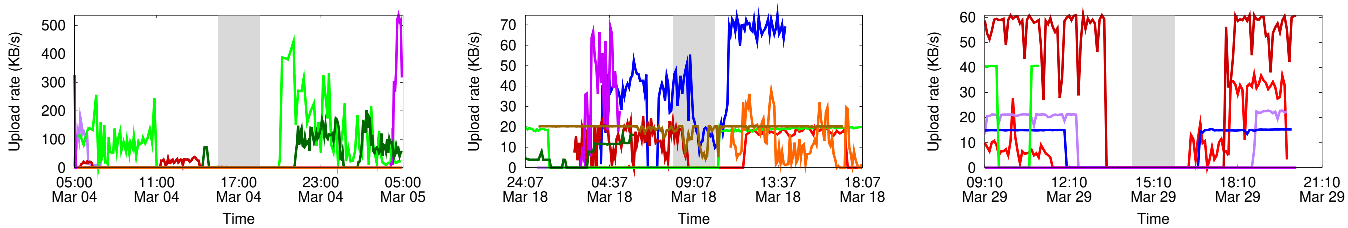
To evaluate the accuracy of our approach we compare its results against labeled network problems from two ISPs, our ground truth. For the purpose of comparing these datasets, if an event was detected within 2 hours of a reported time, we count it as being the same event (based on reporting delays in our case study).

For BT Yahoo, of the 181 events detected by our approach, 54 are confirmed network problems – covering nearly 80% of the labeled events. Our edge-based approach detected an additional 127 events; although these are not confirmed problems, we caution against inferring false positive rates, as the reported events are based on those detectable from existing monitoring systems. Still, even in the unlikely case that these events are not real, the average false alarm rate (just over 4 events per day) is manageable.

For a North American ISP, our approach detected a variety of performance events, some of which were confirmed outages. For cases where there was a drop in performance but not an outage, we were not able to obtain ground truth information. Figure 7 shows a sample of three cases of events detected by our approach: (a) an confirmed outage, (b) a non-outage performance event (unconfirmed) and (c) an unconfirmed outage.[5] Table 3 presents summary results for this ISP. Our approach was able to detect half of the largest outages (column 3). In column 4, we show the number of outages that appeared to affect monitored hosts, but not in sufficient numbers to validate the event. In addition to these events, our approach detected 41 events during the 1-month period. Unfortunately, the ISP did not have sufficient information to confirm or deny them.

[4] `http://www.grenouille.com`

[5] An *outage* refers to loss of network connectivity; *non-outages* indicate connectivity with significantly reduced performance.



(a) Confirmed outage. (b) Unconfirmed non-outage event. (c) Unconfirmed outage.

Figure 7: Timelines depicting events (centered in the figures) affecting transfer rates for peers in a North American ISP. Throughput drops are misaligned due to periodic data reports being dropped during the outages. Best viewed in color.

Affected customers	Pct of total events	Detected	Possible
$C \geq 10000$	53%	50%	38%
$10000 > C \geq 1000$	40%	0%	67%
$C < 1000$	7%	0	0

Table 3: Comparison with events from a North American ISP.

(a)

ISP	Users	Events
Deutsche Tel.	6760	69
HTP	3652	112
HanseNet	3216	17
Neuf Cegetel	2821	108
Arcor	2245	29
Cableuropa	1999	245
Proxad/Free	1769	176
France Tel.	1688	31
Tel. Italia	1651	20
Telefonica	1337	27

(b)

ISP	Users	Events
Cableuropa	1999	245
BTnet UK	1277	182
Proxad/Free	1769	176
HTP	3652	112
Neuf Cegetel	2821	108
Deutsche Tel.	6760	69
Telewest	237	50
Pakistan Tel.	729	46
Comunitel	197	45
Mahanagar Tel.	454	42

Table 4: Top 10 ISPs by users (a) and by events (b).

5.2 Coverage

Edge-system event detection requires a sufficient number of peers to concurrently use a network for the purpose of corroboration. To evaluate whether using a popular P2P service as a host application can offer sufficient coverage for edge-system event detection, we calculated the maximum number of peers simultaneously online (and active) for each network in our dataset. Figure 8 plots a CDF of these values for each routable prefix and ASN. On average, the number of simultaneous online peers per routable prefix is 3 and per ASN is 7. Even though our installed base of users represents less than 0.4% of all BitTorrent clients, we find that this offers sufficient coverage (three or more peers concurrently online) for more than half of the ASNs that we study.

5.3 Worldwide events

Having shown the effectiveness of NEWS, this section characterizes network events that detected worldwide, using a threshold $LR = 2$ (as guided by Fig. 6). For a one-month period, our approach detected events in 38 countries across five continents, emphasizing how edge-based detection can achieve broad network coverage worldwide. In Table 4(a), we list the top 10 ISPs in terms of the number of users participating in our study, and the number of events detected in each of these ISPs.

Because different networks provide different quality of service [22], increasing the number of peers should not necessarily increase the number of events detected. As the table shows, there is indeed little correlation between the number of vantage points in a network and the number of detected performance events.

Relationship	Min. ASNs	# cases	# countries
Customer-Provider	2	370	5
Customer-Provider	3	7	2
Peer-Peer	2	487	7

Table 5: Number of cross-network events as inferred from single-network events. The first column indicates the AS relationship; the following columns specify the minimum number of ASes and countries affected.

Time (GMT)	Provider(s)	Affected ASes	Country
Apr 16, 13:35	8218	15557,12876,12322	FR
Apr 17, 12:40	1267	16338,3352,6739	ES
Apr 30, 01:15	10396,7910	12357,16338,12715	ES

Table 6: Example cross-network events corresponding to the second row of Table 5.

Table 4(b) shows the top 10 ISPs in terms of the number of events detected, covering ISPs of varying size in Europe and Asia. We note that with the exception of the top three ISPs, our approach generates fewer than four detected events per day. Thus, a deployment of our approach should report events at a reasonable rate – one that will not overwhelm network operators and users.

5.4 Cross-network events

A unique advantage of an edge-system based monitoring approach like NEWS is its ability to detect network problems affecting multiple ISPs, e.g., due to provider or peering link issues. The following paragraphs describe cross-network events detected during the period of study.

We focus on events due to issues with upstream providers or peers. To this end, we find events that occur in multiple ASes at the same time, then determine which of these ASes have a peering relationship or identical providers (based on the AS topology generated by Chen et al. [5]). Events that occur within 30 minutes of each other are considered the same, and we conservatively consider AS relationships only for those ASes in the same country.

Table 5 summarizes our results. The first row indicates that when searching for at least two ASNs with the same provider, there are 370 cases in five countries. In the second row, we use a more restrictive search where we require that at least three ASNs having the same provider see synchronized network events – such events are, as expected, much rarer; a sample of them is provided in Table 6. Finally, the last row indicates that there is a significant number of peering ASNs that see synchronized problems.

5.5 Robustness

As discussed in Sec. 4.3.2, the likelihood ratio (LR) can be seen as a parameter for distilling *network* events from locally detected

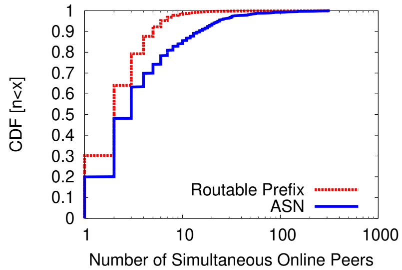
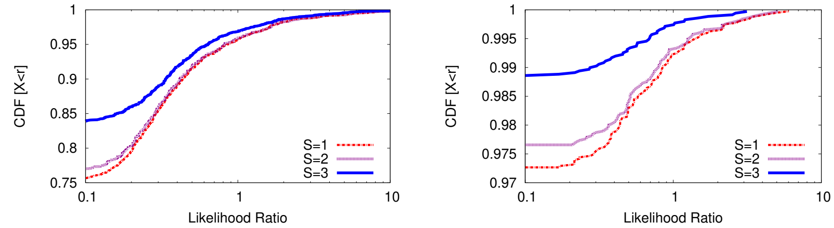


Figure 8: Concurrent online peers in this study.



(a) Dev.=1.5σ, Win.=10 (b) Dev.=2.2σ, Win.=20

Figure 9: Likelihood ratios are robust to various parameter settings; they detect network problems at most 2.15% of the time for small deviations and window sizes (a) and at most 0.75% of the time for larger ones (b).

ones. As such, the number of network events detected using an LR threshold should not significantly change with different local detection settings.

Fig. 9 plots CDFs of LR values for BT Yahoo during one month. In Fig. 9(a), we plot LR values for $W = 10$ and $\sigma = 1.5$ and Fig. 9(b) plots the same for $W = 20$ and $\sigma = 2.2$. Settings for small deviations and small window sizes logically yield a larger number of ratio values greater than one (2.15% of the time) whereas larger deviations and windows yield a smaller number (0.75%). Generally, such cases (where concurrent events occur more often than chance) are rare for different detection parameters, suggesting that LRs are indeed robust to detection settings.

6. DEPLOYMENT DETAILS

The NEWS plugin for Vuze is written in Java and the core classes for event detection comprise ≈1,000 LOC. Released under an open-source (GPL) license, our plugin has been installed over 45,000 times since its release in March, 2008. In the rest of this section, we discuss details of our NEWS implementation in its current deployment. In addition to providing specific algorithms and settings that we use for event detection, our discussion includes several lessons learned through deployment experience.

Local detection. NEWS detects local events using the moving average technique discussed in Sec. 4.3.1, which uses the window size (w) and standard-deviation multiplier (t) parameters to identify edges in BitTorrent transfer rate signals. NEWS currently uses $w = 10, 20$ samples and $t = 2.0, 2.5, 3.0$ in parallel, dynamically configurable settings that have shown to be most effective.

In practice, we found that BitTorrent often saturates a user's access link, leading to stable transfer rates and small σ. As a result, a moving-average technique may detect events in the throughput signals even when there are negligible *relative* performance changes. We address this issue in NEWS by including a secondary detection threshold that requires a signal value to change by at least 10% before detecting an event.

Throughput signals also undergo phase changes, during which a moving average detects consecutive events. NEWS treats these as one event; if enough consecutive events occur, we assume that the signal has undergone a phase change, and reset the moving average using only signal values after the phase change.

After detecting a local event, NEWS generates a report containing the user's per-session ID, w, t, a bitmap indicating the performance signals generating events, the current event detection rate (L_h), the time period for the observed detection rate, the current time (in UTC) and the version number for the report layout. The current report format consumes 38 bytes.

The plugin disseminates these reports using Vuze's built-in Kademlia-based DHT [24], a key-value store that maintains multiple values for each key. To facilitate group corroboration of locally detected events, we use network locations as keys and the corresponding event reports as values.

In our deployment we found variable delays between event detection and reporting, in addition to significant clock skew. To address these issues, NEWS uses NTP servers to synchronize clocks once per hour, reports event times using UTC timestamps and considers any events that occurred within a five-minute window when determining the likelihood of a network event occurring.

Group corroboration. After a NEWS peer detects a local event, it performs corroboration by searching the DHT for other event reported in each of the host's regions – currently its BGP prefix and ASN.[6] Before using a report from the DHT for corroboration, NEWS ensures that: (1) the report was not generated by this peer; (2) the report was generated recently; and (3) the standard-deviation multiplier and window size for detecting the event match a local detection setting.

If these conditions are met, the report's ID is added to the set of recently reported events for the corresponding detection setting. If a peer finds events from at least three other concurrent peers (a configurable threshold), it uses Eq. 3 to determine the likelihood of these events happening by coincidence. Using the information gathered from events published to the DHT over time, the peer can calculate the likelihood ratio described in Sec. 4.3.2. If the likelihood ratio is greater than two (also configurable), the monitor issues a notification about the event.

NEWS peers read from the DHT after detecting a local event in order to corroborate their finding. To account for delays between starting a DHT write and its value being available for reading, NEWS sets a timer and periodically rechecks the DHT for events during a configurable period of interest (currently one hour).

Last, our likelihood ratio calculation requires access to the local detection rate for each online peer. To ensure it is available, each peer writes its local detection rate to distributed storage at least once per hour, regardless of whether it has yet detected a local event during its current session.

Third-party interface. Beyond end-users, network operators should be notified to handle service-level events. With this in mind, we have implemented a DHT crawler (*NEWS Collector*) that any third party can run to gather in and analyze local event reports.

To demonstrate its effectiveness, we built *NEWSight* – a system that accesses live event information gathered from NEWS Collector and publishes detected events through a public Web interface. NEWSight allows network operators to search for events and register for event notifications. Operators responsible for affected networks can confirm/explain detected events.

[6]Vuze already collects the host's prefix and ASN; we are currently adding support for whois information.

Whereas NEWS crowdsources event detection, NEWSight can be viewed as an attempt at crowdsourcing network event labeling. Confirmed events can help to improve the effectiveness of ours and similar approaches – addressing the paucity of labeled data available in this domain [28]. We are currently beta-testing this interface with ISPs; the interface and its data are publicly available.

Overhead for participating hosts NEWS passively monitors performance and uses low-cost event-detection techniques, so there is negligible overhead for detecting local events. The primary sources of overhead are calculating the union probability (CPU/memory) and sharing locally detected events (network). We now demonstrate that these overheads are reasonably low.

For determining the union probability, the formula in Eq. (3) specifies ${}_nC_{n/2}$ (n choose $n/2$) operations, where n is the number of hosts in the network having a nonzero probability of detecting an event.[7] We use Miller's algorithm [25], an optimal trade-off between memory, $O(n)$, and computation, $O(n^3)$. While a substantial improvement over a naïve implementation, its processing overhead can still be significant for a large n (e.g., $n > 50$). To bound this, we limit the number of hosts used in the computation to the H hosts with the largest L_h. In this way, we conservatively estimate an upper bound for P_u for the full set of n hosts.

The other source of overhead is using distributed storage to share locally detected events. While this overhead is variable and dependent on factors including the target network and detection settings, we found it to be reasonably low for many settings. For example, our analysis shows that read and write operations are performed by each host with average frequencies on the order of several minutes, and in the worst case once every 30 seconds (less than 4 B/s for each peer in the BT Yahoo network).

7. RELATED WORK

The problem of detecting network events (or anomalies) has attracted a large number of research efforts. In this context, CEM is a framework for online detection of network events that impact performance for applications running on end systems. This section classifies key properties of event detection systems and describes how CEM relates to previous work in these areas.

Crowdsourcing. Central to our approach is the idea of crowdsourcing event detection to ensure good coverage and accuracy at the scale of hundreds of thousands of users. This model has successfully enabled projects that include solving intractable [33] or otherwise prohibitively expensive problems [1] using human computation. Unlike these examples, our system passively monitors network activity from each member of a crowd, but it does *not* require human input. Dash et al. [10] use a similar model to improve the quality of intrusion detection systems in an enterprise network and demonstrate its effectiveness through simulation using traffic data from 37 hosts inside their enterprise network.

Event types. A class of previous work focuses on detecting network events in or near backbone links, using data gathered from layer-3 and below [13, 18, 19, 21, 29]. While these monitors can accurately detect a variety of events, they may miss silent failures (e.g., incompatible QoS/ACL settings) and their impact on performance. Other work focuses on detecting network events from a distributed platform [2, 16, 20, 36]. These solutions do not correlate these events with user-perceived performance, and their detection is limited by their network visibility and/or the overhead for probing large numbers of networks. The goal of CEM is to detect *service-level network events* and correlate their impact on application performance from the perspective of end users.

Monitoring location. CEM targets events that impact user-perceived application performance, by running on the *end systems* themselves. While several researchers have proposed using end-host probing to identify routing disruptions and their effect on end-to-end services [12,16,35,37], they have focused on GREN [20,36] or enterprise [10, 15, 26] environments and have not looked at the impact of network events on application performance nor addressed the issues of scalability when running on end systems. Some commercial network monitoring tools generate flows that simulate protocols used by edge systems (e.g., Keynote and IneoQuest[8]). While these can indeed detect end-to-end performance problems, these tools require controllable, dedicated infrastructure and are inherently limited to relatively small deployments in PoPs. Our CEM approach does not require any new infrastructure, nor control of end systems, and thus can be installed on systems at the edge of the network. Several research efforts have investigated the idea of active and passive network measurement from end users, e.g., DIMES [31] and Neti@home [32], but have not explored the use of their monitoring information for online network event detection.

Measurement technique. CEM focuses on *passive monitoring* of popular applications to detect events, which allows our approach to scale to the vast numbers of users at the edge of the network while still detecting events quickly. In a similar vein, previous work has suggested that the volume and breadth of P2P systems' natural traffic could be sufficient to reveal information about the used network paths without requiring any additional measurement overhead [9,36]. PlanetSeer [36] uses passive monitoring of a CDN deployed on PlanetLab, but relies on active probes to characterize the scope of the detected events. Casado et al. [3] and Isdal et al. [14] use opportunistic measurement to reach these edges of the network, by leveraging spurious traffic or free-riding in BitTorrent. Unlike these efforts, CEM takes advantage of the steady stream of natural, (generally) benign traffic generated by applications. Approaches that use active monitoring (e.g., [2, 16]) are limited by the overhead for detection, which grows with the number of monitored networks and services. While CEM could be combined with limited active probes to assist in characterizing and localizing network events, it does not require them.

8. CONCLUSION

The user experience for networked applications is becoming an important benchmark for customers and network providers. To assist operators with resolving such issues in a timely manner, we argued that the most appropriate place for monitoring service-level events is at the end systems where the services are used. We proposed a new approach, called *CEM* for Crowdsourcing Event Monitoring, based on pushing end-to-end performance monitoring and event detection to the end systems themselves. We presented a general framework for CEM systems and demonstrated its effectiveness using a large dataset of diagnostic information gathered from peers in the BitTorrent system, along with confirmed network events from two different ISPs. We showed that our crowdsourcing approach enables worldwide event detection, including events spanning multiple networks. Finally, we designed, implemented and deployed a BitTorrent extension that performs online event detection using our approach – currently installed more than 45,000 times. Having demonstrated the feasibility and effectiveness of this approach, we are investigating opportunities for porting it to other host applications such as VoIP and streaming video.

[7] When $L_h = 0$ for a host, it does not contribute to the union probability. Thus n is the number of hosts seeing at least one event.

[8] http://www.keynote.com and http://www.ineoquest.com/

Acknowledgments

We thank our shepherd Kevin Jeffay and the anonymous reviewers for their insightful comments. We also thank Kobus van der Merwe, Jennifer Yates, Lorenzo Alvisi, Yan Chen and Peter Dinda for their valuable advice and John Otto and Zach Bischof for early feedback on the paper. We are especially grateful to the users/adopters of our software for their invaluable data. This work was supported in part by the National Science Foundation under grants CNS 0644062 and CNS 0917233.

9. REFERENCES

[1] AMAZON. Amazon mechanical turk. `http://www.mturk.com/`.

[2] ANDERSEN, D., BALAKRISHNAN, H., KAASHOEK, F., AND MORRIS, R. Resilient overlay networks. In *Proc. ACM SOSP* (2001).

[3] CASADO, M., GARFINKEL, T., CUI, W., PAXSON, V., AND SAVAGE, S. Opportunistic measurement: Extracting insight from spurious traffic. In *Proc. HotNets* (2005).

[4] CASTRO, M., DRUSCHEL, P., GANESH, A., ROWSTRON, A., AND WALLACH, D. Secure routing for structured peer-to-peer overlay networks. In *Proc. USENIX OSDI* (2002).

[5] CHEN, K., CHOFFNES, D., POTHARAJU, R., CHEN, Y., BUSTAMANTE, F., AND ZHAO, Y. Where the sidewalk ends: Extending the Internet AS graph using traceroutes from P2P users. In *Proc. ACM CoNEXT* (2009).

[6] CHEN, X., ZHANG, M., MAO, Z. M., AND BAHL, P. Automating network application dependency discovery: Experiences, limitations, and new solutions. In *Proc. USENIX OSDI* (2008).

[7] CHOFFNES, D., AND BUSTAMANTE, F. Pitfalls for testbed evaluations of Internet systems. *SIGCOMM Comput. Commun. Rev. 40*, 2 (April 2010).

[8] CHOFFNES, D. R., AND BUSTAMANTE, F. E. Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *Proc. ACM SIGCOMM* (2008).

[9] COOKE, E., MORTIER, R., DONNELLY, A., BARHAM, P., AND ISAACS, R. Reclaiming network-wide visibility using ubiquitous endsystem monitors. In *Proc. USENIX ATC* (2006).

[10] DASH, D., KVETON, B., AGOSTA, J. M., SCHOOLER, E., CHANDRASHEKAR, J., BACHARCH, A., AND NEWMAN, A. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proc. AAAI* (2006).

[11] DISCHINGER, M., MARCON, M., GUHA, S., GUMMADI, K. P., MAHAJAN, R., AND SAROIU, S. Glasnost: Enabling end users to detect traffic differentiation. In *Proc. USENIX NSDI* (2010).

[12] FEAMSTER, N., ANDERSEN, D., BALAKRISHNAN, H., AND KAASHOEK, M. F. Measuring the effect of Internet path faults on reactive routing. In *Proc. ACM SIGMETRICS* (2003).

[13] IANNACCONE, G., NEE CHUAH, C., MORTIER, R., BHATTACHARYYA, S., AND DIOT, C. Analysis of link failures in an IP backbone. In *Proc. ACM IMW* (2002).

[14] ISDAL, T., PIATEK, M., KRISHNAMURTHY, A., AND ANDERSON, T. Leveraging BitTorrent for end host measurements. In *Proc. PAM* (2007).

[15] KANDULA, S., MAHAJAN, R., VERKAIK, P., AGARWAL, S., PADHYE, J., AND BAHL, P. Detailed diagnosis in enterprise networks. In *Proc. ACM SIGCOMM* (2009).

[16] KATZ-BASSETT, E., MADHYASTHA, H. V., JOHN, J. P., KRISHNAMURTHY, A., WETHERALL, D., AND ANDERSON, T. Studying black holes in the Internet with Hubble. In *Proc. USENIX NSDI* (2008).

[17] KEYNOTE. http://www.keynote.com/.

[18] LABOVITZ, C., AHUJA, A., AND JAHANIAN, F. Experimental study of Internet stability and wide-area backbone failure. Tech. Rep. CSE-TR-382-98, U. of Michigan, 1998.

[19] LAKHINA, A., CROVELLA, M., AND DIOT, C. Diagnosing network-wide traffic anomalies. In *Proc. ACM SIGCOMM* (2004).

[20] MADHYASTHA, H. V., ISDAL, T., PIATEK, M., DIXON, C., ANDERSON, T., KIRSHNAMURTHY, A., AND VENKATARAMANI, A. iPlane: an information plane for distributed systems. In *Proc. USENIX OSDI* (2006).

[21] MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. Understanding BGP misconfiguration. In *Proc. ACM SIGCOMM* (2002).

[22] MAHAJAN, R., ZHANG, M., POOLE, L., AND PAI, V. Uncovering performance differences among backbone ISPs with Netdiff. In *Proc. USENIX NSDI* (2008).

[23] MAHIMKAR, A., GE, Z., SHAIKH, A., WANG, J., YATES, J., ZHANG, Y., AND ZHAO, Q. Towards automated performance diagnosis in a large IPTV network. In *Proc. ACM SIGCOMM* (2009).

[24] MAYMOUNKOV, P., AND MAZIERES, D. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. IPTPS* (2002).

[25] MILLER, G. D. Programming techniques: An algorithm for the probability of the union of a large number of events. *Commun. ACM 11*, 9 (1968).

[26] PADMANABHAN, V. N., RAMABHADRAN, S., AND PADHYE, J. NetProfiler: profiling wide-area networks using peer cooperation. In *Proc. IPTPS* (2005).

[27] RABINOVICH, M., TRIUKOSE, S., WEN, Z., AND WANG, L. Dipzoom: The internet measurements marketplace. In *Proc. IEEE INFOCOM* (2006).

[28] RINGBERG, H., SOULE, A., AND REXFORD, J. WebClass: adding rigor to manual labeling of traffic anomalies. *SIGCOMM Comput. Commun. Rev. 38*, 1 (2008).

[29] ROUGHAN, M., GRIFFIN, T., MAO, Z. M., GREENBERG, A., AND FREEMAN, B. IP forwarding anomalies and improving their detection using multiple data sources. In *Proc. of the ACM SIGCOMM workshop on Network troubleshooting* (2004).

[30] SCHULZE, H., AND MOCHALSKI, K. Internet study 2008/2009, 2009.

[31] SHAVITT, Y., AND SHIR, E. DIMES: let the Internet measure itself. *SIGCOMM Comput. Commun. Rev. 35*, 5 (2005).

[32] SIMPSON, JR, C. R., AND RILEY, G. F. Neti@home: A distributed approach to collecting end-to-end network performance measurements. In *Proc. PAM* (2004).

[33] VON AHN, L. Human computation. In *Proc. DAC* (2009).

[34] VUZE, INC. Vuze. http://www.vuze.com.

[35] WU, J., MAO, Z. M., REXFORD, J., AND WANG, J. Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network. In *Proc. USENIX NSDI* (2005).

[36] ZHANG, M., ZHANG, C., PAI, V., PETERSON, L., AND WANG, R. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *Proc. USENIX OSDI* (2004).

[37] ZHANG, Y., MAO, Z. M., AND ZHANG, M. Effective diagnosis of routing disruptions from end systems. In *Proc. USENIX NSDI* (2008).