NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-09-19
October 2, 2009

Using the Crowd to Monitor the Cloud:
Network Event Detection from Edge Systems

David R. Choffnes, Fabián E. Bustamante, Zihui Ge

Abstract

The user experience for networked applications is becoming a key benchmark for customers and network providers when comparing, buying and selling alternative services. There is thus a clear need to detect, isolate and determine the root causes of network events that impact end-to-end performance and the user experience so that operators can resolve such issues in a timely manner. We argue that the most appropriate place for monitoring these service-level events is at the end systems where the services are used, and propose a new approach to enable and support this: Crowdsourcing Cloud Monitoring (C2M).

This paper presents a general framework for C2M systems and demonstrates its effectiveness using a large dataset of diagnostic information gathered from BitTorrent users, together with confirmed network events from two ISPs. We demonstrate that our crowdsourcing approach allows us to detect network events worldwide, including events spanning multiple networks. We discuss how we designed, implemented and deployed an extension to BitTorrent that performs real-time network event detection using our approach. It has already been installed more than 34,000 times.

Keywords: network events, anomaly detection, edge system monitoring, BitTorrent

Using the Crowd to Monitor the Cloud:
Network Event Detection from Edge Systems

David R. Choffnes[†], Fabián E. Bustamante[†], Zihui Ge[*]

[†]Dept. of EECS, Northwestern University [*]AT&T Labs - Research

Abstract

The user experience for networked applications is becoming a key benchmark for customers and network providers when comparing, buying and selling alternative services. There is thus a clear need to detect, isolate and determine the root causes of network events that impact end-to-end performance and the user experience so that operators can resolve such issues in a timely manner. We argue that the most appropriate place for monitoring these service-level events is at the end systems where the services are used, and propose a new approach to enable and support this: Crowdsourcing Cloud Monitoring (*C2M*).

This paper presents a general framework for C2M systems and demonstrates its effectiveness using a large dataset of diagnostic information gathered from BitTorrent users, together with confirmed network events from two ISPs. We demonstrate that our crowdsourcing approach allows us to detect network events worldwide, including events spanning multiple networks. We discuss how we designed, implemented and deployed an extension to BitTorrent that performs real-time network event detection using our approach. It has already been installed more than 34,000 times.

1 Introduction

The Internet is increasingly used as a platform for diverse distributed services such as VoIP, 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 [1].

Perceived user experience is in large part determined by the frequency, duration and severity of network events 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 resolve such issues in a timely manner, minimizing their impact on revenue and reputation.

We argue that the most effective way to detect service-level events is by monitoring the end systems where the services are used. *In this work, we develop a practical approach to monitoring that enables real-time detection of network events impacting the user experience for services that reach the network edge.*

Most previous work focuses on monitoring core networks [2–5] or probing from global research and education network (GREN) environments [6, 7]. While effective at detecting events that affect large numbers of customers and services, these approaches can miss silent failures (e.g., incompatible QoS or ACL settings) and their impact on services for customers [8]. Further, existing end-to-end monitoring approaches require active measurements that do not scale to the vast number of elements at the edge of the network.

Detecting service-level network events from end systems at the network edge poses a number of interesting challenges. First, any practical approach must address the scalability constraints imposed by collecting and processing information from potentially millions of end systems [9]. Second, to assist operators in addressing problems promptly, events should be detected quickly (i.e., within minutes) and isolated to specific network locations (e.g., BGP prefixes). Finally, the approach must facilitate a broad (Internet-scale) deployment of edge-system monitors, ensure user privacy and provide trustworthy event detection information.

We address these challenges through a new approach to network event detection – pushing end-to-end performance monitoring and detection to the end systems themselves. We call this approach *C2M* for Crowdsourcing Cloud Monitoring. By crowdsourcing network monitoring, participating hosts can handle the magnitude of data required for detecting events in real time, at the scale of millions of monitors. In addition, using end systems provides flexibility in the types of monitoring software that can be installed inside or alongside services, facilitating immediate and incremental deployments. Finally, we discuss general techniques to ensure the reliability of detection results without violating user privacy.

This paper makes the following contributions. Sec. 2 identifies challenges faced by any edge-system monitoring approach and discusses potential solutions. Next, we address the general problem of how to detect network performance events from the edge (Sec. 3). Specifically, we develop a framework for our C2M approach in which each end system performs a significant portion of event detection locally, then uses a distributed approach for corroborating these events.

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 running the Ono plugin [10] for the Vuze BitTorrent client. Guided by confirmed network events that they

observed, we design and implement the *Network Early Warning System (NEWS)*, a BitTorrent extension that performs real-time event detection.

We evaluate the effectiveness of our approach in Sec. 5. In addition to comparing NEWS-detected events with confirmed ones, we demonstrate that our crowdsourcing approach allows us to detect 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 34,000 times, demonstrating not only the feasibility of our approach for a real application, but also that there are appropriate incentives for widespread adoption beyond BitTorrent (Sec. 6). We are currently working with developers of popular software to instrument additional applications. To assist with quickly resolving problems causing detected network events, we have implemented *NEWSight*[1] – a system that accesses live event information and publishes its results in real time. We are beta-testing this public interface with ISPs.

2 C2M Advantages and Challenges

The user experience for networked applications (e.g., Web sites, VoIP and video streaming) is becoming an important benchmark for customers and network providers when comparing, buying and selling alternative services [1]. Monitoring service-level events – ones that impact end-to-end performance and the user experience – is thus important for users, service providers and network operators. Further, correcting these issues in a timely manner requires that operators know where, when and why they are occurring in real time.

To detect service-level events, we propose using monitoring software that runs inside or alongside applications that use or provide the corresponding services – on end systems. In particular, our goal is to detect service-level events from the edges of the network in real time.

Table 1 summarizes how the C2M approach differs from previous work in network event detection. The vast majority of previous work focuses on detecting network events in or near backbone links, using data gathered from layer-3 and below [2, 4, 5, 11, 12]. While such device-level monitoring can detect many types of events (e.g., outages or packet loss), existing techniques require active measurements that do not scale to the large number of elements at the network edge. Further, these monitors may miss silent failures (e.g., incompatible QoS/ACL settings) and their impact on performance. In contrast, our approach focuses on end-to-end performance problems, which vary among applications.

Other research projects have proposed monitoring network performance from end systems; however, because they rely at least in part on active measurements they are limited in scale and scope to GREN [6, 13] or enterprise [14, 15] environments. Our approach relies on passive measurements of running services so that it can scale to end systems located at the edge of the Internet.

Approach	Event type	Coverage	Online?
ISP monitoring			
Failures [3, 7, 16, 17]	Network	Core	Real time
Chronic events [8]	Network	Core	Offline
IPTV [18]	Network/Service	Core-Edge	Offline
GREN monitoring			
All pairs (active) [19]	Network/Service	GREN	$O(h)$ time
All pairs (passive) [6]	Service	GREN	Real time
Distributed probes [7]	Network	GREN-Edge	$O(n)$ time
C2M			
Services/OS (passive)	**Service**	**Edge-Edge**	**Real time**

Table 1: Comparison of detection approaches. For systems where detection times depend on system size, h is the number of monitors and n is the number of monitored networks.

Finally, some network monitoring tools generate flows that simulate protocols used by edge systems [1]. While these can indeed detect end-to-end performance problems, current deployments require controllable, dedicated infrastructure and are inherently limited to relatively small deployments in PoPs. Our C2M approach does not require any new infrastructure, nor control of end systems, and thus can be installed on end systems at the edge of the network.

There is a number of important issues that must be addressed in the context of C2M.

Scalability. As one moves toward the edge of the network, the number of network elements – and thus the opportunities for failures – rapidly increase. 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. Extending existing network monitoring approaches to edge systems is nontrivial: deployments in network edge devices (e.g., DSL modems) are difficult or impossible without vendor support; moreover, gathering and processing data for detecting events in real time may require costly dedicated infrastructure [20].

We propose a decentralized approach to event detection that relies on each system detecting local service-level performance problems as potential network events. By processing performance data at the edge systems, our approach facilitates an immediately deployable, scalable monitoring system.

Granularity. Any real-time network monitoring system must 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. By gathering and

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

processing performance information locally at each end system, C2M can detect events with fine granularity (on the order of seconds) and relatively low CPU and memory overhead. To isolate the scope of network events, we use multiple locally detected events from the same network location. These network locations can include publicly available locations such as 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.

Adoption. Any network event detection approach is limited by the coverage of its deployment. In the case of C2M, there is no cost to deploy and there are essentially no limitations as to where participating hosts can be located; however, the main challenge is gaining widespread adoption. One can address this issue by 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 [21], altruism [22] and mutual benefit [10]. Based on the success of Ono [10], we propose using a mutual benefit model, which has been sufficient for a prototype implementation of C2M already installed over 34,000 times.

In the next section, we address many of these challenges with a general approach to performing service-level network monitoring from edge systems.

3 C2M Framework

Our C2M approach relies on *edge system monitors* (ESMs) installed on end systems to detect service-level problems associated with one or more networks. We assume that each ESM has access to one or more sources of performance information (e.g., transfer rates, latency jitter and dropped packets). We further assume that each ESM can connect to a distributed storage system to share information about detected events.

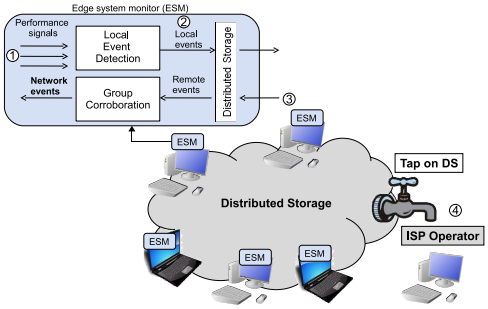


Figure 1: Schematic view of our edge detection approach.

Fig. 1 depicts the C2M architecture. As we discussed in the previous section, it is infeasible for edge systems to publish detailed performance data for scalability and privacy reasons. To address this issue, our approach detects events using locally gathered performance data at each ESM (step (1) of the figure) – we discuss this in Sec. 3.1.

Local event detection presents new design challenges for determining the scope and severity of events. C2M 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) in Fig. 1), allowing any other participating host to examine these aggregate events. Sec. 3.2 discusses how C2M determines the likelihood that a set of these locally detected problems corresponds to a *network* event.

In our architecture, network events can be detected by the monitors themselves or via third-party analysis. Each participating host can use the distributed store to capture events corresponding to its network (step (3) in Fig. 1), then determine whether these local events indicate a network event. Alternatively, a third-party system (e.g., run by an ISP) could use the distributed store to perform the analysis (step (4) in Fig. 1). 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 C2M is to analyze local performance information to determine whether the monitored host is experiencing a problem. In this section, we discuss the types of available performance signals and techniques for detecting local performance events.

3.1.1 Performance Signals

By pushing detection to end systems located at the edge of the network, C2M can use a wide variety of service-level information to diagnose local performance problems (Table 3). 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 whether there was a gap in voice playback. 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 C2M 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 Local Event Detection

C2M 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 the problem, i.e., whether the problem is local or network-related. To this end, the output of local detection is a summary for 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 choice of event detection technique is strongly dependent on the service being monitored. For instance, when monitoring end-to-end throughput for a host (e.g., for video streaming), edge detection can identify drops in transfer rates potentially caused by a network issue like congestion. In the domain of IPTV [18], video quality (among other factors) may indicate problems with the network. 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 *network* problems. In a P2P file-sharing 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, C2M determines whether other hosts in the same network are seeing the same problem – this requires hosts to share local event detection results. 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, C2M 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 as the same time in the same network, our approach must determine if these events are likely to be due to the network.

There is 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 isolated to the service driving network activity, e.g., performance from a Web server or from a swarm of P2P users sharing content. Finally, simultaneous events can occur by chance, e.g., due to multiple users experiencing interference on separate wireless routers.

In the following paragraphs, we discuss how C2M 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, C2M can determine the *relative likelihood* of the detected problem being caused by the network instead of by chance.

Eliminating confounding factors. The first step in

the likelihood analysis is to determine the probability that each host detects local problems independently. Thus, for each host h we produce a series $A_h = \{a_{h,i}, a_{h,i+1}, ..., a_{h,j}\}$ for the time period $T = [i, j]$, such that at time t, $a_{h,t} = 1$ if a local event was detected and $a_{h,t} = 0$ 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:

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

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 a single logical 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.

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

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

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 the number of hosts detecting the event and the number of 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 one signal, L_{h1}, is chosen uniformly at random in the range $0.005 \leq L_{h1} \leq 0.05$. Similarly, the probability of witnessing a local event concurrently in two signals, L_{h2}, is chosen uniformly at random from the range $0.005 \leq L_{h2} \leq L_{h1}$ and the range for three signals, L_{h3} is $0.005 \leq L_{h3} \leq L_{h2}$. 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.

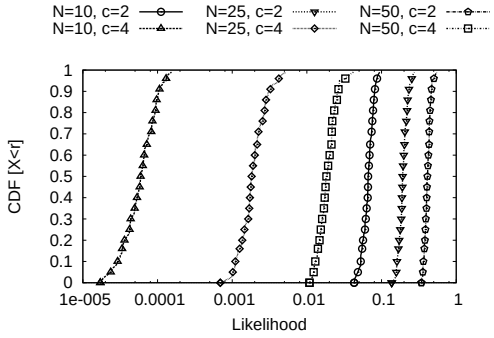


Figure 2: CDF indicating how increasing the number of hosts corroborating an event decreases the likelihood of it occurring by chance.

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, C2M 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 Localization

After detecting a local event, our approach can use information published in event reports to determine the scope of the problem. If many hosts in a network detect an event at the same time, it is likely a problem best addressed by the responsible network operators. In such cases, our approach should be able to identify the network affected by the event so as to provide the necessary information for operators to determine the root cause and fix the problem [9].

Our approach can localize problems using structural information about the organization of networks and their geolocations. 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, C2M 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 C2M

The previous section described our C2M approach for detecting events from edge systems. Designing, deploying and evaluating C2M 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 [23]. 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 [6, 24].

Based on these advantages, we choose to design and evaluate a prototype implementation of C2M in a large P2P system. To guide our design 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 [25], to facilitate adoption and to piggyback on the application's large user base.

Category	Number (Pct of total)
Number of users	700,000 (3% of Vuze users)
Countries	200 (78%)
IP addresses	3,100,000
Prefixes	46,685
Autonomous systems (ASes)	7,000
IPs behind middleboxes	$\approx$ 82.6%

Table 2: 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 monitored end system may run other applications that interfere with the P2P application and event detection. Finally, some event detection techniques require access to privileged system calls and information not accessible to a P2P application. In the rest of this section, we show that despite these challenges NEWS can detect network events simply by passively monitoring BitTorrent, thus validating our C2M approach.

4.1 Datasets

4.1.1 BitTorrent traces

The realization of NEWS is guided by measurement data from the Ono plugin for Vuze. Ono implements a biased peer selection service aimed at reducing the amount of costly cross-ISP traffic generated by BitTorrent without sacrificing system performance [10]. Beyond assisting in peer selection, the software allows subscribing volunteers to participate in a monitoring service for the Internet. With over 700,000 users today, distributed in over 200 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 2.

Data collected. While observing downloads, Ono samples transfer rates for each connection once every 5 seconds and cumulative transfer rates (over all connections) once every 30 seconds. Besides transfer rates, the system records 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, as well as information about the availability of data for each download. The complete list of collected signals is in Table 3.

This data is reported to a data-collection infrastructure, which converts data timestamps to a universal time zone. *This collection is for our design and evaluation; it is* ***not*** *part of NEWS event detection.*[2]

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 300,000,000 peer IPs; collectively, its users monitor more than 17 million paths per day. Ono users covered 500 prefixes within its first two weeks of deployment, and grew to over 40,000 prefixes (covering nearly every country) in under two years. Collectively, these users have established connections to peers in about 222,000 routable prefixes and 21,800 ASNs.

Besides monitoring many paths, our dataset covers 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. Further, Chen et al. [26] show that these peers monitor more than 20,000 AS links not visible to public topology views, and their flows cross 40% more peering AS links than seen from public views.

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 in a network – is the most common [27].

As one example, we use publicly available event reports from the British Telecom (BT Yahoo) ISP[3] in the UK. This site identifies the start and end times, locations and the nature of network problems. During the month of April, 2009 there were 68 reported problems, which include both 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 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 whether confirmed events are detected by our approach, but we cannot draw strong conclusions about false positive and negative rates.

[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.

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

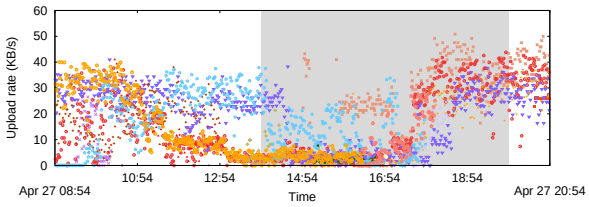


Figure 3: Upload rates for peers in a routable prefix owned by British Telecom during a confirmed disruption (shaded region).

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 the following problem in BT Yahoo: On April 27, 2009 at 3:54 PM GMT, the network status page reported, "*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 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

In this section, we 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.

4.3.1 Local Event Detection

Any C2M 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. Monitoring for

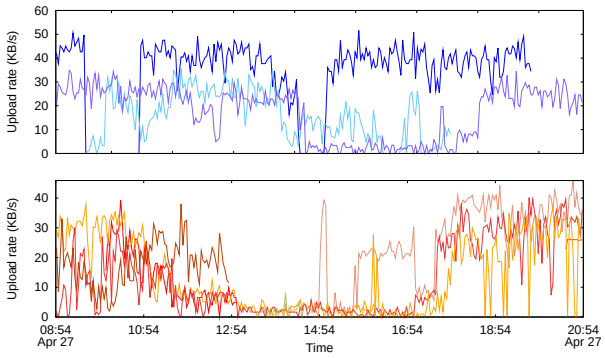


Figure 4: Moving averages facilitate identification of separate network events affecting transfer rates for two groups of peers during the same period shown in Fig. 3.

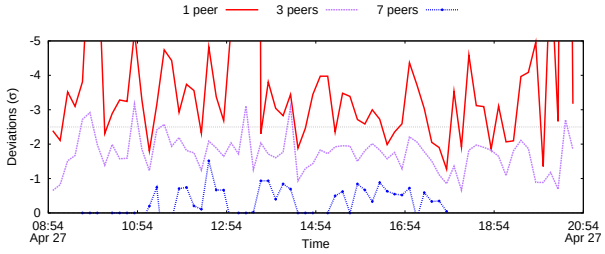


Figure 5: Timeline of the maximum performance drops for at least n peers (moving average window size of 10, $n = 1, 3, 7$). Deviations for any one peer are highly variable; those for seven peers rarely capture any performance drops. The peaks in deviations for three peers correspond to confirmed events.

this type of event corresponds to detecting edges in the throughput signal; specifically, we detect downward edges 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 peers 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 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. To help understand how to set this threshold, 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). If our approach to local detection is viable, there should be some threshold ($t \cdot \sigma$) for identifying peers' local events that correspond to network ones.

The top curve, where $n = 1$, shows that the maximum deviations from any one peer produces a noisy signal that is subject to 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 all see performance drops simultaneously, so features in this signal are not useful for detecting events during this period. Last, the middle curve, where $n = 3$, 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. Downward edges in the throughput signal provided by a host BitTorrent application are 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 3 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

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 3: Signals available when monitoring from BitTorrent.

to the same torrent are treated as the same logical 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.

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 problem. If a host's connections were biased toward a remote AS, for example, then 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 by each peer's connections during a 19-day period. We find that the vast majority of peers (90%) connect to peers in five or more prefixes during the period; the median is 169. This range indicates that it extremely 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, C2M 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 in real time.

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, $\mathrm{LR} = \mathrm{P_e}/\mathrm{P_u}$, as described in Sec. 3.2.1. For this analysis, we use one month of data to determine P_e and P_u.

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.

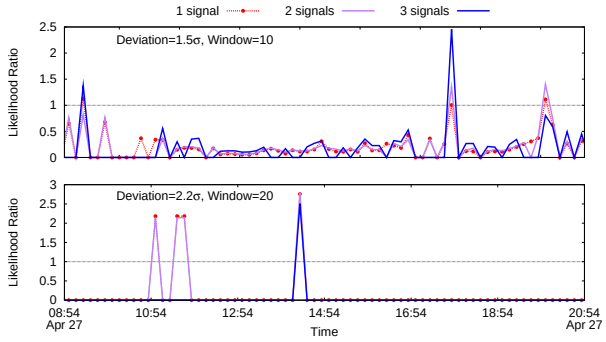


Figure 6: Timeline showing the likelihood ratio for different moving average settings. In each case, there are few events with $LR > 1$, and nearly all correspond to confirmed events.

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.

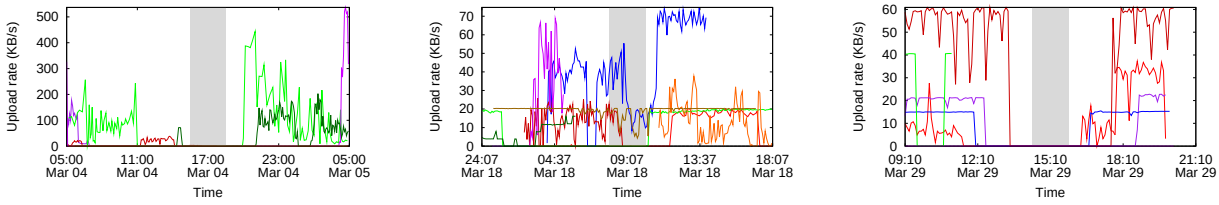
Fig. 6 (bottom) uses a detection threshold of 2.2σ and window size of 20. As expected, the larger threshold and window size detect fewer events in the observation window. In this case, *all of the three values that appear above* $\mathrm{LR} = 1$ *correspond to the known network problems*, and they are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

These examples demonstrate that our approach is able to reliably 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. As such, the likelihood ratio can be seen as a single parameter that selects detection settings that reliably detect network problems.

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 information that can be used to personally identify the user. 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



(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.

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 [28].

4.3.4 Adoption 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 [10], we propose using a similar *mutual benefit* incentive model in which 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 installed over 34,000 times.

5 Evaluation

We use one month of data gathered from BitTorrent users to answer key questions about large-scale edge-system network event detection. We first demonstrate the effectiveness of our approach 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. We close the section with an evaluation of the robustness of our approach to parameter settings and an analysis of the overhead for participating hosts.

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 4: Comparison with events from a North American ISP.

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.

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 additional 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. Table 4 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.

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

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

P2P application as a host application can offer sufficient coverage for edge-system event detection, we calculated the maximum number of peers simultaneously online 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 it detects worldwide, using a threshold $LR = 2$ (as guided by Fig. 6). For a one-month period, NEWS 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 (second column), and the number of events detected in each of these ISPs (third column).

Because different networks provide different quality of service [29], 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 performance events that our approach detects.

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 (or annoy) network operators and users.

5.4 Cross-network events

An advantage to our C2M approach is that it is uniquely positioned to detect network problems affecting multiple ISPs, e.g., due to provider or peering link issues. As we discussed in Sec. 3.2.2, one can use AS relationships and geolocation information to isolate the scope of cross-network events.

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

Table 6: Number of cross-network events (and countries affected) as inferred from single-network events. The first column indicates the AS relationship and the second column specifies the minimum number of affected ASes.

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

We focus on cross-network events due to issues with upstream providers or peers. To this end, we find events that occur in multiple ASNs at the same time, then determine which of these ASNs have a peering relationship or identical providers (based on the AS topology generated by Chen et al. [26]). Events that occur within 30 minutes of each other are considered the same, and we conservatively consider AS relationships only for those ASNs located in the same country.

Table 6 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 much rarer; a sample of them is provided in Table 7. Finally, the last row indicates that there is a significant number of peering ASNs that see synchronized problems. Discovering such potential problems is unique to our approach – these links are often invisible to existing Internet monitoring techniques that do not rely on edge-system monitors [26].

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 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 window sizes yield a larger number of ratio values greater than one (2.15% of the time) whereas larger deviations and

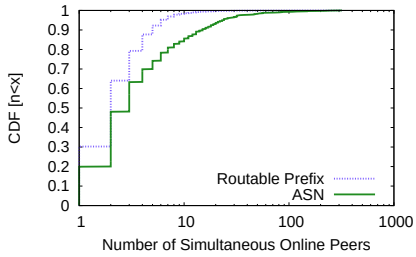
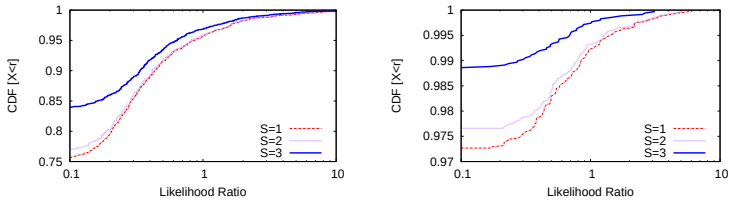


Figure 8: CDF of concurrent online peers in this study.



(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).

window sizes yields a smaller number of them (0.75%). Generally, such cases (where concurrent events occur more often than chance) are extremely rare for significantly different detection parameters, suggesting that LRs are indeed robust to detection settings.

5.6 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.[4] We use Miller's algorithm [30], 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 large n (in our experience, $n > 50$). To bound this overhead, 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.

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 34,000 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. In practice, we found that BitTorrent often saturates a user's access link, leading to stable transfer rates and small σ. As a result, edges in the performance signal occur 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 the Kademlia-based DHT [31] built into Vuze. This DHT is a key-value store that stores 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

[4] 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.

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 NEWS detects a local event, it performs corroboration by searching the DHT for other event reports in each of its regions – currently the host's BGP prefix and ASN.[5] Before using a report from the DHT for corroboration, NEWS ensures that: (1) the report was not generated by this host; (2) the report was generated recently; and (3) the standard-deviation multiplier for detecting the event was not less than the one used locally.

If these conditions are met, the report's ID is added to the set of recently reported events. If a peer finds events from three or more other peers at the same time (a configurable threshold), it then 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 2 (also configurable), the monitor issues a notification about the event.

NEWS peers read from the DHT only after detecting a local event, in order to corroborate their finding. To account for delays between starting a DHT write and the corresponding 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).

Third-party interface. Following our incentive model, NEWS keeps end-users informed about detected service-level events (Sec. 4.3.4.) Beyond end-users, network operators should be notified to assist in identifying and fixing these problems. With this in mind, we have implemented a DHT crawler (*NEWS Collector*) that any third party can run to collect 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 its detected events through a public Web interface. NEWSight also allows network operators to search for events and register for notifications of events detected in their networks. Operators responsible for affected networks can confirm/explain detected events.

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 our approach and other similar ones – addressing the paucity of labeled data available in this domain [27]. We are currently beta-testing this interface with ISPs; the interface and its data are publicly available.

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

7 Related Work

As an approach to detecting service-level network events from end systems located at the edge of the network, C2M is related to a variety of prior work. Most previous efforts on network event detection have focused on core networks and GREN environments [2, 11, 32, 33].

Several researchers have proposed using end-host probing to identify routing disruptions and their effect on end-to-end services [3, 7, 16, 17]. A number of recent efforts are exploring new monitoring techniques using distributed research platforms (e.g., PlanetLab or NIMI2) as vantage points. These approaches are inherently limited by the relatively small number of nodes available for experimentation and the fact that they are not representative of the larger Internet. While most of these hosts are deployed in GREN environments, often close to the core, much of the Internet's growth occurs beyond their reach, such as behind NAT boxes and firewalls or in regions of the Internet not exposed by public BGP feeds [26, 34, 35]. C2M uses a fundamentally different approach that pushes detection to the edge-systems where services are used.

NEWS passively monitors BitTorrent to identify service-level network events. 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 [6, 24]. PlanetSeer [6] 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. [35] and Isdal et al. [36] use opportunistic measurement to reach these edges of the network, by leveraging spurious traffic or free-riding in BitTorrent. Unlike these efforts, NEWS takes advantage of the steady stream of natural, (generally) benign traffic generated by BitTorrent and does not require any active measurement. While NEWS shares many goals with DIMES [22] and Neti@home [37], it uses immediate incentives to ensure significantly wider adoption than what is possible with a purely altruistic model.

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 *C2M* for Crowdsourcing Cloud Monitoring, based on pushing end-to-end perfor-

mance monitoring and event detection to the end systems themselves. We presented a general framework for C2M 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 demonstrate that our crowdsourcing approach allows us to detect network events worldwide, including events spanning multiple networks. Finally, we designed, implemented and deployed a BitTorrent extension that performs real-time event detection using our approach – currently installed more than 34,000 times.

References

[1] keynote, "keynote: The Mobile and Internet Performance Authority," http://www.keynote.com/, 2009.

[2] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proc. ACM SIGCOMM*, 2004.

[3] Y. Zhang, Z. M. Mao, and M. Zhang, "Effective diagnosis of routing disruptions from end systems," in *Proc. USENIX NSDI*, 2008.

[4] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *Proc. ACM SIGCOMM*, 2002.

[5] G. Iannaccone, C. nee Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proc. ACM IMW*, 2002.

[6] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, "PlanetSeer: Internet path failure monitoring and characterization in wide-area services," in *Proc. USENIX OSDI*, 2004.

[7] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson, "Studying black holes in the Internet with Hubble," in *Proc. USENIX NSDI*, 2008.

[8] A. Mahimkar, J. Yates, Y. Zhang, A. Shaikh, J. Wang, Z. Ge, and C. T. Ee, "Troubleshooting chronic conditions in large IP networks," in *Proc. ACM CoNEXT*, 2008.

[9] C. R. Kalmanek, Z. Ge, S. Lee, C. Lund, D. Pei, J. Seidel, J. van der Merwe, and J. Yates, "Darkstar: Using exploratory data mining to raise the bar on network reliability and performance," in *Proc. DRCN*, 2009.

[10] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems," in *Proc. ACM SIGCOMM*, 2008.

[11] M. Roughan, T. Griffin, Z. M. Mao, A. Greenberg, and B. Freeman, "IP forwarding anomalies and improving their detection using multiple data sources," in *Proc. of the ACM SIGCOMM workshop on Network troubleshooting*, 2004.

[12] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of Internet stability and wide-area backbone failure," U. of Michigan, Tech. Rep. CSE-TR-382-98, 1998.

[13] H. V. Madhyastha, T. Isdal, michael Piatek, C. Dixon, T. Anderson, A. Kirshnamurthy, and A. un Venkataramani, "iPlane: an information plane for distributed systems," in *Proc. USENIX OSDI*, 2006.

[14] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed diagnosis in enterprise networks," in *Proc. ACM SIGCOMM*, 2009.

[15] V. N. Padmanabhan, S. Ramabhadran, and J. Padhye, "NetProfiler: profiling wide-area networks using peer cooperation," in *Proc. IPTPS*, 2005.

[16] J. Wu, Z. M. Mao, J. Rexford, and J. Wang, "Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network," in *Proc. USENIX NSDI*, 2005.

[17] N. Feamster, D. Andersen, H. Balakrishnan, and M. F. Kaashoek, "Measuring the effect of Internet path faults on reactive routing," in *Proc. ACM SIGMETRICS*, 2003.

[18] A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao, "Towards automated performance diagnosis in a large iptv network," in *Proc. ACM SIGCOMM*, 2009.

[19] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. ACM SOSP*, 2001.

[20] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk, "Gigascope: a stream database for network applications," in *Proc. ACM SIGMOD*, 2003.

[21] M. Rabinovich, S. Triukose, Z. Wen, and L. Wang, "Dipzoom: The internet measurements marketplace," in *Proc. IEEE INFOCOM*, 2006.

[22] Y. Shavitt and E. Shir, "DIMES: let the Internet measure itself," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, 2005.

[23] ipoque, "Internet Study 2007: Data about P2P, VoIP, Skype, file hosters like RapidShare and streaming services like YouTube," Nov. 2007.

[24] E. Cooke, R. Mortier, A. Donnelly, P. Barham, and R. Isaacs, "Reclaiming network-wide visibility using ubiquitous endsystem monitors," in *Proc. USENIX ATC*, 2006.

[25] Vuze, Inc., "Vuze," http://www.vuze.com, 2008.

[26] K. Chen, D. Choffnes, R. Potharaju, Y. Chen, F. Bustamante, and Y. Zhao, "Where the sidewalk ends: Extending the Internet AS graph using traceroutes from P2P users," in *Proc. ACM CoNEXT*, 2009.

[27] H. Ringberg, A. Soule, and J. Rexford, "WebClass: adding rigor to manual labeling of traffic anomalies," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 1, 2008.

[28] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proc. USENIX OSDI*, 2002.

[29] R. Mahajan, M. Zhang, L. Poole, and V. Pai, "Uncovering performance differences among backbone ISPs with Netdiff," in *Proc. USENIX NSDI*, 2008.

[30] G. D. Miller, "Programming techniques: An algorithm for the probability of the union of a large number of events," *Commun. ACM*, vol. 11, no. 9, 1968.

[31] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. IPTPS*, 2002.

[32] C. Logg, J. Navratil, and R. L. Cottrell, "Correlating internet performance changes and route changes to assist in trouble-shooting from an end-user perspective," in *Proc. PAM*, 2004.

[33] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and identification of network anomalies using sketch subspaces," in *Proc. IMC*, 2006.

[34] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using PlanetLab for network research: Myths, realities and best practices," *ACM SIGOPS Op. Sys. Rev.*, vol. 40, no. 1, pp. 17–24, 2006.

[35] M. Casado, T. Garfinkel, W. Cui, V. Paxson, and S. Savage, "Opportunistic measurement: Extracting insight from spurious traffic," in *Proc. HotNets*, 2005.

[36] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Leveraging BitTorrent for end host measurements," in *Proc. PAM*, 2007.

[37] C. R. Simpson, Jr and G. F. Riley, "Neti@home: A distributed approach to collecting end-to-end network performance measurements," in *Proc. PAM*, 2004.