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

David R. Choffnes and Fabián E. Bustamante
Department of Electrical Engineering & Computer Science, Northwestern University
{drchoffnes,fabianb}@cs.northwestern.edu

November 2, 2009

Abstract

Whether streaming videos, making VoIP phone calls or simply downloading large files, we expect to receive good performance – often beyond the best-effort guarantees that the Internet provides. 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. 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 believe that the most effective way to detect service-level events is by monitoring the end systems where the services are used. This document describes an implementation of this approach for BitTorrent called NEWS (Network Early Warning System), a system that provides real-time detection of network events impacting the user experience for peer-to-peer file sharing. We use probability theory, extensive network traces from users and ground-truth information from ISPs to design and build a system that detects network problems effectively, quickly and reliably. We also discuss several key features of its current implementation for BitTorrent, called the Network Early Warning System (NEWS), which has been installed more than 30,000 times.

1 Introduction and Background

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.

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. This document describes an implementation of this approach for BitTorrent called NEWS (Network Early Warning System), a system that provides real-time detection of network events impacting the user experience for peer-to-peer file sharing.

Most previous work focuses on monitoring core networks or probing from global research and education network (GREN) environments such as PlanetLab. 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. 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. 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.

NEWS address these challenges through an approach to network event detection that pushes end-to-end performance monitoring and detection to the end systems themselves. 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. We discuss our event detection framework in Section 2.

To demonstrate the effectiveness of our edge-based approach, we use a large dataset of diagnostic information from edge systems running the Ono plugin [1] for the Vuze BitTorrent client (Section 3). Finally, we discuss several key NEWS implementation details in Section 4.

This document provides a high-level discussion of key features of NEWS. For a more technical treatment of the topic, please see the associated technical report [2].

2 NEWS Overview

Our event detection approach relies on NEWS monitors installed on end systems (at the edge of the network) to detect service-level problems associated with one or more networks. NEWS monitors have access to one or more sources of performance information (e.g., transfer rates, latency jitter and dropped packets) and connect to a distributed storage system to share information about detected events.

Fig. 1 depicts the architecture for NEWS. An important challenge in this approach is that it is infeasible for edge systems to publish detailed performance data for

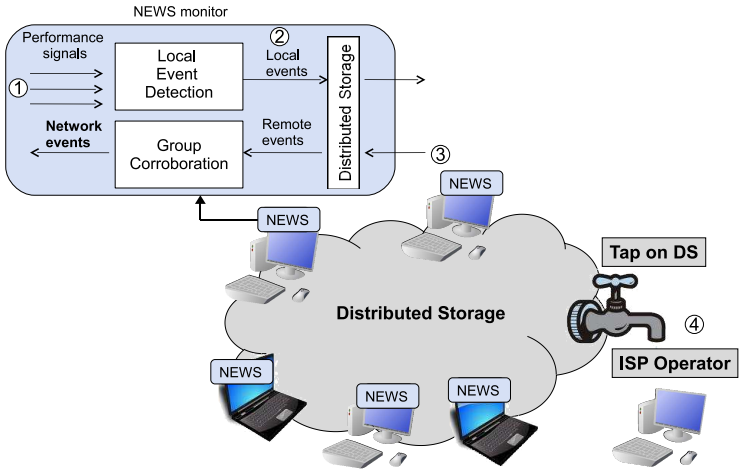

Figure 1: Schematic view of our edge detection approach.

scalability and privacy reasons. To address this issue, our approach detects events using locally gathered performance data at each monitor (step (1) of the figure). These events can be unexpected drops in transfer rates for P2P file sharing or choppy video playback for video streaming.

Local event detection presents new design challenges for determining whether a serious network problem is occurring and what part(s) of the network are affected. NEWS 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.

Locally detected events may indicate a network problem, but each local view alone is insufficient to determine if this is the case. When multiple hosts detect a problem at the same time in the same network, we must determine whether these problems are due to the *network* and not simply happening by coincidence. To quantify this, we use a *likelihood ratio*, i.e., the ratio of the observed probability of concurrent events to the probability of concurrent events happening independently.[1]

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.

[1]Likelihood ratios are commonly used in medicine as a way to interpret diagnostic tests; e.g., the likelihood that a given test result would be expected in a patient with a certain disorder compared to the likelihood that same result would occur in a patient without the target disorder. (Source: Wikipedia)

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 1: Summary of our P2P vantage points.

3 NEWS Effectiveness: A Case Study

Designing, deploying and evaluating an edge-based network event detection system 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. Thus, 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. We used this data to design the *Network Early Warning System (NEWS)*, our prototype edge-based event detection system that uses BitTorrent as a host application.

Our traces consist of BitTorrent performance information gathered from the Ono plugin for Vuze.[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 [1]. 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 1.

Case study. 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.

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.

We now 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. 2 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

[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/

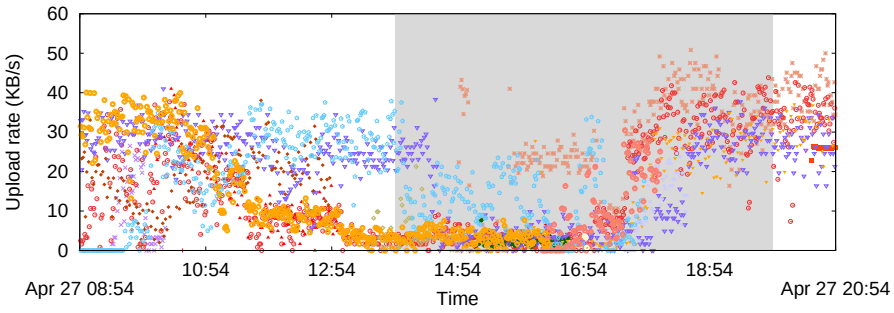

Figure 2: Upload rates for peers in a routable prefix owned by British Telecom during a confirmed disruption (shaded region).

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.

Any network event detection 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 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. 3 plots the 10-minute averages of upload rates for two groups of peers from Fig. 2. 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 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. 4 shows how deviations behave over time for peers experiencing the network problems illustrated in Fig. 3,

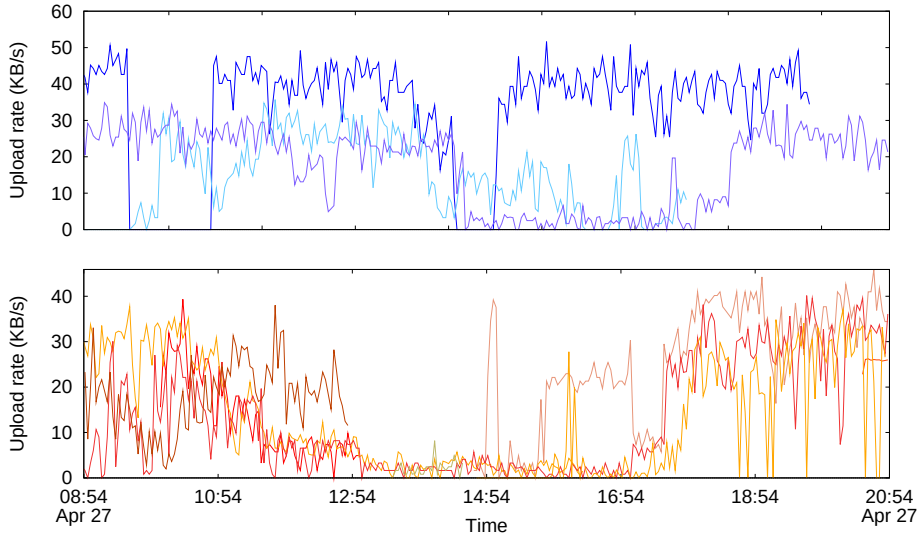

Figure 3: Moving averages facilitate identification of separate network events affecting transfer rates for two groups of peers during the same period shown in Fig. 2.

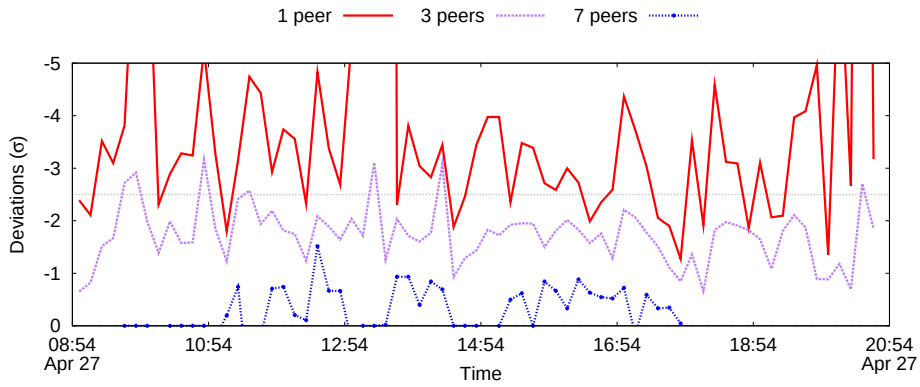

Figure 4: Timeline of the maximum performance drops for at least n peers (moving average window size of 10, $n = 1, 3, 7$). Deviations for any one peer are highly variable; those for seven peers rarely capture any performance drops. The peaks in deviations for three peers correspond to confirmed events.

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. We now show how we use likelihood ratios to extract probably network events from local events detected using these settings.

Group Corroboration As discussed in the previous section, after detecting local events, NEWS 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.

To derive this ratio, NEWS 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, NEWS 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. We now apply this likelihood analysis to the events in BT Yahoo.

Figure 5 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. 5 (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. 5 (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.

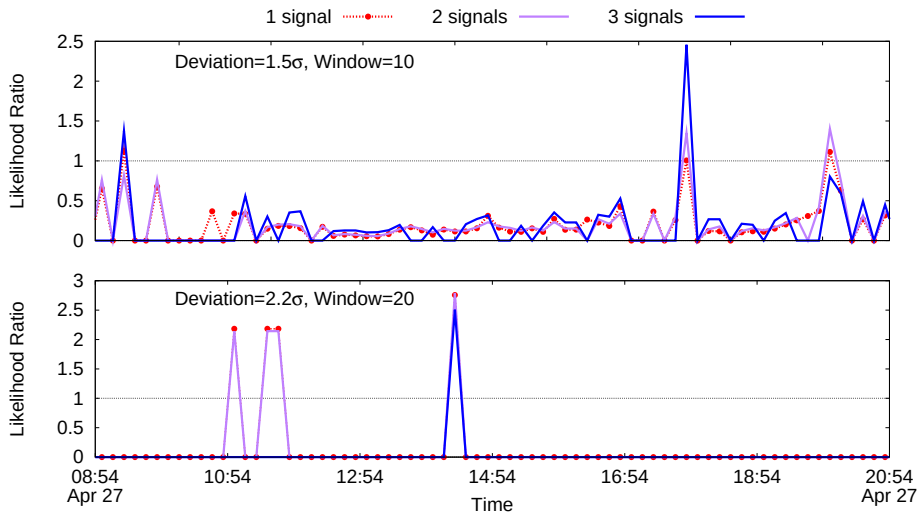

Figure 5: Timeline showing the likelihood ratio for different moving average settings. In each case, there are few events with $LR > 1$, and nearly all correspond to confirmed events.

4 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 a moving average technique, which takes the window size (w) and standard-deviation multiplier (t) as 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 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 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.[4] 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 the union probability 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, LR. 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. To provide incentives for users to install the software, NEWS keeps end-users informed about detected service-level events. 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. We are currently beta-testing this interface with ISPs; the interface and its data are publicly available.

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

5 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. Our NEWS software implements this idea, pushing end-to-end performance monitoring and event detection to the end systems themselves. We demonstrated the effectiveness of NEWS using a large dataset of diagnostic information gathered from peers in the BitTorrent system, along with confirmed network events. Finally, we discussed implementation details for our NEWS BitTorrent extension, which is currently installed more than 34,000 times.

References

[1] 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.

[2] D. Choffnes, F. Bustamante, and Z. Ge, "Using the crowd to monitor the cloud: Network event detection from edge systems," Northwestern Univeristy, Tech. Rep. NWU-EECS-2009-xx, October 2009.