Service-Level Network Event Detection from Edge Systems

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

David R. Choffnes

EVANSTON, ILLINOIS

June 2010
ABSTRACT

Service-Level Network Event Detection from Edge Systems

David R. Choffnes

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

This dissertation proposes addressing these issues by pushing monitoring to applications on end systems and using their collective view to detect network events and their impact on services – an approach called Crowdsourcing Event Monitoring (CEM). This work presents a general framework for CEM systems and demonstrates its effectiveness for a P2P application using a large dataset gathered from BitTorrent users, together with confirmed network events
from two ISPs. We discuss how we designed and deployed an extension to BitTorrent that implements CEM. This is the first system that performs real-time service-level network event detection through passive monitoring and correlation of perceived performance in end-users’ applications. It has already been installed more than 44,000 times as of May, 2010.
Acknowledgements

None of this work would be possible without the support of my advisor, Fabián Bustamante. Working with Fabián has been a privilege. I was told (and I continue to tell others) that it’s not what your work on, but who you work with, that determines your success in grad school. In that respect, I don’t think I could have ended up with a better advisor. He shelters me from administrative hassles, makes sure I have the resources I need for my work, gets out the way once I get going and yet stays involved enough to make sure our work doesn’t jump the rails. His enthusiasm for research is contagious and his carefully targeted optimism has carried us through rough patches on numerous occasions. More than just the last name on my publications, he has been – and always will be – a wise mentor and a close friend.

I would also like to thank my thesis committee for their valuable feedback and encouragement through the years. In my early years at Northwestern, Peter Dinda served as both as a mentor and a guide to help expand my horizons in the area of systems research. Collaborations with Yan Chen and his group helped me to plunge into the details of the dark underbelly of the Internet. Last, Lorenzo Alvisi has provided nothing less than enthusiastic support during my thesis work, and his unique perspective has been critical to making the work accessible to a broader audience.

I have had the privilege of working with a large number of graduate and undergraduate students throughout my time here. Stefan Birrer helped me get started way back in 2004, while Mario Sánchez, Zach Bischoff and John Otto have helped with managing and mining our large
dataset from P2P users. I also must thank Nikola Borisov, for taking on (and completing) every task we gave him, often beyond our expectations.

Besides my colleagues at Northwestern, I am grateful for the numerous research opportunities made available by my mentors from industry. Mark Astley (formerly IBM Watson) gave me the opportunity to finally work on a real operating system, and his unfiltered advice remains useful to this day. Jennifer Yates and Zihui Ge from AT&T Labs Research helped tremendously with motivation and validation of this work. Kobus van der Merwe took me under his wing during my 2009 summer in Florham Park, and I will continue to benefit from his perspective and vision for many years to come.

Last but not least, I must thank my wife Erin, for putting up with the lifestyle of graduate student for six years and for keeping me grounded. I also could not have made it this far without the unconditional support of my parents Albert and Susan, and thoughtful advice from my brother Daniel.
This material is based upon work supported by the National Science Foundation under Grant No. 0644062. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>3</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>5</td>
</tr>
<tr>
<td>Preface</td>
<td>7</td>
</tr>
<tr>
<td>List of Tables</td>
<td>11</td>
</tr>
<tr>
<td>List of Figures</td>
<td>12</td>
</tr>
<tr>
<td>Chapter 1. Introduction</td>
<td>15</td>
</tr>
<tr>
<td>1.1. Overview</td>
<td>16</td>
</tr>
<tr>
<td>1.2. Contributions</td>
<td>19</td>
</tr>
<tr>
<td>1.3. Related Work</td>
<td>22</td>
</tr>
<tr>
<td>1.4. Roadmap</td>
<td>24</td>
</tr>
<tr>
<td>Chapter 2. Network Monitoring at the Edge</td>
<td>27</td>
</tr>
<tr>
<td>2.1. Detecting Events</td>
<td>27</td>
</tr>
<tr>
<td>2.2. A Case for Edge Monitoring</td>
<td>28</td>
</tr>
<tr>
<td>2.3. Challenges</td>
<td>30</td>
</tr>
<tr>
<td>Chapter 3. CEM Framework</td>
<td>34</td>
</tr>
<tr>
<td>3.1. Architecture</td>
<td>34</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Comparison of event detection approaches. 29

3.1 Signals available when monitoring applications. 40

4.1 Summary of Ono’s P2P vantage points. 57

6.1 Comparison with events from a North American ISP. 86

6.2 Top 10 ISPs by users. 89

6.3 Top 10 ISPs by events. 89

6.4 Cross-network events detected. 90

6.5 Example cross-network events. 90
List of Figures

3.1 Schematic view of the proposed event detection approach. 35
3.2 Effects of key parameters on relative likelihood. 46
4.1 Data-collection system architecture. 55
4.2 Number of installed clients over time. 59
4.3 Growth in unique prefixes covered by Ono. 59
4.4 P2P traffic volume not mapped to public paths. 61
5.1 Upload rates during confirmed network event. 64
5.2 Moving averages facilitate identification of network events. 66
5.3 Timeline of the maximum performance drops for at least $n$ peers. 67
5.4 P2P network coverage, based on number of hosts per BGP prefix. 69
5.5 Timeline of likelihood ratio for different moving average settings. 73
5.6 Timeline of performance in India during disruption. 78
5.7 Timeline for performance in Israel during disruption. 80
5.8 Timeline for performance in Egypt during disruption. 82
6.1 Diagram indicating the portion of reported events detected by CEM. 84
6.2 Timelines for performance during events in a North American ISP. 85
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>Number of peers online simultaneously in this study.</td>
</tr>
<tr>
<td>6.4</td>
<td>Robustness of likelihood ratio to parameters.</td>
</tr>
<tr>
<td>6.5</td>
<td>Number of detected events over time.</td>
</tr>
<tr>
<td>6.6</td>
<td>Number of event reads and writes over time.</td>
</tr>
<tr>
<td>6.7</td>
<td>Cumulative number of DHT reads and writes over time.</td>
</tr>
<tr>
<td>7.1</td>
<td>Flowchart diagram of the NEWS corroboration procedure.</td>
</tr>
<tr>
<td>7.2</td>
<td>Screen shot of NEWS plugin view.</td>
</tr>
<tr>
<td>7.3</td>
<td>Screen shot of NEWS alert message.</td>
</tr>
<tr>
<td>7.4</td>
<td>Architectural diagram of NEWS Collector.</td>
</tr>
<tr>
<td>7.5</td>
<td>Screen capture of NEWSight interface.</td>
</tr>
</tbody>
</table>
## Listings

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Pseudocode for CEM</td>
<td>36</td>
</tr>
<tr>
<td>5.1</td>
<td>Pseudocode for NEWS local detection</td>
<td>70</td>
</tr>
<tr>
<td>5.2</td>
<td>Simplified code sample for NEWS corroboration</td>
<td>74</td>
</tr>
<tr>
<td>7.1</td>
<td>Pseudocode for NEWS corroboration procedure</td>
<td>101</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

Monitoring network performance is a complex, challenging problem. Due in part to the explosive growth of broadband Internet access worldwide, today’s networks consist of large numbers of (often) heterogeneous devices connecting Internet Service Providers (ISPs) to one another according to ever-changing business relationships. Further, many ISPs rely on manual configuration to implement policies (e.g., class of service and virtual private networks) across collections of geographically dispersed routers – creating significant opportunities for misconfigurations leading to disruptions and/or reduced performance. Finally, as the number of points of failure grows with the size and complexity of a network, monitoring the entire network quickly becomes intractable and consequently many failures go undetected for long periods of time.

As end users increasingly rely on their providers for Internet services such as video streaming, voice over IP (VoIP) and content sharing, minimizing the duration and severity of network problems is essential for ISPs to retaining subscribers and their revenue. Toward this goal, network providers commonly use a combination of actively gathered path measurements (e.g., latencies and loss rates) and passively gathered network-layer information (e.g., Netflow statistics and BGP data). However, because the volume of active path measurements grows with the square of the number of endpoints to monitor, existing systems must limit the number of links they cover and the frequency with which they are probed. Passive measurements such as those from Netflow and BGP snapshots provide a large volume of data, but gathering, storing
and extracting meaningful data online (i.e., within small numbers of seconds) from this information is challenging. As such, network performance outside of providers’ core links is rarely monitored, leaving ISPs with little to no view of the edges of the network – where most of the users are located and most of the failures occur.

This thesis posits that an effective way to address these limitations is to detect changes in network state that impact performance (i.e., network performance events) by pushing network monitoring to hosts at the edge of the network. In this approach, each host identifies potential network events using its local view of network performance, then gathers events reported by hosts in the same network to corroborate them. By reusing network performance data gathered passively from running applications and publishing only summaries of suspected problems, this technique addresses the scalability issues with network monitoring, thereby allowing online detection and isolation of events that impact performance for networked services.

The remainder of this introductory chapter is organized as follows. An overview of this dissertation is in Section 1.1. Next, Section 1.2 lists the major contributions of this work. Section 1.3 summarizes key related research in network monitoring and a roadmap for the remainder of the dissertation is in Section 1.4.

1.1. Overview

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 [37].
Perceived user experience is in large part determined by the frequency, duration and severity of network events that impact a service. Unfortunately, these events – which include congestion, fiber cuts and routing misconfigurations – are commonplace in today’s large and diverse networks. There is thus a clear need to detect, isolate and determine the root causes of these service-level network events (i.e., those that impact applications) so that operators can resolve such issues in a timely manner (within seconds or minutes), minimizing their impact on revenue and reputation.

While today’s networks generally implement sophisticated infrastructure that detects and issues alarms when core network elements fail, there remains a class of failures that often go undetected – the so-called silent failures (in that they are silent to operators, but not to subscribers). Configuration errors (e.g., incorrect ACL settings), routing anomalies (e.g., routing loops), and router bugs (simply because routers are incapable of detecting their own internal failures) are common causes for silent failures that can impact performance for services. In addition, in-network alarms fail to provide information to help operators understand the impact of network events. Despite efforts to use infrastructure to monitor end-to-end performance for customers, the cost of deployment, the number of services to monitor and the volume of data generated by these solutions limit their scalability, response times and effectiveness.

Given these limitations, a natural solution would be to detect service-level events by monitoring the end systems where the services are used. However, detecting events from 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), isolated to specific network locations (e.g.,
BGP prefixes) and this information should be made available to ISPs for further troubleshooting and problem mitigation. Finally, the approach must facilitate a broad (Internet-scale) deployment of edge-system monitors, ensure user privacy and provide trustworthy event detection information.

This work proposes a solution that addresses these issues through *crowdsourcing event monitoring (CEM)*. CEM does this by pushing service-level event monitoring to the end systems where the services are used. Building on end systems has a number of clear advantages. First, the approach provides flexibility in the types of monitoring software that can be installed inside or alongside services, facilitating immediate and incremental deployments. Second, by leveraging the unique perspective of participating end systems, it offers the potential for broad network visibility into an increasingly opaque Internet. Finally, its collaborative model enables a highly robust and more scalable system by drawing from every node’s resources and avoiding any centralized components.

First, this dissertation describes the challenges faced by any edge-system monitoring approach and discuss potential solutions. This first problem is addressing the general problem of how to detect network performance events from the edge. Specifically, this work develops a framework for the CEM 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. This work addresses the issue using a large dataset of diagnostic information from edge systems, gathered from users running the Ono plugin [17] 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 online event detection.

To evaluate the effectiveness of the approach, this work compares NEWS-detected events with confirmed ones, and demonstrates that the CEM crowdsourcing approach detects network events worldwide, including events spanning multiple networks. The approach is robust to various parameter settings and incurs reasonably low overhead.

NEWS has already been installed 44,000 times, demonstrating not only the feasibility of the CEM approach for a real application, but also that there are appropriate incentives for widespread adoption. Ongoing work includes working with developers of popular software to instrument additional applications and services, such as VoIP and IPTV. 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. As of May, 2010, this public interface is undergoing beta-testing by ISPs.

### 1.2. Contributions

This work makes the following contributions.

**Edge system monitoring.** To push network event detection to the edge of the network, one must determine how to efficiently monitor performance from potentially millions of hosts in a completely decentralized manner. Chapter[^2] identifies the key challenges for addressing the problem in general, and formulates the approach used in this work to solve them. Next, we address the general problem of how to use edge-system monitoring to detect network performance events from the edge (Chapter[^3]). One of the primary challenges in this context is to decouple the cost of detecting network events from the number of hosts performing detection worldwide.

and from the number of networks that need to be monitored in the Internet. Toward this goal, we develop a framework for the CEM approach in which each end system performs a significant portion of event detection locally using passive monitoring, then publishes information only about events corresponding to its network. This approach enables a system with overhead that scales with the number of events impacting a service in a network.

**Framework for edge system event detection.** While performance monitoring at scale is an important first step toward crowdsourcing event detection, this work must also address the question of how to use the views of network performance gathered by uncontrolled (and potentially untrustworthy) hosts to reliably detect network problems. To this end, Chapter 3 presents a framework for a CEM approach in which each end system performs a significant portion of event detection locally, then uses a distributed approach for corroborating these events. Local event detection poses a separate set of challenges in determining whether multiple, concurrently detected events in the same network correspond to a network-wide event. As such cases may be due to the applications being monitored (e.g., the service is the cause of the event) or coincidence, CEM requires a technique for identifying when a set of local problems in a network corresponds to a network-wide event. This work uses a likelihood ratio technique, often used in the medical field for evaluating treatment effectiveness, for identifying likely network events.

**Acquiring network-wide views.** 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. Chapter 4 addresses this issue using a large dataset of diagnostic information from edge systems running the Ono plugin for the Vuze BitTorrent client, comprising more than 970,000 users as of May, 2010. In addition to using the data to guide the design of the CEM approach, we are making anonymized versions of this dataset available to
the community [2]. To date, our data-collection infrastructure has recorded more than 14 TB of raw performance data from users in over 200 countries.

**CEM performance evaluation.** Chapter 6 evaluates the effectiveness of the CEM approach by comparing the events NEWS detected with those made available by two large ISPs. This comparison requires a subset of events that both CEM and ISPs are designed to detect, and this work shows that most of the confirmed events fall in this category. In addition to comparing CEM-detected events with confirmed ones, the results demonstrate that CEM’s crowdsourcing approach allows us to detect network events worldwide, including events spanning multiple networks. The chapter also evaluates the cost of the approach in terms of the number of events detected and how much network overhead they generate. The results indicate that the local detection technique for CEM is robust to various parameter settings, generating a reasonable number of confirmed events across a wide range of detection sensitivity. The analysis further indicates that the overhead for detection is reasonably low over time and across networks.

**Prototype implementation.** The effectiveness of any monitoring and detection approach is largely determined by the success of its implementation and the utility of its deployment. To demonstrate that the CEM approach to event detection is practical, we design and implement the *Network Early Warning System (NEWS)*, a BitTorrent extension that performs online event detection. This extension has been publicly released with source code included, and it has been installed 44,000 times as of May, 2010. In addition to crowdsourcing event detection through passive monitoring of BitTorrent, the implementation addresses important issues of privacy by corroborating events without needing personally identifiable information, and ensures broad deployment through an explicit incentive model for users to adopt the software.
While end users can use this software to detect network events, often only network operators can fix them. To assist such operators with quickly resolving problems causing detected network events, we have also implemented NEWSight – a system that accesses live event information and publishes its results in real time. This public interface is under beta testing with ISPs.

1.3. Related Work

The problem of detecting network events (also called network anomalies) has attracted a large number of research efforts. This section present a brief review of related work; a full discussion of related work can be found in Chapter 8. In the context of network monitoring, CEM is the first approach for detecting, in an online manner, network events that impact performance for applications running on end user systems. The rest of this section describes how CEM relates to previous work, classified according to key properties of event detection systems.

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

Event types. A class of previous work focuses on detecting network events in or near backbone links, using data gathered from layer-3 and below. While these

\[\text{http://aqualab.cs.northwestern.edu/projects/news/newsight.html}\]
monitors can accurately detect a variety of events, they may miss silent failures (e.g., incompatible QoS/ACL settings) and their impact on performance.

Other work focuses on detecting network events from a distributed platform \cite{8,36,44,76}. Such work can probe only parts of the network made publicly visible – as shown later in this work, such public views capture a surprisingly small amount of real network flows. Further, these approaches are limited by the overhead of monitoring these paths, a cost that is in the worst case quadratic and in the best case superlinear with the number of networks.

Unlike CEM, these solutions do not correlate these events with user-perceived performance. CEM is the first approach to detect service-level network events and correlate their impact on application performance from the perspective of end users.

**Monitoring location.** CEM detects events that impact user-perceived application performance, by running on the end user systems themselves. While several researchers have proposed using end-host probing to identify routing disruptions and their effect on end-to-end services \cite{27,36,73,77}, they have focused on global research and education network (GREN) \cite{44,76} or enterprise \cite{25,35,54} environments and thus have not considered the impact of network events on application performance nor addressed the issues of scalability when running on end user systems.

Some commercial network monitoring tools generate flows that simulate protocols used by edge systems (e.g., Keynote \cite{37}). 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. The CEM approach does not require any new infrastructure, nor control of end systems, and thus can be installed on systems at the edge of the network. Several research efforts have investigated the idea of network measurement (actively...
and passively) from end users, e.g., DIMES [64] and Neti@home [65], but have not explored the use of their monitoring information for online network event detection.

**Measurement technique.** CEM detects events using *passive monitoring* of popular applications, which allows the CEM approach to scale to the vast numbers of users at the edge of the network while still detecting events in an online manner. In a similar vein, previous work has suggested that the volume and breadth of P2P systems’ natural traffic could be sufficient to reveal information about the used network paths without requiring any additional measurement overhead [22,76]. PlanetSeer [76] uses passive monitoring of a CDN deployed on PlanetLab [56], but relies on active probes to characterize the scope of the detected events. Casado et al. [12] and Isdal et al. [33] use opportunistic measurement to reach these edges of the network, by leveraging spurious traffic or free-riding in BitTorrent. Unlike these efforts, CEM takes advantage of the steady stream of natural, (generally) benign traffic generated by applications. Approaches that use active monitoring (e.g., [8,36]) are limited by the overhead for detection, which grows with the number of monitored networks and services. While CEM could be combined with limited active probes to assist in characterizing and localizing network events, it does not require them.

### 1.4. Roadmap

The remainder of this dissertation is organized as follows. Chapter 2 motivates the CEM approach for crowdsourcing network event detection by discussing the advantages of edge-system monitoring. The chapter details the challenges that any solution must overcome to bring the approach to fruition.
Chapter 3 addresses these challenges through a generic framework for crowdsourcing event detection. In particular, it presents an architecture wherein each host performs passive local event detection, publishes information only about suspected events to distributed storage, then uses reports of events from other hosts in the same network to determine whether a network-wide event is occurring.

Evaluating any system intended to work at an Internet scale must address the challenge of acquiring and using a representative dataset with which to evaluate its effectiveness. Chapter 4 describes the data collected by our Ono extension [17] to the Vuze BitTorrent client, and demonstrate that it is sufficient in scale and coverage to evaluate CEM. It also describes the set of confirmed network events that are compared with those detected by the CEM approach.

Chapter 5 describes how we use these datasets to design a system, called the Network Early Warning System (NEWS), for using BitTorrent to crowdsource network event detection. The chapter illustrates key features of NEWS using case studies from public events. Chapter 6 provides an evaluation of this approach in the wide area. In addition to comparing NEWS-detected events with those confirmed by ISPs, the chapter presents results from detecting events worldwide, including cross-network events. The results show that NEWS does not generate unduly large numbers of events across a wide range of local event detection settings, and that the overhead for corroborated network events is reasonably low.

Chapter 7 discusses how we implemented and deployed a prototype that implements the CEM approach for the Vuze BitTorrent client. This software has been installed more than 44,000 times as of May, 2010, validating the incentives for the CEM approach. The chapter also describes NEWSCollector, a service that crawls distributed storage for events and NEWSight, a public interface for presenting globally detected events and allowing experts to confirm them.
A discussion of related work can be found in Chapter 8, a summary of this dissertation’s contributions in Chapter 9 and we conclude in Chapter 10 by discussing the role of this work in its broader research agenda and the topics we would like to address as part of future work in this area.
CHAPTER 2

Network Monitoring at the Edge

A natural consequence of the Internet’s continued explosive growth is that service disruptions are the rule, not the exception. Since the level of reliability in a network is dependent on the time to identify and isolate problems, a great deal of previous work focuses on detecting changes in network state indicative of a potential problem. This work refers to such changes as network events. Such problems can be caused by congestion, link failures, configuration errors or malicious activity. Once an event is detected, information about its time and location are provided to a network operator for the purpose of root-cause analysis and resolution.

2.1. Detecting Events

In general, network event detection consists of monitoring a number of signals and using a detection algorithm to determine when there may be a problem. Examples of commonly used signals include round-trip time (RTT) latencies for sending packets between network devices, the rate of packet loss along a path, the volume of data flowing through a router, the paths used to route traffic inside a network and the set of Internet paths advertised by external networks through the Border Gateway Protocol (BGP).

Given a time series of these values, an event is detected when there is an abnormal change in one or more values. For latency and loss, the technique may be to simply set a threshold (e.g., more than 3% packet loss is an event). When evaluating aggregate flows at routers, principal component analysis and subspace detection has yielded promising results [42]. Likewise,
edge detection using moving averages has been proposed for detecting sudden drops in throughput [9].

Network providers use a variety of techniques to monitor the state of their core links and detect most issues affecting their corresponding performance. Despite the effectiveness of these solutions, there remains a class of failures that often goes undetected – the so-called silent failures. Configuration errors (e.g., incorrect ACL settings), routing anomalies (e.g., routing loops), and router bugs (simply because routers are incapable of detecting their own internal failures) are common causes for silent failures that can impact performance for services.

In addition to silent failures, providers cannot detect problems in regions of the network that they do not monitor. These regions are usually located at the edges of network (for example, last-mile links), where most subscribers are located.

### 2.2. A Case for Edge Monitoring

To address these limitations, we propose detecting service-level events, as they occur, through monitoring software that runs inside or alongside applications on the end systems where they are used. Rather than replacing existing (and effective) monitoring infrastructures currently in place, CEM is designed to extend event detection to the kinds of outages currently visible only to end-to-end monitoring platforms (e.g., silent failures).

Table 2.1 summarizes how the CEM approach differs from previous work in network event detection. At a high level, CEM offers the following key differences and advantages. First, it reveals the application’s view of performance, while most previous work focuses on aggregated
views and inferred end-to-end performance. Acquiring this service-level view of network performance also allows us to rule out sudden changes in performance that are expected behavior for an application.

By using a decentralized approach to detection that runs on end systems, CEM avoids the need for new infrastructure nor access to proprietary ISP information (e.g., router-level views). Last, because the CEM approach is designed to run in (or alongside) popular networked applications installed at the edge of the network, its coverage of network events grows naturally with the Internet itself. As a result, CEM provides visibility into regions of the network currently hidden from public views such as BGP feeds, and it can detect events that span multiple administrative domains.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Event type</th>
<th>Coverage</th>
<th>Detection time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISP monitoring</td>
<td>Failures [27, 36, 73, 77]</td>
<td>Network Core Online</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chronic events [48]</td>
<td>Network Core Offline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IPTV [47]</td>
<td>Network/Service Core-Edge Offline</td>
<td></td>
</tr>
<tr>
<td>GREN monitoring</td>
<td>All pairs (active) [8]</td>
<td>Network/Service GREN Online</td>
<td>O(h) time</td>
</tr>
<tr>
<td></td>
<td>All pairs (passive) [76]</td>
<td>Service GREN Online</td>
<td>O(n) time</td>
</tr>
<tr>
<td></td>
<td>Distributed probes [36]</td>
<td>Network GREN-to-Visible Edge Online</td>
<td></td>
</tr>
<tr>
<td>CEM</td>
<td>Services/OS (passive)</td>
<td>Service Edge-to-Edge Online</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1. Comparison of network event 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.
2.3. Challenges

While promising, CEM poses a number of important design and implementation challenges. As discussed below, the issues of scalability and detection granularity are general to any monitoring system. In the context of pushing monitoring to end systems, CEM must also address the issues of privacy, reliability and adoption.

**Scalability.** When pushing the coverage of a monitoring system toward the edge of the network, the number of included 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, managing data for online event detection may require costly dedicated infrastructure [24].

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, the CEM approach facilitates an immediately deployable, scalable monitoring system. Specifically, the overhead incurred by CEM (e.g., CPU, memory and network consumption) should scale with the number of events in a network, as opposed to the number of hosts performing monitoring or the number of signals being monitored. By publishing only compact summaries of locally detected events, the amount of network activity generated by the approach is minimal.
**Granularity.** Any online 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 that potentially indicates a problem. By gathering and processing performance information locally at each end system, CEM can detect events with fine granularity (on the order of seconds) and relatively low CPU and memory overhead. By comparison, in-network active monitoring tools commonly provide five-minute granularity, while those launched from distributed research platforms can take tens of minutes [36] if not hours.

To isolate the scope of network events, CEM uses multiple locally detected events from the same network location. Similar to many distributed monitoring tools, CEM can take advantage of publicly available network regions such as BGP prefixes and AS numbers. More importantly, CEM can incorporate any structured network information. For example, this work will show how we have successfully used whois information to localize groups of hosts and exploited richer information such as AS relationships and detailed topologies allow us to detect 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. By passively monitoring performance signals and processing them locally, CEM obviates the need for publishing information that could reveal the details of user activity. In fact, CEM need
only distinguish when locally detected event reports come from different users. As discussed in Chapter[7] this allows a CEM deployment to remain effective without publishing any PII (or even IP addresses).

**Trust.** Most existing network event detection approaches are implemented as closed, controlled 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. Section[5.2] proposes several ways to harden an implementation against such attacks.

**Deployment model.** Any network event detection approach is limited by the coverage of its deployment. ISPs can fully control the coverage of their monitoring systems in the sense that they can place monitoring devices and software at arbitrary locations in their network. In practice, however, ISPs are limited by the fact that the cost of deploying and maintaining monitoring devices scales with the size of the monitoring deployment. Further, devices installed at the edge of the network (e.g., cable modems) are often provided by third-party vendors that limit opportunities for monitoring.

Adding monitoring software to distributed research platforms such as PlanetLab is typically “free”, but is limited by the portions of the network visible to participating hosts. As we have shown in previous work[16], this view misses a large portion of the Internet that is actually used by edge systems.

As an application-layer approach, there is no cost to deploy CEM and there are practically 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
operating system, providing it as a separate application running 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 [58], altruism [64] and mutual benefit [17].

Based on the success of Ono [17], we use a mutual benefit model where providers and subscribers both gain from participation. In this instance, subscribers (i.e., those running monitoring software) benefit from immediate notification and logging of network performance problems while network providers receive a more detailed view of their network for improving the quality of their service. This has been sufficient for a prototype implementation of CEM already installed over 44,000 times as of May, 2010. While the current CEM prototype is designed to run on top of BitTorrent, the approach generalizes to any application with large numbers of active users (e.g., VoIP and IPTV).

The next chapter addresses many of these challenges with a general approach to performing service-level network monitoring from edge systems.
CHAPTER 3

CEM Framework

This chapter develops a general framework for crowdsourcing network event monitoring. The next section presents a high-level architecture for the system, and is followed by a discussion of how to use locally detected events to diagnose widespread issues.

3.1. Architecture

As discussed in the previous chapter, the CEM approach can be deployed as part of an operating system, a separate application running as a background service or inside a monitored application. Regardless of the deployment model, the approach consists of edge system monitors (ESMs) installed on end systems to detect service-level problems associated with one or more networks. The approach assumes that each ESM has access to one or more sources of performance information (e.g., transfer rates, changes in latency and dropped packets) and that information can be gathered to form a time series for event detection. Another assumption is that each ESM can connect to a distributed storage system to share information about detected events. Examples include distributed hash tables (DHTs) and cloud-based storage systems.

Figure 3.1 depicts the CEM architecture and Listing 3.1 presents Java-style pseudocode for high-level aspects of how CEM operates. In the figure, the bottom right portion of the diagram depicts hosts in a distributed system, each running CEM software depicted as shaded rectangles.
As 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 affecting each ESM using only locally available performance data, gathered mostly through passive monitoring (step (1) of the figure; line 8 in pseudocode). In Section 3.2, we discuss how CEM processes performance information to detect events (line 10) and rules out changes in performance that are normal application behavior (line 13).

Local event detection presents new design challenges for determining the scope and severity of events. Specifically, CEM must enable participating hosts to acquire network-wide views to corroborate events detected through their local view. CEM addresses this through a decentralized approach to disseminating information about detected events and the network(s) they impact.
In particular, each edge system publishes its locally detected events to distributed storage (step (2) in Fig. 3.1 line 21 in pseudocode), allowing any other participating host to examine these aggregate events. Distributed storage additionally offers the appealing properties of scalability and resilience, facilitating the retrieval of event information even during problems such as network partitions.

Of course, multiple hosts may detect local events at the same time by coincidence, and not because of a network event. Section 3.3 discusses how CEM determines the likelihood that a set of these locally detected problems actually corresponds to a network event.

In the CEM 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. 3.1 line 25 in pseudocode), then determine whether these local events indicate a network event (line 48). Additionally, a third-party system (e.g., one maintained by an ISP or other organization) could use the distributed store to perform the analysis (step (4) in Fig. 3.1). 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. A prototype implementation of such a service is presented in Chapter 7.

Listing 3.1 presents the high-level pseudocode corresponding to the key aspects of this architecture. The while loop (starting on line 6) consists of the logic for local event detection, while the remoteEventCallback (line 32) corresponds to the functionality for corroborating locally detected events. The following sections expand on the above architecture and pseudocode for edge detection, beginning with how to detect performance events at end systems.

Listing 3.1. Pseudocode for CEM.
/* global state */

// performanceSignals: collection of performance signals to monitor
// groupings: collection of network locations for local host
// sampleInterval: time between signal samples

while (true) {
    /* iterate over each performance signal */
    for (String signal : performanceSignals){
        /* perform event detection on signal */
        boolean eventDetected = detectPerformanceDrop(signal);
        /* use application level information to determine if event was normal behavior */
        if (eventDetected && !isExpectedDrop(signal))
            addToLocalEvents(signal);
    } // end for each performance signal
    /* if event detected, publish it and corroborate it */
    if (localEventCount>0){
        /* perform for each network grouping (e.g., BGP prefix) */
        for (String networkGrouping : groupings){
            /* publish the event */
            publishLocalEventSummary(networkGrouping);
            /* retrieve events detected by remote hosts in the same network group; remoteEventCallback (defined below) enables asynchronous operation */
        }
    
}
retrieveRemoteEvents(networkGrouping, remoteEventCallback);
}
} // end if there is a local event

/* perform once per sample interval */
sleep(sampleInterval);
} // end signal sampling loop

void remoteEventCallback()
{
    /* after retrieving remote events, perform 
       corroboration per network grouping */
    for (String networkGrouping : groupings){
        /* for each local event to corroborate */
        for (long timestamp : getLocalEventTimes(networkGrouping)){
            /* grab remote events detected at same time */
            Collection<Event> remoteEvents = getRemoteEvents(timestamp);
            /* determine relatively likelihood that the observed events are 
               occurring compared to the probability they would happen by 
               coincidence */
            double relativeLikelihood =
                observedDetectionRate(remoteEvents) /
                probabilityCoincidence(remoteEvents);
            /* if detected events occur sufficiently more often than chance, 
               event is corroborated as likely network event */
            if (relativeLikelihood > likelihoodThreshold)
3.2. Local Detection

The first step in CEM is to analyze local performance information to determine whether the monitored host is experiencing a problem. This section discusses the types of available performance signals and techniques for detecting local performance events.

3.2.1. Performance Signals

By pushing detection to end systems located at the edge of the network, CEM can use a wide variety of service-level information to diagnose local performance problems (Table 3.1). Examples of these performance signals available to any monitored application include flow and path-quality information such as throughput, loss, and latencies. The approach can also incorporate service-specific information to distinguish normal performance changes from potential network events. For instance, P2P file-sharing systems can provide information about whether a transfer has completed and a VoIP application can indicate when there is silence. The approach additionally can use system-level information for local event detection. For example, the operating system can provide information about throughput consumed by all running applications, allowing CEM to account for the performance impact of concurrent applications. Because these types of information can be gathered passively, they can be sampled frequently so that events are detected as soon as they occur.
<table>
<thead>
<tr>
<th><strong>Signals Generally Available</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall upload rate</td>
<td>Overall download rate</td>
</tr>
<tr>
<td>Per-connection upload rate</td>
<td>Per-connection download rate</td>
</tr>
<tr>
<td>Connected hosts</td>
<td>RTT latencies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Service-Specific Signals</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BitTorrent</strong></td>
</tr>
<tr>
<td>Availability</td>
</tr>
<tr>
<td>Number available leechers</td>
</tr>
<tr>
<td>Number active downloads</td>
</tr>
</tbody>
</table>

| **VoIP**                    |
| Jitter                      | Voice packet loss            |

| **IPTV**                    |
| Dropped frames              | Channel changing delay       |

Table 3.1. Signals available when monitoring applications.

It is important to note that CEM does not require active measurements for detecting events. However, to assist with diagnosing network problems, the approach can incorporate limited active measurements such as traceroutes, pings and available-bandwidth probes. The design and deployment of such hybrid measurement systems is left as future work.

### 3.2.2. Local Event Detection

CEM uses signals described in the previous section to detect local performance events. The goal of local detection is to identify performance problems, then provide sufficient information for determining the scope of the problem, i.e., whether the problem is local (isolated to single ESM) or network-related. To this end, CEM must incorporate a detection technique that identifies performance problems from the set of available signals. The output of local detection is a summary for each event describing its type (e.g., unexpected drop in throughput, lost video frame), the time of detection, where in the network it was discovered and how it was detected.
The types of events that can be detected and the appropriate technique to detect them are dependent on the service being monitored. For instance, when monitoring end-to-end throughput for a host (e.g., for video streaming), we show that moving averages can identify sudden drops in transfer rates potentially caused by a network issue like congestion. In the domain of IPTV [47], video quality (among other factors) may indicate problems with the network. Alternatively, a P2P file-sharing application may detect dropped connections for a subset of its peers, indicating a network event impacting only certain prefixes (e.g., due to routing issues). Finally, a VoIP application may experience sudden jitter that impacts call quality. CEM is agnostic to how these events are detected, so long as they correspond to service-level problems.

Correlating local events. Performance changes for monitored services do not necessarily indicate widespread problems. In a P2P 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 one removes these confounding factors from the analysis, the confidence that a detected problem is independent of the monitored service improves. Similarly, concurrent events occurring in multiple performance signals for a service (e.g., download and upload rates), further increases the confidence that the event is independent of the service. As the following chapter shows, incorporating this service-level information improves the quality of event detection.

Publishing local events. After detecting a local event, CEM determines whether other hosts in the same network are seeing the same problem at the same time, which requires hosts to share
local event detection results. Because the approach must scale to potentially millions of hosts, distributed storage (e.g., a DHT) is a viable medium for sharing these events.

### 3.3. 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 source.

#### 3.3.1. Corroboration or Coincidence?

To identify events impacting a particular network, CEM first gathers a list of events reported by monitors in that network. This can be done periodically or on demand (e.g., in response to events detected by an ESM). If multiple events occur at the same time in the same network, CEM 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.

The following paragraphs discuss how CEM accounts for service-specific dependencies and correlated events that occur by coincidence. After accounting for service dependencies, CEM tests the null hypothesis that each host experiences events *independently at random* and not due to network problems. By comparing this value to the observed rate of local events occurring
concurrently for hosts in a network, CEM can determine the relative likelihood of the detected problem being caused by the network instead of by chance.

**Eliminating confounding factors.** To test the null hypothesis that groups of hosts experience events independently at random, one must first determine the probability that each host detects local problems independently. Given a set of participating hosts $H$, each host $h$ produces a series $A_h = \{a_{h,i}, a_{h,i+1}, \ldots, a_{h,j}\}$ 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 $T$, the observed detection rate is the estimate of the probability of host $h$ detecting a local event at any given time:

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

If two hosts’ performance are mutually dependent at a given time (e.g., because they are transferring data with each other), the independence assumption mentioned above does not hold. Thus, to control for such dependencies in concurrently detected events, any set of hosts whose performance is mutually dependent during a time interval $(i - 1, i]$ are treated as the same host during that interval for the purpose of the analysis. In this case, such hosts do not corroborate each other’s events. To give a more concrete example, in the context of 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.\(^1\)

\(^1\)Note that if the goal is to detect problems caused by a service-provider disruption instead of a network event, one need only change the order of grouping. This work, however, focuses on detecting service-level problems caused by network events.
After this step, CEM 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:

\[
P\left(\bigcup_{h=1}^{N} L_h\right) = \sum_{h=1}^{N} P(L_h) - \sum_{j>h=1}^{N} P(L_h \cap L_j) + \ldots + (-1)^{n-1} P(L_1 \cap \ldots \cap L_N)
\]

(3.1)

Given that the null hypothesis is that the events are independent, one can simplify the union probability:

\[
P\left(\bigcup_{h=1}^{N} L_h\right) = \sum_{h=1}^{N} P(L_h) - \sum_{j>h=1}^{N} P(L_h)P(L_j) + \ldots + (-1)^{n-1} P(L_1)\ldots P(L_N)
\]

(3.2)

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. Calculating this entails peeling off the first \( n - 1 \) terms of Equation 3.2. For example, the probability that at least two hosts will see concurrent events is:
The next step in the analysis is to use these equations to calculate the likelihoods of groups of hosts seeing events at the same time by coincidence, and show that this probability quickly decreases as the number of corroborating hosts increases.

**Effect of corroboration.** Intuitively, the confidence in a detected event being due to the network increases with the number of hosts detecting the event and the number of independent performance signals indicating the event. The following paragraphs quantify the impact of these factors through a simulation of a region of interest (e.g., a BGP prefix) with \( N \) hosts.

In the simulation, each of these hosts provides multiple performance signals as described in Section 3.2.1. The probability of host \( h \) witnessing an event in one signal, \( L_h^1 \), is chosen uniformly at random in the range \( 0.005 \leq L_h^1 \leq 0.05 \). Similarly, the probability of witnessing a local event concurrently in two signals, \( L_h^2 \), is chosen uniformly at random from the range \( 0.005 \leq L_h^2 \leq L_h^1 \) and the range for three signals, \( L_h^3 \), is \( 0.005 \leq L_h^3 \leq L_h^2 \). This approach simulates the typical scenario that events in multiple independent performance signals are less likely than events in any single signal.

These simulations can be used to determine the probability of \( c \) hosts (1 < \( c \) ≤ 5) seeing an event by coincidence for networks with \( N = 10, 25, 50 \) hosts, then compare this value with the
Figure 3.2. The effects of key parameters on the likelihood of correlated events when compared to uncorrelated events. Relative likelihoods decrease as the number of corroborating hosts and signals increases.
probability of any one host seeing an event. For each setting, the graphs are based on aggregate results from 100 randomly generated networks.

Figure 3.2(a) 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. There is a similar effect when varying the number of signals detecting local events (Fig. 3.2(b)) – the more signals experiencing performance events concurrently, the less likely it is 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 discussed at the beginning of this section, a goal of CEM is 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. Likelihood ratios are often used, for example, in the field of medicine for diagnostic testing to determine the probability that a condition (e.g., a disease) is present.

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

The larger the value for $LR$, the more confident is the conclusion that detected problems are due to the network. As Chapter 5 shows, a value $LR > 2$ is sufficient for detecting confirmed events in a large ISP. In fact, the threshold value for $LR$ serves as a single “tuning knob” that operators and users can modify to control the frequency with which network events are detected. The impact of this setting is discussed in Chapter 6.

### 3.3.2. Problem Isolation

After detecting a local event, CEM can use information published in event reports to determine the scope of the problem. If the event is local to a host (i.e., no other hosts report problems), it is likely an isolated issue (e.g., a home router malfunction). However, 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, CEM 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.

CEM supports localization of problems using structural information about the organization of networks and their geographic locations. For instance, it can use events detected by hosts in the same routable BGP prefix or ASN, and use geographic information to localize events to cities and countries. Further, CEM can use an AS-level Internet graph to localize network issues to upstream providers or a router-level graph to isolate problematic routers and links. Finally, CEM could use a variety of other sources of location information, ranging from identifying codes to data gathered from whois listings. In the end, the granularity of detected
problems is directly tied to the level of detail in the available localization information. As the following chapters show, even publicly available information such as BGP prefixes is sufficient for detecting problems.
The previous section described the CEM approach for detecting events from edge systems. Designing, deploying and evaluating CEM, however, poses interesting and significant challenges given the absence of a platform for experimentation at the edge of the network or 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 \[32\]. By avoiding the need to deploy additional infrastructure and offering hosts that are already cooperating \[29\], these systems are an appealing vehicle for monitoring – one that grows naturally with the network \[22,76\].

Based on these advantages, we designed and evaluated a prototype implementation of CEM in a large P2P system. To guide its design and evaluate its effectiveness at scale, our analysis takes 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 used for evaluation, and a particular case study that motivates our implementation. The section concludes with a description of the Network Early Warning System (NEWS), a 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 \[70\], to facilitate adoption and to piggyback on the application’s large user base.
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. The next chapter shows that despite these challenges NEWS can detect network events simply by passively monitoring BitTorrent, thus validating the CEM approach.

The remainder of this chapter describes the edge-system dataset used in this work, how the data is collected and the uniqueness of the view it provides. These section closes by describing the confirmed network problems used to validate NEWS.

4.1. Data Collection Infrastructure

The realization of NEWS is guided by measurement data gathered 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 [17]. Because this service can, in fact, improve performance for end users, it enjoys a large-scale adoption that has provided an unparalleled opportunity for network measurement and, with it, enormous challenges for data collection. Before discussing the details of this collection infrastructure, the following paragraphs provide some brief details about the Ono client instrumented to collect performance measurements.

**Taming the Torrent.** Peer-to-peer (P2P) systems, which provide a variety of popular services, such as file sharing, video streaming and voice-over-IP, contribute a significant portion
of today’s Internet traffic. By building overlay networks that are oblivious to the underlying Internet topology and routing, these systems have become one of the greatest traffic-engineering challenges for Internet Service Providers (ISPs) and the source of costly data traffic flows. In an attempt to reduce these operational costs, ISPs have tried to shape, block or otherwise limit P2P traffic, much to the chagrin of their subscribers, who consistently finds ways to eschew these controls or simply switch providers.

Ono addresses this problem through an approach to reducing this costly cross-ISP traffic without sacrificing system performance. A key feature of Ono is that it recycles network views gathered, at low cost, from CDNs to drive biased peer selection without any path monitoring or probing. Following from the observation that CDN redirections are driven primarily by latency [67], Ono is derived from the hypothesis that if two peers exhibit similar redirection behavior, they are likely to be close to one another. In many cases, we expect that these nearby peers will be mostly within the same ISP, thus avoiding cross-ISP traffic and optimizing clients’ performance by avoiding most network bottlenecks [6].

Unlike previous oracle-based proposals [5][10], our CDN-based approach does not require new infrastructure and does not depend on cooperation between ISPs and their subscribers. This work is a concrete example of the use of “recycled” information, gathered by long-running services such as CDNs, in building more efficient services — one instance of a negative feedback loop essential to Internet scalability.

To validate our approach, we made Ono freely available as an extension to the popular Azureus BitTorrent client beginning in April 2007. Using results collected from a deployment in BitTorrent with over 120,000 users in nearly 3,000 networks, we showed that our lightweight approach significantly reduces cross-ISP traffic and over 33% of the time it selects peers along
paths that are within a single autonomous system (AS). Further, we find that our system locates peers along paths that have two orders of magnitude lower latency and 30% lower loss rates than those picked at random, and that these high-quality paths can lead to significant improvements in transfer rates. In challenged settings where peers are overloaded in terms of available bandwidth, our approach provides 31% average download-rate improvement; in environments with large available bandwidth, it increases download rates by 207% on average (and improves median rates by 883%).

As of May, 2010, the software has been installed more than 970,000 times in 200 countries and typically has between 10,000 to 20,000 users online per day. In addition to implementing our scalable biased peer selection technique, the software performs network measurements and records information about file-transfer performance (as described in Section 4.2).

Collecting Internet-scale measurements with spare parts. The design, implementation and deployment of a hardware and software infrastructure to support this data collection is the topic of an ongoing systems project. To date, we have recorded on the order of 14 TB of raw performance data; each day we must support hundreds to thousands of concurrent data-reporting connections from users, perform online archival and ensure that user performance is not negatively affected by data collection.

The design of any data collection infrastructure must be guided by the scale of its contributing hosts. While testing the Ono software in a controlled environment, active clients never numbered more than a few hundred (based on the size of the PlanetLab deployment). In this context, each client simply reported its measurements directly to database servers using a MySQL interface built into the Ono software.

\footnote{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.}
To evaluate the effectiveness of Ono in the wild, we added a link to the software from a Wiki page describing the BitTorrent client that supports it. Within weeks we attracted thousands of users who installed the client and reported data to centralized servers. Despite using a server-class machine to host the database, there were simply too many connections established (and maintained) in parallel, and a more scalable solution was required. In particular, we require an infrastructure that supports thousands of concurrent clients establishing short-lived connections, and a data recording solution that permits regular archival of reported data without interrupting ongoing data recording. Finally, and perhaps most importantly, the solution must ensure that any issues with data collection on the server side are invisible to clients of our software. If these issues affect the user experience (e.g., causing annoyance), users are likely to uninstall or otherwise disable the software.

Our data-collection system is a three-tier Web service, depicted in Figure 4.1. On the client side, Ono software continuously records data and periodically flushes it to our data collection infrastructure. In particular, Ono uses a gzip stream to reduce the network overhead for transmitting data and does so at regular intervals to prevent the need to transmit large volumes of data at the end of a user session. In fact, because users may restart their BitTorrent client (e.g., due to software upgrades) Ono must not impose an unreasonable delay while reporting data at the end of a session. Ono thus reports data for at most one minute at the end of a session and do so in small batches to ensure the maximum possible partial information is recorded.

The data is reported to one of several Web servers running Apache and using PHP scripts to parse and store the data. Using Web servers allows us to dynamically change how and where the data is actually stored, and to make these changes in response to variations in load. To achieve good performance, each server is configured with extensive optimizations for data
Figure 4.1. High-level architecture of our data-collection system. The top of figure illustrates the first tier, which corresponds to clients located throughout the Internet. The bottom of the figure represents the server-side portion of our infrastructure, located in private networks.

collection, including network settings to tune connection performance and PHP opcode caching to minimize CPU consumption and processing time.

The last tier of our infrastructure is a set of servers running MySQL. Again, our system supports an arbitrary number of database servers that can be changed on demand, thus allowing us to adjust in response to the dynamic loads that Ono users generate.
Our system collects between 20 and 30 GB of raw data per day. To ensure manageable-sized archives, our system uses a cron job to perform a complete backup of our dataset every day. Once the day’s data has been compressed and replicated, it is deleted from the production database servers.

It is important to note that an archival discipline can interfere with ongoing data collection if there is contention for the same data. In this case, contention is the rule: there is always a client writing to a table at the same moment that the archival job needs to read from it. To address this issue, the data-collection servers maintain two replicas of the same database schema on our servers – an active and inactive database. The Web service uses values in a separate location to determine which database to use for reporting data. Thus, when the archival task begins, if first swaps the identities of the active and inactive databases, then it safely reads from the (now) inactive database without any contention.

During the past two years, we have successfully scaled this platform to support the increasing numbers of clients reporting data. To assist the community with data collection in other environments of similar scale, we will make our system design, and the software that supports it, publicly available.

4.2. BitTorrent Traces

Nearly 1,000,000 users as of May, 2010, distributed in over 200 countries, Ono is the largest end-system monitoring service. The following paragraphs describe the data collected; summary information about Ono users is in Table 4.1.
Table 4.1. Summary of Ono’s P2P vantage points.

<table>
<thead>
<tr>
<th>Category</th>
<th>Number (Pct of total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users</td>
<td>970,000 (3% of Vuze users)</td>
</tr>
<tr>
<td>Countries</td>
<td>200 (78%)</td>
</tr>
<tr>
<td>IP addresses</td>
<td>3,600,000</td>
</tr>
<tr>
<td>Prefixes</td>
<td>62,827</td>
</tr>
<tr>
<td>Autonomous systems (ASes)</td>
<td>8,000</td>
</tr>
<tr>
<td>IPs behind middleboxes</td>
<td>≈ 82.6%</td>
</tr>
</tbody>
</table>

**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 first six rows of Table 3.1 contain the complete list of signals collected from our deployment.

In addition to the above data, which is collected passively, the software performs traceroute measurements for evaluation purposes. The following section uses this data to build detailed topologies that highlight the advantages of an edge-based monitoring approach compared to current testbed deployments. Ono uses each host’s built-in (i.e., OS-level) traceroute command, and the target of each measurement is a randomly selected, connected peers. There is at most one traceroute issued at a time.

This data is reported to our data-collection infrastructure, which converts data timestamps to a universal time zone.
4.3. Uniqueness of Edge Views

Any dataset is subject to limits in the coverage of its measurement hosts. The Ono dataset currently contains connection information from users to more than 380,000,000 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. To put these numbers in context, Casado and Freedman’s extensive study on edge opacity relied on a dataset including IP addresses from about 85,000 prefixes and almost 15,000 ASNs.

The rapid growth in the number of participating peers and network coverage for the Ono service illustrates the promise of an end-system approach with sufficient incentives for adoption. Figure 4.2 plots the number of client installations during the nearly three years that Ono has been publicly available. The figure shows steady growth in the number of vantage points over time, with bows in the curve corresponding to seasonal effects (reduced P2P usage in the summer in the northern hemisphere).

These users implicitly monitor large portions of the Internet, which is ideal for wide area event detection. One way of evaluating this coverage is to examine the peers that users connect to in terms of the number of distinct BGP prefixes over time. Fig. 4.3 plots the number of unique routable prefixes containing users of our BitTorrent extension. These users reported data from more than 10,000 unique routable prefixes within the first 110 days of deployment; as of February 2010, the software has recorded data from more than 62,827 prefixes. As a point of reference, a popular infrastructure-based monitoring platform, PlanetLab, has grown to 470 sites over the course of six years.
Figure 4.2. The number of installed clients over time since the initial release of Ono. Discontinuities in the slope of the curve indicates press releases about the software.

Figure 4.3. Growth in unique prefixes where a P2P service is installed. It currently grows at a rate of 200-300 new prefixes per week.

Besides monitoring many paths from many distinct network locations, this 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 and are thus generally inaccessible to external distributed monitoring systems such
as those in GREN environments. Further, in a recent study [14] we showed 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.

To further emphasize the unique view captured from edge systems and not available to previous approaches using testbed platforms, it is useful to determine how much P2P traffic is covering paths invisible to public views. To accomplish this, the first step is to determine the volume of data transferred over each connection for each host, then map each connection to a source/destination AS pair using the Team Cymru service [68]. The next step is to gather a set of paths from public views and P2P traceroutes [14] and, finally, for each host to determine the portion of its traffic volume that could not be mapped to any AS path in the combined dataset.

Figure 4.4 uses a cumulative distribution function (CDF) to plot these unmapped traffic volumes using only BGP data (labeled BGP) and the entire dataset (labeled All). The figure shows that when using All path information, one can assign P2P traffic to AS paths the vast majority of the time. Specifically, while path information is incomplete for 66% of hosts; the median portion of unmapped traffic (i.e., flows not corresponding to an AS path) is only 0.4%. The figure also shows that cases where large portions of P2P flows are unmapped are extremely rare. For example, 4% of hosts use connections for which at least half of their traffic volumes are unmapped and less than 2% use connections for which there is no path information at all.

When using only BGP information, however, the median volume of unaccounted traffic is nearly 75% – nearly 20 times the same value when using all path information. In fact, complete path information is available for only 7.4% of hosts and 7.3% of hosts use connections for which BGP data provides no path information.

---

2For simplicity, this analysis assumes that publicly announced BGP paths coincide with those that data actually traverses.
One implication of this result is that while peer-based topology data adds 20% more links to the Internet graph, it allows us to map an order of magnitude more P2P traffic than using BGP alone. The discrepancy is not due to a failure of existing path measurements, but rather the limitations of their coverage.

Given these results, the only sound way to design a monitoring system intended for the edge of the network (where most users are located) is to obtain a trace representative of this environment. Using the Ono dataset is a large step forward toward this goal – one that has not been achieved by any prior work in this area.

### 4.4. Confirmed Network Events

In addition to extensive, representative network traces, it is essential to evaluate the effectiveness of a network event detection approach using 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 [59].

For many networks, obtaining a list of labeled events is challenging. Because these events may reveal issues that cause customers to change providers, it is rare to find publicly available lists. Absent this information, researchers usually must turn to nondisclosure agreements with cooperating ISPs. This work focuses on two sources of labeled events: a public list of events from a large British ISP and a proprietary list from a large North American ISP.

The first set of labeled data consists of publicly available event reports from the British Telecom (BT Yahoo) ISP[^1] 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.

The other set of labeled data consists of network problems reported from a large North American ISP. For nondisclosure reasons, this work 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, one can determine whether confirmed events are detected by NEWS, but one cannot draw strong conclusions about false positive and negative rates. This issue affects all research in network event detection.

The next two chapters use the above set of confirmed events to motivate, illustrate and evaluate the implementation of NEWS.

CHAPTER 5

Implementing CEM

This chapter discusses key design aspects of NEWS, a prototype edge-system monitor that implements CEM for BitTorrent. The following discussion uses a confirmed event to explain NEWS design choices.

5.1. The View from BitTorrent

To assist with the presentation of NEWS, this section focuses on one of the events in British Telecom (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.

It is important to note two key features of this event description that tends to be common in labeled data. For one, the description indicates that the ISP “became aware” of a problem at 3:54 PM. Though it is unclear exactly how they became aware, such events are either detected by in-network monitoring or, in many cases, reported through subscriber phone calls. Thus, the start time for the event should be interpreted as an approximation – one that is likely later in the day than the actual start time for the event. Similarly, the time at which the problem is marked as resolved is at best an estimate based on the time at which an operator validated the repair.

Figure 5.1 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

\footnote{All times reported in this case study use Greenwich Mean Time (GMT).}
region. Each point in the graph represents an upload-rate sample for a single peer; different point shapes/colors represent signals for different peers.

There are several key features of this graph that validate the CEM approach to event detection. For one, the samples take on a wide range of values and each host is present for an uncontrolled duration. Taken alone, each host’s view is noisy and incomplete, but in aggregate they can provide good coverage of this network over time.

In addition, the range of values for each host differs significantly – some hosts achieve at most 30 KB/s while others can achieve 50 KB/s. Simply aggregating these views and performing a moving average analysis on the ensemble would omit valuable information about each host’s baseline performance and make it difficult to distinguish network events from normal variations in performance among hosts.

5.2. Network Monitoring from BitTorrent

This section discusses key design aspects of NEWS using the confirmed BT Yahoo event in Fig. 5.1. With respect to the design challenges listed in Chapter 2, local event detection and
group corroboration address scalability and granularity; the remaining issues of privacy, trust and adoption are covered in the subsequent sections. Note that Chapter 7 presents low-level implementation details.

5.2.1. Local Event Detection

Any CEM implementation must define a service-level event that could be due to a network problem. In NEWS, these are defined as unexpected drops in end-to-end throughput for BitTorrent, which corresponds to steep drops in the time series formed by BitTorrent throughput samples. Monitoring for this type of event corresponds to detecting edges in the throughput signal; specifically, NEWS detects 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, \( \mu_i \), and the standard deviation, \( \sigma_i \) 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 at time \( i + 1 \).

To demonstrate visually how moving averages facilitate edge detection, Fig. 5.2 plots the 10-minute averages of upload rates for two groups of affected peers extracted from Fig. 5.1. Using these averages, it becomes clear from the top graph that there is a correlated drop in performance among a group of three peers at 14:54, 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. 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, it is clear that there were two distinguishable network problems corresponding to the single generic report.

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 the estimate of $\sigma$ and thus detection accuracy. In general, however, $\sigma$ varies over time, so increasing the window size reduces responsiveness to changes in $\sigma$. 

Figure 5.2. Moving averages facilitate identification of separate network events affecting transfer rates for two groups of peers during the same period shown in Fig. 5.1. Best viewed in color.
Figure 5.3. 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.

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 \( \sigma \) naturally ties the threshold to the variance in the signal, it is difficult \textit{a priori} to select a suitable value for \( t \). If our approach to local detection is viable, however, there should be some threshold \((t \cdot \sigma)\) for identifying peers’ local events that correspond to network ones. To demonstrate this is the case, Fig. 5.3 shows how deviations from the moving average for upload rates behave over time for peers experiencing the network problems illustrated in Fig. 5.2 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, it is difficult to compare them. NEWS addresses this problem by normalizing them using the standard deviation for the window (\( \sigma \)). If this 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 $2.5\sigma$ correspond to real network problems. This suggests that there are moving-average settings that can detect confirmed problems in this network. Section 5.2.2 shows how NEWS can extract network events from a variety of settings, using the likelihood analysis from Section 3.3.

**Confounding factors.** A drop in the throughput signal provided by a host BitTorrent application is not necessarily due to a network event (Section 3.3). For example, when a peer completes downloading a torrent, the download rate drops to zero for that torrent. Alternatively, when there is one or more pieces of a torrent missing from all members of a swarm, transfer rates will eventually all drop to zero even though none of the torrent transfers are complete. Thus, when monitoring BitTorrent it is essential to use service-specific information to distinguish expected behavior from network events.

NEWS uses several of the performance signals listed in Table 3.1 to eliminate well known confounding factors. For instance, NEWS tracks the transfer states of torrents and accounts for the impact of download completion. To eliminate performance problems due to the application (as opposed to the network), such as missing torrent data or high-bandwidth peers leaving a swarm, all peers connected to the same torrent are treated as the same 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.
Figure 5.4. CDF of the minimum, mean and maximum number of unique BGP prefixes seen per hour for each measurement vantage point during a 6-day period. The vast majority of P2P vantage points (99%) connect to peers in four or more prefixes during an average hour-long period; the median number is 137. This indicates that detected problems using BitTorrent are extremely unlikely to be due to remote networks.

NEWS also requires multiple performance signals to see concurrent events before publishing an event. As discussed in Section 3.3, improving the confidence that the event is independent of the application also improves the confidence that it is caused by the network. For example, a sudden drop in upload rates may indicate a network problem, but a sudden drop in both upload and download rates is an even stronger indication.

When detecting an event, NEWS must not only determine that there is a problem with a network, but specifically identify the host’s network as the one experiencing the problem. If a host’s connections were biased toward a remote AS, for example, then it would be unclear if detected problems were specific to the host’s AS or the biased one. One way to explore this issue is to determine the number of unique routable prefixes that each peer connects to per hour. Figure 5.4 uses as CDF to depict this result for the minimum, mean and maximum number of
hourly unique remote prefixes during a 6-day period in April, 2009. A point \((x, y)\) on the graph means that \(y\) percent of peers saw at most \(x\) unique prefixes.

The figure shows that, in an average hour, the vast majority of BitTorrent users (99%) connect to peers in four or more prefixes; the median is 137. Even in the worst case, when looking at the curve for the minimum number of prefixes that a peer connects to during any given hour, only 10% of P2P hosts ever connect to only one prefix per hour. The median value for this statistic is 24, meaning the vast majority of hosts connect to a large number of prefixes during any hour they are online. These results indicate that it extremely unlikely that problems in remote networks would be falsely interpreted as problems in the host’s network.

Listing 5.1 illustrates NEWS local detection using psuedocode. At a high level, local detection consists of updating the moving average for each signal (line 6), identifying significant deviations from the average in a signal (line 17), then determining whether that deviation can be explained as normal BitTorrent behavior (line 19). If not, a local event is detected and published (line 28).

---

**Listing 5.1. Pseudocode for NEWS local detection.**

```java
1 // signals: set of monitored performance signals
2 // stdDevThresh: current deviation threshold
3
4 /* sample each signal to update moving averages */
5 for (String signal : signals ){
6   updateMovingAverage(signal);
7 }
8```
/* check for anomalous state and account for dependencies */
for (String signal : signals){

    /* determine if an event in that signal has occurred */
    if (getMovingAverage(signal).isAnomalous(stdDevThresh)){
        if (isTransferRate(signal)){
            /* make sure a significant drop has occurred */
            if (getMovingAverage(signal).getDeviationPercent() < MIN_DEVIATION_PCT) continue;

            if (isNormalBehavior(signal)) continue;

            addAnomalousRate(signal);
        }
    }

    addAnomalousSignal(signal);
}

/* local event detected only if more than one performance signal has anomaly and one of the signals is a transfer rate */
if (getNumberOfRateEvents > 0 && getNumberOfTotalEvents>1){
    publishLocalEvent(); // publish event details

    retrieveRemoteEvents(); // get remote events
5.2.2. Group Corroboration

As discussed in Section 3.3 after detecting local events, CEM determines the likelihood that the events are due to a network problem. Thus, once a local event has been detected, NEWS publishes local event summaries to distributed storage so that participating hosts can access them as soon as they are detected.

The following paragraphs apply this likelihood analysis to the events in BT Yahoo as described in Section 5.1. Recall that the goal is to detect synchronized drops in performance that are unlikely to have occurred by chance. To that end, NEWS determines the likelihood ratio, \( LR = \frac{P_e}{P_u} \), as described in Section 3.3.1. This analysis uses one month of data to determine \( P_e \) and \( P_u \).

Figure 5.5 depicts values for LR over time for BT Yahoo using different local event detection settings. In both figures, a horizontal line indicates 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 one can attribute to the problem not being the application.

Figure 5.5 (top) uses a detection threshold of \( 1.5\sigma \) and window size of 10. Such a low threshold not surprisingly leads to many cases where multiple peers see problems at the same time (nonzero LR values), but they are not considered network problems because LR < 1. Importantly, there are few values above LR = 1, and the largest corresponds to a performance drop potentially due to TCP congestion control, since it occurs when peers have simultaneously saturated their allocated bandwidth after the confirmed network problem is fixed.
Figure 5.5. 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.

Figure 5.5 (bottom) uses a detection threshold of $2.2\sigma$ 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 $LR = 1$ 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 NEWS 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.
Finally, Listing 5.2 illustrates group corroboration using pseudocode. For each time that a local event is detected, NEWS finds the set of remote events detected at the same time in the same network (line 2). It then determines the likelihood ratio for n hosts (line 25) and confirms the event as likely due to the network if the ratio is sufficiently large (line 29). Based on the analysis above, NEWS uses a threshold LR value of at least 2; as discussed in the next section, tuning this threshold value changes the sensitivity for event detection and thus allows users to control the event detection rate.

Listing 5.2. Simplified code sample for NEWS corroboration.

```plaintext
remoteEvents = getRemoveEvents(currentTime);

/* proceed only if enough peers are corroborating an event */
if (remoteEvents.size()<MIN_CORROBATION) return;

/* get independent probabilities for each host seen online
   in the recent past */
independentProbabilities = getAllIndependentProbabilities();

/* determine relative likelihood of n peers
   seeing event at same time */
for (n = 3; n<MAX_CORROB; n++) {
    /* find union probability of n peers seeing event */
    unionProb = findUnionProbability(independentProbabilities, n);
```
/* empirical probability is the number of times n hosts saw an
// event at the same time divided by total number of observations */
empiricalProb = getNumberCorroboratedEvents(n)/observations;

/* compute ratio only if empirical probability is
greater than zero */
if (empiricalProb>0){
    likelihoodRatio = empiricalProb/unionProb;

    /* if sufficiently large, announce confirmed event */
    if (likelihoodRatio > ratioThreshold){
        announceConfirmedEvent();
    }
}

5.3. Deployment Considerations

This section discusses aspects of an event detection system that are specific to a deployment in an open, uncontrolled environment and how NEWS addresses them.

5.3.1. Privacy and Trust

Any implementation of a network monitoring service is subject to important considerations such as privacy and trust. For instance, NEWS relies on corroboration among multiple vantage points
to confirm and isolate events. This requires users to share information about their network views. To ensure user privacy, NEWS does not publish any information that can be used to personally identify the user (e.g., IPs or download activity). Rather, it reports only detected events and assigns per-session, randomly generated IDs to distinguish events from different users.

While this approach to ensuring privacy is appealing for its simplicity, it opens the system to attack by malicious parties. For example, one ISP may wish to “poison” the system by introducing false event reports for a competitor’s ISP. There are several ways to harden an implementation against such attacks. First, NEWS includes each host’s $L_h$ in the event reports, and recall that larger $L_h$ leads to a smaller contribution to the likelihood (as defined in Equation (3.3)). 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 [13]. Though such an approach eliminates anonymity by assigning identities to users, the privacy of the details of their network activity is maintained.

5.3.2. Participation Incentives

In general, CEM does not require incentives for adoption, e.g., if applications are deployed with instrumentation by default. For the NEWS prototype system in BitTorrent, however, the deployment model relies on users installing third-party software.
Based on the success of Ono [17], NEWS uses a similar mutual benefit incentive model. The incentive for users to install Ono is based on users’ selfish behavior – the software offers potentially better download performance while at the same time reducing cross-ISP traffic. To encourage NEWS adoption, the software offers users the ability to ensure they receive the network performance they pay for. Similar incentives have been successfully used by the Grenouille project[2] (currently more than 20,000 users in France) and various network neutrality projects (e.g., Glasnost [26], which has been run more than 100,000 times).

Specifically, NEWS users contribute their network view (at essentially no cost) in exchange for early warnings about network problems that impact performance. As these problems may indicate changes in ISP policies, violations of SLAs or ISP interference, such warnings provide a mechanism for users to ensure that the Internet service they pay for is properly provided. This has been sufficient incentive for NEWS, which has already installed over 44,000 times as of May, 2010.

5.4. 2008 Mediterranean Cable Disruption

This section uses the 2008 Mediterranean cable disruption [57] as another example to further highlight the advantages of CEM. After describing the event, this section demonstrates the importance of detecting such problems using live traffic flows from the perspective of edge systems inside the affected networks.

Beginning on January 30th, 2008 and continuing for several days, submarine Internet cables providing connectivity to the Middle East and Asian subcontinent were either cut or otherwise failed. The result was that Internet connectivity in the affected regions was significantly reduced.

http://www.grenouille.com
or entirely lost. It is particularly appealing to study this event because unlike minor intra-ISP disruptions, it was well-publicized, previously studied [23,57] and is easy to confirm.

The PingER project [23] was able to identify the Mediterranean submarine cable disruption using lightweight active monitoring in the form of periodic pings among hosts worldwide. Through their observed ICMP latencies and loss rates they estimated throughput in various countries affected by the outage. In particular, they identified India as suffering from an order-of-magnitude drop in throughput as a result.

Figure 5.6 shows a timeline of upload and download performance averaged over all peers in the Ono dataset that are located in India. The left and right y-axes represent transfer rates and the x-axis represents time in UTC. As expected, performance significantly declined at 10:55 UTC on January 30th, which is when the first Indian disruption occurred, but initially the impact on end users was approximately a 50% drop in performance. Later in the timeline, there is another event that does result in an order-of-magnitude reduction in throughput from the pre-disruption levels, but only for a short while.
**Detecting the disruption.** The following paragraphs demonstrate how NEWS is able not only to characterize the impact of network anomalies but also to isolate them in space and time. Figure 5.7(a) presents a timeline of aggregate upload and download performance for users in Israel. Though the graph depicts aggregate performance for visualization purposes, the events were detected using *individual peer perspectives*. Each gray vertical bar represents an event detected by CEM, with darker shades of gray representing smaller window sizes for the moving average. Upon visual inspection, most of the flagged events indeed appear to be valid, as they correspond to significant drops in both the averages of upload and download rates.

Figure 5.7(b) plots a timeline of transfer rates for the same region, but uses moving-average event detection on the *aggregate performance* averaged over all peers in the country. In this case, a large number of known events are not identified, while other events that do not correspond to known outages (and are likely to be false positives) are found. Intuitively, using the aggregate performance data for such a broad region should yield poor results due to the variety of bandwidth allocations and user data-transfer behavior.

One way to validate the detected events is to compare their timestamps with those of prefix outages detected by Popescu et al. [57]. These outages can be seen as ground truth because the analysis was performed by operators of networks in the affected region. The first NEWS-detected event appears at approximately 04:05 UTC on January 30th – in strong agreement with the first known reports of outages from the cut. Interestingly, it is difficult to classify this as an event based on the aggregate transfer rates – in fact, the rates seem to have hardly changed at all. Once again, this underscores the unique benefits of using individual peer perspectives for event detection.
Figure 5.7. Timelines showing download and upload rates in Israel during the week of the submarine cable disruption. The top figure, which uses individual peer views and corroboration, successfully diagnoses events corresponding to known Internet outages, while using aggregated views (bottom) misses true events and generates false positives.

All the remaining events are confirmed by spikes in prefix outages as shown by Popescu et al. [57], with the only exception being the two events just before February 1, 14:00 UTC. As a
side note, the reason for these false positives is that our data-collection server was affected by an operating-system bug\(^3\) that limited – but did not entirely prevent – recording of measurements.

Finally, Figure 5.8 demonstrates how CEM detects events in Egypt, another country affected by the cable disruption. It is immediately clear that the event impacted Egypt much worse than Israel, despite the relative proximity of the two countries. The initial disruptions are successfully detected, with the exception of the one that caused complete lack of connectivity to our data-collection servers on January 30th. This appears on the graph as zero values for upload and download transfer rates; because there was no data from individual peers, the offline analysis did not flag an event.

This raises the question of how the NEWS approach performs during disconnections. It is important to note that NEWS peers \textit{inside} the affected network should be able to classify the sudden loss of connectivity as an event, and both users and operators \textit{inside} the partitioned network need to be informed of the problem. This highlights the importance of a distributed storage that is resilient to network partitions (e.g., a DHT). Though peers \textit{outside} the affected network would not be able to access this event information during the disconnection, those that need it most (i.e., users and operators \textit{inside} the affected region) can still exchange information about the event.

\[^3\text{This was caused by a race condition in the network stack for an early version of Solaris 10, causing the server to drop connections under heavy loads.}\]
Figure 5.8. Timeline showing download and upload rates in AS8452 (Egypt) during the cable disruption. NEWS successfully diagnoses events corresponding to known Internet outages (shaded regions).

5.5. Summary

This chapter illustrated the implementation of NEWS for detecting network events using BitTorrent, using confirmed network events in England and the Mediterranean. The next chapter evaluates its performance and effectiveness based on results from hundreds of networks worldwide.
CHAPTER 6

Evaluation

This chapter uses one month of data gathered from BitTorrent users to answer key questions about large-scale edge-system network event detection. The first part of the evaluation demonstrates the effectiveness of the approach using confirmed events from two large ISPs. The results show that using a popular P2P service as a host application can offer sufficient coverage for edge-system event detection. Next, this chapter includes a summary of results from running the NEWS detection algorithm on networks worldwide. The chapter concludes with an evaluation of the robustness of NEWS to parameter settings and an analysis of the overhead for participating hosts.

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

6.1. Effectiveness

To evaluate the accuracy of NEWS, the following analysis compares NEWS event detection results against labeled network problems from two ISPs, the ground truth in this section. For the purpose of comparing these datasets, if an event was detected within 2 hours of a reported
time, it counts as being the same event. Figure 6.1 shows the set of events detected by NEWS and its relationship with those reported in the BT Yahoo logs.

For BT Yahoo, of the 181 events detected by NEWS, 54 are confirmed network problems – covering nearly 80% of the labeled events. This edge-based approach detected an additional 127 events; although these are not confirmed problems, one must use caution when 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, the available list of events corresponded only to network outages, not other types of service disruptions where connectivity remained. In the same network, NEWS 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, there was no ground truth information. Figure 6.2 shows a sample of three cases of events detected by NEWS: (a) an confirmed outage, (b) a non-outage performance event (unconfirmed) and (c) an unconfirmed outage.

Figure 6.1. Diagram indicating the portion of reported (ground truth) events detected by NEWS for the BT network.
Figure 6.2. Timelines depicting events (centered in the figures and shaded) affecting transfer rates for peers in a North American ISP.
In addition to the location and duration of outages, the ISP’s event list specifies the number of users affected by the outage. If the Ono dataset does not contain any users in an affected region of the network, naturally NEWS could not have detected it, so this analysis does not include these events. Given this set of events, the following analysis classifies them according to the number of affected subscribers and compares the NEWS-detected events with those from the ISP.

Table 6.1 presents summary results for this ISP. NEWS was able to detect half of the largest outages (column 3). Column 4 shows the number of outages that appeared to affect monitored hosts, but not in sufficient numbers to validate the event. This occurs when there are fewer than 3 hosts active at the same time in the network during an event – in this case, the sample size is not large enough to draw strong conclusions through corroboration. In addition to these events, NEWS detected 41 events during the 1-month period. Unfortunately, the ISP did not have sufficient information to confirm or deny them. As part of future work, we will explore opportunities to input these events into a root-cause analysis tool to better diagnose the reason they were detected.

### 6.2. Coverage

Edge-system event detection requires a sufficient number of peers to concurrently use a network for the purpose of corroboration. This section evaluates whether this is the case when
using a popular P2P application as a host application for event detection. One way to quantify this is to calculate the maximum number of peers simultaneously online for each network in the one-month dataset – this sets an upper bound for the number of networks NEWS can currently reach.

Figure 6.3 plots a CDF of these values for each routable (BGP) prefix and ASN. On average, the number of simultaneous online peers per routable prefix is 3 and per ASN is 7. Even though the Ono installed base of users represents less than 0.4% of all BitTorrent clients, the graph shows that this offers sufficient coverage (three or more peers concurrently online) for more than half of the ASNs in this study.

6.3. Worldwide events

The previous sections showed the effectiveness of NEWS when compared to confirmed events, and that NEWS offers broad coverage. This section now describes the network events
that it detects worldwide, using a threshold $LR = 2$ (as guided by Figure 5.5). For a one-month period, NEWS detected events in 38 countries across five continents, highlighting how edge-based detection can achieve broad network coverage worldwide.

Table 6.2 lists the top 10 ISPs in terms of the number of users participating in this study (second column), and the number of events detected in each of these ISPs (third column). This table uses the ISP names associated with each prefix after combining those prefixes that are clearly from the same ISP. As one example, the analysis groups together listings such as “CCCH-AS1 - Comcast Cable...” and “CCCH-AS2 - Comcast Cable...” as a single network named “Comcast”.

If NEWS is detecting real network events, the number of events witnessed in an ISP should be related to the ISP’s quality of service [46], not necessarily the number of peers monitoring the network. As such, 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 NEWS detects.

Note that the networks listed in the tables are located primarily in Europe. This is a natural consequence of the user population for the Ono dataset (and BitTorrent generally) – the majority of BitTorrent users are located in Europe. Specifically, the distribution of peers per continent is approximately 57% in Europe, 17% in North America, and 16% in Asia, with the remainder distributed in small percentages.

Table 6.3 shows the top 10 ISPs in terms of the number of events detected, covering ISPs of varying size in Europe and Asia. Importantly, with the exception of the top three ISPs, NEWS generates fewer than four detected events per day. Thus, a NEWS deployment should report events at a reasonable rate – one that will not overwhelm (or annoy) network operators and users.
<table>
<thead>
<tr>
<th>ISP</th>
<th>Users</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deutsche Telekom</td>
<td>6760</td>
<td>69</td>
</tr>
<tr>
<td>HTP</td>
<td>3652</td>
<td>112</td>
</tr>
<tr>
<td>HanseNet</td>
<td>3216</td>
<td>17</td>
</tr>
<tr>
<td>Neuf Cegetel</td>
<td>2821</td>
<td>108</td>
</tr>
<tr>
<td>Arcor</td>
<td>2245</td>
<td>29</td>
</tr>
<tr>
<td>Cableuropa Ono</td>
<td>1999</td>
<td>245</td>
</tr>
<tr>
<td>Proxad/Free ISP</td>
<td>1769</td>
<td>176</td>
</tr>
<tr>
<td>France Telecom</td>
<td>1688</td>
<td>31</td>
</tr>
<tr>
<td>Telecom Italia</td>
<td>1651</td>
<td>20</td>
</tr>
<tr>
<td>Telefonica</td>
<td>1337</td>
<td>27</td>
</tr>
</tbody>
</table>

Table 6.2. Top 10 ISPs by users.

<table>
<thead>
<tr>
<th>ISP</th>
<th>Users</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cableuropa Ono</td>
<td>1999</td>
<td>245</td>
</tr>
<tr>
<td>BTnet UK</td>
<td>1277</td>
<td>182</td>
</tr>
<tr>
<td>Proxad/Free ISP</td>
<td>1769</td>
<td>176</td>
</tr>
<tr>
<td>HTP</td>
<td>3652</td>
<td>112</td>
</tr>
<tr>
<td>Neuf Cegetel</td>
<td>2821</td>
<td>108</td>
</tr>
<tr>
<td>Deutsche Telekom</td>
<td>6760</td>
<td>69</td>
</tr>
<tr>
<td>Telewest Broadband</td>
<td>237</td>
<td>50</td>
</tr>
<tr>
<td>Pakistan Telecom.</td>
<td>729</td>
<td>46</td>
</tr>
<tr>
<td>Comunitel Global</td>
<td>197</td>
<td>45</td>
</tr>
<tr>
<td>Mahanagar Telephone</td>
<td>454</td>
<td>42</td>
</tr>
</tbody>
</table>

Table 6.3. Top 10 ISPs by events.

6.4. Cross-network events

An advantage to the CEM approach is that it is uniquely positioned to detect network problems affecting multiple ISPs, e.g., due to provider or peering link issues. As discussed in Section 3.3.2, one can use AS relationships and geolocation information to isolate the scope of cross-network events.
<table>
<thead>
<tr>
<th>Relationship</th>
<th>Min. ASNs</th>
<th># cases</th>
<th># countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer-Provider</td>
<td>2</td>
<td>370</td>
<td>5</td>
</tr>
<tr>
<td>Customer-Provider</td>
<td>3</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Peer-Peer</td>
<td>2</td>
<td>487</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 6.4. 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.

<table>
<thead>
<tr>
<th>Time (GMT)</th>
<th>Provider(s)</th>
<th>Affected ASes</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr 16, 13:35</td>
<td>8218</td>
<td>15557,12876,12322</td>
<td>FR</td>
</tr>
<tr>
<td>Apr 17, 12:40</td>
<td>1267</td>
<td>16338,3352,6739</td>
<td>ES</td>
</tr>
<tr>
<td>Apr 30, 01:15</td>
<td>10396,7910</td>
<td>12357,16338,12715</td>
<td>ES</td>
</tr>
</tbody>
</table>

Table 6.5. Example cross-network events corresponding to the second row of Table 6.4.

This section demonstrates this feature of CEM by focusing on cross-network events due to issues with upstream providers or peers. The first step of the analysis is to identify 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. [14]). Events that occur within 30 minutes of each other are considered the same, and the results conservatively include AS relationships only for those ASNs located in the same country.

Table 6.4 summarizes the results. The first row indicates that when searching for at least two ASNs with the same provider, there are 370 cases in five countries. The second row presents results from a more restrictive search that requires events detected at the same time in at least three ASNs having the same provider – such events are much rarer; a sample of them is provided in Table 6.5. 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 CEM – these links are often invisible to existing Internet monitoring techniques that do not rely on edge-system monitors [14,16].
6.5. Robustness

As discussed in Section 5.2.2, 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.

Figure 6.4 plots CDFs of \(LR\) values for BT Yahoo during one month. Figure 6.4(a) plots \(LR\) values for \(W = 10\) and \(\sigma = 1.5\) and Figure 6.4(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 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.

6.6. 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). This section now demonstrates that these overheads are reasonably low.

Computational overhead. For determining the union probability, the formula in Equation 5.3 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\(^1\). Fortunately, the terms in the formula are highly repetitive and thus offer much room for optimization.

\(^1\)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.
Figure 6.4. 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 (top) and at most $0.75\%$ of the time for larger ones (bottom).

NEWS uses Miller’s algorithm [50], 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 $n$ (based on experience, $n > 50$). To bound this overhead, NEWS limits the number of hosts used in the computation to the $H$ hosts with the largest $L_h$. In this way, NEWS conservatively estimates an upper bound for $P_u$ for the full set of $n$ hosts.

**Network overhead.** 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, it is reasonably low for many settings.

As an example, the following paragraphs analyzes two prefixes in the BT Yahoo network that from the preceding case study. This represents the higher end of a medium-sized network, and as such is neither a worst-case nor a best-case scenario. Regardless, this network allows us to illustrate the dynamics of NEWS overheads for a network with confirmed problems.

Recall that NEWS can employ multiple local event detection settings in parallel, and use the likelihood ratio to determine which settings are detecting meaningful events. Given this scenario, the results are based on simulating the overhead that NEWS users would experience in the month of April 2009, using three standard deviation thresholds (2.0, 2.5 and 3.0) and two window sizes (15 minutes and 30 minutes). These correspond to a range of detection settings that include those that are both sensitive and relatively insensitive to changes in throughput. As the following paragraphs show, even using these six combinations of detection settings does not incur unreasonable overhead.

The rest of this section focuses on the 86.128.0.0/16 and 81.128.0.0/16 networks in the BT Yahoo ISP. Figure 6.5 depicts the number of local events detected by users in these networks over time using 100-second sample intervals. As the figure shows, most of the time there are zero or one event detected, and the maximum number of concurrent events detected is 6. This
demonstrates that while NEWS generates a significant number of events, the rate at which it does so is low.

Another way to evaluate overhead is in terms of the number of event-report read and write operations that occur over time. To this end, Figure 6.6 plots the number of these reads and writes over time for NEWS hosts in these networks. This analysis assumes that if a host writes a value to a key, it persists at that key for one hour. Thus, when a host subsequently reads a key, it will receive all values written during the previous hour.

Again the figures show that for this network, there is a significant, but not altogether unreasonable amount of read/write traffic that the NEWS system generates. For the smaller network, there is no more than 150 reads or writes per interval, and for the larger one there is rarely more than 400. An interesting feature of this plot is that there are diurnal patterns in the numbers of read and write operations. Because this data is drawn from a single time zone, and because users tend to use BitTorrent during evening and weekend hours (thus providing more monitoring points during those times), this pattern is not surprising.

Finally, Figure 6.7 shows how these read and write operations grow over time. On the x-axis is time and the y-axis is the cumulative number of read and write operations, on a log scale. In particular, each point (x,y) means that from the beginning of this timeline until point x, there were y operations.

One clear feature of the graph is that the reads grow much faster than writes, but within a constant factor. This occurs because each time a host writes a value to a key it also reads values from that key for the purpose of corroboration. Since the distributed store caches entries for a period of time, hosts tends to read several more values than they write. Also note that the ripples
Figure 6.5. Number of detected events over time.
in the curve correspond to large numbers of local events being detected at the same time, likely corresponding to network-wide events.

In the larger network, there were about 140,000 reads and fewer than 10,000 writes. To put this in context, this corresponds to less than one read every 10 seconds and about one write every two minutes on average throughout the period – spread across several hundred users. For
each user, this corresponds to about 4 B/s network overhead – a vanishingly small portion of capacity even for low-bandwidth hosts.

Again, this indicates that NEWS generates reasonably low overhead over time and per user. Regardless, note that the cost of read operations (which are larger in number) is in practice amortized by the Kademlia DHT that Vuze uses. In particular, the amount of network overhead to read a key decreases with the popularity of the key because Kademlia caches and replicates content that is frequently read.

This section concludes by comparing the network overhead of NEWS with two strawman alternatives. The first strawman is a centralized solution where each host sends its performance information to a centralized server for event detection. In the current implementation, this would require sampling 13 signals every 15 seconds. At 4 bytes per signal sample, the result is 52 bytes every 15 seconds, or 4 KB/s for 1,000 hosts. More generally, this overhead scales linearly with the number of supported hosts. While this amount of network overhead may seem manageable, it ignores many practical issues. For example, it is unclear who should host such a data-collector effort, nor does it account for the CPU and memory requirements for performing event detection on this data. Last and perhaps most importantly, it ignores the privacy issues for event detection.

The second strawman alternative alleviates the issue of centralized data collections by sharing performance signals among all participating hosts. In this case, sharing incurs $O(N^2)$ cost, so 52 bytes of data generated by each host’s signals every 15 seconds translates to 34.6 MB/s of overhead for 1,000 hosts. As these cases demonstrate, alternative approaches that eschew local detection at each host incur orders of magnitude greater overhead and raise significant privacy concerns not present in the CEM approach.
Figure 6.7. Cumulative number of DHT reads and writes over time using a semilog scale.

6.7. Summary

This chapter demonstrated the effectiveness of NEWS using both confirmed events and a characterization of system performance using a range of detection settings. The next chapter discusses key details of the deployed NEWS implementation for the Vuze BitTorrent client.
CHAPTER 7

Implementation

In the previous chapters, we used extensive BitTorrent traces to motivate the design of a system for detecting network events from the edge of the network and to evaluate the effectiveness of our approach. In this chapter, we discuss how we implemented these ideas to support NEWS in popular BitTorrent client. We also describe key aspects of our third-party interface for accessing network events detected by these BitTorrent users.

7.1. BitTorrent Extension

This chapter discusses key aspects of the NEWS implementation that has been deployed for the Vuze BitTorrent client. The NEWS plugin for Vuze is written in Java and contains more than 12,500 method lines of code, though the core classes for event detection comprise $\approx 1,000 \text{ LOC}$. Released under an open-source (GPL) license, this plugin has been installed more than 44,000 times between March, 2008 and May, 2010. The rest of this section discusses details of the NEWS implementation in its current deployment. In addition to providing specific algorithms and settings used for event detection, the discussion includes several lessons learned through deployment experience.

\footnote{A large portion of the total LOC is dedicated to Vuze-specific mechanisms (e.g., the GUI) and statistics reporting.}
7.1.1. Local detection

NEWS detects local events using the moving average technique discussed in Section 5.2.1, which uses the window size ($w$) and standard-deviation multiplier ($t$) parameters to identify edges in BitTorrent transfer rate signals. NEWS currently uses $w = 10, 20$ samples and $t = 2.0, 2.5, 3.0$, dynamically configurable settings that were most effective based on the previous offline analysis.

In practice, we found that BitTorrent often saturates a user’s access link, leading to stable transfer rates and small $\sigma$. As a result, a moving-average technique may detect events in the throughput signals even when there are negligible relative performance changes. NEWS addresses 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, a value that works well in deployment.

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 (specifically, $w/2$ events), the signal has undergone a phase change, and NEWS resets the moving average using only signal values after the phase change.

NEWS uses a randomly generated per-session ID to distinguish events from different hosts, thus avoiding the need to reveal any PII in its reports.

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.
It is important to note that NEWS hosts may not detect any local events for long periods of time, even though they have done so in the past (and thus their $L_h > 0$). Because the DHT does not provide reliable storage, the most recent report for host $h$, containing its $L_h$ may no longer be cached in the DHT. Further, this value may be stale. To ensure reasonably accurate information is available in the DHT for each other host to perform the likelihood ratio analysis (which requires $L_h$), each NEWS host periodically refreshes this information in the DHT using a “no-op” report. In particular, the host uses its per-session ID, invalid $w$, $t$ and bitmap values, and includes its current value for $L_h$. NEWS currently performs this no-op operation once per hour.

The plugin disseminates these reports using the Kademlia-based DHT [49] 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, NEWS use network locations as keys and the corresponding event reports as values.

In the NEWS deployment there are variable delays between event detection and reporting, in addition to significant skew in users’ local clocks. 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.

---

**Listing 7.1. Pseudocode for NEWS corroboration procedure.**

```
1 // myId: per-session local peer identifier
2 // myRegion: local peer’s region (e.g., BGP prefix)
3 // t: local event detection threshold as discussed above
4 // lrThreshold: likelihood ratio threshold
```
// RECENCY_THRESHOLD: currently set to one hour

// read all event data
EventReport reports[] = ReadEventRecord();

for ( EventReport report : reports ) {
  // check for valid record
  if ( report.Id != myId &&
       report.timestamp < RECENCY_THRESHOLD &&
       report.stdDevMult >= t ) {
    CacheReport( report ); // cache record
  }
}

// check for actionable item
if ( LikelihoodRatio( myRegion ) >= lrThreshold ) {
  AlertActionableItem(); // notify user/operator
}

7.1.2. 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. The corroboration procedure for low-level regions is illustrated in pseudocode in Listing 7.1 and illustrated in Figure 7.1. Before using a report from the DHT for corroboration, NEWS ensures that: (1) Vuze already collects the host’s prefix and ASN; we are currently adding support for whois information.
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 Equation 3.3 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 Section 5.2.2. If the likelihood ratio is greater than 2 (also configurable), the monitor issues a notification about the event. In this way, NEWS keeps end-users informed about detected service-level events, following the incentive model in this work (Section 5.3.2).

Figure 7.2 shows the NEWS plugin view that allows users to configure detection settings and view a history of detected events. In particular, the top half of the UI allows users to override default detection settings to be more or less sensitive to network events. Though the setting options are describing (“More sensitive”, “Very sensitive”), changing the value in fact changes
the likelihood ratio threshold. When an event is detected, a nonintrusive message appears in the bottom right corner of the screen with details about the event (Figure 7.3). In the case that such warnings in fact become annoying, the UI also allows users to disable notifications for events without disabling the detection itself. Finally, the bottom half of the UI contains a list of events detected during the current session.
NEWS peers read from the DHT only after detecting a local event, in order to corroborate their finding. Specifically, a peer reads event information from the DHT using the low-level regions it belongs to as keys. The process of corroboration continues until it is determined that a sufficiently large number of other peers have indeed recently seen a similar event or when the peer has exhausted all of its search keys. 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).

### 7.2. Third-party interface

As discussed in Chapter 1, online identification of network events is essential for network providers to maintain good network performance, satisfied subscribers, and their associated revenue. The CEM approach is designed to meet these requirements; however, so far this work focused on the network subscriber. This section presents NEWSight, a public Web service for notifying network operators (and the general public) about network events as they occur.

While this work demonstrates the effectiveness of NEWS for the networks where both extensive traces and confirmed network problems (i.e., labeled data) were available, the research community currently suffers from a paucity of labeled data with which to design, evaluate and compare alternative detection techniques. In particular, sound evaluations of research in this area requires a set of events labeled by a domain expert (e.g., a network operator) to form a “ground truth” \[59\].

There are many challenges to obtaining such labeled data. For one, manually labeled data is labor-intensive and can be time-consuming for many (otherwise busy) experts. In part to address this limitation, Ringberg et al. describe WebClass \[59\], a tool to assist with inspecting
timelines of network performance and manually labeling network events. Users specify the type of event to identify and WebClass displays performance timelines for anomalous time periods while allowing the user to label the event (or declare it a false positive). While the goal of this approach is a valuable contribution, a significant barrier to the utility of this tool is that experts have little incentive to access historical network event data.

To address this issue, one can again turn to a mutual benefit incentive model for encouraging manual labeling of the events that NEWS detects. As the following section describes, NEWSight allows network operators to register for immediate e-mail notifications when NEWS detects events in their networks. The e-mail contains a link to the NEWSight interface where operators can view the ongoing event and label the event as soon as it happens.

There are several advantages to this approach that should encourage labeling. For one, researchers can improve event detection techniques through better labeled data. With better detection, operators receive more useful notifications which in turn should improve their work performance. Second, operators are sent directly to the site only for events that impact their networks, as they occur. It is reasonable to believe that operators are much more likely to label events in small numbers within a short time of their occurring, rather than setting aside time after the fact to comb through large swaths of historical data. Last, the events that NEWS detects are made available to anyone, including potential subscribers. As these events may indicate the relative quality of service from different providers, corresponding operators should be encouraged to correct and/or explain as many events as possible.

The rest of this chapter describes NEWSight – the front-end of a public interface to events detected by NEWS. The next section describes NEWS Collector, a service that gathers and analyzes this event reports for efficient retrieval by NEWSight.
7.2.1. NEWS Collector

The goal of NEWS Collector is to gather event reports generated by participating hosts, determine whether any local events correspond to likely network wide ones and cache this information for efficient display. Depicted as the tap on distributed storage in Figure 3.1, the NEWS Collector connects to the same store used by NEWS clients and gathers the active event reports. Figure 7.4 depicts an architectural diagram of the NEWS Collector.
As discussed in Section 7.1.2, NEWS uses the Kademlia DHT that is built into the Vuze BitTorrent client for storing and retrieving local event reports. Thus NEWS Collector uses a modified version of Vuze that uses the same DHT codebase. The main different is that the tool is designed exclusively to crawl the DHT for events (and not participate in BitTorrent file sharing).

The Kademlia DHT is a key-value store; when reading a specified key the DHT returns zero or more values, each of which corresponds to a value that some user stored at the key. Thus, the DHT can contain more than one DHT value for each key, and multiple users can each store multiple values. Values are removed from the store when they expire (after a fixed duration) or when the hosts storing all replicas of a value go offline. Given the relatively high rates of churn in P2P systems, it is thus critical for NEWS Collector to periodically and frequently crawl the DHT to obtain as many reports as possible.

To retrieve event reports from the DHT, the crawler requires a set of relevant keys to search. Because NEWS currently uses BGP prefixes and ASNs as keys for storing event reports, NEWS Collector can simply exhaustively search all possible values in each category. However, as there are hundreds of thousands of prefixes and tens of thousands of ASNs, performing this brute-force search not only wastes resources but also significantly reduces the frequency with which the crawler visits each key.

NEWS Collector addresses this issue by exploiting the fact that all NEWS users report their IP address to centralized servers upon initialization. Using this set of IPs, the collector generates a list of all prefixes and ASNs that they belong to and search only this subset of keys. Currently the crawler takes tens of minutes to visit each active prefix and only a few minutes to

---

[3] This is done for tracking the number of unique users.
visit each active ASN. By contrast, an exhaustive search of all possible keys would take on the order of hours to complete.

The Vuze DHT interface is asynchronous, meaning that the crawler need not block when reading values. However, once a value is retrieved from the DHT, NEWS Collector must store and process the associated data. There is a number of challenges that must be addressed in this context.

First, the crawler may retrieve the same event report multiple times before it expires. Thus, when storing the event, NEWS Collector must ensure that it maintains only unique reports. This is accomplished using the per-session randomly generated ID that each host uses to distinguish its own records from those of others. In particular, the collector drops a newly read event if it has already read an event report with the same ID, timestamp and detection settings.

After collecting a set of unique event reports, the collector must process their information to determine whether a network event is occurring (or has occurred). Because clients use NTP to synchronize clocks (see Section 7.1.1), the crawler does not need to adjust the timestamps of reported events; however, hosts may detect events at different times (based on their local sampling period and start times). To account for these variations, NEWS Collector places reported events into time buckets, where any report with a timestamp within a time range (greater than the detection period) is assumed to correspond to the same event.

After grouping the events by time, NEWS Collector performs the likelihood ratio calculation as described in Section 5.2.2. The results of this calculation are cached for fast retrieval by the NEWSight interface.

The next section discusses how NEWSight allows network operators (and users) to register for e-mail notifications when corroborated network events are detected. To enable this,
when NEWS Collector detects a corroborated network event, it checks whether any users are registered to receive notifications for the corresponding network. If so, the system e-mails the subscribed users.

The collector currently runs on centralized servers that interact with the NEWSight front-end, but they can also run on multiple hosts using Aqualab infrastructure or run in-house and tailored to specific networks that a user or operator wishes to monitor. The system uses a database to store events and cache results of event likelihood analysis; using a centralized database additionally allows one to run any number of Collector instances (to increase the chances of obtaining complete, fresh event reports) in parallel – using database locking mechanisms and uniqueness constraints to ensure correctness of parallel operation.

7.2.2. NEWSight

To demonstrate the effectiveness of a third-party interface to the events that NEWS detects, we built and deployed 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 and/or explain detected events. The following paragraphs describe the key components of this Web interface for accessing events detected by NEWS.

The front page of the NEWSight interface, which is publicly available is depicted in Figure 7.5. The primary portion of the Web page consists of a Flash animation that contains graphs of network events over time, separated by country, ASN and BGP prefix. The y-axis of these

http://barbera.cs.northwestern.edu/newsight/release/Newsight.html
graphs represents time, with the default period set to one week. The x-axis represents the number of locally detected events at each time step, or optionally the likelihood ratio of a corroborated event at each time step. The interface also provides a slider component that allows the user to dynamically specify the range of time for the graph to depict. As the user moves the slider, content is dynamically fetched from the NEWS Collector database and the scale of the graph is adjusted.

In addition to changing the period to observe in the timeline, NEWSight allows users to specify the regions to inspect. By default, NEWSight displays information for the US, an ASN in the US an a prefix in the ASN. Future work includes adding a feature that displays information for the user’s country and network. When the user changes one of the higher level regions (e.g., an ASN), the list in the lower level region is adjusted to contain only those subregions belonging to the parent region (e.g., only BGP prefixes belonging to the ASN).

On the right-hand side of each graph is a list of corroborated network events and their timestamps. Whereas NEWS crowdsources event detection, NEWSight can be viewed as an attempt at crowdsourcing network event labeling. A registered network operator can select events from the list and click the “Confirm” button to access the interface for manually labeling events. That interface allows the administrator to confirm (or deny) the event, along with a
description of the event. This interface is being beta-tested with ISPs as of May, 2010; the interface and its data are publicly available.
CHAPTER 8

Related Work

Ensuring high availability and a positive user experience is a critical goal for network providers as they compete for subscribers and revenue. Achieving this goal requires a solution for detecting, isolating and repairing issues that affect network performance. As a result, there is a great deal of previous work that focuses on detecting such network events (also called network anomalies).

CEM distinguishes itself from other research in this area in three key ways. First, CEM is specifically designed to detect network problems at the edges of the Internet; as such, it must scale to tens of thousands, if not millions of participating hosts. Second, CEM detects service-level network events (i.e., those impacting application performance) that are critical to user-perceived network performance and that can be invisible to existing in-network monitoring tools. Last, the NEWS implementation of CEM has been tested and deployed at the scale of tens of thousands of end systems worldwide.

The remainder of this chapter discusses several categories of prior research that relates to this dissertation. At a high level, we group this related work according to how and where monitoring is done. One key distinction is whether a system relies primarily on active or passive measurement. The key trade-off is that active measurements provide potentially finer-grained control for detecting problems, but they do not scale to large deployments. Another key distinction is whether a solution detects events inside a single ISP or across multiple ISPs. The former tend to require specialized hardware and access to proprietary information, but provide detailed
network views. The latter can be deployed anywhere and correlate information across multiple networks (e.g., to detect cross-network outages) but can probe only portions of networks made publicly visible.

Based on this characterization, CEM is primarily a passive monitoring tool that runs across multiple networks. Unlike previous work in this category, CEM acquires application-layer views of performance from the edge of the network.

8.1. In-Network Monitoring

In-network monitoring focuses on data available only inside a single administrative domain, as it requires access to routers or other infrastructure inside a provider’s network. While these approaches and CEM may detect the same problem (e.g., in the case of a link failure), they are generally complementary because they target different types of failures.

Active measurements. At a minimum, most ISPs instrument their core network with some type of active measurements to ensure connectivity and determine whether routers and links are overloaded. Although network providers tend not to share the details of their diagnostic network measurement activities and their frequency, one can infer this information based on discussions on network operator mailing lists (e.g., NANOG [52]). These messages indicate that operators tend to use ad-hoc network management system (NMS) solutions [51,53] that include ping latency and reachability measurements in addition to determining application-layer connectivity for a set of specified hosts and exchanging SNMP messages with them [52].

While these approaches are useful for detecting problems with critical network infrastructure, their reliance on active measurements limits their scale. As a result, most ISPs cannot extend their measurements to the edges of the network without sacrificing probe frequency and
thus the speed with which problems are detected. In addition, such tools often rely on operator-specified thresholds for anomalous data that might indicate a network problem \[9\]. While loss of connectivity is a simple condition for alert, identifying “high latency” indicative of a network problem is difficult (if not impossible) with a simple threshold.

Lastly, while these tools provide access to an enormous volume of data, they do not facilitate extraction of meaningful information. Thus, the effectiveness of these tools is limited to cases where network events are obvious (e.g., a cable disruption); while operators may ignore other subtle, yet important, performance signals that simply generate too much noise.

**Passive measurements.** In lieu of relying strictly on active measurements, a large body of event detection research focuses on combining multiple sources of network status information. These include both active data as described above and passively gathered data such as Netflow \[20\] records and BGP tables from routers.

Researchers have used a variety of signal processing techniques to detect abnormal behavior in aggregate traffic statistics, typically using core or border routers. For example, Barford et al. \[9\] use deviation-based signal analysis on aggregate flows to detect anomalies in traffic on a campus border router. They show that techniques such as wavelet analysis and moving-average forecasting using Holt-Winters \[19\] are effective at distilling significant events from a large volume of timeseries data. The authors also note, however, that determining threshold deviations for these events in general is a challenging problem. Krishnamurthy et al. \[38\] also propose performing signal analysis to detect anomalies, but focus on using a sketch data structure to more efficiently manage the number of signals available to a detection system.

Lakhina et al. \[41\],\[42\] and Li et al. \[43\] propose using principal component analysis (PCA) to detect network problems based on historical timeseries of byte counts, packet counts, and
IP-flow counts gathered from routers reporting Netflow data. In particular, their approach is to create a matrix of the time series for multiple routers, then use PCA to determine the set of principal components for this matrix. Based on the hypothesis that a large amount of the variance is captured by a small number of components, and that these components correspond to normal traffic, they focus on the remaining components as a filter for detecting abnormal traffic. A traffic anomaly is defined as time periods with relatively large values in signals using those remaining components. The authors show how this analysis reveals a large number of confirmed traffic anomalies. More recently, however, Ringberg et al. [60] demonstrate that tuning this technique to run effectively over time and in multiple networks is a challenging problem that limits its effectiveness.

Other work in network event detection focuses on routing messages as signals indicating problems. In particular, they study Border Gateway Protocol (BGP) messages and tables to determine if there is an outage, routing loop or other problem that impact network performance. Zhang et al. [75] use signatures and statistical learning models to detect unexpected changes in the behavior of BGP update messages. Feldmann et al. [28] simulate link failures and use the result to develop a technique for isolating these failures based on the BGP messages generated in response. Wu et al. [73] uses signatures of known BGP events that indicate performance failures, correlates them in time and prefix space and estimates their impact on traffic. Last, Roughan et al. [62] propose combining traffic-flow data and BGP data to more accurately detect network problems.
8.2. Cross-Network Monitoring

Cross-network monitoring includes approaches that attempt to detect problems across multiple administrative domains, which requires a platform for measuring multiple networks. Toward this goal, a number of recent efforts are exploring new monitoring techniques using distributed research platforms (e.g., PlanetLab [56] or NIMI2 [4]) as vantage points, to address parts of Clark et al.’s [21] broad vision of a knowledge plane for the Internet [44, 72, 76]. 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 [16]. 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 [12, 14, 66]. CEM uses a fundamentally different approach that pushes detection to the edge-systems where services are used. When deployed as part of a P2P system, we have shown that this approach reaches hidden corners of the Internet [14].

Active measurements. Several researchers have proposed using end-host probing to monitor various network properties. For example, Paxson [55] uses traceroute measurements from 37 vantage points to study end-to-end routing behavior in the Internet. Similarly, the PingER project [23] uses periodic pings among all sites in a distribute research platform to monitor link liveness and variations in latency. As with in-network monitoring, the overhead of active-measurement approaches limit their scale and thus their visibility into the network.

A number of distributed measurement systems attempt to identify routing disruptions and their effect on end-to-end services. For example, Zhang et al. [77] use distributed traceroute measurements to detect routing events and employ a scheme that reduces the number of probes
required to cover the paths visible to their platform. Feamster et al. [27] uses all-pair ping measurements, packet-loss-induced traceroute probes and BGP information to understand how and why Internet path faults occur and how they related to BGP instability. Madhyastha et al. [44] propose iPlane, a system for measuring and predicting Internet performance, using traceroutes, pings and bandwidth capacity measurements from the PlanetLab testbed. Katz-Bassett et al. [36] use periodic, distributed measurements to detect regions of the Internet that unexpectedly become unreachable (i.e., “black-holed”). All of these efforts use GREN platforms for monitoring. While such deployments facilitate controlled experimentation across distributed hosts, we showed in Section 4.3 that these vantage points miss significant portions of the network compared to those traversed by P2P traffic. Further, such approaches that use active monitoring (e.g., [8], [36]) are limited by the overhead for detection, which grows with the number of monitored networks and services. While CEM could be combined with limited active probes to assist in characterizing and localizing network events, it does not require them.

In light of the limitations with testbed environments, several research groups in our community are exploring opportunities to incorporate edge-based network measurements. For instance, DIMES [64] and Neti@home [65] perform active measurements (typically traceroutes) from software installed at end hosts. Unlike this work, these projects do not focus on using their measurements to perform online detection, and their incentive for adoption is altruism. While CEM shares many goals with these projects, it uses immediate incentives to ensure significantly wider adoption than what is possible with a purely altruistic model. In a similar vein, Rabinovich et al. propose addressing the software adoption issue through matchmaking and direct economic incentives in DipZoom [58], a project that attempts to create a marketplace for network measurements.
In the domain of detecting network interference, a number of efforts have turned to active measurement to identify service differentiation, including ShaperProbe [3] and Glasnost [26]. While these software share our goal of increasing transparency for network service, this dissertation focuses on detecting network events in general rather than cases of interference from ISPs.

The C’MON [1] project (in collaboration with the Grenouille project) attempt to capture the view of performance in home networks through user-installed clients that perform periodic bandwidth measurements to evaluate the level of service provided by French ISPs. While this project shares our goal of increasing transparency in network performance, the two approaches are fundamentally different. Specifically, CEM is a passive measurement system that does not rely on any centralized services, while Grenouille requires active FTP transfers to a small set of servers to evaluate performance. These issues have limited the coverage of their current approach to a single country.

The “How’s My Network” (HMN) project [61] uses Java and Javascript code executed through user interaction with a Web site to perform network measurements. Similar to CEM, this project reuses an existing, widespread service in unintended ways to perform network measurement. In contrast, CEM piggybacks on long-running services, allowing it to gather longer-term views of network performance necessary for event detection.

Some commercial network monitoring tools generate flows that simulate protocols used by edge systems (e.g., Keynote [37]). 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. The CEM approach does not require any new
infrastructure, nor control of end systems, and thus can be installed on systems at the edge of the network.

**Passive measurements.** 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 \[22\,\[76\]. Seshan et al. \[63\] and Padmanabhan et al. \[54\], among others, have argued that many of the disadvantages of active measurements, including the introduction of unnecessary network traffic, can be avoided by the sharing of information collected through passive monitoring. PlanetSeer \[76\] 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. \[12\] and Isdal et al. \[33\] 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.

Typically, passive monitoring infrastructures are deployed below the application layer, e.g., on routers \[20\] or in end-user operating systems \[64\]. While these systems can detect many properties of the network, they tend to do so only in aggregate views that lose valuable semantic information about their corresponding flows \[22\]. Application-layer monitoring, on the other hand, does not require supervisor privileges and it improves user privacy by restricting monitoring to only those applications that are instrumented, thus lowering the barrier to adoption. CEM performs event detection at the application layer, using knowledge of expected behavior to more confidently identify network problems.
8.3. Service monitoring.

While CEM uses application-layer views of network performance to detect events, a large body of research focuses on using these views to dynamically adapt to changes in performance impacting high-end distributed systems, grids and virtual machine networks. In this domain, for example, one can use passively gathered application-layer and OS-layer information to determine that communication patterns among nodes in a distributed system could be optimized by moving tasks to different locations in the system [30]. Performing this online adaptation requires a scalable monitoring systems that informs resource scheduling and placement decisions; Zanikolas et al. provide a thorough survey of monitoring approaches [74].

Perhaps the most closely related work to CEM in this domain is the Network Weather Service [71], which aims to measure and predict dynamically changing performance characteristics for resources available to distributed systems. In addition to measuring and predicting CPU load and network throughput, the authors discuss how to make this information available to the system efficiently using a hierarchical abstraction called cliques. Similar to NWS, CEM assumes that historical performance can predict future performance; however, this dissertation focuses on deviations from such predictions as potential network events.

8.4. Crowdsourcing.

Central to the CEM approach is the idea of crowdsourcing event detection to ensure good coverage and accuracy at the scale of hundreds of thousands of users. This model has successfully enabled projects that include solving intractable [69] or otherwise prohibitively expensive problems [7] using human computation. Unlike these examples of crowdsourcing, NEWS passively monitors network activity from each member of a crowd, but it does not require human
input. Dash et al. [25] use a similar model to improve the quality of intrusion detection systems in an enterprise network and demonstrate its effectiveness through simulation using traffic data from 37 hosts from within their enterprise network. In contrast, this work shows the effectiveness of crowdsourcing network monitoring at the scale of tens of thousands of users.
CHAPTER 9

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, this work argues that the most appropriate place for monitoring service-level events is at the end systems where the services are used. This dissertation proposed a new approach, Crowdsourcing Event Monitoring (CEM), based on pushing end-to-end performance monitoring and event detection to the end systems themselves. We presented a general framework for CEM systems and demonstrated its effectiveness using a large dataset of diagnostic information gathered from peers in the BitTorrent system, along with confirmed network events from two different ISPs. We demonstrated that this 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 online event detection using CEM – installed more than 44,000 times as of May, 2010.
CHAPTER 10

Generalization and Future Work

CEM is part of a larger research agenda covering topics that include measurement reuse in large-scale systems, identifying and encouraging mutually beneficial incentive models to enable new participatory services, and acquiring a better understanding of how distributed systems behave and perform in the wild. In addition to the Ono and NEWS projects, we have also demonstrated how to reuse CDN redirections to drive high-performance detouring [15]. Further, we have used Ono’s extensive measurement dataset to uncover significant regions of the Internet topology that are invisible to public views [14], and to re-evaluate existing network positioning systems [18].

In this broader context, there is a number of research issues for monitoring events at the scale of Internet users worldwide that are outside the scope of this dissertation and that are possible paths for future work.

Other applications. This work demonstrated the feasibility of detecting network events using a popular P2P application, and showed that there are sufficient incentives for users to adopt the software in large numbers. In this case, the availability of an extensible, open-source and popular software distribution facilitated research in this area. We would like to explore implementing this approach in other applications, such as VoIP and IPTV, where there is a similar scale of users but generally closed-source, non-extensible software.
Labeled data. In this work, we used publicly available and proprietary sources of confirmed network events to validate CEM. To better understand the effectiveness of event detection approaches, one needs even more labeled network data. As part of this effort, we created NEWSight, an interface for allowing operators to view and annotate events detected by NEWS. Future work entails more interaction with operators to ensure the effectiveness of the tool and its interface, and to make the corresponding labeled data publicly available.

Comparison with other detection approaches. This work demonstrated the effectiveness of a simple local event detection technique and a probability-based analysis for detecting network events. As part of future work, we would like to investigate the relative effectiveness of other techniques (e.g., Bayesian belief networks) and their suitability for deployments at an Internet scale.

Root cause analysis. While detecting events is a critical aspect of providing reliable network service, pinpointing the source of the events is essential for resolving these issues as quickly as possible. As such, future work entails exploring opportunities for end hosts to assist in root cause analysis for the events they detect. In this domain, a promising research direction is to investigate how to incorporate limited, triggered active measurement to complement CEM’s extensive passive measurements, for the purpose of isolating the location and characterizing the nature of CEM-detected events.

Service events. This approach focuses on detecting network events as evidenced by their impact on applications running at the edge of the network. As discussed in Section 3.2, CEM’s ability to detect network events derives from grouping hosts located in the same network. If, however, one changes the grouping of users from “same network” to “same service” then one
can detect events that impact service reliability independent of the network. As users increasing rely on Internet services, and disruptions in those service become increasingly difficult for administrators and operators to determine (i.e., whether they are due to network problems or service problems), this should be a valuable service.

**Edge measurement.** As part of this dissertation work, we highlighted the degree to which the information provided by today’s research measurement platform is incomplete when compared to the rich and diverse of information available from systems running at the edge of the network \[16\]. Because measurements from the edge reveal fundamentally different network properties that impact system design, we believe that edge-based measurement and monitoring is essential to the success of future Internet research.

Based on the results of our work, we find that a promising way to address this issue is to build immediately deployable systems with appropriate incentives for adoption and with instrumentation – both to evaluate performance in the wild and also to inform the design and deployment of new research systems. To generalize this model, we envision a positive feedback loop wherein researchers provide services to users in return for measurement data that in turn is used to refine existing systems and guide the design and implementation of new ones.

While extending the reach of the CEM approach, future work includes investigating opportunities for integrating monitoring into the operating system, to facilitate and coordinate retrieval of suspected events across multiple, possibly concurrent applications. We believe that this monitoring platform can serve as the basis to extend our model for incentive-compatible research.
References


