Mining Web Logs to Debug Distant Connectivity Problems

Emre Kıcıman, David A. Maltz, Moises Goldszmidt and John C. Platt
Microsoft Research

ABSTRACT

Content providers base their business on their ability to receive and answer requests from clients distributed across the Internet. Since disruptions in the flow of these requests directly translate into lost revenue, there is tremendous incentive to diagnose why some requests fail and prod the responsible parties into corrective action. However, a content provider has only limited visibility into the state of the Internet outside its domain. Instead, it must mine failure diagnoses from available information sources to infer what is going wrong and who is responsible.

Our ultimate goal is to help Internet content providers resolve reliability problems in the wide-area network that are affecting end-user perceived reliability. We describe two algorithms that represent our first steps towards enabling content providers to extract actionable debugging information from content provider logs, and we present the results of applying the algorithms to a week's worth of logs from a large content provider, during which time it handled over 1 billion requests originating from over 10 thousand ASes.

1. INTRODUCTION

Networks by themselves have little value. Rather, their value comes from a user's ability to contact servers that provide the content they want. In this way, content providers are, in essence, the public face of the Internet. In particular, they are in the uncomfortable position of being labeled by users as "unreliable" when a failure occurs at a network provider between the user and the content provider's servers. Moreover, there is a tremendous financial incentive for content providers to ensure that user requests are able to reach them, since it is the flow of requests that brings a flow of money (ad generated, e-commerce or otherwise) to the content provider—missing requests means lost income.

Ultimately, we endeavor to provide content providers with information they can use to improve the experience of their users. We assume that content providers can make a limited number of "phone calls" to badger unreliable network providers, so they must correctly target their energy at the most appropriate network providers. The likelihood of the network provider fixing a problem increases as the content provider provides more information about the problem, motivating content providers to do as much "remote diagnosis" as possible to avoid the "it's working now, what do you want us to do?" response, only to have the failure recur later.

In this paper, we begin to address this goal by framing and then answering the question: How can a content provider mine information already available from their systems and the Internet infrastructure to improve their end-to-end reliability in the eyes of their clients? We study this question through analysis of web servers, as they are the most prevalent form of service provided over the Internet today. The contributions of this paper include: (1) examining the feasibility of using web log analysis (processing the records of HTTP request successes and failures) to survey and improve the end-to-end reliability of an Internet service. (2) a technique for identifying when user-affecting events started and stopped; and (3) a technique for attributing failed requests to potential causes of failures, including network failures, broken client-side software, or server-side outages.

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

2. THE PROBLEM

Our ultimate goal is to help Internet content providers resolve reliability problems in the wide-area network that are affecting end-user perceived reliability. The goal is not necessarily to find all reliability problems affecting end-users, but to prioritize and detect the most significant and actionable ones. This is especially true considering the wide disparity in size and reliability of the various Autonomous Systems (ASes) through which end-users receive Internet connectivity.

As an example of how the goal of finding significant and actionable problems differs from that of simply finding the largest failure rate, consider the following two cases: Case (I): a small AS with failure rates that make the service effectively unavailable from that AS. Case (II) a large AS, from which emanates many requests, that has a small failure rate. Even though the failure rate is high in one case and low in the other, there is significant business value in fixing both of these cases. Having the customers of an entire AS unable to reach a service negatively affects the reputation of the content provider in a broad way, even if the number of affected customers is small. Whereas for a large AS, even a small failure rate can indicate many unhappy customers.

2.1 Available Information and Actions

Figure 1 shows how requests for web content flow from a user's web browser through the network to the content provider's servers and back. At the transport and application layer, requests originate on a client machine as the client uses DNS to resolve the name of the desired web site. The DNS response may specify a server owned by the content provider, or that of an infrastructure provider operating between the client and the content provider (*e.g.*, Aka-

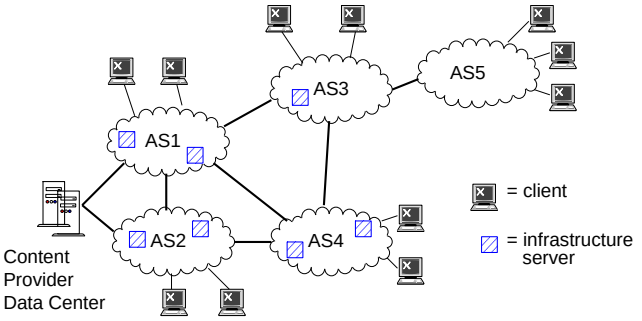


Figure 1: A content provider's view of the network, with clients connected via distant ASes, potentially with network infrastructure servers in between. DNS servers not shown.

mai [1]). When the client opens a TCP connection to transmit its request, the connection may be directed through a proxy, to an infrastructure server, or directly to the content provider. If an infrastructure provider or proxy is involved, they may internally route the request through several hops and/or DNS lookups. For each of the transport and application layer steps described above, packets might need to flow across and between multiple ASes. In each AS and at each peering point, routing policy or congestion may cause packets to be delayed or lost, and the request to fail.

An ideal set of HTTP request logs would be: 1) *complete*, containing records of all HTTP requests, whether they were successful or not, 2) *deep*, capturing information about requests' dependencies on infrastructure components, including client-side, network and server-side systems, and any observations of failures in these components; and 3) *accessible*, meaning the logs are easily available to content providers and fast to collect. While this idealized web log does not exist, today's typical content providers can capture part of this information from three different vantage points, each with its advantages and disadvantages:

Client-side logs: Internet services often have a subset of customers (such as paid or volunteer beta-testers, 3rd parties that measure site reliability, etc.) who have agreed to log and report their view of the service. These logs can have very detailed information about the end-to-end reliability of the service from these clients' viewpoint, but these clients are often a poor sample of the overall population. For example, many enthusiasts might be concentrated in a particular region of the world, or have particularly reliable Internet connections.

CDN logs: Content-distribution networks (CDNs), such as Akamai, record logs of request success and failure of every request that passes through their proxies, even if wide-area network failures keep these requests from reaching the Internet service itself. However, these logs still do not contain information about any request that fails to reach the CDN proxy. Moreover, because of various geographic distribution, load, and engineering constraints, these logs are often not accessible for hours after requests initially occur.

Central logs: These logs contain records of every request that reaches the web servers at an Internet service. These logs are generally very easy to access, but do not contain direct information about requests that never reach the service because of network failures.

The actions a content provider can take after determining that its clients are seeing poor performance varies with the type of problem. Common actions include:

Reengineering connectivity: If congestion or routing policy appear to be a problem, contact the operators of the AS involved and ask them to move the site's traffic to a different peering point, add more capacity to the link to the AS's peer or perhaps add a link directly to the AS hosting the content provider's site.

Targeted nagging: If a misconfiguration, such as a proxy problem, appears to be the issue, locate the operators of the misconfigured system in question (*e.g.*, the IP address block) and ask them to fix their system.

Content changes: If a browser version incompatibility appears to be the issue, contact the content producers with a request to support the browser.

2.2 Challenges

Serious challenges confront the content provider who is trying to help its customers see better service.

Defining failure in the presence of noise: A request may be marked as "failed" for many reasons that are unrelated to network infrastructure problems. For example, client-side browser failures, where some version of a browser, robot, or malicious worm consistently makes invalid HTTP requests. A converse example would be a server-side problem in the content provider's systems. Normal user behavior, like canceling requests or "clicking away" before a response arrives, may also cause a request to be logged as a failure. This may indicate an impatient user or a serious slowdown in the network infrastructure. Working with request status logs requires techniques to separate "background" odd behavior from serious failures. Section 3.1 describes a technique that can separate out several classes of noise from network infrastructure failures.

Coalescing of failures into incidents: While content providers are very alert to major interruptions in connectivity, minor network issues are more likely to be dealt with after-the-fact, prioritized in accordance with their recurrence and longer-term impact. To avoid overwhelming operators with all the individual failures that occurred over the course of a day or week, we must be ready with techniques that can coalesce groups of related failures into single incidents. Section 3.2 presents one such a technique for determining the boundaries at which incidents begin and end.

Incomplete information: Packets can be lost anywhere, for many reasons, and many kinds of failures can prevent requests from reaching a logging point. Given the few points of visibility for the content provider, failures can result in seeing fewer requests, rather than requests that fail. We must be able to infer the existence of particular failure from absences of requests, and are addressing this challenge in future work.

Experimental challenges: While perhaps less of an issue for content providers, as researchers one of our major challenges is finding ways to validate our techniques generally. This has proven difficult as the ground truth cause of failure incidents is hard to obtain. For this "first-impressions" paper, results were validated through manual investigation that ruled out other causes. An alternate approach might be to implement a test service on PlanetLab so that known faults could be experimentally injected, but results from such tests would be suspect as not replicating the full complexity of a content provider's service, the infrastructure providers, and the diaspora of clients. Conducting solid evaluation remains a topic of current work.

3. APPROACH AND DATA ANALYSIS

We first attack the issue of finding likely locations of failures in the network, while explicitly accounting for failures due to DoS attacks, broken problematic clients, etc. Secondly, we address the challenge of identifying when incidents begin and end.

We focus our analysis on 3-hours of logs bracketing a problematic period one afternoon in the fall of 2005, and present summary results from analyzing a full week of web logs from January, 2006.

The logs we analyze were recorded at servers in Akamai's CDN, co-located in over 1000 networks close to end-users. All requests answered locally by the Akamai servers were pruned from the logs, leaving only those that traversed the wide-area network to the content provider and back. This pruning was done to focus our analysis on faults occuring in the wide-area network between the Akamai servers and the MSN data centers. A future analysis will focus on failures, such as network problems between clients and Akamai proxies, that would affect requests satisifed by Akamai proxies. For each request, the log records whether the request succeeded or failed. We consider "abandoned requests"—where a user cancels a request before receiving a response—to be failures as well. During this week, we saw over 1 billion requests coming from over 10k distinct ASes. During these periods, there were no known system failures, nor did our analyses point to any system failures.

3.1 Where are the problems?

Fundamentally, our goal is to help Internet content providers identify and fix any problems in the network infrastructure that are affecting end-user perceived reliability. Our first step is to estimate the failure probability of each piece of the network infrastructure (including the client's browser and the content provider's servers). When a serious problem occurs, our estimate of the failure rate of some piece of the infrastructure should increase, and the Internet content provider can contact the owner of that infrastructure and encourage them to repair it. Since contact information is generally available for each Autonomous System (AS), our first cut at identifying failed network infrastructure operates at the granularity of ASes.

A naïve method of estimating the failure rate of ASes would be to simply count the fraction of of successful and failed requests from every AS. However, as pointed out earlier, a request could fail for many reasons other than network infrastructure problems. A failure at MSN's servers or the sudden spread of a worm sending broken HTTP requests would unfairly penalize the estimated failure rate of many ASes. A more accurate method must take into account these alternate explanations for failures, in effect estimating the failure rate of other possible causes of problems as well as the failure rate of network infrastructure. We call each potential cause of a failure a *candidate*.

General Algorithm: Formally, to estimate the failure rates of candidates given the success/failure of requests, we use a noisy-OR model for root cause finding, a technique first used in the context of medical diagnosis [2, 3]. That is, we assume that if any one of the candidates on which a request depends fails, then the request fails as well. Alternatively, we can state that when an HTTP request succeeds, every associated candidate has also "succeeded." But, when an HTTP request fails, we only know that *at least one* of the candidates involved with the request has failed.

We have a novel method for performing approximate inference on the noisy-OR model. We apply stochastic gradient descent (SGD) to the data likelihood to create on-line estimates of the underlying probability that each candidate is the cause of observed failures. This model was inspired by the noisy-OR boosting of [4], but does not require boosting or that the underlying causes are tied in any way. Unlike previous inference techniques, we maintain current estimates of candidate probabilities that update as new observations arrive. Previous inference techniques for noisy-OR solve the cause inference problem for a single fixed set of observations, essentially using known values for the parameterization of the q_j below.

First, for each request we determine which candidates *might* cause that request to fail. This is equivalent to determining the set of candidates on which a request depends. In our experiments, we consider three types of candidates: 1 the specific Internet site being contacted (*i.e.*, the site's hostname); (2) the client's AS; and (3) the client's browser type. We are currently working to integrate BGP feeds into our analysis, so that we can also explicitly consider transit ASes between the servers and clients. We label the set of candidates associated with each request i as C_i.

Then, we calculate the probability p_i that any given request i is going to fail as a noisy-OR of the probabilities q_j that any of the candidates $j \in C_i$ associated with the request fails:

$$p_i = 1 - \prod_{j \in C_i} (1 - q_j) \tag{1}$$

We parameterize q_j to be a standard logistic function of the log-odds z_j:

$$q_j = \frac{1}{1 + e^{-z_j}} \tag{2}$$

For every new request i, we can update our estimates of the failure probabilities of the candidates associated with the request. This update is in the direction of the gradient of the log of the binomial likelihood of generating the observations given the failure probabilities:

$$D = y_i log(p_i) + (1 - y_i) log(1 - p_i) \tag{3}$$

$$\Delta z_j = \eta \frac{\partial D}{\partial z_j} = \eta \frac{q_j (y_i - p_i)}{p_i} \tag{4}$$

where η is a weight that controls the impact of each update, and $y_i \in \{0, 1\}$ indicates the observed success ($y_i = 0$) or failure ($y_i = 1$) of a request i.

We use an initial value of $z_j = -5$ for all candidates j. For each request i, updates are applied only to the candidates j involved in that request. Since not all candidates are involved with each request, as requests are processed, the posterior probabilities of each candidate j diverge from each other.

Empirically, we have found that using a relatively high value of $\eta = 0.1$ and applying an exponential smoothing function on the gradient, Δz_j, provides a good trade-off between responsiveness to failures and stability in reported values. Thus, we calculate a smoothed gradient, $\tilde{\Delta} z_j$, at time t as:

$$\tilde{\Delta} z_j^t = \alpha \tilde{\Delta} z_j^{t-1} + (1 - \alpha) \Delta z_j^t \tag{5}$$

Interpretation and Motivation: We interpret the resultant probabilities q_j as follows. An estimated failure probability approaching 100% implies that all the requests dependent on the candidate j are failing, while a probability approach 0% implies that no requests are failing due to candidate j. If the estimated probability of failure is stable at some value between 0% and 100%, then that implies that the candidate j is experiencing a partial failure, where some dependent requests are failing while others are not. For example, an AS that drops half of its outgoing connections would have a failure probability estimate approaching 50%.

The SGD technique was selected for several reasons. First, it avoids the problems of the naïve approach described earlier that only detect large failure rates (thereby missing the small failures in large ASes). Second, the algorithm produces a refined estimate of the probablity each candidate has failed after processing each request. This on-line/incremental nature should make the algorithm well suited for ongoing monitoring (we currently use it off-line, but

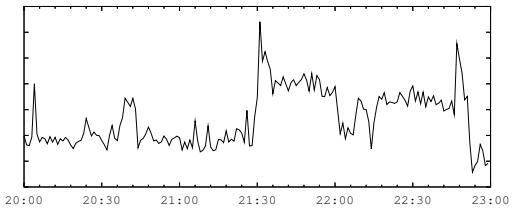


Figure 2: The timeline of observed system-wide failure rates during a 3-hour period. We redact the scale of the error rate to avoid disclosing sensitive information.

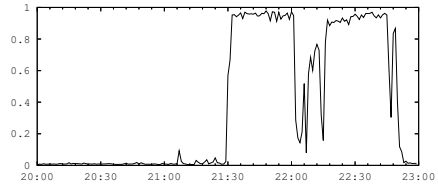
are examining on-line applications). Third, it is a very expressive framework to which it is easy to add additional candidates. Fourth, we believe it makes maximal use of all the available information.

Results: To test how well the success/failure of HTTP requests, together with this SGD analysis, can estimate the failure probabilities of network infrastructure components, we examined our 3-hour logs from fall 2005. Figure 2 shows the observed system-wide failure rates during this period. The graph begins with a low rate of background failures occurring due to broken browsers and problems at small ASes. At 21:30, we see that an incident occurs, and the failure rate increases for approximately 85 minutes. Our goal is to see whether and how quickly SGD analysis can localize these failures to some network infrastructure component.

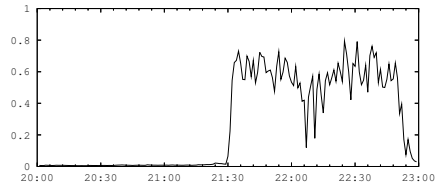
We highlight SGD's failure probability estimates for several candidates in Figure 3. Here, we show the timeline of estimated probabilities for 1) two ASes that are associated with the incident, 2) an AS that suffers a short-lived failure at time 21:07 and at 21:30, 3) a browser-type, ColdFusion, that is failing almost continuously throughout the 3-hour period, and 4) an AS that is not associated with any failure during this period.

In Figure 4, we show the observed failure rates of the two client ASes most closely correlated by the SGD analysis with the major incident. Upon inspection, we find that these two ASes together account for almost all of the additional error-load that occurred during the time-period of the incident. The graph illustrates a strength of our technique in that AS 2 is correctly identified as being associated with the incident, even though AS 2 is small enough that the total number of errors it contributes is dwarfed by the failures from AS 1. According to their WhoIs entry, both of these ASes are located in the same geographic area, leading us to believe that they share some relationship in the network topology, and a single failure caused both to be unable to reach the MSN service. SGD's failure probability estimates for these ASes were immediately affected, and rose to 95% within 2-3 minute of the incident's start.

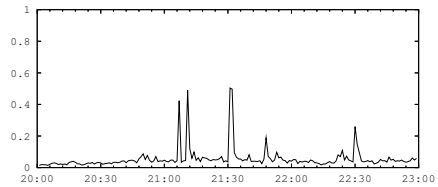
We applied our SGD analysis to a full week of web logs, and we analyzed the resultant failure probability estimates to better understand how many different ASes a content provider might have to contact and work with to resolve a problem during a week. First, we found that most ASes are quite reliable: almost 75% of ASes have a peak estimated failure probability less than 1% (in other words, greater than 99% reliability in their worst minute). Moreover, 99% of ASes maintained a mean estimated reliability of 99%, over a fifth maintained 99.9% reliability, and many maintained better than "five 9's" of reliability. Only 2.9% of ASes had a peak estimated failure probability greater than 50%, and of these ASes, only 19% had peaks greater than 50% for more than 10 minutes out of the week.



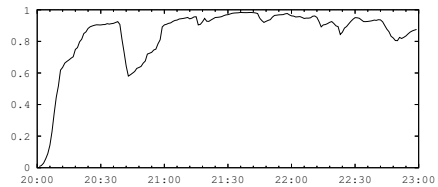
(a) Bad AS 1 failure probability



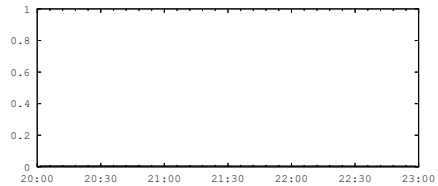
(b) Bad AS 2 failure probability



(c) Bad AS 3 failure probability



(d) Bad browser type failure probability



(e) Good AS failure probability

Figure 3: A highlight of the estimated failure probabilities resulting from our SGD analysis for two ASes associated with the major incident during this time, a third AS that is failing separately, a browser type that is almost continuously failing, and a good AS that sees no failures during this time.

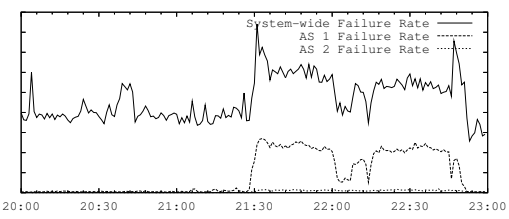


Figure 4: A timeline of the observed failure rate from the two candidate ASes most correlated with the incident shown in Figure 2.

3.2 Identifying incidents

Our goal in this subsection is to develop analysis techniques for web logs that can discover when incidents affecting client-perceived reliability begin and end. This information is important because it can help focus further investigation efforts on alerts, maintenance activities, logs, etc. occurring at about the same time as the incident. While humans are very good at manually analyzing data and recognizing the pattern-changes that indicate that such an incident has occurred, we wish to automate this procedure for two reasons: 1) humans cannot scale to analyzing the volumes of data in days of web logs of client requests from thousands of ASes; 2) while humans are good at detecting major incidents, the many minor incidents can get lost as "noise," especially when looking at such large amounts of data.

A simple but naïve algorithm is to detect incidents by looking for failure rates to rise above a predetermined threshold, or by monitoring for high minute-to-minute changes in the failure rate. However, because of the level of background noise, bursty failures, etc., these techniques generate too many false positives and negatives—failures that begin slowly might not be noticed, and an incident that caused very noisy or bursty changes in failure rate would be classified as many incidents instead of just one.

General Algorithm: We approach this problem as one of segmenting a time-series of failure rates into regions, where the time-series values within each region are generally similar to each other, and generally different from the time-series values in neighboring regions. This is equivalent to finding the *change points* in the time series [5], and this particular offline approach is originally due to Fisher [6]. In this model, a transition boundary between two regions then represents abrupt changes in the mean failure rate, and thus, the potential beginning or end of one or more incidents. This problem is mathematically equivalent to the v-Optimal histogram problem, described in [7].

More formally, given a time-series of failure rates $x_1, \cdots x_n$, we attempt to find a segmentation of the time-series into k regions, such that we minimize the total distortion:

$$D = \Sigma_{m=1}^{k} \Sigma_{i=s_{m-1}+1}^{s_m} (x_i - \mu_m)^2 \quad (6)$$

where s_m represents the time-series index of the boundary between the m^{th} region and the $(m+1)^{th}$ region, $s_0 = 0$, $s_k = n$, and $\mu_m = \frac{\Sigma_{i=s_{m-1}+1}^{s_m} x_i}{s_m - s_{m-1}}$, the mean value of time-series throughout the m^{th} region. We implement a dynamic programming algorithm to find the set s of boundaries that minimize D.

To fit the parameter k, we can use one of the many model fitting techniques described in the literature of statistical pattern recognition and statistical learning [8, 9]. In the analyses performed in this paper, we iterated over k, generating a curve of distortion rates. We select the value of k associated with the knee in the distortion curve, as this balances our desire to fit the boundaries to the data while avoiding the problem of overfitting (since overall distortion approaches 0 as $k \rightarrow n$ and every time period becomes its own region).

Results: In Figure 5, we show the result of applying this technique to the problematic period from Fall 2005, segmenting the system-wide failure rate into five pieces as indicated by the knee in the distortion curve. We have applied this segmentation analysis to system-wide failure rates, and are currently experimenting with applying it to timelines of failure rates of individual ASes and other candidates.

We should be clear that each segment found by this algorithm does not correspond to either an incident or an incident-free period. Rather, since many independent incidents might be occurring simultaneously, each segment boundary corresponds to the beginning or end of one or more incidents. To better understand such occurrences, we are currently working to combine the information we gain from our segmentation analysis with the results of our failure rate estimations for candidates. By comparing the sets of candidates with high failure rates before and after each segment boundary, we expect to be able to generate a list of incidents that have started or stopped at each boundary.

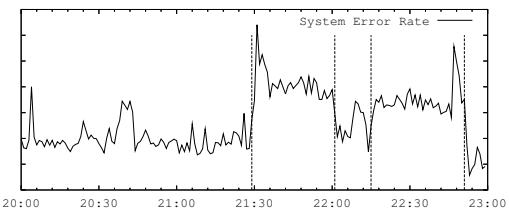


Figure 5: The 3-hour fall 2005 incident, as segmented by our incident boundary technique. The segment boundaries correspond to when an failure incident begins or ends.

4. RELATED WORK

There is a large body of prior work measuring client perceived performance of web servers. Two of the most related are WIND and WAWM. The WIND project[10] uses wavelets to process packet-level traces of interactions with web servers to find "interesting" portions of the data. Our work differs in two ways. First, our segmentation technique identifies event boundaries rather than individual bins of time that contain interesting data, thereby reducing the effort required from human operators. Second, we concentrate on attributing problems to a *location* in the network topology or an alternate candidate (*e.g.*, browser type), where WIND focuses on finding network *paths* with issues but not the location of the issue along the path. WAMN [11] uses packet-level traces collected at clients and web servers to "assess the relative impact of server delay, network delay, packet loss, etc. on transfer latency." Our work seeks to determine where in the network lie the problems causing failed requests. This forces us to develop techniques that cope with a wider range of causes for failed requests, including AS or DNS failures that prevent the transfer from even starting.

Other researchers have addressed the issue of distributed network measurement [12, 13] and failure diagnosis in IP networks [14] and DNS infrastructure [15], but here we are concerned with how a single content provider running a small number of data centers can leverage the data available to it. Incorporating information from public distributed network measurement services to refine the predictions of our techniques would be interesting future work. Similar to these systems are the commercial services, such as Keynote and Alexa, that "rate" the performance of content providers by accessing them from a large number of distributed clients.

Recent research has also applied various statistical and machine learning analyses to detect and diagnose failures within the internal systems of an Internet service [16, 17]. In general, these projects are complementary to our own, though some challenges are shared across domains: one such challenge is the visualization of algorithmic results and the consideration of operator confidence in the techniques [18].

5. DISCUSSION AND FUTURE WORK

This paper presents our initial steps to analyze the logs of requests made to Internet content providers with the goal of identifying and debugging wide-area network problems that interfere with clients using the content provider. While we have begun by attacking two specific problems—estimating the failure rates of network infrastructure elements and identifying the start and end of incidents—there are many open, challenging problems in this area, such as recognizing recurring problems and discriminating between classes of incidents (*e.g.*, distinguishing router issues from DNS problems).

Currently, we are applying our analyses to longer periods of time to better characterize and explore the problems occurring in the wide-area network. To extend our visibility further into the network infrastructure, we are integrating into our analyses more sources of data, including centralized web logs, client-side web logs and BGP feeds. We are also extending our analyses to better deal with missing information, such as occurs when a network failure prevents a request from reaching a proxy or server that would log its failure.

Finally, we are investigating the different kinds of results that are useful to Internet content providers as they attempt to debug failures in wide-area network infrastructure and work with third-parties to repair and resolve these problems. Our hope is that by providing the right tools to the parties with the resources and incentives to resolve end-to-end connectivity issues, we will be able to improve the reliability of the Internet as a whole.

6. REFERENCES

[1] http://www.akamai.com/. last visited 4/17/06.

[2] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper, "Probabilistic diagnosis using a reformultation of the INTERNIST-1/QMR knowledge base: Part 1. the probabilistic model and inference algorithms," vol. 30, pp. 241–255, 1991.

[3] B. Middleton, M. Shwe, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper, "Probabilistic diagnosis using a reformultation of the INTERNIST-1/QMR knowledge base: Part 2. evaluation of diagnostic performance," vol. 30, pp. 256–267, 1991.

[4] P. A. Viola, J. Platt, and C. Zhang, "Multiple instance boosting for object detection.," in *Proceedings of NIPS*, 2005.

[5] M. Basseville and I. Nikiforov, *Detection of Abrupt Changes: Theory and Applications*. Prentice Hall, 1992.

[6] W. Fisher, "On grouping for maximum homogeneity," *Journal of the American Statistical Association*, vol. 53, pp. 789–798, December 1958.

[7] H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel, "Optimal histograms with quality guarantees," in *Proc. of the 24th Intl Conf on Very Large Databases (VLDB)*, August 1998.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience, 2nd ed., October 2000.

[9] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. Springer, July 2003.

[10] P. Huang, A. Feldmann, and W. Willinger, "A non-instrusive, wavelet-based approach to detecting network performance problems," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 213 – 227, 2001.

[11] P. Barford and M. Crovella, "Measuring web performance in the wide area," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 2, pp. 37 – 48, 1999.

[12] R. Wolski, N. Spring, and J. Hayes, "The network weather service: A distributed resource performance forecasting service for metacomputing," in *Journal of Future Generation Computing Systems*, vol. 15, pp. 757–768, October 1999.

[13] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, "Planetseer: Internet path failure monitoring and characterization in wide-area services," in *Proceedings of the Sixth Symposium on Operating Systems Design and Implementation*, 2004.

[14] S. Kandula, D. Katabi, and J. P. Vasseur, "Shrink: A tool for failure diagnosis in IP networks," in *Proceedings of ACM SIGCOMM Workshop on Mining Network Data*, August 2005.

[15] V. Pappas, P. F. ltströ m, D. Massey, and L. Zhang, "Distributed DNS troubleshooting," in *Proceedings of ACM SIGCOMM Workshop on Network Troubleshooting*, August 2004.

[16] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *Proceedings of the 1st IEEE International Conference on Autonomic Computing*, 2004.

[17] I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons, "Correlating Instrumentation Data to System States: A Building Block for Automated Diagnosis and Control," in *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2004.

[18] P. Bodik, G. Friedman, L. Biewald, H. Levine, G. Candea, K. Patel, G. Tolle, J. Hui, A. Fox, M. Jordan, and D. Patterson, "Combining visualization and statistical analysis to improve operator confidence and efficiency for failure detection and localization," in *Proceedings of the 2nd IEEE International Conference on Autonomic Computing*, 2005.