Internet Service Performance Failure Detection

Amy Ward Peter Glynn
Engineering Economic Systems &
Operations Research Department
Stanford University
Stanford, CA 94305
{*award,glynn*}*@leland.stanford.edu*

Kathy Richardson
Network Systems Laboratory

Compaq Computer Corporation
Palo Alto, CA 94301
kjr@pa.dec.com

Abstract

The increasing complexity of computer networks and our increasing dependence on them means enforcing reliability requirements is both more challenging and more critical. The expansion of network services to include both traditional interconnect services and user-oriented services such as the web and email has guaranteed both the increased complexity of networks and the increased importance of their performance. The first step toward increasing reliability is early detection of network performance failures. Here we consider the applicability of statistical model frameworks under the most general assumptions possible. Using measurements from corporate proxy servers, we test the framework against real world failures. The results of these experiments show we can detect failures, but with some tradeoff questions. The pull is in the warning time: either we miss early warning signs or we report some false warnings. Finally, we offer insight into the problem of failure diagnosis.

1 Introduction

The goal of failure detection is to preserve network and service reliability by identifying possible problems before they impact end user services. Early discovery opens the possibility of corrective action. This is a two-fold problem as we must both alert the network manager to the fact that a failure is imminent and convey its probable cause. Here we quantify the failure detection problem and identify an approach for the isolation problem.

Proxy and Web servers are affected by internal network failures, Internet failures, proxy server and proxy application failures, domain name service problems, and local ISP failures. Each of these types of failures results in lowered network performance. Performance failures range in severity from increased service time to total service denial. Total service denial is easy to detect; degradations in service time are difficult. However, increased service times cause equally significant problems for service applications. From a business's perspective, slower web service results in decreased revenues and fewer repeat customers. From the user's perspective, slow request processing means idle computer time and, if dramatic enough, reason to forego connection requests entirely.

Part of the difficulty inherent in failure detection is the definition of a performance failure. In order that Quality of Service requirements can be placed on network applications, we must have a reliable mechanism for fault management [HJ]. Here we define a failure as a user-problematic departure from expected operating conditions [MO93]. In our environment, we use the number of requests processed at our proxy servers as a mechanism for tracking performance. Noticeable deviations in the number of requests processed for a specific time interval often indicate undesirable operating conditions. Summarizing, our general approach to the failure detection problem is to first establish a measure that reflects operating conditions whose general expected behavior we can discern and then determine when measurements deviate from that behavior.

Changing network conditions and continually evolving traffic patterns mean different expected behavior for performance measures over time. However, for a mature network, it is reasonable to assume that for a limited time span we can model regular behavior patterns and use this model to spot deviant behaviors as an indication of performance failures [Max90]. Here we present methodology for identifying and tracking a process that reflects network performance and give results for the process we tracked on our network. Though different processes on different networks may exhibit different behavior, the ideas generalize to any network upon which performance measurements can be taken and which exhibit regular patterns over some time interval.

2 Measurement Setup

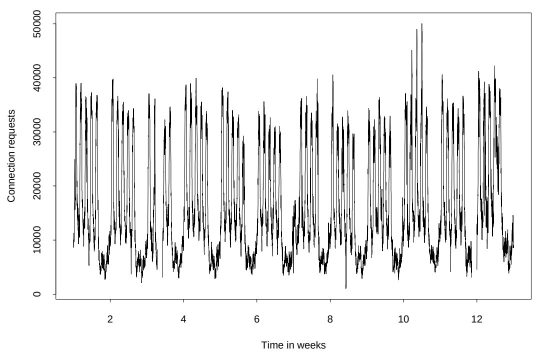
Measurements were taken for a 12 week time interval, between June 21, 1997 and October 12, 1997, at the two proxy servers through which requests for Internet connections from machines within Digital Equipment Corporation are routed. These two proxy servers service upwards of 10,000 client machines and, during the busiest hours of the day, typically process 30,000 - 40,000 requests during a 15 minute time span. Total daily traffic is upwards of two million requests per day.

We concentrate on one measurement variable for the failure detection process: the total number of requests processed every 15 minutes. However, the methodology presented in the data analysis section can be generalized to apply to measurements from any network process meeting the stated assumptions. We further consider the following three measurement variables for the failure identification process:

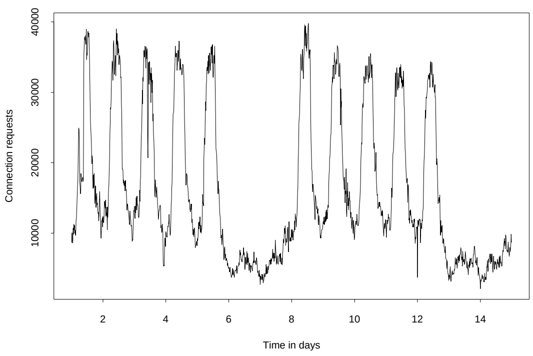
1. Number of TCP_IP connections in the `Established` state
2. Number of TCP_IP connections in the `Syn_Sent` state
3. Number of TCP_IP connections in the `Syn_Rcvd` state

3 General Observations

Throughout a given day, we see a certain pattern of connection requests. This pattern corresponds to a lower number of requests early in the day, a higher number of requests during the middle of the day, and a return to the lower number in the evening. The pattern is most pronounced on work days and is similar, though much less pronounced, on holidays. Observe the pattern in figure 1 (a) and figure 1 (b). One shows the entire data series



(a)



(b)

Figure 1:
(a) Connection requests for both proxy servers, 07/21-10/12
(b) Connection requests for both proxy servers in a 2 week interval

and the other shows a two week interval in order to clearly illustrate the time-of-day and day-of-week patterns. These patterns are general network traffic patterns and the heuristics are supported by measurements made by the vBNS engineering department at MCI within Internet MCI's backbone [TMW97]. Abnormalities in this pattern usually correspond to some type of failure. For example, a sudden drop in the number of connection requests is cause for alarm. Figure 3 shows a day in which a problem did occur. The sharp downward drop in connection requests in the middle of the day details the time at which the failure occurred and the following upward swing indicates the return to normal network operation. We also see a failure at the end of the day, again corresponding to the sharp downward drop. The question is how large this drop must be to merit immediate investigation.

Another important feature of the data is its correlations. Even after de-trending the data so as to compensate for the daily and weekly patterns mentioned above, we still find correlations extending beyond consecutive time slots equal to between 0.2 and 0.3. We hypothesize the cause to be request service time. Overall network

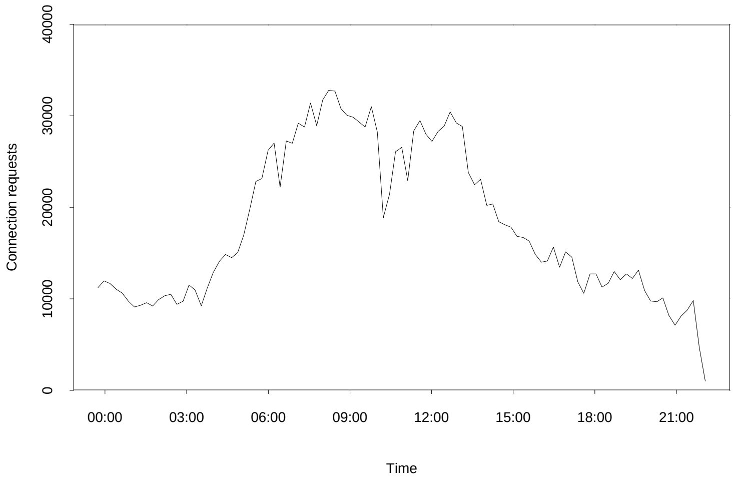


Figure 2: Example of connection requests in a day when a failure occurs

performance has a large effect on request service time and remains relatively stable across short time periods. Request service times then affect request rates [Gla94], accounting for the data correlations. Usually, the faster requests are serviced, the faster users make them. For example, consider browsing web pages. The more quickly these pages are requested depends directly upon how quickly the pages are received after a request is made. Because slow or fast service times during one time period will also occur during the next time period, we see correlations in the measurements.

The most important consequence of having correlated data is many statistical models depend upon a series of independent observations. The longer range data dependencies we find invalidates most traditional statistical modeling frameworks. Thus we first consider what results we can obtain with minimal assumptions about the underlying process producing the measurements.

4 Data Analysis

The general methodology presented here depends upon the following assumptions.

1. The process is stationary over some time interval. In order to accurately analyze the process's behavior at some future point in time, we must assume that its past is representative of its future. Specifically, let $\vec{X}_i$ be the vector of observations on workday i. Then we assume the sequence $\vec{X}_n, n \geq 0$ has the same distribution as the shifted sequence $\vec{X}_{n+k}$ for each $k \geq 1$. We make the equivalent assumption for holidays.

2. We have a strong law of large numbers: i.e., the mean of the observations we see at a given time of the day converges to the expected number of connection requests at that time. Let X_i^j be the jth observation on the ith workday in the data series. Then we assume $E[X_i^j] \approx \frac{1}{n}\sum_{i=1}^{n} X_i^j$ for large n. Again, we make the equivalent assumption for holidays.

3. Deviant behavior of the process can reflect network failures. The process may behave "strangely" at times when no failure occurs. However, this requirement states that, when a failure occurs, it must behave strangely. The tough question is to identify the times at which the measurements reflect failures and the times at which they are fluctuations of the process itself.

Notice that our assumptions about the underlying process are minimal. Here we explore the boundaries of how much information is necessary for informative conclusions.

4.1 Methodology

We propose the following steps as a first pass procedure for detecting network faults based on observed measurements.

1. Discard all observations from the data series during which a known fault occurred.
2. Use the remaining observations to estimate the distributions for the measurements from the process at each time period.
3. Determine mean and variance for the process at each time period.
4. Adjust the sensitivity level for identifying deviant observations.

Step 1 is critical. All subsequent analysis assumes that the observations we consider accurately reflect the process under desired operating conditions.

In step 2, we determine that our data is normally distributed in each time period. In general, it may or may not be true that the distribution for the observations is normal. However, in the case that the measurements record N(t), the total number of counts up to time t, and the arrival distribution for these counts are independent and identically distributed, Donsker's Theorem for renewal processes [Bil68] states that for large enough time scales, N(t) is approximately normal. In our case, N(t) counts the number of processed connection requests during a 15 minute time interval. We do not automatically assume either that our time scale is large enough or that the connection request arrivals occur independently. Rather we observe that it is likely our observations could be normally distributed and perform statistical tests to confirm this supposition. The statistical test we performed was the test developed by Anderson and Darling [Pet77], and the test confirms the normality assumption. The mean and variance of the process at each time period can now be used to identify deviant observations; i.e., observations from the tail of the distribution. The criterion for the observation to be deviant should be adjusted such that rarely is a failure identified when no failure occurred and that all behavior possibly indicative of a failure is reported.

Figure 4.1 shows the results for our data series for step 3. Specifically, figure 4.1 tracks the estimated mean plus or minus 1 standard deviation for the process for a normal workday. Notice the rise in connection requests in the morning, as workers arrive, the heightened fluctuation of the process during mid-morning to mid-afternoon, and the decline in connection requests as workers leave. The early rise and fall in connection requests is due to the fact that the bulk of DEC's business operations is centered on the East coast but their connection requests are processed at proxy servers on the West coast.

Step 4 uses the expected behavior of the number of processed connection requests found in step 3 to find possible failures through anomalous behavior. To identify deviant observations, we first transform each observation as follows:

$$Z_t = \frac{X_t - \overline{X_t}}{\overline{\sigma_t}}$$

where $\overline{X_t}$ is the estimated mean of the observations at time t and $\overline{\sigma_t}$ is the estimated standard deviation of the observations at time t. For large time t, Z_t looks approximately like a standard normal random variable. We then consider both "too high" observations and "too low" observations because often a large increase in the number of connection requests overloads the servers on the network and results in a performance failure. Further, deviant high loads can signal attempted break-ins and, therefore, may identify security threats. "Too low" observations reflect performance failures since they indicate the network is operating under decreased loads. There is an asymmetry between how high an observation must be and how low one must be to be indicative of a failure. The reason for this phenomenon is that an observation above the bounds of "normal" behavior for the process must be far enough above to reflect an overloaded network. The performance failure then results from slow response times to service requests. The exact bounds we use follow in the next section.

4.2 Results

The experiment we performed to test our methodology uses the first two-thirds of the data series to train the algorithm and the last third to measure its performance. We adjust the algorithm's parameters so as maximize the number of failures we correctly detect and minimize the number of false failures we identify. (A false failure is defined as a failure found by our algorithm but one that does not correspond to a known performance failure.)

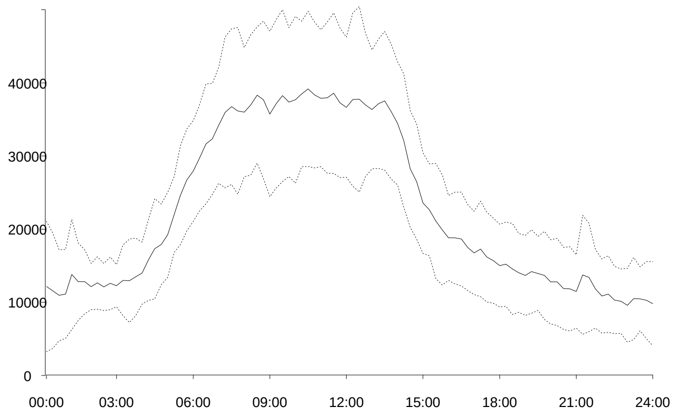


Figure 3: Estimated mean plus or minus 1 standard deviation for a 24 hour workday

The bounds we use to classify the observation Z_i as deviant are: $Z_i < -2.3$ or $Z_i > 3.00$. We then use these same bounds on the last third of the series to identify deviant observations and calculate algorithm performance metrics.

Our results demonstrate a working algorithm for failure detection. In the first two-thirds of the data series, we identify 80% of the failures and 2 false failures. Using the same parameters, we then identify 90% of the failures in the last 3rd of the series with only 2 false failures.

4.3 Detection Tradeoffs

Ideally, we would like to detect early warning signs of significant performance failures in order to pro actively react to avoid them. This can be abstracted to the problem of predicting an 'extremely deviant' observation in a general stochastic process once the words 'extremely deviant' are defined. Currently, our algorithm gives little or no warning that a failure will occur. One remedy we investigate is to use a window of observations to identify early failure warning signs. We can then take advantage of known patterns we observe before significant performance failures. The general pattern of warning signs for a failure is either several extremely high observations indicative of an overloaded network or several moderately low and declining observations indicative of a physical problem worsening performance. We incorporate this information into another algorithm given below.

Specifically, let Z_n denote the current observation and $Z_n, Z_{n-1}, \ldots, Z_{n-N}$ all the observations under consideration. If Z_n is extremely deviant, then we immediately conclude a failure occurred. If Z_n is moderately low or extremely high, then we conclude a failure occurred only if the number of moderately deviant observations in the past N time periods exceeds some threshold value.

Similar to how we tested the methodology first presented, we use the first two-thirds of the data series to determine the appropriate window size and criterion for extremely deviant and moderately deviant observations. We then use the identical parameters on the last third of the data series to calculate algorithm performance results.

This algorithm again found 90% of the failures in the last third of the data series but also found 3 false failures. Most importantly, this algorithm identified failures earlier. When setting the parameters for this algorithm, we found 87% of the failures in the first two-thirds of the data series with 4 false alarms. We used a window size of 5 and a threshold value of 2. Our criteria for a moderately deviant observation was $Z_n < -1.96$ or $Z_n > 3.00$. We used the same criterion for deviant observations as in the methodology first presented.

The advantage to this algorithm is earlier identification of failures; the disadvantage is more frequent false

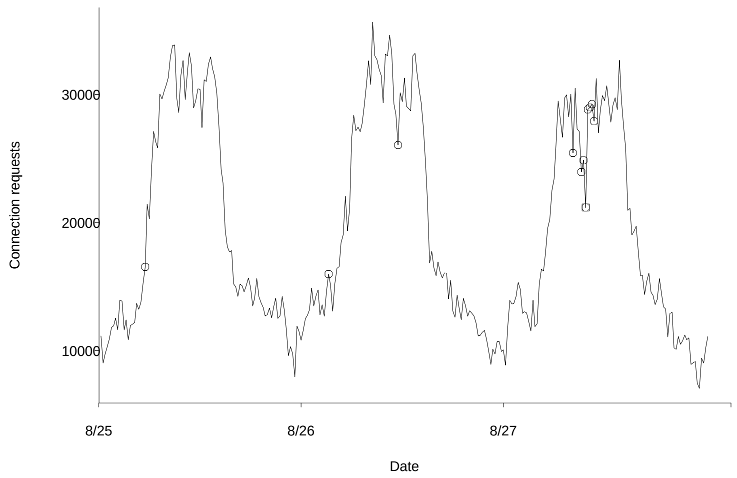


Figure 4: Failures detected using 1 observation (squares) vs multiple observations (circles)

failures. We illustrate the tradeoff between the two algorithms in figure 4.3. The figure shows a 3 day interval in the data series with no failures on the 1st day, one moderate failure on the 2nd, and one severe failure on the 3rd. The circles depict the failures found using additional observations and the squares show the failures found using only one observation. The squares identify the most extreme failure, but miss the early warning signs of it and miss the moderate failure entirely. The circles identify both failures, and warn of the extreme failure but also detect a false alarm in the morning both on 8/25 and 8/26. This is representative of the larger problem of knowing which moderately deviant observations are early warning signs of a failure and which are merely fluctuations in the process.

5 Failure Diagnosis

Once it is known that a failure has occurred, the problem remains to determine its cause. We use 4 measurement variables, as detailed in the Measurement Setup section, to isolate the problem:

1. Proxy throughput (what we have been studying)
2. Number of TCP_IP connections in the `Established` state
3. Number of TCP_IP connections in the `Syn_Sent` state
4. Number of TCP_IP connections in the `Syn_Rcvd` state

The TCP_IP `Established` state indicates an active connection. The TCP_IP `Syn_Sent` state indicates that the proxy server has requested an outside connection and is waiting for a reply. The TCP_IP `Syn_Rcvd` state indicates that the proxy has replied to a connection request from an internal machine and is waiting for that machine's reply before establishing the connection. Typically, all 4 processes are highly correlated. Failure diagnosis depends upon identifying out-of-sync behavior. For example, an external problem is evidenced by a jump in the ratio of the number of connections in the `Syn_Sent` state to the number in the `Syn_Rcvd` state and a decrease in throughput. This is because the amount of time spent in the `Syn_Sent` state reflects the health of the external internet and the amount of time spent in the `Syn_Rcvd` state reflects the health of the internal network. The throughput drops as well since the proxy is now slow to process outside connection requests.

Consider failure diagnosis for the two failures in figure 4.3. On neither day do we see measurements indicative of external problems (as explained above). However, on August 26, the number of established connection drops in conjunction with the failure in addition to a decrease in the number of TCP_IP `Syn_Rcvd` connections. This is indicative of an internal network problem. On August 27, the `Syn_Sent/Syn_Rcvd` ratio remains fairly constant.

The proxy server application logs show that a reboot occurred on both days. A reboot requires a cache rebuild during which most requests must be serviced externally. This delays service and causes fewer connection requests. Therefore, the reboot alone could explain the drop in connection requests on both days. On the 26th, internal network problems may have contributed to this reboot whereas on the 27th we suspect the fault lied solely at the proxy server.

6 Summary and Future Work

We have introduced a general framework for network performance failure detection and presented results for proxy request rate measurements which fit the required assumptions. Because we use proxy server measurements that reflect network operating conditions, we can identify performance failures as they are occurring and before service applications such as the web are drastically affected. We have further addressed how to use additional measurement series to make a preliminary diagnosis of probable failure cause.

Our results demonstrate that we have a working algorithm for catching extreme performance degradations with a low false identification rate. There is, however, a pull between how early we detect them and the number of false warnings we give. By identifying measurement patterns immediately prior to failure we are able to improve the detection results without increasing the false identification rate drastically. Finally, our results outline an appropriate approach to isolating the reason for the failure.

Future work focuses on time scale granularity, training sample choice, and failure diagnosis automation. First, a 15 minute sampling period may miss important process fluctuations. How fine the time scale must be to accurately capture network performance information is still unclear. Second, the length of a training sample and the duration of its validity requires further investigation. We do not know either the optimum time span of past data that should be used to define deviant behavior or how long the definition is valid. Finally, we would like to better understand the movement of different network measurement variables in parallel. This understanding will spur the establishment of methodology that automates the failure diagnosis process.

References

[Bil68] Patrick Billingsley. *Convergence of Probability Measures.* Wiley, 1968.

[Gla94] Steve Glassman. Proceedings of the first international world wide web conference. In *A caching relay for the world wide web*, pages 69–76. IEEE Computer Society, 1994.

[HJ] C.S. Hood and C. Ji. Automated proactive anomaly detection.

[Max90] Roy A. Maxion. 20th international symposium on fault-tolerant computing. In *Anomaly Detection for Network Diagnosis.* IEEE Computer Society, 1990.

[MO93] Roy A. Maxion and Robert T. Olszewski. 23rd international symposium on fault-tolerant computing: Toulouse, france. In *Detection and Discrimination of Injected Network Faults.* IEEE Computer Society, 1993.

[Pet77] A.N. Pettitt. Testing the normality of several independent samples using the anderson-darling statistic. *Applied Statistics*, 26(2):156–161, 1977.

[TMW97] Kevin Thompson, Gregory J. Miller, and Rick Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, November/ December 1997.