NwsAlarm: A Tool for Accurately Detecting Resource Performance Degradation[*]

Chandra Krintz
Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive, Dept 0114, La Jolla CA 92093-0114 USA
ckrintz@cs.ucsd.edu

Rich Wolski
Computer Science Department
University of Tennessee
203 Claxton Complex, 1122 Volunteer Blvd., Knoxville, TN 37996-3450
rich@cs.utk.edu

Abstract

End-users of high-performance computing resources have come to expect that consistent levels of performance be delivered to their applications. The advancement of the Computational Grid enables the seamless use of a multitude of computing resources by these users. The combination of these developments has generated a need for users to monitor the end-to-end performance available to an application. In addition, tools are needed to alert users of degradation in expected performance.

We present the NwsAlarm, a Java-based utility that enables users to monitor performance levels of any resource being monitored by the Network Weather Service. The NwsAlarm is invoked by a user without special privileges with a simple click on a web page link. More importantly, the NwsAlarm allows any user of the NwsAlarm to register and set expected performance levels. When performance levels fall below these thresholds, the registered administrators are immediately notified via email. The NwsAlarm uses prediction of performance measurements to filter false alarm values. We exemplify the importance of and accuracy achieved by the NwsAlarm with real examples of performance degradation caused by routing table changes and loss of service on the Abilene, Internet-2 research network used for experimentation with evolving Grid software technology. On average, 92% *fewer false alarms are raised by the NwsAlarm than if raw measurements are used.*

[*]This work was sponsored by the NASA Information Power Grid project and NASA Ames Research Center.

1 Introduction

As high-performance network connectivity proliferates, end-users have come to expect delivered network performance (and not just trunk capacity) to keep pace. In addition, better end-to-end performance makes it possible to consider the use of distributed computing platforms for applications that previously required expensive, large-scale, and dedicated machines. The Computational Grid [10, 3] is a new and successfully evolving distributed computing metaphor for the seamless and dynamic acquisition of resources from a heterogeneous, federated resource pool. In addition, "peer-to-peer" computing systems such a those developed by Entropia [8], ParaBon [19], and SETI@Home [20] are attempting to harness unused but ubiquitous computer capacity via the burgeoning internetwork of high-performance connectivity.

These recent advances place a premium on the ability to monitor the *performance deliverable to the application end-to-end*. Users need to ensure that the resources, for which they are paying but which they do not own, meet expected performance levels. System and network administrators responsible for appeasing this performance-hungry user-base must be able to detect and, if possible anticipate, deficient performance at the application level. The problem of performance monitoring is further complicated by resource federation. Often, administrative policy prohibits public access to low-level performance information for security and/or proprietary reasons. Even if low-level information is published, however, it is often difficult to translate it into a measure of performance delivered to the user.

In this paper, we describe a performance alarm system based on the *Network Weather Service* (NWS) [25]. The

NWS is a user-level performance monitoring and forecasting system designed to measure end-to-end resource performance in Computational Grid settings. It supports a variety of performance sensors (available CPU capacity, available core memory, end-to-end TCP/IP bandwidth and latency, etc.) and operates completely without privileged user access. Using the NWS as a backbone infrastructure, we have developed a Java-based tool for visualizing continuously generated NWS readings, and automatically triggering an email alarm when observed performance falls outside a specified range. The system draws heavily upon the adaptive statistical forecasting techniques that are part of the NWS [24] and their Java applet implementation [15].

Our results show that the NWS alarm system (NwsAlarm) can accurately detect problems such as routing misconfiguration by dynamically analyzing end-to-end network performance. It does this through its use of the Java implementation of the NWS forecasters. We illustrate these results with examples from the Abilene [1] experimental research network — a network facility deployed, in part, to support Computational Grid research. While we focus on network performance in this paper, our system also works for available CPU and memory, and will accept readings from any other NWS sensors that are configured.

In the next section, we briefly describe the infrastructures from which the NwsAlarm was developed we detail the implementation of the NwsAlarm itself. In Section 3 we provide the experimental methodology used for this study. Section 5, 6, and 7 contain our empirical results, the related work, and our conclusions, respectively.

2 NwsAlarm Implementation

The NwsAlarm monitors performance levels, predicts future performance levels, displays the data graphically, and reports "performance faults" (occasions when predicted performance does not match expected levels) to administrators when they occur. To enable this functionality, the NwsAlarm extends the Network Weather Service [24] and the JavaNws [15].

2.1 The Network Weather Service

The Network Weather Service (NWS) is a distributed, generalized system for producing short-term performance forecasts based on historical performance measurement. The goal of the system is to characterize and forecast dynamically the performance deliverable to the application level from a set of network and computational resources. Such forecasts have been used successfully to implement dynamic scheduling agents for Grid applications [21, 4], and to choose between replicated web pages [2].

The NWS takes periodic measurements of the currently deliverable performance (in the presence of contention) from each resource and uses numerical models to generate forecasts of future performance levels dynamically. Forecast data is continually updated and distributed so that resource allocation and scheduling decisions may be made at run time based on **expected** levels of deliverable performance. The NWS forecasts provide difficult to obtain, statistical estimates of available service quality from each resource of interest, as well as the degree to which those estimates are likely to be accurate [23].

Since the NWS measures and forecasts performance deliverable to the application level, it is implemented using the same communication and computation mechanisms that applications use resulting in forecasts that accurately reflect the true performance an application can expect to obtain. Separate implementations of the NWS have been developed using sockets and for the Globus/Nexus [11] and Legion [12] metacomputing environments, each of which provides a software infrastructure that supports high-performance distributed and parallel computing.

2.2 The JavaNws

The JavaNws is a Java implementation of a subset of the NWS toolkit that provides measurement and prediction for network resources. The JavaNws measures the TCP/IP socket performance (bandwidth and round-trip time) between the user's desktop and the web server from which the JavaNws applet was downloaded. Predicted performance is computed from the measurements by the applet and both are visualized in real-time. The JavaNws enables users to circumvent the need to explicitly install and maintain an NWS network monitoring process and any special-purpose visualization software; NWS measurement and forecast data are delivered to the users web browser in real-time. Previous work with the NWS and Java-based applications indicates that basing transfer decisions on NWS forecast data can dramatically improve execution performance [9, 22].

2.3 The NwsAlarm

Like JavaNws, the NwsAlarm is written in Java and requires no installation or special privileges for execution and access to the vast amount of performance data collected by the NWS. A Java-language implementation is important since it enables security, portability, and instant invocation on the user's desktop using the applet execution model. The NwsAlarm enables visualization of performance for *any* resource currently monitored by the NWS (CPU, memory, networking) as well as the network performance between the web server and the desktop. In addition, administrators can use the NwsAlarm to set performance thresholds and to send alarms when expected performance levels degrade.

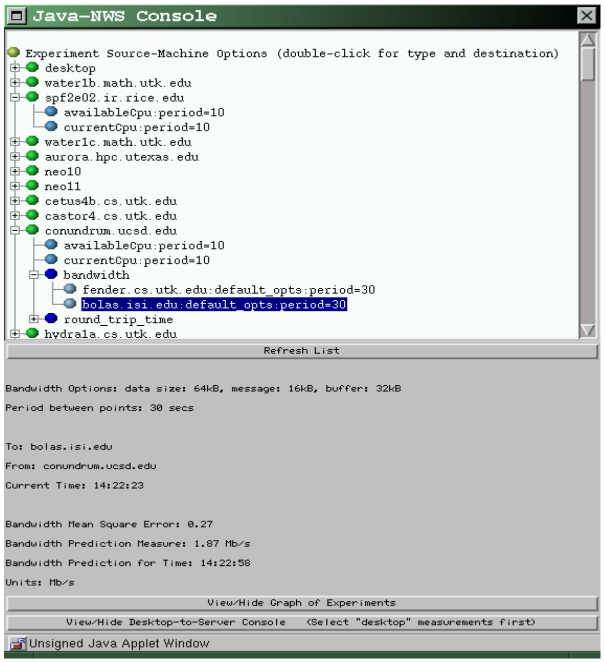
The NwsAlarm consists of two parts: The applet that executes on the user's desktop and the server program located at the machine from which the applet is downloaded. Upon NwsAlarm invocation, the server program, started as a background process, requests and acquires the list of available hosts from an NWS name server. This list is transfered to the NwsAlarm applet on the user's desktop and is displayed as a tree of choices as shown in Figure 1. A user can select any host, any available resource (CPU, memory, network performance, etc.) associated with that host, and the destination host if the network resource is chosen. The list can be refreshed by the user at any time to acquire a new list updated with any, dynamically added, resources.

The selection made by the user is communicated by the NwsAlarm applet to the server program which obtains and returns the associated measurement from the NWS name server. If the selection is the desktop, then a series of experiments are performed to measure the connectivity between the desktop and server, just as in JavaNws [15]. For any selection, the resulting measurement is given to the NwsAlarm forecasters (a Java implementation of the NWS forecasters) to predict future performance of the resource. The measurements and predictions are then displayed graphically for the user as in Figure 2.

2.4 NwsAlarm: Degradation Detection

The NwsAlarm also provides users with a mechanism to alert administrators of degradation in performance. The administrator sets performance thresholds and registers his/her email address with the NwsAlarm. When performance drops below a threshold, the administrator is notified via email.

Two types of performance thresholds are available in the NwsAlarm. The first is a performance value that must be maintained; if a measurement is less than the given value, it is considered a degradation. The administrator can indicate the number of such events that must occur before he/she is alerted. The second type of threshold is the number of communication errors between the desktop and the server and the server and the NWS name server. If the number of errors exceeds the given threshold the administrator will be notified. Such errors occur if either the server from which the NwsAlarm applet was downloaded or the name server becomes unavailable due to network partition, other catastrophic failure, or transfer timeout. For the remainder of this paper, we focus on network resources, however, any resource the NWS can access can be monitored by NwsAlarm.


Java-NWS Console
Refresh List
View/Hide Graph of Experiments
View/Hide Desktop-to-Server Console (Select "desktop" measurements first)
Unsigned Java Applet Window

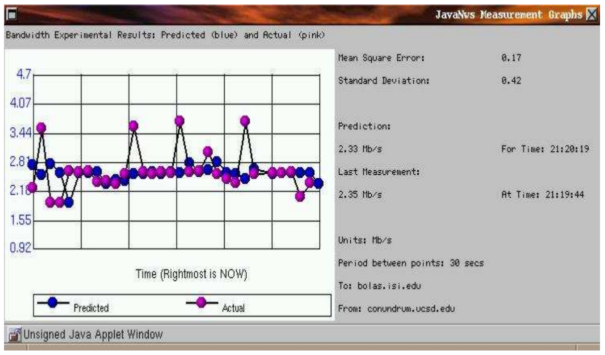
```
Experiment Source-Machine Options (double-click for type and destination)
+ desktop
+ water1b.math.utk.edu
- spf2e02.ir.rice.edu
    availableCpu:period=10
    currentCpu:period=10
+ water1c.math.utk.edu
+ aurora.hpc.utexas.edu
+ neo10
+ neo11
+ cetus4b.cs.utk.edu
+ castor4.cs.utk.edu
- conundrum.ucsd.edu
    availableCpu:period=10
    currentCpu:period=10
  - bandwidth
      fender.cs.utk.edu:default_opts:period=30
      bolas.isi.edu:default_opts:period=30
  + round_trip_time
+ hydra1a.cs.utk.edu
```

```
Bandwidth Options: data size: 64kB, message: 16kB, buffer: 32kB
Period between points: 30 secs

To: bolas.isi.edu
From: conundrum.ucsd.edu
Current Time: 14:22:23

Bandwidth Mean Square Error: 0.27
Bandwidth Prediction Measure: 1.87 Mb/s
Bandwidth Prediction for Time: 14:22:58
Units: Mb/s
```

Figure 1. The NwsAlarm console. The console provides users with a click-able, refreshable tree menu of machines for which NWS resource data is configured. For each machine, a list of resource types is given (network bandwidth and round-trip time, CPU availability and load, memory usage, etc).



```
Bandwidth Experimental Results: Predicted (blue) and Actual (pink)

Mean Square Error:            0.17
Standard Deviation:           0.42

Prediction:
2.33 Mb/s                     For Time: 21:20:19
Last Measurement:
2.35 Mb/s                     At Time: 21:19:44

Units: Mb/s
Period between points: 30 secs
To: bolas.isi.edu
From: conundrum.ucsd.edu
```

Figure 2. NwsAlarm performance visualization. When a user makes a resource selection from the console, the measurement data (light-colored points) and predicted performance (dark points) are displayed. The y-axis indicates measurement values in the units associated with the resource type (here the resource is bandwidth and the units are Mb/s) and x-axis is time. Summary data is provided to the right of the graph.

3 Experimental Methodology

For the results described in this paper, we gathered data between a machine at the University of Tennessee (UT) and the University of California, San Diego (UCSD). The predominant network technology between these two hosts is Abilene [1]. Abilene is an advanced backbone network that supports the development and deployment of the new applications being developed within the Internet2 community. Abilene connects regional network aggregation points, called gigaPoPs, to support the work of Internet2 universities as they develop advanced Internet applications. It is characterized by high-bandwidths and relatively high round-trip times induced by large geographic distance. When Abilene fails or routing tables change, the link can degrade to the use of the common carrier between the hosts.

Measurements of link performance were made from May 7th, 1999 through September 25th, 2000. We used the NWS to collect the data [1] The measurements were made at approximately 30 second intervals. We collected both bandwidth and round-trip time values.

In addition to these measurements, we logged traceroute [14] data between the two machines at 1-hour intervals. Traceroute is a UNIX utility that uses the IP protocol to provide a trace of the network route between two machines. This data is used in our results section (Section 5) to confirm that performance faults detected by NwsAlarm correspond to incorrectly initialized routing tables.

4 Degradation Discovery Using Prediction

Since end-to-end network performance is highly variable from one moment to the next, we must ensure that the NwsAlarm is able to distinguish between random fluctuations and true performance trends so that alarms are raised accurately. Network performance, in particular, is highly variable from one moment to the next. If an alarm were triggered every time a low performance measurement occurs, many false alarms will be generated. To enable accurate alarm detection, the NwsAlarm compares "predicted" performance data thresholds set by the NwsAlarm user. The thresholds represent the performance expectation that the user has for the monitored resources. The forecasts represent the expected performance for the resource based on past history. The role of forecasting in this setting is to remove the random noise from the measurement history to reveal the "true" performance signal. A fault is defined to be when this true signal falls outside the specified range.

The use of predicted values is the key difference between this system and all others. Prediction enables the NwsAlarm to identify events that are imperceptible if the trace data is graphed and observed visually. In this section we provide two cases, the first in which fault occurrences are obvious and a second in which they are not, to motivate the function of the NwsAlarm.

A common event that causes disruptions in network performance is a routing table change. Often, it is difficult for local network administrators and backbone service providers to keep routing tables synchronized. When the routing tables are incorrectly set, connectivity may be disrupted entirely. This type of fault is easy for local administrators to detect since users will begin calling the hapless administrators almost immediately to discuss the network outage and to constructively suggest possible courses of action. However, it is also possible for the routing tables to be set incorrectly causing network traffic to take a functioning but heavily congested path. In this case, connectivity quality is degraded, but since users expect a certain amount of performance variation (which is difficult to quantify) they may not report such problems to the overworked networking staff.

An example of this second type of routing table problem is illustrated in the following output generated by the traceroute utility.

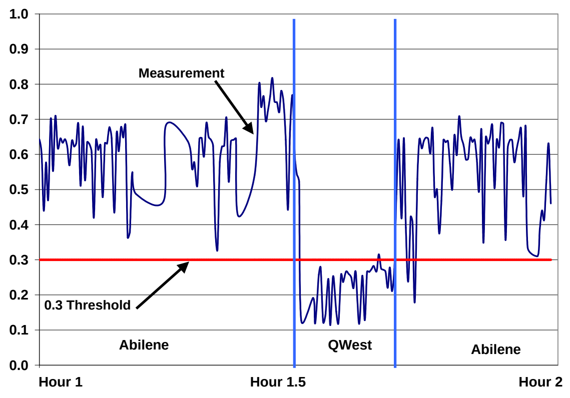
```
Wed May 10 00:30:09 EST 2000
 1  R5HM01V277.NS.UTK.EDU (128.169.92.1)  0.937 ms  0.745 ms  0.804 ms
 2  192.168.101.3 (192.168.101.3)  2.296 ms  1.366 ms  1.588 ms
 3  UTK-GATECH.NS.UTK.EDU (128.169.50.246)  33.318 ms  33.190 ms  32.945 ms
 4  atla.abilene.sox.net (199.77.193.2)  33.475 ms  33.017 ms  34.511 ms
 5  hous-atla.abilene.ucaid.edu (198.32.8.33)  46.454 ms  45.876 ms  45.739 ms
 6  losa-hous.abilene.ucaid.edu (198.32.8.21)  77.904 ms  77.352 ms  77.955 ms
 7  USC--abilene.ATM.calren2.net (198.32.248.85)  78.006 ms  78.311 ms  77.959 ms
 8  UCSD--USC.POS.calren2.net (198.32.248.34)  81.943 ms  81.173 ms  81.286 ms
 9  sdsc2--UCSD.ATM.calren2.net (198.32.248.65)  83.004 ms  87.349 ms  93.498 ms
10  cse-rs.ucsd.edu (132.239.254.45)  83.513 ms  83.264 ms  83.408 ms
11  conundrum.ucsd.edu (132.239.55.213)  91.528 ms *  91.058 ms

Wed May 10 01:30:17 EST 2000
 1  R5HM01V277.NS.UTK.EDU (128.169.92.1)  0.783 ms  0.801 ms  0.681 ms
 2  192.168.101.3 (192.168.101.3)  1.612 ms  1.794 ms  1.471 ms
 3  R7SM99.NS.UTK.EDU (128.169.54.8)  1.988 ms  2.281 ms  1.977 ms
 4  205.171.49.165 (205.171.49.165)  20.043 ms  20.200 ms  20.449 ms
 5  atl-core-02.inet.qwest.net (205.171.21.45)  20.042 ms  20.600 ms  20.267 ms
 6  wdc-core-03.inet.qwest.net (205.171.5.241)  30.911 ms  31.092 ms  30.964 ms
 7  wdc-core-01.inet.qwest.net (205.171.24.10)  30.988 ms  31.422 ms  30.979 ms
 8  chi-core-02.inet.qwest.net (205.171.5.227)  54.913 ms  56.092 ms  55.025 ms
 9  chi-core-03.inet.qwest.net (205.171.20.30)  55.234 ms  55.718 ms  55.063 ms
10  chi-brdr-01.inet.qwest.net (205.171.20.66)  55.479 ms  55.740 ms  55.463 ms
11  s2-0-1.chi-bb1.cerf.net (134.24.103.153)  71.576 ms  71.121 ms  72.423 ms
. . .
17  pos1-0-0-155M.san-bb1.cerf.net (134.24.29.190) 143.320ms 141.121ms 140.459ms
18  sdsc-gw.san-bb1.cerf.net (134.24.12.26)  189.463 ms  367.079 ms  149.953 ms
19  bigmama.ucsd.edu (192.12.207.5)  122.431 ms  130.265 ms  121.961 ms
20  cse-rs.ucsd.edu (132.239.254.45)  105.015 ms  112.668 ms  103.970 ms
21  conundrum.ucsd.edu (132.239.55.213)  104.508 ms *  133.320 ms
```

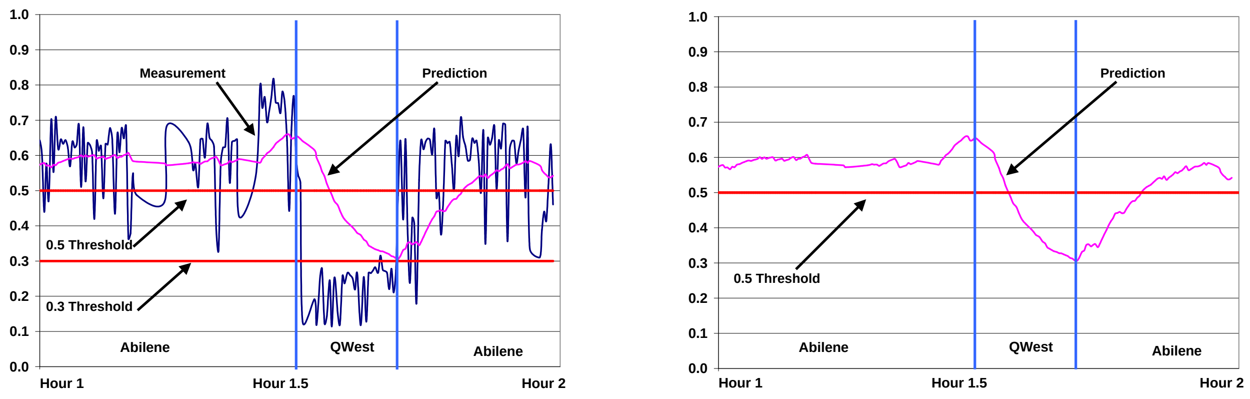
This trace was generated by a pair of systems that are intended to route packets between themselves over Abilene at all times. Abilene provides more consistent performance, less contention, and, as can be seen from the output, fewer hops in many cases. A loss of Abilene service can impact the end-to-end performance experienced by users. If an administrator is aware of the loss of service he/she may be able correct the problem before users are inconvenienced. The NwsAlarm is designed to be used in this setting to alert administrators and users impacted by a change in network performance.

Figure 3(a) shows a two hour trace in which a routing table change occurs. Bandwidth (in Mb/s) was measured

[1]Our prior work shows that there is little, if any, significant difference between measurements gathered using Java and those generated by a C program [16].



(a)



(b)

Figure 3. Sub-trace from 5-month Abilene bandwidth trace data. The data is a 2-hour trace Friday, June 9 starting at approximately at 5:00am. (a) contains measurement values only, (b) contains measurement and predicted values, and (c) contains just the predicted values. Two vertical lines indicating a routing table change in the associated traceroute data from the same period are also included. Horizontal threshold lines indicated the bandwidth below which the routing table change can be detected. In this case, it is obvious from the measurement data when the change takes place. However, we show that this is rarely the case. Using prediction, with the NwsAlarm (predicted values), fewer false alarms are raised and a tighter threshold can be set. False alarms occur when the value drops below the threshold while the Abilene link is in use in this scenario.

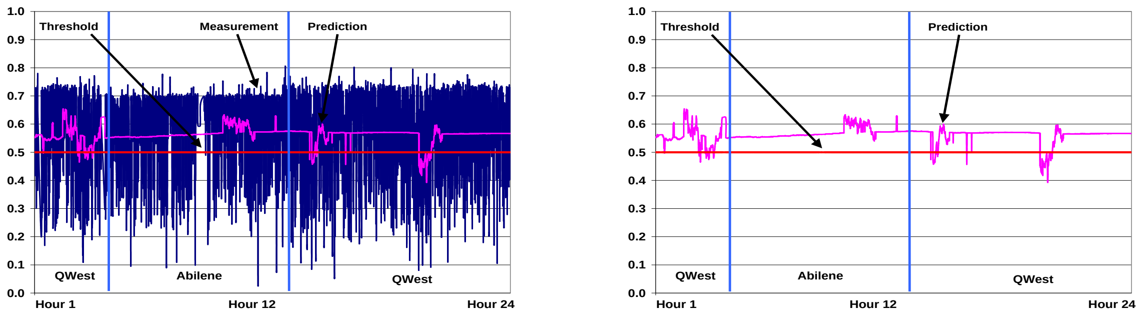


Figure 4. Sub-trace from 5-month Abilene bandwidth trace data. The pair is a 24-hour trace Friday, June 1 starting at approximately midnight. The left graph contains measurement and predicted values, and the right only the predicted values. Two vertical lines indicating a routing table change in the associated traceroute data from the same period are also included. A horizontal threshold line indicate the bandwidth below which the routing table change should be detected. It is difficult using a human eye and measurement values to determine when the change occurs. The NwsAlarm using prediction, however, can effectively and accurately raise alarms only when the common carrier (QWest) is in use.

between two hosts, one at the University of Tennessee, Knoxville, the other at the University of California, San Diego. The y-axis for this and all other graphs in this paper is time and the x-axis is bandwidth in Mb/s. (a) contains measurement values only, (b) contains measurement (dark) and predicted (light) values, and (c) contains predicted values only, for clarity. Two vertical lines indicating a routing table change in the associated traceroute data from the same period are also included. Horizontal threshold lines indicated the bandwidth below which the routing table change can be detected. In this case, it is obvious from the measurement data when the change takes place.

The measurement data alone indicates that approximately midway through the trace there is a loss in performance on the link. Traceroute data collected for the same period confirms that the routing table changed from Abilene to common carrier (QWest in this case). The routing table changes are indicated by two vertical lines within the graph with the textual link type (Abilene or common carrier (QWest)) given in each section of the resulting divided graph. Using a threshold of 0.3Mb/s (horizontal red line on the graph) we are able to visually identify the occurrence of the event. That is, when the bandwidth measurements fall below 0.3Mb/s, they indicate, in this scenario, that a routing table change occurred. The NwsAlarm, using predicted values also discovers the routing table change and is able to do so using an even tighter threshold of 0.5Mb/s.

However, consider the data shown in Figure 4. This 24-hour sub-trace of bandwidth data is between the same pair of hosts during different time period. In the left graph, both measurements (dark) and predicted (light) values are shown. The right graph contains only the predicted values. In this example, it is very difficult to detect visually when the routing tables were correctly initialized, and when they were set erroneously using only measurement data.

The NwsAlarm (prediction) values, however, effectively and accurately indicate when changes occur. Accuracy is determined by the number of alarms that are falsely sent; in this case, when a value is below threshold and the Abilene network is in use. Raising many false alarms makes it difficult for administrators to efficiently distinguish when problems actually occur. The right graph exemplifies the accuracy of the NwsAlarm: the predicted values only fall below threshold when the common carrier is in use. In particular, the 0.3Mb/s threshold value that worked for measurement data in Figure 3 is ineffective as a threshold in this latter example. The reverse is not true, however. In both cases, using a 0.5Mb/s threshold and the NWS forecasts (instead of the measurements) accurately detects the routing faults.

It should be pointed out that traceroute data alone can be used to discover such faults. There are several advantages to using end-to-end measurements taken at the application level over lower-level mechanisms such as traceroute. First traceroute is a setuid program which makes it inappropriate for many security settings. Indeed, access to low-level monitoring features is often carefully controlled and is difficult to manage. Application-level performance, however, must be measurable or applications will not function. More importantly, however, the NwsAlarm methodology is general. In the case of network faults, we can call upon traceroute to verify the efficacy of the system, but traceroute itself might prove a better choice in some settings. For resources without analogous low-level measurement utilities (i.e. non-paged real memory on Unix systems) NwsAlarm is also applicable (although its accuracy is more difficult to verify). It is our conjecture that since the performance fault detection methodology we describe in this paper is effective in cases where it can be verified, it will also be effective in the cases where it can't.

5 NwsAlarm Validation

The NwsAlarm is able to identify accurately events that cause changes in expected performance levels. It does this by monitoring changes in forecasted values as opposed to raw data measurements. In this section, we verify this accuracy by comparing the number of false alarms that are raised when measurement data alone is compared against performance thresholds, and when forecast data is used instead.

In our first example we monitored the bandwidth on an ISDN link between between the University of Tennessee and the home of a local user in Knoxville, Tennessee. This data is displayed in Figure 5. We show both the measurement and the forecasted data taken at 10 second intervals and collected over a period of 5 hours. The left graph shows both the measurement (dark) and forecasted (light) data together. The right graph contains only the forecasted values (from the left graph) for clarity. In addition, each graph contains an NwsAlarm threshold line (in red for colored version) at 0.4Mb/s. This indicates an arbitrary threshold set by an administrator. For the measurement data case, a measured value below this threshold causes an alarm to be triggered. Similarly, for the forecast case, an alarm is triggered when the forecast value falls below 0.4Mb/s.

During the measurement period, four large transfers were made causing a reduction in available bandwidth. In addition, the network failed in the 3rd and 4th hours (as indicated on each graph). The NwsAlarm was used to indicate when failures or low bandwidth availability occurred. The total number of alarms that should have be sent in this scenario is 136. Using measurements to evaluate threshold limits cause 32 alarms to be falsely sent; using the NwsAlarm predicted values, only 2 false alarms were sent. Unlike our other examples, the NwsAlarm is used in this scenario to distinguish events that have no other low-level measurement facility, namely, the loss of bandwidth due to contention. If the link was intended to be free of other traffic, the alarms would have been indicative of either a routing problem (i.e. other traffic was erroneously being routed over the link) or a security breech.

The NwsAlarm can also be used to alert administrators to loss in Abilene service, as described in the previous section. If Abilene becomes unavailable, either due to catastrophic failure or routing table misconfiguration, users, expecting the quality of service Abilene provides, can be alerted using the NwsAlarm.

To empirically evaluate the accuracy of the NwsAlarm in this situation, we present four different traces of Abilene data from the link between the University of Tennessee, Knoxville, and the University of California, San Diego (UCSD) in Figures 6. Bandwidth values are shown in Mb/s (y-axis) at approximately 30 second intervals. The length of the traces varies for each pair of graphs, but is given along the x-axis. Two graphs are shown for each trace period.

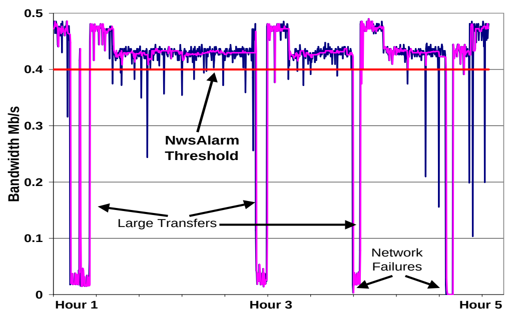
The left graph of each pair again shows the measurement (dark) and predicted (light) data. The right graphs help to distinguish the two series by providing only the predicted data. A NwsAlarm threshold value of 0.5 Mb/s was used in this study and is indicated by the horizontal (red) line on each graph. Each time a value is below the threshold line, it indicates that an alarm has been sent to an administrator of the link. To verify that the NwsAlarm accurately determines routing table changes, we have imposed two vertical lines on each graph indicating when such events occurred in our traceroute data logged over the same period.

The goal of the NwsAlarm in this scenario is to send an alarm only when the Abilene service degrades to common carrier. Common carrier is indicated on the graphs as "QWest". Once Abilene service has resumed, no further alarms should be sent. Obviously, if raw measurements are used to determine when to send an alarm, many false alarms occur. Using the NwsAlarm results in far fewer false alarms. These counts are shown in Table 7. On average, 92% fewer false alarms are sent using NwsAlarm with NWS-predicted values.

6 Related Work

Much research has gone into the measurement and prediction of resource performance. For network performance specifically, the authors in [6] describe characteristics and theoretical predictability but do no on-line analysis as is provided by the NwsAlarm. Carter et.al. perform dynamic probing of networks with bprobe [5], and use basic forecasting techniques to predict short term performance. The prediction utilities of NWS-based tools are more sophisticated than those used in bprobe, although it is possible that even simple forecasting techniques will be effective. Bprobe, however, is not designed to detect and signal performance faults in the way NwsAlarm does.

In [7], Downey describes the effectiveness and limitations of using pathchar [13], a tool for measurement of bandwidth, round-trip time, average queue length, and loss



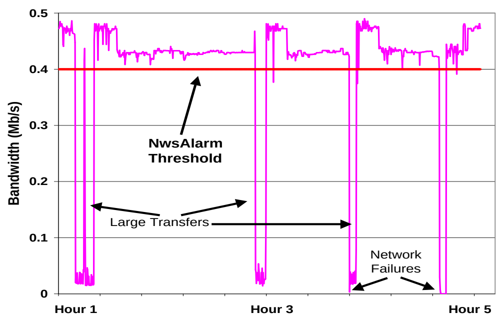


Figure 5. 5-hour ISDN bandwidth trace data. The left graph shows both the measurements (dark-colored series) and NWS predicted values (light-colored series) taken at 30 second intervals. The right graph shows only the predicted values for clarity. The x-axis is time and the y-axis is bandwidth in Mb/s. Three large transfers occurred during the trace and two network failures. The NwsAlarm is used to identify these events. A horizontal line is shown at 0.4 Mb/s, this value is the NwsAlarm threshold value. Alarms are sent when predicted values fall below this threshold. Using actual measurements to send alarms cause inaccurate, false alarms.

rate, to predict Internet link characteristics. Pathchar is implemented using ICMP echo and/or port-unreachable packets and require super-user privileges. While the tool and Downey's analysis of its use are exceptional, he points out that in many wide area settings (such as Abilene) pathchar may yield erroneous readings. In particular, the predictions it makes for application-deliverable bandwidth performance can be substantially in error. Since the NWS uses end-to-end measurements, it does not suffer from these inaccuracies. An advantage of pathchar, however, is that it does not require "hard collaboration", but the NWS does.

Dinda et.al. articulate the predictability of CPU load in [18]. The NwsAlarm can also predict CPU load and availability using the same forecasters as those used for network performance prediction. NWS-based tools differ in that the forecasters are computationally less intensive while offering similar accuracy. In [17], the authors use raw transfer time and CPU load of mirrored World Wide Web servers to determine which server sites should be selected at any given time. This work differs from the NwsAlarm for the same reasons noted above. The NwsAlarm can be used to visualize raw and predicted data between a user's desktop and any server, mirrored or otherwise, at which the NwsAlarm is installed. This way, users can dynamically determine which server (if mirrored) to use and change his/her decision when alerted by the NwsAlarm.

7 Conclusion

Knowledge of end-to-end performance deliverable to an application enables users to make informed decisions about the use of available resources. Tools are needed to aid users by measuring and visualizing available performance and by alerting users when expected performance degrades. In this paper we present one such utility, the NwsAlarm, which displays this available performance (CPU, memory, or network performance), reports short-term performance forecasts, and alerts users to unexpected degradations. Administrators of Grid-computing infrastructures can use the latter to maintain expected performance levels or to inform users when they are unable to do so.

We illustrate the utility of the system by demonstrating how it is able to detect erroneous routing table configurations by dynamically analyzing end-to-end performance measurements. By comparing forecasts to user-specified thresholds, the NwsAlarm accurately identifies periods of time during which the routing tables are correctly configured between a pair of hosts, and periods when they are misconfigured causing a performance degradation. In addition, the system correctly detects link contention and of course, link outage.

To investigate the efficacy of forecasting, we compare the number of false performance alarms that are generated when raw measurement data is used as a trigger, and when NWS forecasts are used to trigger and alarm. Since the forecasting techniques effectively filter random fluctuations from the performance traces, the NwsAlarm, on average

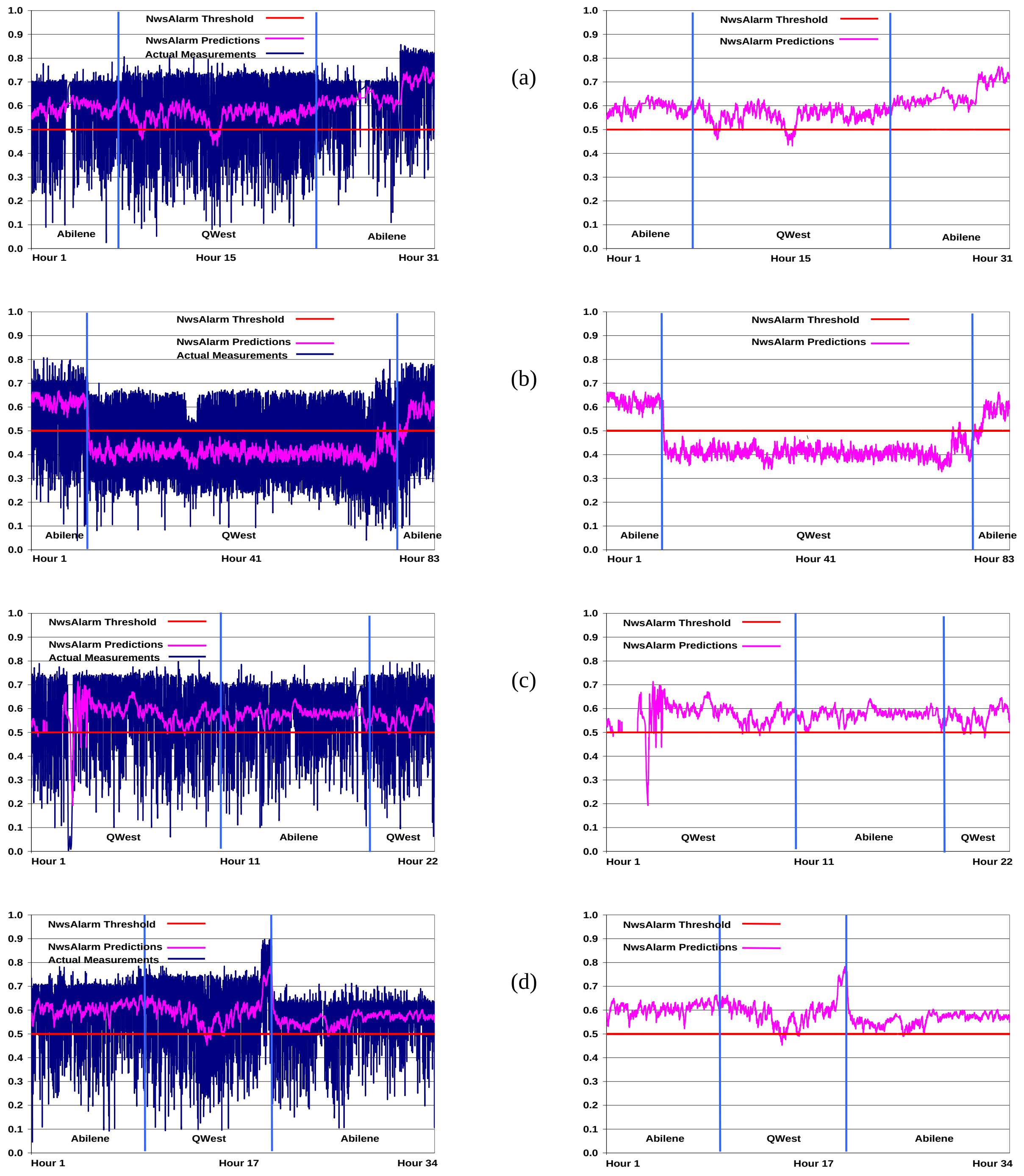


Figure 6. Sub-traces from 5-month Abilene bandwidth (Mb/s) trace data. Pair (a) is from a 31-hour trace starting June 1 at approximately 5:30am; (b) from a 83-hour trace starting August 11 at approximately 12:38pm. Pair (c) is from a 22-hour trace starting May 24 at approximately 3:21am; (d) from a 83-hour trace starting May 26 at approximately 3:55am. The left graph shows both the measurements (dark-colored series) and NWS predicted values (light-colored series) taken at 30 second intervals; the right graphs shows predictions only. The NwsAlarm was used to determine when Abilene connectivity degraded to common carrier (QWest) and when service resumed. A horizontal line is shown at 0.5 Mb/s, this value is the NwsAlarm threshold value. Alarms are sent when predicted values fall below this threshold. Using actual measurements to send alarms cause inaccurate, false alarms as indicated by the data. NwsAlarm accurately indicates loss and restoration of Abilene service.

Table 1. Comparison of false alarm count using NWS-predicted values and raw measurement data in the NwsAlarm. Use of predicted values enable more accurate error detection.

Sub-trace	Predictions False Alarms	Raw Measurements False Alarms
Figure 6a	0	298
Figure 6b	112	477
Figure 7a	0	250
Figure 7b	13	494
Avg	31	380

raises 92% fewer alarms than if raw bandwidth measurements are used to detect performance changes.

8 Acknowledgements

Thanks to the referees and others who provided valuable feedback on this work.

References

[1] Abilene. `http://www.ucaid.edu/abilene`.

[2] D. Andresen and T. McCune. Towards a hierarchical scheduling system for distributed www server clusters. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, July 1998.

[3] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, L. J. Dennis Gannon, K. Kennedy, C. Kesselman, D. Reed, L. Torczon, , and R. Wolski. The grads project: Software support for high-level grid application development. Technical Report Rice COMPTR00-355, Rice University, February 2000.

[4] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application level scheduling on distributed heterogeneous networks. In *Proceedings of Supercomputing 1996*, 1996.

[5] R. Carter and M. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical Report TR-96-007, Boston University, 1996. `http://cs-www.bu.edu/students/grads/carter/papers.html`.

[6] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In *Proceedings of the 1996 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, 1996.

[7] A. Downey. Using pathchar to estimate internet link characteristics. In *SIGCOMM '99*, 1999.

[8] Entropia. `http://www.entropia.com`.

[9] M. Faerman, A. Su, R. Wolski, and F. Berman. Adaptive performance prediction for distributed data-intensive applications. In *Proceedings of SC99*, November 1999.

[10] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.

[11] I. Foster, C. Kesselman, and S. Tuecke. The nexus approach to integrating multithreading and communication. *Journal of Parallel and Distributed Computing*, 1997.

[12] A. S. Grimshaw, W. A. Wulf, J. C. French, A. C. Weaver, and P. F. Reynolds. Legion: The next logical step towrd a nationwide virtual computer. Technical Report CS-94-21, University of Virginia, 1994.

[13] V. Jacobson. Pathchar: A tool to infer characteristics of internet paths. `http://www.caida.org/tools/utilities/others/pathchar`.

[14] V. Jacobson. Traceroute: A tool for printing the route packets take to a network host. available from `ftp.ee.lbl.gov/nrg.html`.

[15] C. Krintz and R. Wolski. JavaNws: The Network Weather Service for the desktop. In *Proceedings of ACM JavaGrande 2000*, June 2000.

[16] C. Krintz and R. Wolski. Using JavaNws to Compare C and Java TCP-Socket Performance. In *The Journal of Concurrency and Computation: Practice and Experience*, dec 2000.

[17] A. Myers, P. Dinda, and H. Zhang. Performance characteristics of mirror servers on the internet. In *Proceedings of Infocom '99*, March 1999.

[18] D. O. P. Dinda. An evaluation of linear models for host load prediction. In *Proceedings of the Eighth IEEE International Symp osium on High Performance Distributed Computing HPDC8*, August 1999.

[19] ParaBon. `http://www.parabon.com`.

[20] Seti At Home. `http://setiathome.ssl.berkeley.edu`.

[21] N. Spring and R. Wolski. Application level scheduling: Gene sequence library comparison. In *Proceedings of ACM International Conference on Supercomputing 1998*, July 1998.

[22] A. Su, F. Berman, R. Wolski, and M. Strout. Using AppLeS to schedule a distributed visualization tool on the computational grid. *International Journal of High Performance Computing Applications*, 13, 1999.

[23] R. Wolski. Dynamically forecasting network performance to support dynamic scheduling using the network weather service. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, August 1997.

[24] R. Wolski. Dynamically forecasting network performance using the network weather service. *Cluster Computing*, 1998. also available from `http://www.cs.utk.edu/~rich/publications/`.

[25] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 1999. `http://www.cs.utk.edu/~rich/publications/nws-arch.ps.gz`.