

# Enhancing NWS for use in an SNMP Managed Internetwork

Robert E. Busby, Jr.  
AT&T Network Operations  
12976 Hollenberg Drive  
Bridgeton, MO 63044  
bobusby@att.com

Mitchell L. Neilsen, Daniel Andresen  
Dept. of Computing and Information Sciences  
Kansas State University  
234 Nichols Hall  
Manhattan, KS 66506  
{neilsen, dan}@cis.ksu.edu

## Abstract

*The Network Weather Service (NWS) is a distributed resource monitoring and utilization prediction system, employed as an aid to scheduling jobs in a metacomputing environment [9, 10]. We have enhanced the Network Weather Service by providing a broader metric set, forecast trend visualization, and asynchronous notification. Three new components have been added to the NWS system. The first of these is an SNMP Sensor that may be used to collect data being tracked by SNMP agents. Secondly, a Notification Process provides asynchronous events to interested applications when metrics tracked by NWS are expected to exceed established normal ranges. Finally, an extended SNMP agent daemon has been developed to provide SNMP-enabled applications with the ability to retrieve information from NWS without the need for custom programming. The new modules help to position NWS as an integral component for performing QoS monitoring and fault prediction in an SNMP-based fault management architecture. We also demonstrate that the additional functionality adds a minimal amount of overhead to the system, and reported bandwidth utilization correlates closely with results from the invasive NWS bandwidth sensor.*

## 1 Introduction

The task of network management has many facets. Most often, engineers break the task down into the five “Specific Management Functional Areas” defined by the International Standards Organization (ISO). The first area defined by ISO is “Fault Management” - the detection and correction of network faults. Fault Management can be classified as either proactive or reactive [7].

The reactive model of fault management relies heavily upon asynchronous events sent by network components to notify the management server(s) of problems. This can only be taken so far: if a device fails, it cannot send an event indicating that it failed. In a reactive model, faults cannot be avoided because they are never detected before they occur. Once detected, it is too late to avoid impacting the users of the network.

For these reasons, virtually every network management application relies, at least to some degree, on the proactive model. Implementing this model of fault management involves actively contacting the network components of interest, and verifying their continued functionality. This process is known as polling, and may be as simple as sending an ICMP “ping.” Often this is not sufficient.

Polling can be effective in discovering equipment that has become non-functional. However, in its simplest form, polling cannot detect the so-called “soft failures,” where Quality of Service (QoS) falls short of expectations due to resource over utilization. The Simple Network Management Protocol (SNMP) is often employed to retrieve operating data from network elements [2]. SNMP defines many variables that network components may elect to report. Some of these variables can be used to make inferences about quality of service.

Most network management architectures have facilities to track SNMP variables, and allow its administrators to define the range in which these variables are considered normal. If the value of a retrieved variable falls outside of this range, a QoS problem may be indicated, and an alert might be raised. The challenge in setting these thresholds is to avoid entering an alert state prematurely while still recognizing the condition before it can severely impact the network’s operation. If a threshold violation routinely occurs when no problem is present, network management staff will quickly

learn to ignore the alert, which may eventually lead to an unrecognized problem in the network. On the other hand, if the alert occurs after the users have been impacted, the threshold fails to be a useful, proactive tool.

Ideally, the threshold should be set to a value known to indicate a problem. Then, an alert is raised when trend analysis indicates that the variable will cross this threshold in the foreseeable future unless corrective action is taken. Transient spikes in the variable's value should not easily skew the value of this forecast.

The Network Weather Service (NWS) [9, 10] is a distributed resource monitoring system, originally developed by Rich Wolski, Neil Spring, and Jim Hayes of the University of California, San Diego. It is employed as an aid to scheduling jobs in a metacomputing environment, such as the Globus project [3]. The system tracks utilization of key system and network resources, and provides short-term predictions as to the future utilization of these resources. A scheduler process interested in a particular, critical resource can query NWS for a forecast of the utilization of that resource, and so select a host system that will likely provide the resource in sufficient quantities for the job to be scheduled.

While originally designed as a metacomputing scheduling aid, NWS has broader applications, particularly in the field of proactive system and network management. With currently developed applications making stronger and stronger demands on the resources of distributed systems, there is a need for these applications to recognize and react to adverse changes in the "Network Weather Forecast" in such a manner that the services they provide are, at worst, gracefully degraded. With our enhancements, Network Weather Service is a tool that can collect SNMP data, perform trend analysis on these data to provide a forecast as to what the future value of these data might be, and alert the network management staff should this forecast exceed a specified threshold.

The NWS uses intrusive sensors to determine available network bandwidth through periodically transferring a file and measuring the achieved throughput. Various other mechanisms for determining network bandwidth exist. For example, the SWEB++ project inferred available bandwidth through measuring network activity in a LAN [1].

With the Remos system [4, 5], Lowekamp, Gross, et. al. provide a resource monitoring and prediction system similar to NWS. Remos utilizes SNMP queries to provide summarized data concerning high-level network metrics such as available bandwidth and latency. Unlike NWS, the Remos system also uses SNMP to infer network topologies, but it does not provide the

other capabilities discussed in this paper such as asynchronous notification or expanding the metric set.

This project attempts to enhance the Network Weather Service in a number of ways:

- **Broader Metric Set:** While NWS already provides tracking for a few very important metrics, there are many more that are currently being tracked by operating systems and SNMP agents that may be of interest to client applications. For instance, network configuration, utilization, status, and error conditions are tracked by embedded SNMP agents in most networking equipment, and SNMP agents track system resource utilization. A standards-based approach is needed to provide a method to harvest these and future metrics without the need for custom programming.
- **Forecast and Trend Visualization:** NWS clients consist not only of computer processes, but also human network managers and system operators. These humans need to visualize the metrics and forecasts provided by NWS. With the proper tools, human network managers can recognize and react to trends in resource utilization far more effectively than even the most sophisticated software. NWS should allow users to utilize the visualization tools that are already in the network management arsenal, as well as allow for the easy utilization of new tools that may be released in the future.
- **Asynchronous Notification:** Requiring the application to constantly monitor a resource may be too much of a burden. A more convenient approach is to provide a persistent service that client programs may utilize to monitor the resource externally, and provide an interrupt to the client when a notable change is recognized.

To achieve these goals, three new components have been added to the NWS system. The first of these is an SNMP Sensor that may be used to collect data being tracked by SNMP agents. This data is written to the NWS database, allowing predictions to be made about a much wider variety of network and system metrics. Secondly, a Notification Process provides asynchronous events to interested applications when metrics tracked by NWS are expected to exceed established normal ranges. Finally, an extended SNMP agent daemon has been developed to provide SNMP-enabled applications with the ability to retrieve information from NWS without the need for custom programming.



### 2.3. SNMP Agent

In addition to the goal of incorporating SNMP variables into the forecasting framework of NWS, this project also strives to provide NWS measurements and predictions to external, SNMP-enabled applications of the type found in most network management architectures. The extended SNMP agent provides this ability. An NWS MIB maps SNMP variables in an NWS private enterprise namespace to values within the NWS registry and experiment database. The extensible features of the UCD SNMP agent are utilized to provide the transport for delivery of these values via the SNMP protocol [8].

When the agent receives an NWS variable request, it creates a new instance of the NWS agent program. It is the responsibility of this new agent to interpret the OID and SNMP request type, acquire the requested information, and return it to the main agent. The SNMP daemon then transmits an SNMP GET RESPONSE PDU to the requesting process. The first time a particular category of variable is requested, the appropriate server is spawned (if it is not already running). The server then gathers as much information as possible by issuing an appropriate single command to an existing NWS utility program. Caching of variable data is used within the server to enhance efficiency.

## 3. Experimental Results

Two types of experiments were conducted utilizing the new NWS modules. The first explored the CPU resource requirements of the code. The second was performed to evaluate the correlation between variables retrieved in a passive manner by the SNMP Sensor with the measurements taken in an active manner by the original NWS network sensor.

### 3.1. System overhead

Each performance test was run on the same system: a 200 MHz Pentium with 64MB of memory, running Linux kernel 2.0.34 (the Redhat 5.1 release). The system was intentionally configured with the standard set of Internet daemon processes. Individual components were tested for impact on the system which was already running the NWS server. System resource utilization was not significantly impacted.

The primary test involved running all of the new components at once. These included the baseline NWS processes, the SNMP Sensor, the Notifier process servicing one registration each of an SNMP and a TCP

notification, and two instances of a web-based applet/server pair (the NWSGraph application) which plotted periodic samples of NWS measurements in two graphs being displayed by an instance of Netscape Communicator 4.5 on the system under test. The notifications registered were for CPU availability for the system under test with a poll period of 20 seconds. At no time during the test were any notifications actually delivered. The NWSGraph applets also requested CPU availability, one for each system in the network clique, and each with a poll period of 60 seconds. As shown in Figure 2, we saw a significant impact to system resource utilization when all components were running simultaneously. The overall average CPU availability drops to around 94%, with bursts down to around 70%. Even so, only two of the bursts exceed the maximum utilization of the baseline test. Even in combination, it appears that the CPU resource requirements of the components of this project are reasonable.

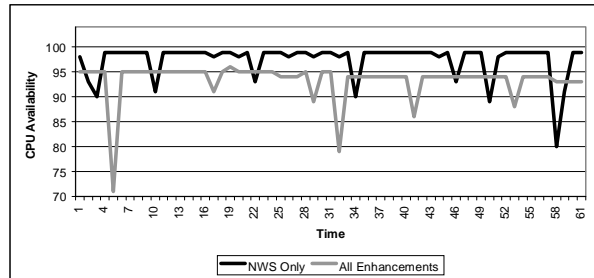


Figure 2. Comparison of NWS CPU Utilization

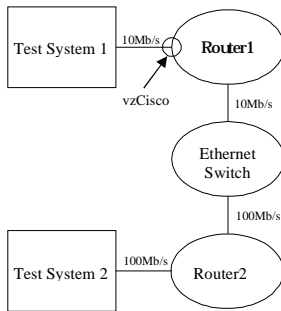
### 3.2. Functional Test

One benefit of the new SNMP NWS sensor is to provide a passive method for measuring variables that previously required an active resource experiment. One such variable is TCP network bandwidth. The traditional NWS network sensor measures bandwidth by sending significant amounts of data between systems. A certain amount of the resource being measured is consumed in the process of the measurement.

Our hypothesis was that total bandwidth between any two systems will be limited by the total utilization of the lowest-capacity network segment. SNMP agents in IP routers typically track, among many other things, the number of octets received and transmitted on each interface. With previous knowledge of which segment is the limiting segment, we should be able to utilize the SNMP sensor to retrieve these values. We would expect total octets routed on the limiting segment to be

inversely correlated with TCP bandwidth, as reported by the NWS network sensor.

Several tests were performed in an effort to prove this hypothesis. Each test was conducted on the same portion of the Kansas State University production IP network, shown in Figure 3. Test System 1 is the 200Mhz system mentioned previously; System 2 is a 143Mhz Sun Ultra 1 running Solaris 2.6. In each case, bandwidth was measured between the same two systems, where all segments were rated at 100 Mbps or more with the exception of one, which was rated at 10 Mbps. The SNMP sensor was configured to retrieve the number of octets sent and received by the router interface facing the lower capacity Ethernet segment. During the same period, the traditional NWS network sensor was utilized to measure actual observed bandwidth between the two systems.



**Figure 3. Test configuration for bandwidth measurement**

Correlating router interface utilization with TCP/IP throughput proved to be more difficult than we had originally anticipated. We found that, in order to observe a significant degree of correlation between the SNMP and the bandwidth experiments, a reasonably short poll interval is required, and the network must be driven close to saturation. One possible reason for this requirement might be rooted in the different nature of the two measurements. Every byte flowing through the router causes the appropriate octet count to increase, whether the network is saturated or not. The bandwidth measurement, on the other hand, is a momentary snapshot of the network conditions. The observed transfer times will vary to some degree depending upon how many other data transfers are occurring on the network at that particular instant in time. If the overall load is significantly below the level of network saturation, the bandwidth observations may be completely unaffected.

Our most successful functional test used a sensor poll period of 10 seconds. The test was carried out

during early morning hours to avoid daytime network traffic. The sensors were allowed to monitor the network in its idle state for approximately the first five minutes of the test. Next, a single 171 MB file was transferred from the first system in the sensor clique to the second system. This transfer lasted approximately five minutes. A five-minute idle period followed, and then the file was again transferred from the first system to the second system. The test was allowed to monitor the idle network for approximately 5 more minutes before completing.

	Minimum (Mb/sec)	Maximum (Mb/sec)	Mean (Mb/sec)	Correlation w/Bandwidth
Bandwidth	1.8369	6.6673	4.4404	
ifInOctets	0.0706	5.5836	1.9690	-0.5294
ifOutOctets	0.0038	0.7588	0.1365	-0.1574
ifInOctets + ifOutOctets	0.0839	5.7379	2.1055	-0.5313

**Table 1. Third Functional Test Results**

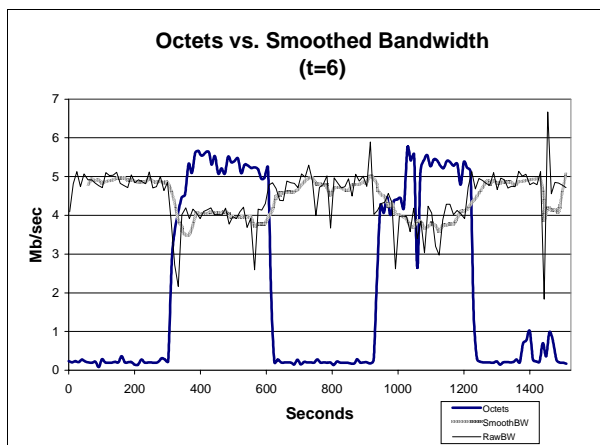
Statistical analysis of the final functional test showed a mild degree of inverse correlation for the comparison of total octets routed and bandwidth observed. Significant reductions in bandwidth can be seen in Figure 4. These reductions correspond with the time intervals of the two file transfers (which are obvious from the Octets graph). The bandwidth observations from this test showed significant variability, both during the idle periods and during the periods in which a file transfer was in progress. In an attempt to eliminate the effects of this variability, time smoothing was performed on the bandwidth samples. The results of this analysis are displayed in Table 2.

These results indicate that the sum of ifInOctets and ifOutOctets (as reported by the SNMP sensor) are inversely correlated to a strong degree with average bandwidth (as reported by the NWS network sensor). The total number of octets routed showed strong inverse correlation with the time-smoothed bandwidth values for both the maximum and average smoothing operations. The maximum operation showed the best correlation when performed with a window size of 3 samples. The average time-smoothed bandwidth was most highly correlated with total octets at a window

	t=3	t=4	t=5	t=6	t=7
Smoothed Minimum	-0.5445	-0.5403	-0.5339	-0.5147	-0.4841
Smoothed Maximum	-0.7663	-0.7427	-0.7140	-0.6807	-0.6607
Smoothed Mean	-0.7603	-0.8171	-0.8526	-0.8678	-0.8677

**Table 2. Coefficients of Correlation for Time-Smoothed Bandwidth vs. Raw Octet Counts (window size t)**

size of 6 samples. This particular comparison (which displayed the highest correlation overall) is depicted graphically in the chart in Figure 4.



**Figure 4. SNMP observed activity (Octets) compared with NWS reported bandwidth, smoothed (SmoothBW) and unsmoothed (RawBW).**

## 4. Conclusion

The addition of these new modules helps to position NWS as an integral component for performing QoS monitoring and fault prediction in an SNMP-based fault management architecture.

The Notification Process relieves client programs of the burden of retrieving NWS forecasts and measurements on a regular basis. After registering a threshold with the Notification Process, an application can simply wait for an asynchronous event to arrive to announce a change in conditions. Since alerts are generated in response to forecasted threshold violation, applications and system managers have time to react to the condition before it can have an adverse impact.

The Extended SNMP Agent provides enhanced visibility of data provided by NWS. This allows for a more transparent incorporation of the system into the SNMP management framework.

Finally, the SNMP Sensor serves as an accurate, passive measurement instrument. It provides a method for the inclusion of a wide variety of new metrics into the NWS database. It also allows NWS to monitor equipment that it previously was unable to inspect, including most commonly utilized network elements.

We have shown that the SNMP sensor, configured to monitor router interface octet transfers, can be used

to reason about available network bandwidth in a passive manner. A downward trend in observed available bandwidth by the NWS network sensor might be used to recognize a condition of network over utilization. By monitoring trends in SNMP octet counts, we may be able to predict and prevent this condition by proactively performing network resource reallocation when we notice that the counts are trending toward some predetermined level of network saturation.

More work needs to be done to identify those SNMP variables that can be effectively used as predictors of QoS shortfalls. Other changes to the SNMP sensor, such as combining variable requests into a single GET request, could improve performance and accuracy.

## References

- [1] D. Andresen, T. Yang, O. Ibarra, and O. Egecioglu, Adaptive Partitioning and Scheduling for Enhancing WWW Application Performance, *Journal of Parallel and Distributed Computing (JPDC)*, pp. 57-85, vol. 49, no. 1, February, 1998.
- [2] J. Case, M. Fedor, M. Schoffstall, J. Davin. A Simple Network Management Protocol (SNMP). RFC 1157, May, 1990.
- [3] I. Foster, and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115-128, Summer, 1997.
- [4] B. Lowekamp, N. Miller, D. Sutherland, T. Gross, P. Steenkiste, and J. Subhlok, A Resource Query Interface for Network-aware Applications, *Proc. of the Seventh IEEE Symposium on High-Performance Distributed Computing*, July, 1997.
- [5] B. Lowekamp, D. O'Hallaron, and T. Gross, Direct Queries for Discovering Network Resource Properties in a Distributed Environment, *Proc. of the Eighth IEEE International Symposium on High-Performance Distributed Computing (HPDC8)*, Los Angeles, CA, pp. 38-46, August, 1999.
- [6] K. McCloghrie, M. Rose. Management Information Base for Network Management of TCP/IP-based Internets: MIB-II, RFC 1213, March, 1991.
- [7] M. Miller. Managing Internetworks with SNMP. M & T Books, 1993.
- [8] UCD SNMP, <http://ucd-snmp.ucdavis.edu/>
- [9] R. Wolski, N. Spring, and C. Peterson. Implementing a Performance Forecasting System for Metacomputing: The Network Weather Service. UCSD Technical Report No. TR-CS97-540, 1997.
- [10] R. Wolski, N. Spring, and J. Hayes. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. UCSD Technical Report Number TR-CS98-599, September, 1998.