Journal of Digital Imaging

Proactively Monitoring Departmental Clinical IT Systems with an Open Source Availability System

C. Toland, C. Meenan, M. Warnock , and P. Nagy

The goal of all radiology information technology (IT) support organizations is excellent customer service through the availability of critical clinical information services, such as picture archiving communication systems and radiology information systems. Despite these goals, IT support personnel often act like fire-fighters, reacting to each problem, but unable to prevent or predict other problems. Proactive support is always more desirable than reactive support. Warning signs may exist well before a technical issue becomes system wide or the user is affected. The objective for IT support organizations in health care should be to maximize system uptime by using proactive monitoring systems for failures and to automatically detect failures through systems management tools. We report on the implementation of Nagios, an open source monitoring tool, as an availability management system in a diagnostic imaging department and on customized applications and protocols specific to radiology needs.

KEY WORDS: PACS, availability monitoring, Nagios, open source, management information systems, system management

INTRODUCTION

Picture archiving and communication systems (PACS) and other imaging-related information technology (IT) systems have grown in popularity over the last decade. Initially used within departments as radiology-focused tools with a small number of users, these systems have now grown to become enterprise-class resources for departments outside of radiology. As demand for and reliance on these systems have continued to grow, so has the demand for an always-on infrastructure and system architectures that can support mission-critical availability and uptime.[1]

Meeting these requirements is often a challenge for PACS and radiology IT professionals. Many system architectures delivered as standard offerings by industry vendors lack redundancy and/or fail-over capabilities. Most PACS teams rely on their clinical users to alert them to problems as they occur. This is a less-than-ideal management approach because it requires that a system fail sufficiently to adversely affect users before the PACS team is notified of the problem. Moreover, it places the onus of problem-reporting responsibility on the already time-challenged end-users. Users often grow dissatisfied in this role and either ignore the issue or wait hours to report it, delaying the repair time, allowing the problem to affect growing numbers of their end-user colleagues, and, in worst-case scenarios, adversely affecting patient care.

With the truly mission-critical nature of these systems, the need to understand possible failure modes and react quickly is high. A much preferred and more straightforward approach is proactive monitoring of system availability.[2] System monitoring tools are available to automatically watch over all systems in a department or hospital.[3] These tools can remotely monitor the ports of a server for activity and can alert IT staff if no

From the Department of Radiology, University of Maryland Medical Center, 22 S. Greene St, Baltimore, MD, 21201, USA.

Correspondence to: C. Toland, Department of Radiology, University of Maryland Medical Center, 22 S. Greene St, Baltimore, MD, 21201, USA; Tel: +1-410-3281054 e-mail: ctoland@umm.edu

Online publication 20 September 2007
doi: 10.1007/s10278-007-9063-2

activity is detected. Digital Imaging and Communications in Medicine (DICOM), Health Level 7 (HL7), databases, and Web services all run on different ports and can be monitored separately. By using these tools, PACS support team members can be automatically notified by pager when a possible problem exists and can work to repair issues more quickly. Monitoring tools can also provide useful statistics (e.g., uptime reporting to the system service level), restart failed services, and interface with ticketing systems to perform issue tracking and trending analysis. Implementation of these tools provides the added advantage of having an independent system monitor the availability of a PACS system. This can be quite useful when trying to hold a vendor accountable to an uptime guarantee in a service-level agreement.

Although commercial packages are available, these are often expensive and lack custom tools to monitor imaging-specific criteria (such as DICOM availability). Commercial packages may also lack the requisite flexibility to monitor such criteria as message queues for radiology information systems (RIS). We report here on the implementation of an open source monitoring tool customized to our departmental needs and on our assessment of its utility in routine clinical imaging operations.

MATERIALS AND METHODS

We implemented the open source availability monitoring package Nagios[4] and configured it to be a comprehensive availability monitoring solution in a heterogeneous environment of 68 systems with 108 monitored services. We installed the Nagios packages on an HP DL360 server (Hewlett-Packard, Palo Alto, CA, USA) with dual central processor units and 2 GB RAM running the Gentoo[5] Linux operating system.

The support team's information was entered into the system. Nagios referred to these as "contacts". Each "contact" had a name, e-mail address, pager e-mail address, and availability window for paging. These contacts were split into groups, such as "pacs_team" and "ris_team." The host servers were then set up in the system. Each host was configured with a name, Internet Protocol address, interval for monitoring (24×7 in most cases, but 8×5 for those up only during business hours), and notification options (down, recovered, warning states). The interval functionality was useful for those devices and modalities that were turned off during specific time periods (such as overnight).

Groups were created from the list of hosts and assembled on the basis of function. Some of the groups we formed were pacs_archives, pacs_databases, and ris_servers. Contact groups were assigned to these host groups for alerting.

One of the most powerful features of Nagios was the ability to create customized approaches to search for more than the availability of a single port. Nagios uses a series of programs or scripts for monitoring as part of its standard functionality. When one of the system programs exit, Nagios interprets the exit code. The exit codes used are 0 (OK), 1 (warning), and 2 (critical). One example of the utility of these codes is in a queue checker we created. In this scenario, a script connects to a database and pulls the value of a specific queue. If the value is >50, the program exited with a code of 2 (critical). If it is between 25 and 49, the program exited with a code of 1 (warning). If the value is <25, the program exited normally with an exit code of 0 (OK). One advantage of using an open source application in this role was the relative ease with which codes can be extended to add functionality.

Our system was customized to monitor criteria beyond system availability, to include port monitoring, queue status, and DICOM availability of PACS gateways and subspecialty visualization work stations, and was more broadly used for an overall picture of the health of our information systems. Additional plug-ins were created to restart services

Table 1. Specific Ports and Services Used by Nagios for System Monitoring

Ports and Services
ICMP or Ping for general availability
Port 80 for HTTP Web server availability
Port 443 for HTTPS secure Web server availability (including SSL Certificate expiration)
Port 104 for DICOM (specifically C-ECHO)
Port 53 for Domain Name Service
Port 21 for File Transfer Protocol
Port 25 for Simple Mail Transport Protocol
Port 3306 for MySql database availability
Port 1433 for Microsoft SQL availability
Port 1521 for Oracle Database availability
Port 389 for Lightweight Directory Access Protocol
Web Services health checker status
Screen scraping of queue monitoring Web pages with scripting to enable a restart of interface via URL string

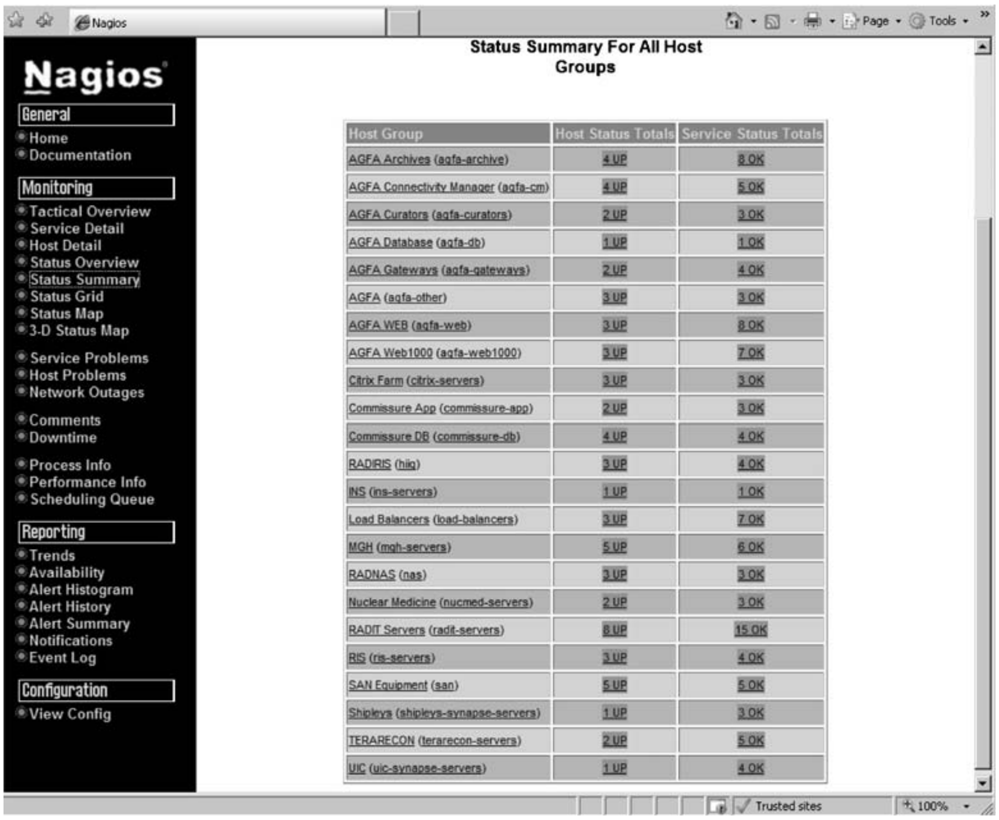
in the event of a failure, to log a ticket in our open source issue tracking system (Radtracker),[6] as well as page our team members in the event of an issue. Specific criteria monitored are listed in Table 1.

We installed this system in 2006 and monitored its usefulness over a 6-month period from December 2006 to May 2007.

RESULTS

The system was extremely effective at providing an overall view of the health level of our systems as well as specific data about availability. Benefits that were realized as a result of this implementation included (but were not limited to):

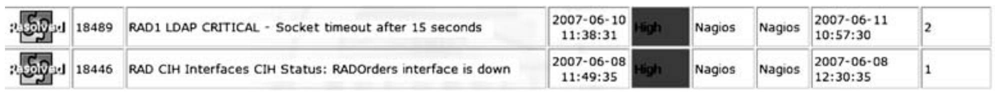
- Automated monitoring with Web-based access to system management data. The software provided a comprehensive view of the systems in our environment, monitoring 68 servers on a 24×7 basis. In addition, remote radiology sites were monitored, giving the added benefit of measuring wide area network (WAN) availability to remote locations (Fig. 1).
- Integration of monitoring with notification. The system was integrated with our inhouse paging system via Simple Mail Transfer Protocol to our local e-mail gateway. By integrating the system in this fashion, alerts could be sent directly to staff members' pagers via e-mail.
- Integration with problem management systems. The system was integrated with our paging system via direct Open Database Connectivity insertion into our local issue tracking system's mysql database (Fig. 2). This allowed the team to track any specific issue as part of the normal call resolution process and provided a knowledge base in the system for root cause analysis as well as a methodology for repair.
- Service Level Agreement reporting. The system provided an overall system uptime view with time-frame customization reporting for availability, with detail down to the service level for each specific host or host group (Fig. 3).



Status Summary For All Host Groups

Host Group	Host Status Totals	Service Status Totals
AGFA Archives (agfa-archive)	4 UP	8 OK
AGFA Connectivity Manager (agfa-cm)	4 UP	5 OK
AGFA Curators (agfa-curators)	2 UP	3 OK
AGFA Database (agfa-db)	1 UP	1 OK
AGFA Gateways (agfa-gateways)	2 UP	4 OK
AGFA (agfa-other)	3 UP	3 OK
AGFA WEB (agfa-web)	3 UP	8 OK
AGFA Web1000 (agfa-web1000)	3 UP	7 OK
Citrix Farm (citrix-servers)	3 UP	3 OK
Commissure App (commissure-app)	2 UP	3 OK
Commissure DB (commissure-db)	4 UP	4 OK
RADIRIS (hiiq)	3 UP	4 OK
INS (ins-servers)	1 UP	1 OK
Load Balancers (load-balancers)	3 UP	7 OK
MGH (mgh-servers)	5 UP	6 OK
RADNAS (nas)	3 UP	3 OK
Nuclear Medicine (nucmed-servers)	2 UP	3 OK
RADIT Servers (radit-servers)	8 UP	15 OK
RIS (ris-servers)	3 UP	4 OK
SAN Equipment (san)	5 UP	5 OK
Shipleys (shipleys-synapse-servers)	1 UP	3 OK
TERARECON (terarecon-servers)	2 UP	5 OK
UIC (uic-synapse-servers)	1 UP	4 OK

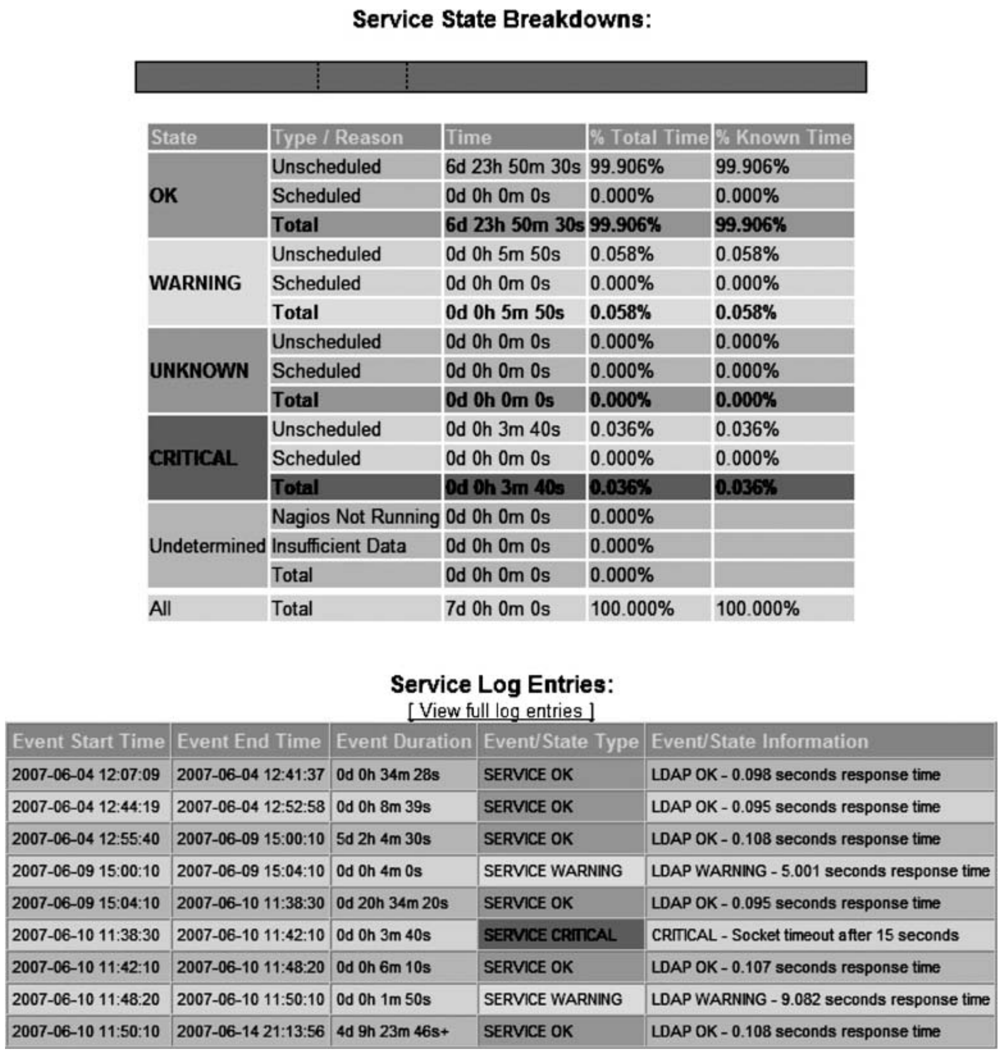
Fig. 1. Status summary for all host groups.



Resolved	18489	RAD1 LDAP CRITICAL - Socket timeout after 15 seconds	2007-06-10 11:38:31	High	Nagios	Nagios	2007-06-11 10:57:30	2
Resolved	18446	RAD CIH Interfaces CIH Status: RADOrders interface is down	2007-06-08 11:49:35	High	Nagios	Nagios	2007-06-08 12:30:35	1

Fig. 2. Service state breakdowns.

- Automated restart of scripted interfaces. For several interfaces in our environment, the capability existed to monitor interface status and queue length via a Web page. The system could monitor these interfaces via a screen scrape of the Web page and restart those interfaces that allowed for a restart when passed via a Uniform Resource Locator string.
- Health check status of systems via Web service. When provided by vendors, the system could also monitor health and other status via Web service capabilities.
- Over the 6-month period, the system generated 461 issues in our issue tracking system. A breakdown of these issues by category is included in Table 2.



Service State Breakdowns:

State	Type / Reason	Time	% Total Time	% Known Time
OK	Unscheduled	6d 23h 50m 30s	99.906%	99.906%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**6d 23h 50m 30s**	**99.906%**	**99.906%**
WARNING	Unscheduled	0d 0h 5m 50s	0.058%	0.058%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**0d 0h 5m 50s**	**0.058%**	**0.058%**
UNKNOWN	Unscheduled	0d 0h 0m 0s	0.000%	0.000%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**0d 0h 0m 0s**	**0.000%**	**0.000%**
CRITICAL	Unscheduled	0d 0h 3m 40s	0.036%	0.036%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**0d 0h 3m 40s**	**0.036%**	**0.036%**
Undetermined	Nagios Not Running	0d 0h 0m 0s	0.000%	
	Insufficient Data	0d 0h 0m 0s	0.000%	
	Total	0d 0h 0m 0s	0.000%	
All	Total	7d 0h 0m 0s	100.000%	100.000%

Service Log Entries:

[View full log entries]

Event Start Time	Event End Time	Event Duration	Event/State Type	Event/State Information
2007-06-04 12:07:09	2007-06-04 12:41:37	0d 0h 34m 28s	SERVICE OK	LDAP OK - 0.098 seconds response time
2007-06-04 12:44:19	2007-06-04 12:52:58	0d 0h 8m 39s	SERVICE OK	LDAP OK - 0.095 seconds response time
2007-06-04 12:55:40	2007-06-09 15:00:10	5d 2h 4m 30s	SERVICE OK	LDAP OK - 0.108 seconds response time
2007-06-09 15:00:10	2007-06-09 15:04:10	0d 0h 4m 0s	SERVICE WARNING	LDAP WARNING - 5.001 seconds response time
2007-06-09 15:04:10	2007-06-10 11:38:30	0d 20h 34m 20s	SERVICE OK	LDAP OK - 0.095 seconds response time
2007-06-10 11:38:30	2007-06-10 11:42:10	0d 0h 3m 40s	SERVICE CRITICAL	CRITICAL - Socket timeout after 15 seconds
2007-06-10 11:42:10	2007-06-10 11:48:20	0d 0h 6m 10s	SERVICE OK	LDAP OK - 0.107 seconds response time
2007-06-10 11:48:20	2007-06-10 11:50:10	0d 0h 1m 50s	SERVICE WARNING	LDAP WARNING - 9.082 seconds response time
2007-06-10 11:50:10	2007-06-14 21:13:56	4d 9h 23m 46s+	SERVICE OK	LDAP OK - 0.108 seconds response time

Fig. 3. Service log entries.

Table 2. Breakdown of Alerts Submitted into Ticketing System over 6 Months

Type of Event	Number of Occurrences
Host availability of Ping	209
Interface or queue problems	104
DICOM C-ECHO	51
Web server failure	40
Web-services-related	37
Lightweight Directory Access Protocol (LDAP) bind failures	8
Misc. other failures	12

DISCUSSION

The system was effective in achieving the desired results. The majority of Nagios issues were related to general system availability. These types of failures were either total system failures or network-related, oftentimes indicating a WAN link failure. Not all notifications were failures, however, because normal network maintenance at our facility was also included. On several occasions, the use of this tool was effective in providing warning notifications to WAN vendors of a network failure or performance degradation before the vendor had been alerted to a problem.

The second most common failure mode was related to interfaces either dropping or having a high number of messages that exceeded a predetermined threshold for that queue. By keeping interfaces available and within guidelines, we were able to keep the flow of data to a more real-time level. By scripting interface restarts and turning these into an automated process, we greatly reduced the impact to users of any interface drop.

The third most common failure mode was related to DICOM availability. Monitoring this failure mode was critical, because it effectively monitors the ability of systems to send images via DICOM to PACS. This functionality does not normally exist in commercial monitoring packages and was of great importance since we were able to effectively monitor our system's ability to function as a DICOM sender or receiver. The Nagios plug-in for this feature was created using the DCMTK DICOM Toolkit[7] to both perform the C-Echo and determine if the association was completed successfully. In one case, we observed that by simply monitoring the DICOM service for a particular host with C-ECHOs at 5-min intervals, we were able to prevent DICOM service failures that had previously been a frequent occurrence on the device.

Other remaining failure modes included Web servers, Web services, and Lightweight Directory Access Protocol bind issues, among others. Monitoring for these failures was a critical addition and a great benefit for our group, because traditional availability-only monitoring (through Internet Control Messaging Protocol or ping alone) would not in most cases have revealed these failure modes as these types of failures most often occur for specific services within servers themselves. In such a scenario, servers would answer a ping and would appear to be available on a network but, in fact, would be experiencing an internal failure that would leave them nonfunctional or in a degraded state and unavailable to users. By capturing these failure modes through our monitoring efforts, we were able to respond immediately to issues that might not be obvious to standard monitoring systems.

CONCLUSION

We found the Nagios availability tool to be effective in the proactive support of mission-critical radiology and other clinical imaging systems. Support staff were notified in real time with monitoring tool alerts to warning signs of possible system failures. Although some PACS and RIS vendors offer their own monitoring packages, we found that the addition of an independent monitoring tool that was outside of vendor control was effective and allowed the monitoring of multiple vendors through the same interface. This comprehensive approach to monitoring departmental and enterprise applications across all support environments provided a comprehensive view of system availability.

We found that open source monitoring tools such as Nagios could provide cost-effective independent monitoring. The benefits of using an open source tool included independent monitoring, agility in adaptation, and an open tool box for customizations. If a new application or system needed to be monitored, custom or prepackaged plug-ins could be utilized without the need of a purchase order or lengthy development turnaround time.

The use of Nagios as a system alerting and availability tool allowed us to monitor our systems for failures and notify our support staff immediately when problems occurred. This allowed them to resolve issues proactively, often before users experienced any problems. The result was enhancement of our continuing efforts to achieve always-on systems and mission-critical delivery of services to our customers.

ACKNOWLEDGMENT

I would like to thank Dr. Nancy Knight from the University of Maryland School of Medicine for her assistance in preparing this manuscript.

REFERENCES

1. Allen A, Kutnick D: Building Operational Excellence. Boston: Intel Press, 2002
2. Schiesser R: IT Systems Management. Upper Saddle River: Prentice Hall PTR, 2002
3. Lyke HL, Cottone D: IT Automation: The Quest for Lights Out. Upper Saddle River: Prentice Hall PTR, 2000
4. Nagios Web site. Available at: www.nagios.org. Accessed on June 15, 2007
5. Gentoo Website. Available at: www.gentoo.org. Accessed on June 15, 2007
6. Radtracker Project. Available at radtracker.sourceforge.net. Accessed June 15, 2007
7. DCMTK Website. Available at dicom.offis.de/dcmtk.php.en. Accessed June 15, 2007