Monitoring Your Enterprise PACS With Nagios®, Cacti And Smokeping

Ron Sweeney, Engineer 2

ClubPACS Western Michigan

Drafted: 09/02/04

Revised: 11/03/04

Index

1 Introduction

1.1 What we are trying to accomplish

The solutions outlined in this white paper will allow you to monitor the overall health of your PACS, trend it, and provide invaluable tools as it relates to the bottlenecks in your network. It will alert you on exceptions within your environment, provide availability reporting, and an overall dashboard to the systems that glue your PACS together. This solution requires little, if at all, of installation of software on your host machines or modalities, just in case you have one of those vendors that does not allow you to change the color scheme of CDE on your server without first qualifying it with them. It will also lend you the benefits of powerful Open Source software to tweak and modify it to your specifications without proprietary constraint. This solution is not limited to an particular PACS vendor, modality vendor, or networking component manufacturer. Still interested? Read on...

1.2 For purposes of this document...

It the spirit of Larry Wall, *TimTowTdi* (there is more than one way to do it). All application versions, Operating System versions, client versions and platforms mentioned with the solution are exceptionally versatile. There is no correct operating system, apache version, etc. that I can recommend, however for purposes of this document, I am going to use the following components for it has "worked for me." Most *NIX like systems can run this solution without exceptional bother. Perspective requirements as it pertains to the applications that are mentioned in this solution should absolutely be adhered to. You may be a bit challenged if you intend on running this solution on a windows platform.

We are running a Linux Fedora Core 1 server running a kernel version of 2.4. The box we are running it on is nothing extravagant, its a Dell with about 1GB of RAM and a 1GHZ Pentium processor and about 30GB that is being managed by LVM on the hard disks (an old Siemens MV300 from the late nineties). It is also performing acceptably for us with an overall PACS footprint of 100 DICOM devices, 140+ workstations, 40 servers and spanning 8 facilities across various WAN links.

This document was drafted on FreeBSD 4.9 STABLE and Open Office 1.0.3 with a Model M keyboard (included here for religious purposes).

On with the show...

2 The Players

This powerful solution is made possible through the efforts of many various members of the Open Source community, distinctively however Tobi Oetiker (RRD Tool and Smokeping), Ian Berry (Cacti) and Ethan Galstad (Nagios®).

Each section will give a brief overview of functionality of each of these applications, installation and configuration notes, and example uses that benefit the support of an Enterprise PACS environment. The installation and configuration notes may be brief, I want to highlight the configurations that are relevant to PACS in this document. It should also be noted that the installation documentation supplied with each of these applications should be used when installing the software. All three of these applications have been deployed on a single server (ie. They play well together).

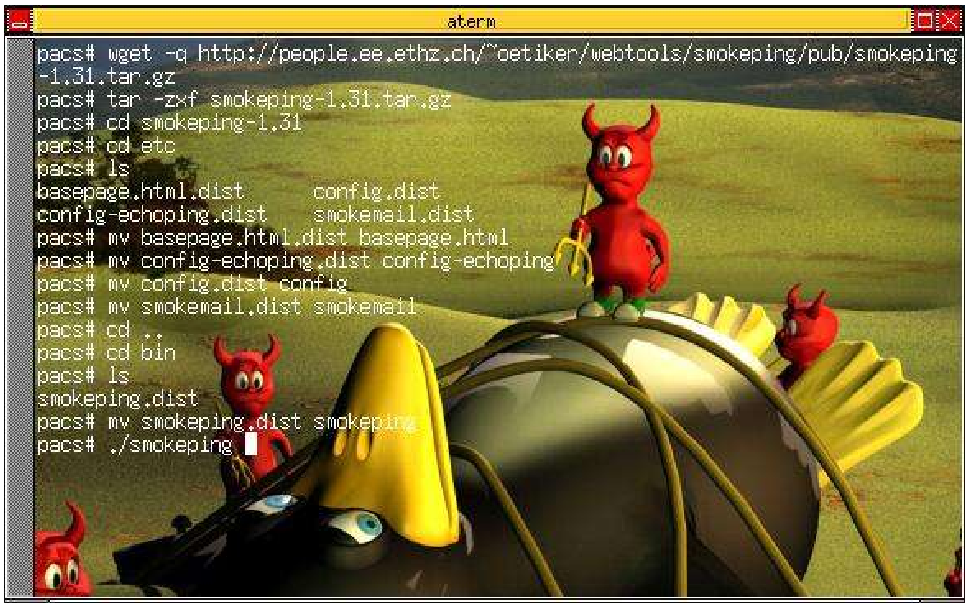
2.1 Smokeping

Overview

With SmokePing you can measure latency, latency distribution and packet loss in your network. SmokePing uses RRDtool (also written by Tobi Oetiker) to maintain a longterm datastore and to draw some pretty cool graphs, giving up to the minute information on the state of each network connection. This tool is extremely useful in detecting network bottlenecks and duplex issues.

Installation and Configuration

Smokeping installation is fairly straight forward. After you have met and installed all the prerequisites(RRDtool 1.0.x, FPing 2.4b2, Perl 5.6.1, SpeedyCgi) its as easy as untarring the distribution, editing the config file and you are off to personify the obsession of ping! Its never that easy, but following the instructions with smokeping is quite straight forward (really).



```
pacs# wget -q http://people.ee.ethz.ch/~oetiker/webtools/smokeping/pub/smokeping
-1.31.tar.gz
pacs# tar -zxf smokeping-1.31.tar.gz
pacs# cd smokeping-1.31
pacs# cd etc
pacs# ls
basepage.html.dist      config.dist
config-echoping.dist    smokemail.dist
pacs# mv basepage.html.dist basepage.html
pacs# mv config-echoping.dist config-echoping
pacs# mv config.dist config
pacs# mv smokemail.dist smokemail
pacs# cd ..
pacs# cd bin
pacs# ls
smokeping.dist
pacs# mv smokeping.dist smokeping
pacs# ./smokeping
```

Once you are off to the races and can point your browser at the `smokeping.cgi` and get dealt a page not littered with errors, you can move to start monitoring some devices, links, or hosts to your desire. You specify the "things" you want to monitor in the `config` file.

The `config` file itself is a little picky on its syntax, I have an example below of the "Targets" section and Ill attempt to highlight the parts to get you going. In this particular example, we want to watch for latency on two different firewalls to troubleshoot connectivity of the hosts on the other side of both of these packet misers.

```
*** Targets ***

probe = FPing

menu = My PACS Network
title = Network Latency Grapher
remark = Welcome to the SmokePing website of My PACS Network \
         This is a troubleshooting tool for latency with our PACS.

+ FWTSHOOT
menu = Firewalls
title = Firewall Latency Analysis

++ FirewallTesting
menu = FirewallTesting
title = Firewalls

+++ WOPIX
menu = WOPIX
title = 192.168.0.2
host = 192.168.0.2

+++ WONOKIA
menu = WONOKIA
title = 192.168.23.2
host = 192.168.23.2
```

Its up! Now what?

Basically, the application is run once every five minutes in cron. It fping's (if you will allow a binary to be expressed as a verb) a host 20 times and measures round trip time and logs packet drops. The "smoke" when visible denotes varying Round Trip Time on the packets either up or down. The color marked by that poll indicates the number of drops. The alchemy of RRD

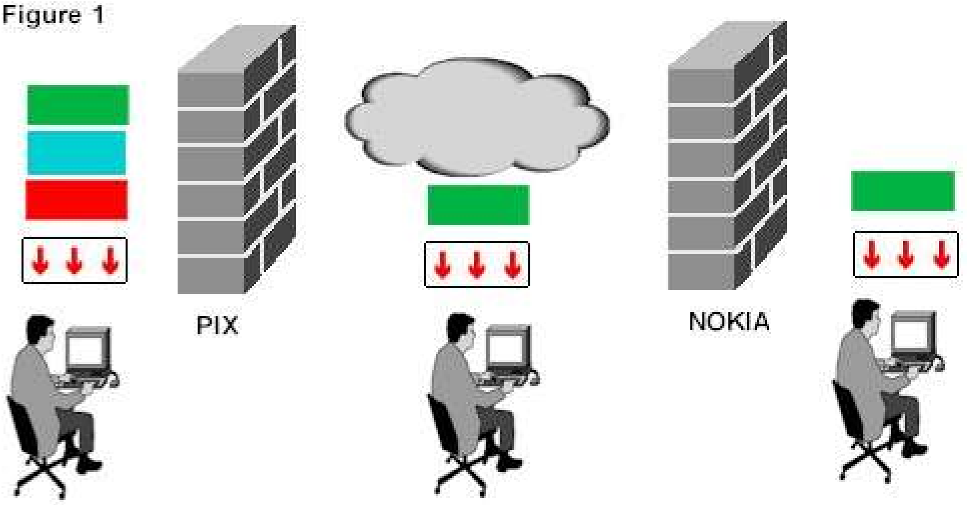
Tool also gives you some pretty statistics based on averages of the now, 10 days ago and more than a year ago.

Real World PACS Examples

You can't be blind to what is going on in your network. You can't dismiss when your Radiologists or Technologists are telling you that “something is not right,” they use the system more than you do and can tell you when something is off kilter, but appears functional. Smokeping has helped in diagnosing slow downs in PACS many o' time, more exclusively it has helped us find duplex problems on Ultrasound units and more recently Firewall problems at our WAN sites.

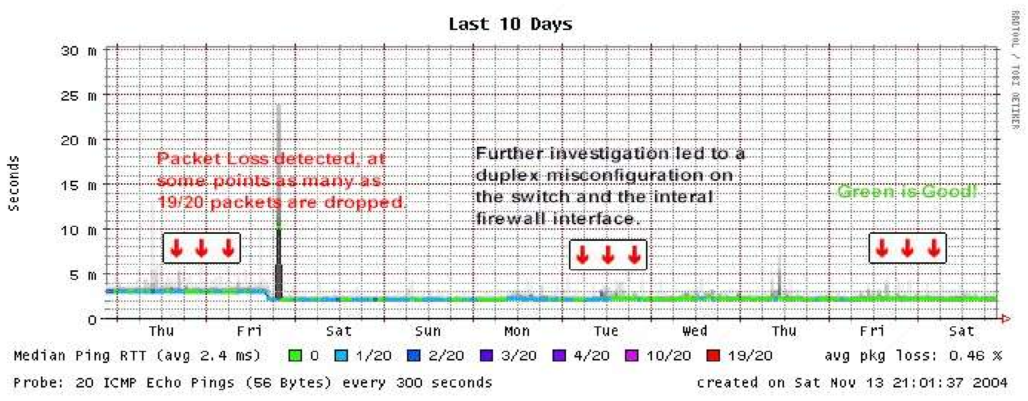
The report was, “It's slow.” If you haven't heard this before in the IT or PACS industry, you are either really good, your using population has not caught up, or you aren't paying attention. We had reports that reading stations at a huge WAN site was having problems with the application initializing and general movement around the GUI. We did our diligence, popped Ethereal on the wire, and immediately blamed Oracle and the Shared Message Block protocol. With more complaints and screaming we decided to do some testing as we felt the two firewalls we were traversing were giving us overhead we didn't need. In order for the transactions to take place, the Radiology PACS application needed to get through a Cisco PIX and then through a Nokia Checkpoint firewall (yep, blame it on the firewall guys). So we moved a workstation in front of one of the firewalls so we had one firewall to content with and did some testing, smokeping was green, and all was good. For good measure, we moved the workstation to the front of the Checkpoint firewall so the application had no firewall to contend with, all was good as well(Figure 1).

Figure 1



Ok, so we have a culprit. Looks like something is wrong on the PIX. After the network engineers enabled some monitoring, they found significant packet loss on the internal interface of the firewall. It was found that a duplex mismatch from the switch to the internal interface was the problem, and after a non-invasive command was issued into IOS, the smokeping started turning green. We checked the application and according to the analysts it had worked better than ever before. After a week had passed we took a look back to smokeping and indeed saw that the change we made, on Tuesday at 12:20 pm had turned things to “better” that even most management can understand (Figure 2).

Figure 2



2.2 Nagios®

Overview

Nagios® is a host and service monitor designed to inform you of network problems before your technologists, radiologists or managers do. The monitoring daemon runs intermittent checks on hosts and services you specify using external "plugins" which return status information to Nagios®. Some of the examples included in this document have experimental “plug-ins” that I have developed to monitor DICOM services (ie. check_dcm). When problems are encountered, the daemon can send notifications out to administrative contacts in a variety of different ways (email, pager, etc). The entire application front end is wrapped up in a slick and intuitive Web Based GUI.

Installation and Configuration

Depending on what way you want to go, installation can be easy. There is different ways you can go about configuring Nagios® when it is up and running, for purposes of this document I chose the “old way” which is template driven.

As far as the install goes it really consists of the following; unpacking the distribution, adding a user, building, configuring apache, and getting/installing the plug-ins. The app is still cute without plug-ins, but unfortunately worthless.

Here are some notes on the configuration of the files in $Nagios®_PATH/etc/, they should get you going and open your eyes up to what can be accomplished with this application.

`Cgi.cfg`

Basically just some path information that has to be set to your install environment.

`Contacts.cfg`

This is the email alias area where we map the who to the notification method. To get the system going, just add your (or a reachable) email address to the to the email and pager sections in the contact block.

`Contactgroups.cfg`

This is where all good people come together. You could break up and map people to particular groups, like “Clinical Engineering” or “PACS Admins” and have them notified differently. This is more or less an alias file.

Example

```
define contactgroup{
        contactgroup_name        PACS-NET # give it a name
        alias                    PACS Support # This a logical Name
         members                  Nagios # people from contacts.cfg
        }
```

`Nagios.cfg`

Nagios.cfg is your main driver, it allows certain things to be turned off and on in the application. I recommend taking a minimalist approach until you get the basics down, such as possibly ping checks on a handful of hosts and having the message end up on your pager. There is path information in here that is relevant to your install.

`Hosts.cfg`

Now, add all of your hosts in your hosts.cfg file that you want to know anything about. The basic format for the file is in the distribution, but a block is added here for referential purposes. You can safely use the defaults for the last four values to get up and going.

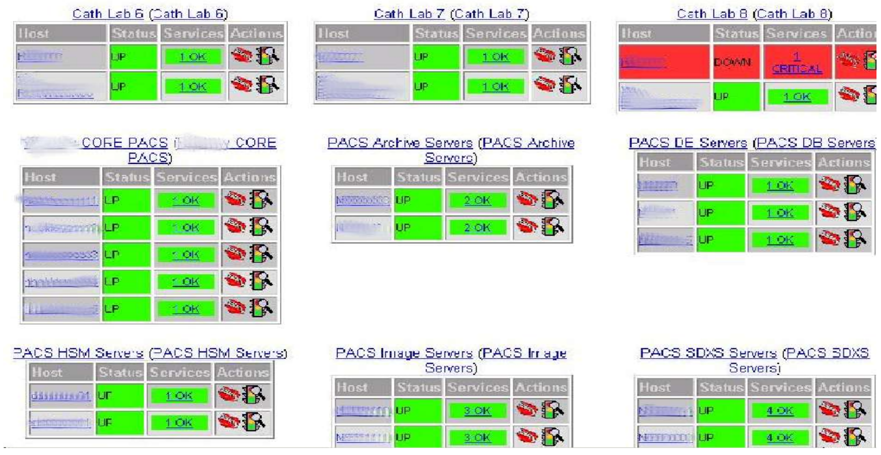
```
define host{
        use                      generic-host
        host_name                WSERV1 # the hostname
        alias                    WEB SERVER 1 # logical name
        address                  172.18.18.4 # IP of node
        check_command            check-host-alive # checkcommands.cfg
        max_check_attempts       10 # maximum times to try
        notification_interval    120 # notify every in minutes
        notification_period      24x7 # business hours
        notification_options     d,u,r # options for notifications
        }
```

`Hostgroups.cfg`

Be sure to group them by type or service type, for example, put all workstations in a block, put all web servers in a block, etc.. This will help you build the hostgroups.cfg. As an example, we put all “Image Servers” with SCP services in a block, “Web Servers” for any referential delivery servers, “Archive Servers” for the archival boxes and “Database Servers” for the

Database Cluster and servers. An example of how we grouped thing is a shot at our Nagios® gui, which if you are successful in your install, you should see something similar(Figure 3):

Figure 3


Cath Lab 6 (Cath Lab 6)
Cath Lab 7 (Cath Lab 7)
Cath Lab 8 (Cath Lab 8)
CORE PACS CORE PACS)
PACS Archive Servers (PACS Archive Servers)
PACS DB Servers (PACS DB Servers)
PACS HSM Servers (PACS HSM Servers)
PACS Image Servers (PACS Image Servers)
PACS SDXS Servers (PACS SDXS Servers)
Host
Status
Services
Actions
UP
DOWN
1 OK
2 OK
3 OK
4 OK
1 CRITICAL

checkcommands.cfg

This is where it starts to get fun, and you realize the value of this application to your PACS. This is where you throw the flags into your plug-ins to get things accomplished. The plug-ins are nothing more than binaries and scripts that perform a certain check and a return a value to Nagios®. The latest build of plug-ins had enough to definitely get you started, I am going to go over the ones in use with our PACS in the real world examples. The idea is you can customize a check based on the flags you want to send to the plug-in. Let's take a look at the following example:

```
define command{
        command_name    check_sdxs # logical name for services.cfg
        command_line    $USER1$/check_tcp -H $HOSTADDRESS$ -p 5550
        }
```

This check is going to check the availability of an HL7 listener within PACS. Im going to use a vanilla plug-in called check_tcp which does nothing more than a socket connect for an open or closed port. It works a lot like netcat or nmap in testing for open ports on a hosts tcp stack. Immediately you can see that will this plug-in you can monitor just about any listening port for availability, though there might be a plugin that is better suited and more advanced for the purpose.

Services.cfg

This is where you group the hosts that will receive the check command in checkcommands.cfg. If you have a group of web servers, and you want to run the check_http check across all of them, then you would include something like the following in services.cfg:

```
define service{
         use                                    generic-service
          host_name                                 WEBSERV0,WEBSERV1,WEBSERV3,WEBSERV4,WEBSERV5 # list
hosts
         service_description                    WebCheck # logical name for service
```

```
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3 # check it this many times
        normal_check_interval           5 # check frequency in minutes
        retry_check_interval            1
        contact_groups                  PACS-NET # recipient of the alert contactgroups.cfg
        notification_interval           120 # alert me every n minutes
        notification_period             24x7
        notification_options            c,r
        check_command                   check_http # name in checkcommands.cfg
        }
```

The rest of the configuration files can probably wait until you have become a little bit more advanced with the application. The ones mentioned here must be configured at a minimum to get Nagios® off the ground.

Its UP! Now what?

Let us walk down the interface and explain *a little* of what can be seen. Here are some shots of the Nagios® interface (figure 5). Number one on the diagram is a tactical overview of all things monitored. The look is really slick and it resembles something out of Star Trek (LCARS). Number two is the service detail all on one page, which gives you the "green is good" feeling across all the checks and services you are looking for a status on. Number 3 in Figure 4 groups all the checks into their particular groups, it gives you a snapshot of a group of say "Image Servers" and gives up the same "green is good" interface in regards to that group. For instance, if you had group "Image Servers" and was showing red on all your importers, there is a good chance that the acquisition side of your pacs is not able to send. Number 4 is a rolling log of all the events over the course of the day. This is one of those pages you can slip a Metatag for refresh on it and let it run in your operations center, giving you up to minute downs, ups, or bottlenecks on your PACS. Lastly (for purposes of this document) Number 5 is an example of the reporting that is available in Nagios® as well. The report I generated, also seen in Figure 5 shows uptime over a queryable time of the DICOM listener on Cardiology PACS server.

Figure 4

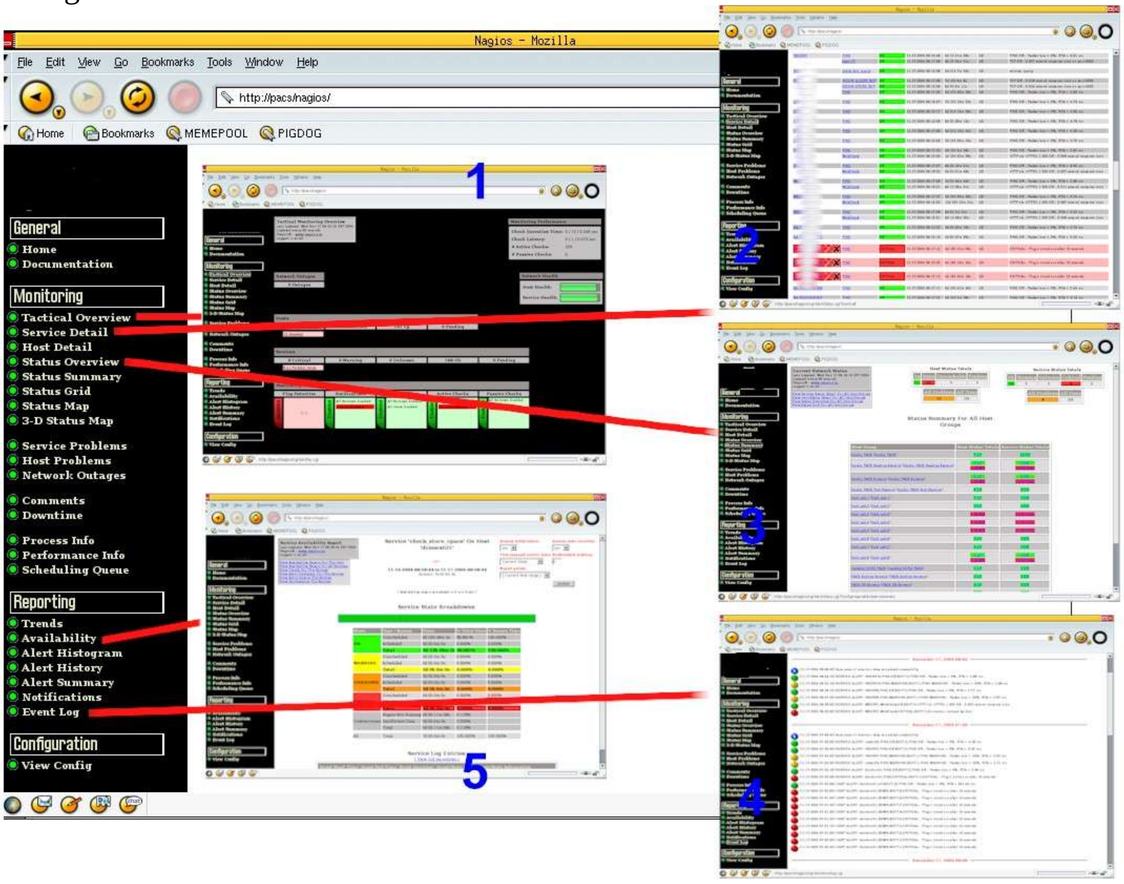
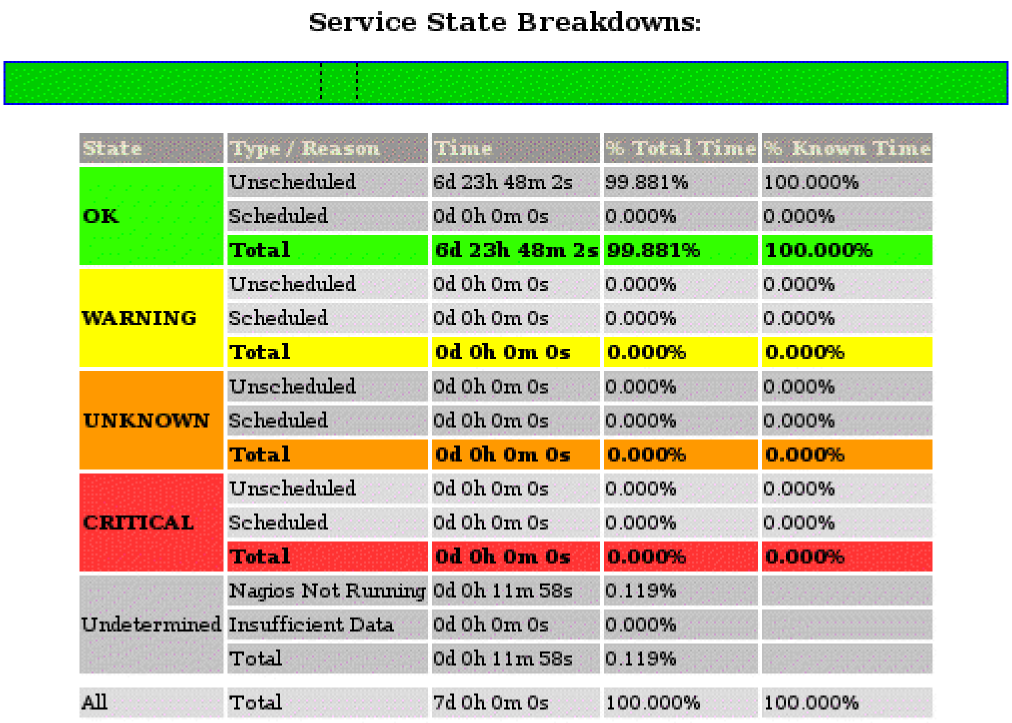


Figure 5



Service State Breakdowns:

State	Type / Reason	Time	% Total Time	% Known Time
OK	Unscheduled	6d 23h 48m 2s	99.881%	100.000%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**6d 23h 48m 2s**	**99.881%**	**100.000%**
WARNING	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%**
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 0m 0s	0.000%	0.000%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**0d 0h 0m 0s**	**0.000%**	**0.000%**
Undetermined	Nagios Not Running	0d 0h 11m 58s	0.119%	
	Insufficient Data	0d 0h 0m 0s	0.000%	
	Total	0d 0h 11m 58s	0.119%	
All	Total	7d 0h 0m 0s	100.000%	100.000%

To reiterate, glance at the side bar of this application and you will realize these examples do not even “graze” the full potential of this application in regards to monitoring your PACS.

Real World PACS Examples

A lot to list here, I will try to lay this out to the point to get you right to the bottom of things with your PACS.

Use Nagios® to monitor SCP and MWL.

This is one that at first I used check_tcp to accommodate the task. I scan for ports 104, 4000, 5000 for various DICOM servers such as Modality Worklist and Store SCP. It worked well, well enough for some, and let me know when these services dropped from availability and usability. However, a lot of the functionality I needed was really missing. I needed something deeper, one that actually checks the ability to associate to the service which gives me a level that up farther in the application layer of OSI. I put together a quick perl script that serves as a wrapper for echoscu and findscu from Offis in their DCMTK package. The check command is called check_dcm, and is called in the following manner.

```
define command{
          command_name    check_dcm_query
          command_line    $USER1$/check_dcm -d query -a QUERY_SCP -n $HOSTADDRESS$ -p 5000
          }
```

At the time of this writing the plugin is experimental. I am going to be posting it to OpenRad in hopes that I can exploit it as a viable ground for development of the plugin, possibly removing the dependency on dcmtk, and adhering to Nagios® plug-in standards. Who knows? If you are interested browse out to Paul Nagy's OpenRad and see if you can contribute!

Use Nagios® to monitor Oracle.

The database platform for our PACS is oracle. I have installed the 10G client on the Fedora Core box and I can now use this plugin to monitor the instance. The basic behavior is a lot like tnsping. It measures if the service is availability and alerts on the latency time for the reply:

```
[root@pacs libexec]# ./check_oracle --tns pacs
OK - reply time 610 msec from pacs
```

But this plugin packs more behind the punch if you want to monitor individual tablespaces or what not, check the switches on the plugin for more details.

Use Nagios® to monitor Dome Cxtra mib values.

This is basically utilizing the SNMP functionality wrapped up in the DOME software and the check_snmp plugin. You can pick from a variety of different values in the management information base (MIB). Some examples are below:

```
 DESCRIPTION     "Current flat panel white level in milli-cd/m^2."
        DESCRIPTION     "Current flat panel temperature in millidegrees C."
        DESCRIPTION     "Backlight time in use in millihours."
        DESCRIPTION     "Desired white level in milli-cd/m^2."
        DESCRIPTION     "Calibrated response function.  Changing the response
        DESCRIPTION     "DICOM part 14 LUM metric in milli-JNDs based on 256
        DESCRIPTION     "Timer interval for alert testing (in hours)."
        DESCRIPTION     "DICOM LUM test active?"
        DESCRIPTION     "Maximum acceptable DICOM LUM value."
```

Use Nagios® to monitor your HL7 listeners.

Using the check_tcp plug-in you can monitor the listening port of your HL7 socket. When orders stop coming in, the worklist stops updating!

```
define command{
        command_name    check_sdxs
        command_line    $USER1$/check_tcp -H $HOSTADDRESS$ -p 5550
        }
```

Use Nagios® to monitor your web servers.

The check_http plugin continually monitors your web servers to verify a status code of 200 comes back to Nagios®. It also manages to report back on the latency at which the request came back (as most plug-ins do).

```
[root@pacs libexec]# ./check_http -H webserver0
HTTP ok: HTTP/1.1 200 OK -   0.011 second response time |time=  0.011
```

Use Nagios® to monitor the CPU on your Image Servers.

With the example below you can monitor an image server for extensive CPU utilization. This has proved quite useful when antivirus has having “issues” traversing thousands of dicom files and takes your CPU out to dinner.

```
[root@pacs libexec]# ./check_snmp -H image2 -o .1.3.6.1.4.1.2.6.71.200.2.1.3.1.3.1.2872348673
-c 70 -C public
```

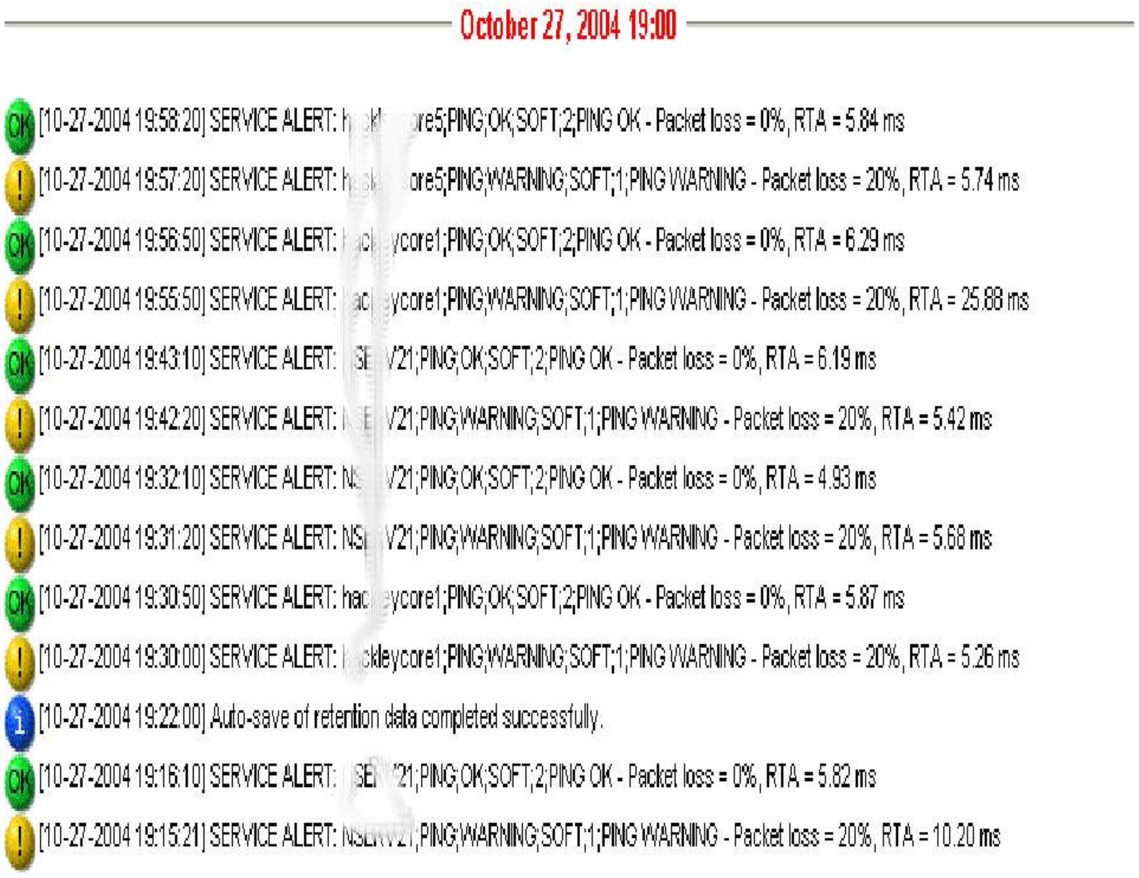
```
SNMP OK - 39
```

The line breaks down as follows, much like the examples above in the DOME check_snmp example. Specify the host, followed by the OID. In this example the OID monitors the CPU on an NT4 server. The critical threshold is set by the -c switch, which was 70 for good measure. The community name of the SNMP agent is specified by the -C switch.

Use Nagios® to monitor host availability and latency

Its not very flashy, but check_ping gives you visibility to the overall health of the PACS by telling you whats UP, and whats DOWN. It also sends back alerts on latency to the ICMP replies too. If you recall the example I gave you from smokeping above, we were having problems with dropped packets to particular devices across a WAN. Not only was smokeping catching the problem, Nagios® continued to WARN us on the packet loss as well! (Figure 6)

Figure 6



```
October 27, 2004 19:00

[10-27-2004 19:58:20] SERVICE ALERT: h[illegible]ore5;PING;OK;SOFT;2;PING OK - Packet loss = 0%, RTA = 5.84 ms
[10-27-2004 19:57:20] SERVICE ALERT: h[illegible]ore5;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 20%, RTA = 5.74 ms
[10-27-2004 19:56:50] SERVICE ALERT: [illegible]ycore1;PING;OK;SOFT;2;PING OK - Packet loss = 0%, RTA = 6.29 ms
[10-27-2004 19:55:50] SERVICE ALERT: [illegible]ycore1;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 20%, RTA = 25.88 ms
[10-27-2004 19:43:10] SERVICE ALERT: [illegible]V21;PING;OK;SOFT;2;PING OK - Packet loss = 0%, RTA = 6.19 ms
[10-27-2004 19:42:20] SERVICE ALERT: [illegible]V21;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 20%, RTA = 5.42 ms
[10-27-2004 19:32:10] SERVICE ALERT: NS[illegible]V21;PING;OK;SOFT;2;PING OK - Packet loss = 0%, RTA = 4.93 ms
[10-27-2004 19:31:20] SERVICE ALERT: NS[illegible]V21;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 20%, RTA = 5.68 ms
[10-27-2004 19:30:50] SERVICE ALERT: hac[illegible]ycore1;PING;OK;SOFT;2;PING OK - Packet loss = 0%, RTA = 5.87 ms
[10-27-2004 19:30:00] SERVICE ALERT: [illegible]ckleycore1;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 20%, RTA = 5.26 ms
[10-27-2004 19:22:00] Auto-save of retention data completed successfully.
[10-27-2004 19:16:10] SERVICE ALERT: [illegible]21;PING;OK;SOFT;2;PING OK - Packet loss = 0%, RTA = 5.82 ms
[10-27-2004 19:15:21] SERVICE ALERT: NSERV21;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 20%, RTA = 10.20 ms
```

These checks should keep your pager rattling off your belt. As you can see by the contents of libexec, there are hundreds of things you can watch, all from one interface, and configurable to your hearts desire. Let's see an HP OpenView do that (for the price).

2.3 Cacti

cacti

Overview

Cacti is a complete network graphing solution designed to harness the power of RRDTool's data storage and graphing functionality. Cacti provides a fast poller, advanced graph templating, multiple data acquisition methods, and user management features out of the box. All of this is wrapped in an intuitive, easy to use interface that makes sense for LAN-sized installations up to Enterprise PACS networks with hundreds of devices.

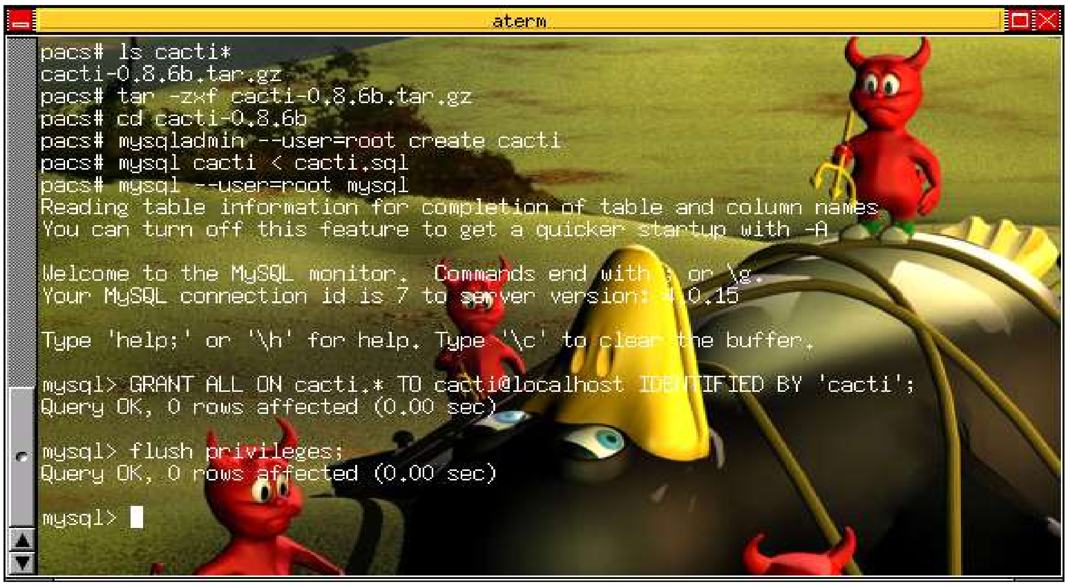
Installation and Configuration

Cacti requires a working mysql server (it has been known to work with others), php, rrdtool, and Net-SNMP. The install of this application is not difficult, and once it is up and running, you do all the configuration of the things you want to watch within the application itself.

Cacti comes with a poller, a php script that does all the data gathering from your PACS hosts. If you plan on monitoring a medium to large sized LAN with this application suggest you go out and grab cactid (http://www.cacti.net/cactid_info.php). It runs WAAAYYY faster than the php poller, and runs clean due to the fact it is coded in C and takes advantage of compilable SNMP librararies etc. The php poller works well however if you have a small number of things you want to trend.

Go out, get the latest build of cacti, create the database and run the sql script to build all of your tables. You are going to have to create a user in mysql and specify a password. In the screenshot below (Figure 7), they are represented by the user "cacti" and the password "cacti." After your db is configured, you will need to edit `include/config.php` and also plug in your database credentials. Its not apparent in the cacti documentation, but you will also have to add a system level user as well to run the poller from cron. Two directories will have to be created, rra/ and log/ for storage of the graphs and logs, they will have to be owned by the system level user you created as well. Either creat a link or copy your configuration into the root of your webserver, point your browser to http://yourbox/cacti/ and we are off to the races!

Figure 7

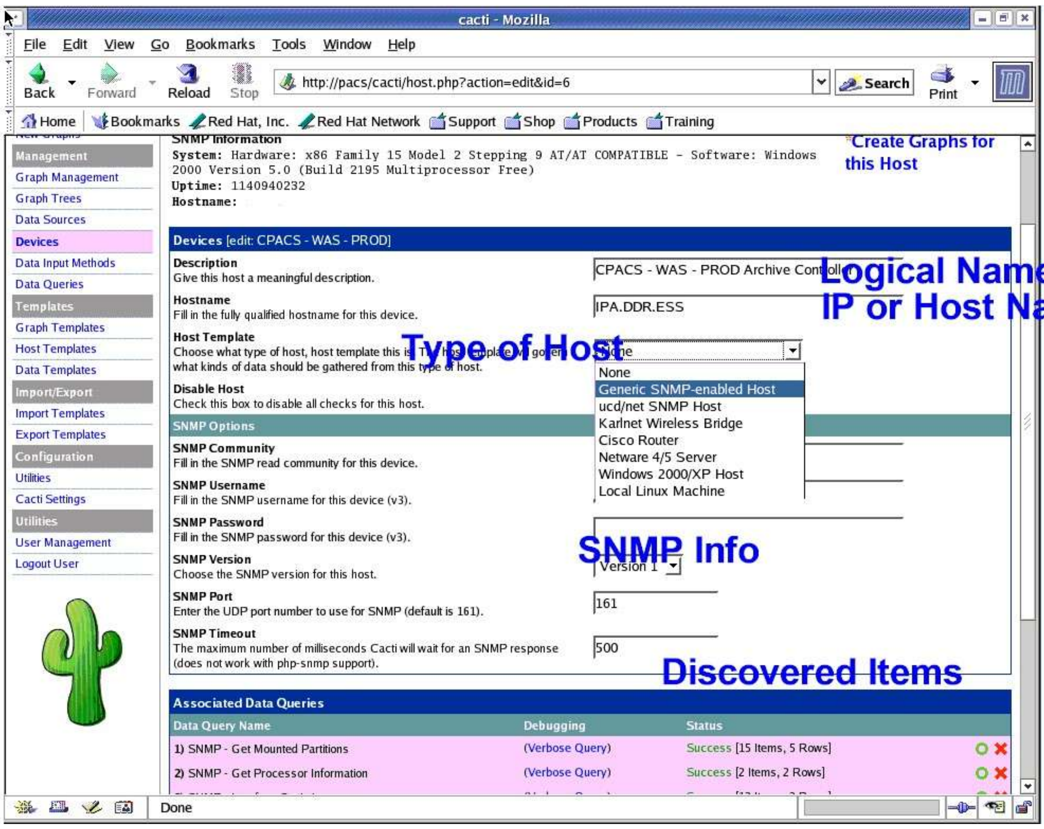


```
pacs# ls cacti*
cacti-0.8.6b.tar.gz
pacs# tar -zxf cacti-0.8.6b.tar.gz
pacs# cd cacti-0.8.6b
pacs# mysqladmin --user=root create cacti
pacs# mysql cacti < cacti.sql
pacs# mysql --user=root mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.0.15

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> GRANT ALL ON cacti.* TO cacti@localhost IDENTIFIED BY 'cacti';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> █
```

Its UP! Now what?

I am a veteran user of Cricket, another RRDTool based poller that generates graphs from config files. The configuration of Cricket is comparable to "eating glass" after you have seen host configuration in Cacti. To get started, login as the Admin, and select "create host" in the Console section. The interface is going to look like the following (Figure 8).

Figure 8



http://pacs/cacti/host.php?action=edit&id=6

```
SNMP Information
System: Hardware: x86 Family 15 Model 2 Stepping 9 AT/AT COMPATIBLE - Software: Windows
2000 Version 5.0 (Build 2195 Multiprocessor Free)
Uptime: 1140940232
Hostname:
```

Devices [edit: CPACS - WAS - PROD]

Description Give this host a meaningful description.	CPACS - WAS - PROD Archive Controller
Hostname Fill in the fully qualified hostname for this device.	IPA.DDR.ESS
Host Template Choose what type of host, host template this is. The host template will govern what kinds of data should be gathered from this type of host.	None ▾ (None / Generic SNMP-enabled Host / ucd/net SNMP Host / Karlnet Wireless Bridge / Cisco Router / Netware 4/5 Server / Windows 2000/XP Host / Local Linux Machine)
Disable Host Check this box to disable all checks for this host.	

SNMP Options

SNMP Community Fill in the SNMP read community for this device.	
SNMP Username Fill in the SNMP username for this device (v3).	
SNMP Password Fill in the SNMP password for this device (v3).	
SNMP Version Choose the SNMP version for this host.	Version 1
SNMP Port Enter the UDP port number to use for SNMP (default is 161).	161
SNMP Timeout The maximum number of milliseconds Cacti will wait for an SNMP response (does not work with php-snmp support).	500

Associated Data Queries

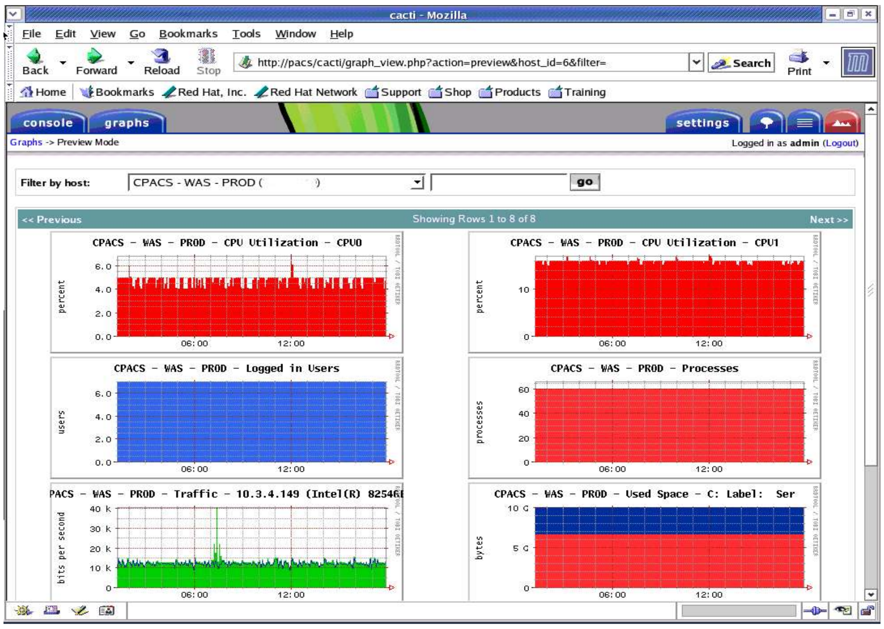
Data Query Name	Debugging	Status
1) SNMP - Get Mounted Partitions	(Verbose Query)	Success [15 Items, 5 Rows]
2) SNMP - Get Processor Information	(Verbose Query)	Success [2 Items, 2 Rows]

Now, just fill out the form!

(Note that not all monitoring in Cacti is done through SNMP, but for purposes of this document, I will adhere to it. You will need to know your SNMP setup on the target machine is actually running, the ACL's are in place for you to walk it, and you know the community string.)

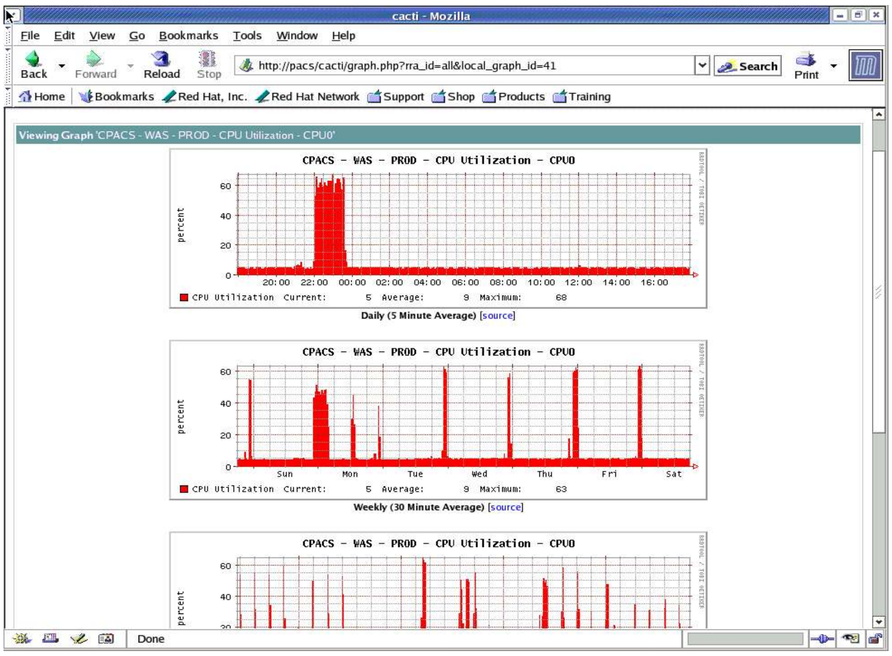
The above screen shot shows the setup of watching an archive device for Cardiology PACS that archives to a downstream Radiology Enterprise Archive. After we created the host, the bottom notes the “items” that cacti discovered... network interfaces, disk drives, processor utilization, logged on users, and memory usage... cool eh? Follow the prompts through the graphing section and wait about 10 minutes, while cacti starts to build you some graphs (Figure 10). Navigate from the console section to the graphs section, select your newly added host and check it out...

Figure 9


cacti - Mozilla
http://pacs/cacti/graph_view.php?action=preview&host_id=6&filter=
console
graphs
settings
Graphs -> Preview Mode
Logged in as admin (Logout)
Filter by host: CPACS - WAS - PROD (
go
<< Previous
Showing Rows 1 to 8 of 8
Next >>
CPACS - WAS - PROD - CPU Utilization - CPU0
CPACS - WAS - PROD - CPU Utilization - CPU1
CPACS - WAS - PROD - Logged in Users
CPACS - WAS - PROD - Processes
PACS - WAS - PROD - Traffic - 10.3.4.149 (Intel(R) 82546E
CPACS - WAS - PROD - Used Space - C: Label: Ser
percent
users
processes
bits per second
bytes
06:00
12:00

Nice! We now have a running graphical representation of system components running on your PACS... If you drill down deeper, selecting one of these graphs, it will give you a complete breakdown over the course of hours, days, weeks, months and years.

Clicking CPU utilization on the archive controller, can you guess what time we archive our Cardiology studies to PACS? 22:00!


cacti - Mozilla
http://pacs/cacti/graph.php?rra_id=all&local_graph_id=41
Viewing Graph 'CPACS - WAS - PROD - CPU Utilization - CPU0'
CPACS - WAS - PROD - CPU Utilization - CPU0
percent
20:00 22:00 00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00
CPU Utilization Current: 5 Average: 9 Maximum: 68
Daily (5 Minute Average) [source]
Sun Mon Tue Wed Thu Fri Sat
CPU Utilization Current: 5 Average: 9 Maximum: 63
Weekly (30 Minute Average) [source]
Done

Real World PACS Examples

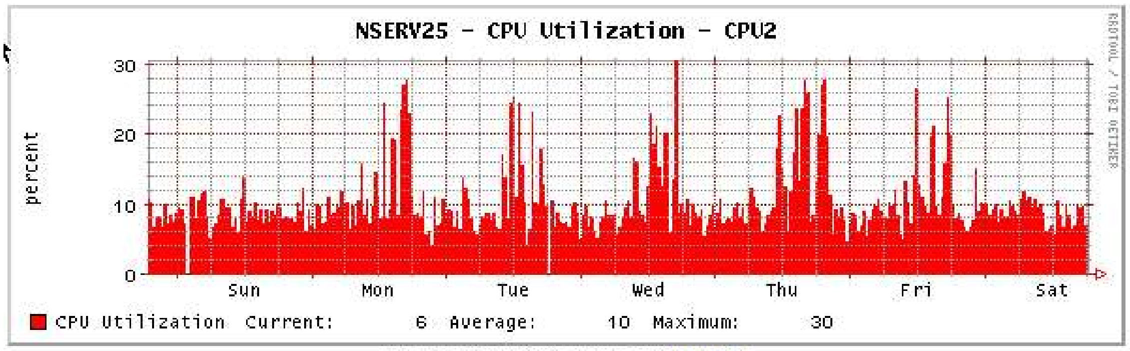
Our PACS systems have a “point system” in regards to servers and the load they can handle, it is based on an algorithm of memory and CPU and how intensive different modality types are on those particular components. For example, an IBM 345 server, can handle a total of 38 points. Multi-Slice CT's, PET/CT are valued as a “6” on the load of the server. The idea here is that you do not want to have any more than 6 (36 points total) of these modalities sending to this particular server or you will overload it. Moreover, it is difficult to guarantee any response time standard to a server that is taxed. Typically a vendor will set a threshold for CPU utilization and guarantee the performance of the box as long as the load does not exceed it. A load of 70% or less is our target for image servers in our environment. How do we keep an eye on the load of a particular server so we do not drop it to its knees by adding another sending modality?

Cacti!

Let's take a peak at an IBM 345,4WAY, SAN attached server with 3.5TB of storage. The sending modalities to this machine are an MRI, and a PET/CT scanner. Without the point system, lets just take a look at the trend in performance on the box.

CPU

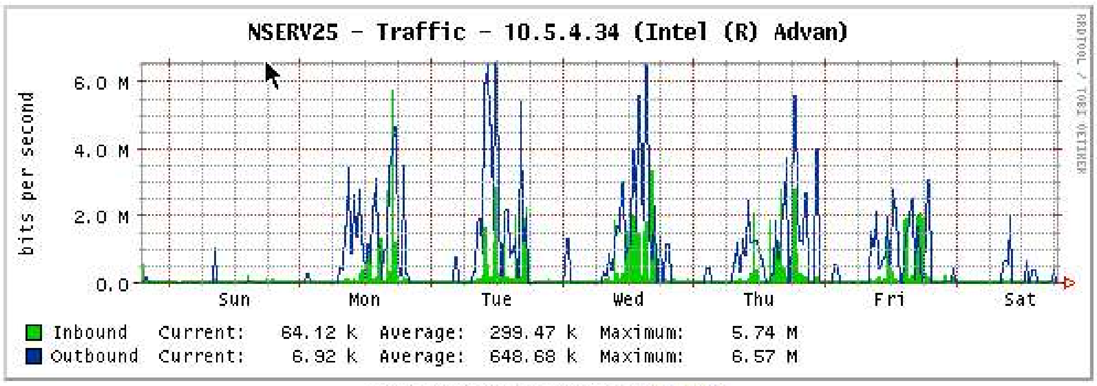
Looks good... maxed out at 30% in the early morning on Wednesday, averages 10% over the week.



Weekly (30 Minute Average) [source]

Network utilization

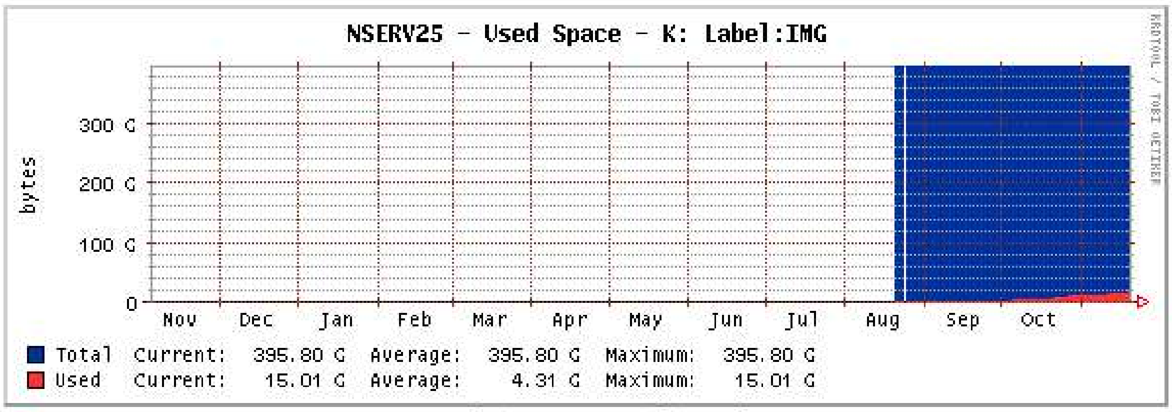
Maximum accepted throughput, approx 6Mb/s...its a over a WAN GigaMAN circuit too.



Weekly (30 Minute Average) [source]

Disk Space

Storage is a huge component of PACS. Cacti can help you monitor how much is used over time and allow you to see at what rate the space is being utilized. These particular LUNS reside on an IBM ESS800 (shark). Looks like this particular "LOCAL" has acquired 15GB of storage since it was turned up in August.



Yearly (1 Day Average) [source]

3 Conclusion

Well, there you have it. These solutions will watch your PACS when you aren't, and provide some tangible metrics to aid in planning or troubleshooting issues, or letting you know something is wrong before your Radiologists or Technologists do. I do not know of a commercial package that offers this type of visibility for the cost, or one that is "open" enough in nature to allow you to hack it to your needs. If you are the type that would want support for each of these products, there is professional services available for both Nagios®, and Cacti from services such as FindOpenSourceSolutions (http://www.findopensourcesupport.com/) or StackSys (http://www.stacksys.com).

Feel free to contact me if you have any questions about this document or the solutions in general. I am more than willing to help a fellow PACS support person exploit Open Source software to make their life easier in the PACS enterprise. I am typically better at hacking on the phone or over email than I am at writing documentation (as you probably figured out), so give me a shout and I will see if I can answer any questions this document didn't or any other questions you may have. I am typically hovering around Club PACS Western Michigan (http://www.clubpacswest-mi.net) or can be reached via email at ron.sweeney@gmail.com.

4 World Wide Web

TimTowTdi:

http://c2.com/cgi/wiki?TimTowTdi

Nagios®:

http://www.nagios.org

Cacti:

http://www.cacti.net

Smokeping:

http://people.ee.ethz.ch/~oetiker/webtools/smokeping/

StackSys:

http://www.stacksys.com

Club PACS:

http://clubpacs.mcw.edu/

Club PACS Western Michigan:

http://www.clubpacswestmi.net

DCMTK:

http://dicom.offis.de/dcmtk.php.en

OpenSolutionSupport

http://www.findopensourcesupport.com/

OpenRad:

http://www.openrad.com

5 Credits, Acknowledgments, License

5.1 Credits

Author : Ron Sweeney <ron.sweeney@gmail.com>

Author of the template : Gianluca Turconi <luctur@openoffice.org>

5.2 Acknowledgments and Thanks

I would like to thank my fellow PACS team members for tolerating the onslaught of notification and heads up pages this solution generates (at all hours of the night).

Many thanks to Tobi Oetiker (RRD Tool and Smokeping), Ian Berry (Cacti) and Ethan Galstad (Nagios®) for their contributions to the Open Source Community.

5.3 License