Thresh – a Data-Directed SNMP Threshold Poller

John Sellens – Certainty Solutions Inc.

ABSTRACT

Thresh is a simple SNMP [1] monitor, written in Scotty [2] (Tcl [3] with the Tnm extensions), which uses the UNIX file system hierarchy for configuration and data storage. Thresh compares SNMP variables to per-device thresholds or values, and issues notifications if any current SNMP variable values are unacceptable (or unexpected). Thresh can be used by itself, or as a complement to other network management and monitoring tools. Thresh can be thought of as fitting in between tools that generate immediate emergency alerts (such as Big Brother [4]) and trending and history tools (such as Cricket [5]).

Introduction

Virtually every computing system and network has some kind of monitoring mechanism in place these days, but in some cases the monitoring system consists only of users phoning and asking if there is something wrong with the network. For those interested in something a little more advanced or automatic, there are quite a few software packages available that do some form of monitoring. They range from the very simple (ping tests, etc.) to the very complex (large commercial packages that map networks, configure devices, and make your lunch), with various alternatives in between.

In the realm of "simple" monitoring software, packages can generally be divided into two types:

- Alarmers – Software that monitors connections, services, and so on, and sends out an alarm (mail, pager, smoke signal) as soon as it detects a problem. Some examples of alarmers are Big Brother [4], nocol [6], and Spong [7].
- Trenders and Trackers – Software that keeps history or trend data for future analysis or review. The most obvious examples here are MRTG [8] and Cricket [5].

A couple of years ago it occurred to me that there was another class of monitoring that didn't seem to be very well addressed – low to medium priority tracking of certain parameters and their values on computing systems and network devices. For example, you might want to track configuration changes, system or device reboots, or network interface status or change time. These are things that you may want to know, but which don't necessarily indicate an immediate problem and which may not be worth waking anyone up to investigate.

Thresh was created to provide this kind of monitoring. It tracks SNMP variables, compares them to threshold, pre-set, or last-observed values, and reports (typically via email) unexpected changes or out of range values.

Why Use Thresh and Not Something Else

I've become convinced that, ugly as it may sometimes seem, the Simple Network Management Protocol (SNMP) [1] should be the basis for just about every monitoring system. Virtually every network connected device either comes with an SNMP agent or can run one with only a modest amount of effort. Most SNMP agents can provide a vast amount of (usually) useful information, and most agents for general purpose computers can be extended to provide just about any data that you might want to have.

Some monitoring systems (such as Big Brother and Spong) rely, to a greater or lesser extent, on separate client agents, running on each system that needs to be monitored, with a system-specific reporting protocol between the agent and the management station. This approach can be somewhat limiting (it's hard to use on things like networking equipment for example), can result in some duplication of services (if you need an SNMP agent for other purposes), and limits both what you can do (they're often not extensible), and where (as your firewall may not be able to pass the particular protocol implemented by the software). Thresh avoids these kinds of problems by using only SNMP for communication.[1]

It's often useful to be able to track certain SNMP variables, but you don't always need to know about changes *immediately* – it's often good enough to hear about them the next time you read your mail. For example, the system.sysUpTime.0 SNMP variable gets reset every time a system or device gets rebooted – if you get notified every time that variable resets, you'll know if you've got a device reliability problem (or an extension cord that people keep tripping over). Similarly, you can generate disk capacity threshold warnings, network bandwidth warnings, network interface up/down notices, and so on.

With many monitoring systems, this kind of low-priority information can be hard to generate. Many of the most common freely available packages tend to have only a small number of notification or alert mechanisms, and are built with the idea that you're

[1]SNMP suffers in some situations from being a UDP based protocol, but its advantages more than make up for its few disadvantages.

monitoring vital services and networks – sometimes everything is assumed to be an emergency.

Thresh was created to address this kind of medium-level monitoring need.

There are a number of commercial monitoring systems available. Some, such as Spectrum [9], HP OpenView [10], and NetCool [11], are widely deployed and very well respected, and most of them can do (or can be made to do) most or all of what thresh does. However, the commercial packages tend to require a much larger monitoring infrastrucutre, and a much larger committment of time and money to implement, operate, and maintain. For small to medium sized sites, small, simple monitoring tools, like thresh, are often the best choice.

Implementation

Thresh was implemented in Tcl [3], using the Tnm network management extensions provided by the Scotty/Tkined [2] software. The Tnm extensions are a toolkit of Tcl procedures that make it easy to perform SNMP operations.

Tcl was chosen because it is well suited to this kind of task, and the Tnm extensions provided just the right functionality. At the time thresh was first contemplated, the SNMP modules for Perl were relatively primitive when compared to Tcl and Tnm.

Thresh has benefitted from the use of a scripting language, and the string and array manipulation routines provided by Tcl. Tcl allows the use of an interactive "shell" for testing and development. Being a traditionalist, I tended to develop incrementally, using the well-known edit, run, repeat cycle, rather than a more "modern" approach to program development. Thresh is currently about 700 lines of Tcl.

In retrospect, I'm still glad to have chosen Tcl in preference to Perl, C, awk, or Visual Basic.

Simplicity

One of the design and implementation goals for thresh was simplicity. That goal has been addressed in the following ways:

- One main program, thresh, and a very small number (currently one) of simple utility programs.
- Implemented in a well-designed scripting language (Tcl [3]), using a well-known and effective set of library routines (Scotty).
- Simple to run – requires no additional daemons (just a crontab entry), and no addtional software is required on remote systems[2].
- Requires no special userids or groups. (It would probably be useful to have a separate userid to run thresh, and perhaps a userid or group to own the config files, but that decision is left up to those installing and using the software.)
- Configuration is relatively straightforward, and involves only two simple file formats. The hierarchical configuration structure makes implementing default settings easy and flexible.

[2] Other than an SNMP agent which you should already have installed and configured anyway, and which likely came with your operating system.

Data Direction and Configuration

Thresh's configuration is "data-directed"[3] – it is configured using a hierarchy of directories and configuration files that is intended to reflect organizational structure and DNS naming conventions.

The default configuration assumption is that each directory in the configuration hierarchy represents an element of the DNS name of the devices being monitored. For example, the sub-directory named com/whizbang/admin/printer1 would usually contain the thresh configuration for the device with the DNS name printer1.admin.whizbang.com. The name configuration directive makes it easy to override this default behaviour, by setting the DNS domain or node name associated with a particular directory.

Configuration Variables

Each directory may contain a DEFAULTS file, which sets the various configuration variables (such as name, notifier, delay, community, etc.) which control thresh's behaviour. Thresh configuration variable names are case sensitive, and all include only lower case letters. Settings in a DEFAULTS file are in effect for that node and those lower in the hierarchy, unless overridden by a lower DEFAULTS file.[4] Thresh configuration variables can also be set on the command line, in which case they override any other settings for the given variables. Figure 1 shows a sample DEFAULTS file.

```
verbose = true
name = mydomain.net
community = hello
mib = /usr/local/mibs/ascend.mib
# big network, long timeout
timeout = 20
notifier = threshmail jsellens
syslog = local1.info
```

Figure 1: A sample DEFAULTS file.

Thresh's configuration variables include:

- walkonly – Walk the data hierarchy, listing the nodes and variable references found, but not querying, reporting, or logging.
- ignore – Defines a list of glob patterns of file and directory names to ignore. Setting

  ```
  ignore = *
  ```

  for example, ends up ignoring the data hierarchy below a given point.

[3] Or, at least, my definition of data-directed – I looked unsuccessfully for a more commonly accepted definition of the term.

[4] One could claim that this is similar to the class inheritance rules in object-oriented programming languages. Or not.

- notifier – The command line to run to send notification messages – the message text is provided as the standard input to the command. This can be set differently for different parts of a hierarchy, which makes it possible to assign responsibility for different nodes to different people or groups.
- frequency – The interval (in minutes) before otherwise identical notifications may be sent.
- describe – Whether or not to include a variable's MIB description field in notification messages. Description fields often end up being fairly generic, but including the description in messages can help make the notifications a little more self-documenting.
- syslog – A facility.level pair for logging messages to syslog.

The other configuration variables are described in the included *threshvars*(5) man page.

SNMP Variables

All other files in a directory are expected to contain a list of SNMP variables to monitor for that particular device, with comparison indicators and expected or threshold values. Thresh currently lacks a file inclusion mechanism, but the use of symbolic links makes it easier to manage the configurations for multiple, similar devices. A sample configuration file is shown in Figure 2.

The SNMP variable names used in a configuration file can be any string that Scotty will recognize as a particular MIB variable. They can be fully-qualified names, such as iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0, unique substings with common prefix elements removed, as in system.sysUpTime.0, or SNMP object identifiers (OIDs), such as 1.3.6.1.2.1.1.3.0. OIDs aren't always the best choice, as they are typically somewhat more cryptic for the casual reader.

The first letter on each configuration line, C, G, I, L, S, or V, indicates the comparison to be made:

- C: Changeable – the variable's value may change, but should be reported each time it changes. This is useful for semi-static data, or for monitoring things such as device interface status changes.
- G: Greater than – the variable is reported if its current value is less than or equal to the specified value.
- I: Increasing – the variable is reported if its current value is less than its previous value. This is handy for watching for reset times, such as the system.sysUpTime.0 variable resetting when a device (or agent) restarts.
- L: Less than – the variable is reported if its current value is greater than or equal to the specified value.
- S: Static – the variable is reported if its current value is not exactly equal to the specified value. If no value is specified, then it is compared against the first-retrieved value of the variable. This is useful for monitoring things that should never (or almost never) change, such as system.sysName.0.
- V: Variable – the value can be anything, but it is queried and tracked to allow for later investigation or review.

The final field on some lines is the threshold value to compare against – a threshold value is required for G and L, optional for S, and not allowed for C, I and V. If a value for an S comparison is not provided in a configuration file, the first value for that SNMP variable retrieved from the device is saved and used as the "normal" value for the variable.

The configuration mechanism has proven to be quite flexible and easy enough to deal with, though some form of file inclusion mechanism would make some configurations simpler to create and maintain.

Notifications

Thresh provides a flexible mechanism for notifications. For each device described in the configuration hierarchy, if thresh determines that something needs to be reported, it formats a message and pipes it into whatever "notifier" program has been specified by the configuration variables. Thresh also keeps a copy of the message for internal reference when it is next

```
# this is a comment
S system.sysDescr.0
S system.sysContact.0
I system.sysUpTime.0
C interfaces.ifTable.ifEntry.ifDescr.5
C interfaces.ifTable.ifEntry.ifAdminStatus.5
C interfaces.ifTable.ifEntry.ifOperStatus.5
G ucdavis.memory.memTotalReal.0                90000
G enterprises.ucdavis.memory.memAvailReal.0    3000
L loadTable.laEntry.laLoad.1                   1.20
L loadTable.laEntry.laLoad.2                   1.50
L loadTable.laEntry.laLoad.3                   2.00
V snmp.snmpInPkts.0
```

Figure 2: A sample configuration file.

run. If thresh observes the same problems (i.e., an identical notification message) the next time it is run, it will only send another notification if the specified frequency time has passed. If it sees a different set of problems when next run, it will report the complete current set of problems, regardless of whether or not the normal delay time has passed.

If the syslog variable is set, thresh will also generate syslog messages for SNMP variables that are out of range, in addition to the normal notifications. Messages formatted for syslog are deliberately terse, and include the node name, comparison type, SNMP variable, the normal or threshold value, and the current value.

Status, History and Logging

For each device in the hierarchy, thresh will create a configuration subdirectory named .thresh, as a convenient location to store the data it generates. Thresh maintains status, history and logging information, both for internal use and to make it possible to review past changes in state. It tracks the previous values of the SNMP variables (in .thresh/status_*), the last notification message sent (in .thresh/last_complaint), the date and time of last contact with the device (in .thresh/last_response), and a log of when variables were found to be outside their "normal" ranges (in .thresh/log_*).

The status and log files are named for the configuration files that they are related to, with a prefix of status_ or log_ added to the configuration file's name.

Currently, there are no really interesting ways to access and use this data – you're more or less stuck with using some paginator program to view the files. I should note that there is no built-in mechanism for log file rotation.

Integration with Other Systems

Thresh can be used effectively as a standalone, isolated monitoring tool, but it can also be integrated with other logging or reporting systems. Thresh's core functionality is polling SNMP variables, comparing against pre-determined thresholds, and generating messages for distribution. Integration with other tools can be accomplished in two ways:

Syslog
: Thresh can be configured to record out of range values to syslog, which provides an easy interface to any existing syslog watcher.

Custom Notifiers
: By setting the notifier configuration variable, thresh's alert messages can be trivially piped through arbitrary custom processes, that can record, mail, or dispatch as appropriate.

The message format is fairly consistent, simple, and relatively easy to parse. Additionally, the internal thresh code which actually generates the messages would be easy to change if a specific output format was required.[5]

Scalability

Thresh is not arbitrarily scalable to huge numbers of devices and variables being monitored. Beyond a certain point, you will start to run into problems such as:

- Configuration complexity – a file and directory based configuration mechanism works fine up to a point, but beyond that you need configuration generators and a real database.
- No inherent parallelism – beyond some number of hosts and variables, you won't have enough time to poll everything you want to poll within the time interval you would prefer. You can deal with this to some extent by running multiple parallel instances of thresh, but that adds administrative complexity.
- The default email notification mechanism can quickly get out of hand, if things start breaking. It is possible to use a smarter notifier, or to send notifications to different users or aliases based on location in the hierarchy, but that can get complicated. It would be possible to use a notifier that inserts messages into your management console, but if you have one of those, you may also already have more advanced commercial monitoring software.

Installation

Installation of thresh is very straightforward.

- You need relatively current versions of Tcl and Scotty (at least 2.1.x) on your monitoring host.
- Set the "hash-bang" line at the top of the thresh script to point to your Scotty installation.
- Install the script and man pages in the "usual" places.
- Set a cronjob to invoke thresh periodically, with a command line something like:

```
thresh topdir=/path/to/data
```

- And simply create a data hierarchy that describes the hosts, devices and variables that you wish to monitor.

The last step is admittedly somewhat more complicated and time-consuming than the others, but that is pretty much unavoidable. The distribution will include some sample configurations.

Enhancements

There are a number of enhancements to thresh that should probably be made:

- Thresh currently only supports SNMP V1 – it would be useful to enable all versions of SNMP supported by the Tnm extension.

[5] The msgformat configuration variable provides a rudimentary and somewhat unsophisticated mechanism for customizing the notification messages.

- Syslog logging is implemented through the external *logger*(1) command – it should be re-implemented using an internal library call.
- There should be some form of ''include file'' mechanism for configuration files, to reduce the need for the use of symbolic links to implement common configuration settings.

Lessons and Problems

Thresh seems to serve to illustrate a few useful lessons:

- Sometimes a variety of tools are needed to implement a complete monitoring ''solution''.
- Simple tools are often quite useful.
- ''Data-directed'' design can be quite effective.
- SNMP can provide all sorts of useful information that can be difficult or impossible to obtain using other mechanisms.
- Tcl and the Tnm extensions are very useful tools.

Some of these also illustrate problems, the most obvious being that you probably need more than one monitoring tool, since no one tool is likely to do everything that you need done.

Generally, thresh has proven useful and fits nicely into an effective monitoring toolkit.

Availability

Thresh is ''freely'' available through http://thresh.sourceforge.net/ or http://www.generalconcepts.com/resources/ .

Author Information

John Sellens is the General Manager for Certainty Solutions in Canada, based in Toronto. (Certainty Solutions was previously known as GNAC – Global Networking and Computing.) Prior to joining Certainty Solutions, he was Director of Network Engineering at UUNET Canada, and was a system administrator at the University of Waterloo for 11 years. He has a master's degree in Computer Science from the University of Waterloo, is a Chartered Accountant, and is a semi-regular contributor to *;login:*. John, Joanne, and their delightful children live in Unionville, Ontario. Contact him at <jsellens@certaintysolutions.com>.

Bibliography

[1] J. Case, M. Fedor, M. Schoffstall, and J. Davin, *A Simple Network Management Protocol (SNMP),* Network Working Group, May 1990, RFC 1157, STD 15.

[2] J. Shönwälder and H. Langendörfer, ''Tcl Extensions for Network Management Applications,'' in *Tcl/Tk Workshop*, pp. 279-288, USENIX and Unisys, Inc., Toronto, Canada, July 6-8, 1995.

[3] John K. Ousterhout, ''Tcl: An Embeddable Command Language,'' in *USENIX Conference Proceedings*, pp. 133-146, USENIX, Washington, D.C., January 22-26, 1990.

[4] Sean MacGuire and Robert-Andre Croteau, *Big Brother FAQ*, http://www.bb4.com/ .

[5] Jeff R. Allen, ''Driving by the Rear-View Mirror: Managing a Network with Cricket,'' in *First Conference on Network Administration (NETA '99)*, pp. 1-10, USENIX, Santa Clara, California, April 7-10, 1999.

[6] Vikas Aggarwal, *NOCOL – Network Operation Center On-Line*, http://www.netplex-tech.com/software/nocol/ .

[7] Stephen L. Johnson, *Spong – Systems and Network Montoring*, http://spong.sourceforge.net/ .

[8] Tobias Oetiker, ''MRTG – The Multi Router Traffic Grapher,'' in *Twelfth Systems Administration Conference (LISA '98)*, p. 141, USENIX, Boston, Massachusetts, December 6-11, 1998.

[9] Aprisma Technologies Inc., *Spectrum Network Monitoring and Management System*, http://www.aprisma.com/ .

[10] Hewlett-Packard Company, *OpenView Monitoring and Management Software*, http://www.openview.hp.com/ .

[11] Micromuse Inc., *Netcool Monitoring and Reporting Suite*, http://www.micromuse.com/ .

[12] University of California, Davis, *UCD-SNMP distribution*, http://ucd-snmp.ucdavis.edu/ .