Self Configuring Network Monitor (SCNM)

for version 0.9-beta, 4-1-2004

Data Intensive Distributed Computing group
Contact: Goujun Jin (g_jin@lbl.gov)
http://dsd.lbl.gov/SCNM/

This manual is for the Self Configuring Network Monitor (SCNM) (version 0.9-beta, 4-1-2004).

It is acknowledged that the U.S. Government has rights to this software under Contract DE-AC03-76SF00098 between the U.S. Department of Energy and the University of California.

This software is provided as a professional and academic contribution for joint exchange. Thus, it is experimental, and is provided "as is", with no warranties of any kind whatsoever, no support, no promise of updates, or printed documentation. By using this software, you acknowledge that the Lawrence Berkeley Laboratory and Regents of the University of California shall have no liability with respect to the infringement of other copyrights by any part of this software.

For further information about this notice, contact:

Brian L. Tierney Bld. 50B, Rm. 2239, Lawrence Berkeley National Laboratory, Berkeley, CA, 94720 email: bltierney@lbl.gov

1 Introduction

The Self-Configuring Network Monitor (SCNM) is designed to allow network users to passively monitor their own traffic stream as it traverses the network without requiring special privileges or intervention of by network administrators. A user wishing to monitor a particular stream sends a special request packet through the network between the same two hosts that are the source and destination of the traffic to monitor. A request packet automatically activates monitors along the path. Since the request is sent to the application destination endpoint not the SCNM monitoring host, the user does not need to explicitly know the locations and identities of the SCNM hosts on the path. All the SCNM monitoring hosts listen for these special UDP request packets on a well-known port. Each SCNM host along the data path capture the activation packet as it travel past its interface. The request packet specifies the characteristics of the traffic to monitor including source, destination, and port(s) of the traffic.

The hardware infrastructure for SCNM is designed to be easy to install and administer securely. Figure 1 shows a typical configuration between two application hosts, or end hosts, across a WAN. A read-only tap is placed on the DMZ between the site border router and the ISP router, and the monitoring host, which we call the SCNM monitoring host, is connected to this tap. Since this is a read-only tap, the SCNM machine will not be able to generate any traffic through this interface. The SCNM monitoring host has an additional network interface (usually on an internal network) used for administering the SCNM host and transmitting monitoring output data. The SCNM host runs the FreeBSD operating system and does not by default run any services an sshd can be configured if desired to allow remote maintenance access. The kernel network drivers in a SCNM host are a modified version we have created to allow us to timestamp packets down on the network card and appropriately moderate interrupts to allow the machine to keep up with high traffic rates. SCNM will work with any type of Ethernet, and has been optimized for GigE over fiber. SCNM will also work with bonded GigE.

SCNM has proven to be very useful for debugging network, protocol, and application performance problems. For sample results see Figure 2.

For more details read our "Passive and Active Monitoring Workshop (PAM) 2003 paper on SCNM:

http://www-didc.lbl.gov/papers/SCNM-PAM03.pdf

2 SCNM SysAdmin Guide

SCNM packet capture deamon only works with FreeBSD. The client-side tools all work with any Unix system.

To get SCNM source code or binaries, email Goujun Jin.

After getting the tar file, follow the instructions in the file README.txt to build and install the software. Please note the LICENSE.txt file in the top-level directory, explaining the terms of use of the software.

2.1 SCNM Capture Host

Passive optical tap

We are currently recommending people use the multimode SX 50/50 tap from Netoptics.

Packet Capture Host

We recommend the following hardware configuration (as of mid-2002)

- one of the DDR266, serverworks chipsets boards such as the X5DL8-GG
- Adaptec AIC-7902 dual channel Ultra 320 SCSI adapter onboard
- Intel Xeon 2.8 GHz CPU
- Four DDR 266 (each 512 MB) 2GB total
- ST373405LW 73GB Cheetah
- one Syskonnect V1.x SK-9844SX (dual ports) or two SK-9843SX (single port) PCI 64-bit GigE NIC

Notes:

- Our testing shows there's an I/O performance penalty for adding a second processor. Even if you run a single processor kernel, you lose something like 5%. If you run a SMP kernel it's even worse. It's ok to use dual processor motherboards (they usually have more slots and other features) but we don't recommend using two processors.
- Normally when specing a packet capture system we go for the best I/O performance, then the fastest memory architecture and then the fastest single processor cpu.
- We recommend against RDRAM. Performance is probably similar to DDR but it's more expensive than DDR (and even intel makes DDR boards now, an indication that RDRAM is on the way out).

For more advice on what hardware to use, email us.

2.2 FreeBSD Modifications and Configuration

The directory ‘SCNM2/bond/kernel.patch/’ contains two patches, one for Berkeley Packet Filter (/sys/net/bpf.*) and one for the Syskonnect driver (/sys/pci/if_sk*).

To install these patches, do the following:

```
cd SCNM2/bond/kernel.patch
./patch-kernel.sh
reboot
```

For more information see: http://www-itg.lbl.gov/Net-Mon/FreeBSD_mods.html

When configuring a new SCNM host or moving a SCNM host to a new location, the ‘/etc/rc.conf’ file needs to be modified. The following fields in ‘/etc/rc.conf’ and related files must be changed:

```
### Network routing options: ###
defaultrouter="DEFAULT_ROUTER_IP"

### Basic network and firewall/security options: ###
hostname="hostname.domain"
ifconfig_bge0="inet IP/NETMASK" # this interface may vary

firewall_enable="YES"
firewall_type="/etc/ipfwrules" # change this file

...
### Network Time Services options: ###
ntpdate_enable="YES"
ntpdate_flags="-sb NTP_SERVER_1,MORE_NTP_SERVER_ARE_SEPARATED_BY,,,"

# 1000 ms is for average packet size around 390-500 bytes.
# For capturing small packets, reduce the interrupt time (minimum 350)
# For capturing large packets, increase it (maxmimum 3000)
sk_interrupt_mod="1000"
```

2.3 SCNM Ports

SCNM uses TCP port 6789 for sending packet traces, and UDP port 5050 for activation packets. Both ports need to be open in both the incoming and outgoing directions.

2.4 Software Installation

SCNM uses two processes: ‘fcd’ (filter control daemon) and ‘pcapd’ (packet capture daemon), to set up a filter for a specific path and monitor its traffic. These are both started using the daemon.bat script.

```
daemon.bat start interface {-cd cache_dir -dddi | -dddd} [-bi ] [-cl #]
```

This SCNM capture daemons should be started at boot time, using the script ‘/usr/local/etc/rc.d/z-scnmd.sh’ To start scnm daemons with default options, do:

```
/usr/local/etc/rc.d/z-scnmd.sh start
```

For example, to start the capture daemons, watching for traffic on interface sk0 in both directions:

```
daemon.bat start sk0 -cd . -dddi -bi
```

Without specifying the interface "sk0", ie., "daemon.bat start", the daemon will monitor the primary interface configured.

Command line options:

'-cd cache directory': used for caching data files between 'pcapd' and 'fcd'. This is for the -dddi method, and the cache directory must be on a large partition that has at least 60 MB/s disk I/O.

'-dddd' daemon deliver data directly to client (synch delivery method). This only works if the NIC (usually fxp0) is on a dedicated network.

'-dddi' daemon deliver data indirectly to client (asynch delivery method). This is the default option.

'-bi' bi-directional capture. Without this option, capture daemon will only monitor the traffic from source to the destination. This is the default.

'-cl capture length': "-cl 0" tells the daemon to record/send variable length data records. The varible length mode is disabled in both -dddd and -dddi methods. The default length is 80 bytes, minimum is 28, and maximum is 128. The default capture length of 80 bytes to suitable for most types of TCP analysis. To ensure capture of the entire TCP header, including all TCP options, use "-cl 86". A 52-byte capture length will be enough for simple TCP options.

To stop the capture daemons:

```
daemon.bat stop
```

To see if any SCNM daemon is running, and what options they are using:

```
daemon.bat stat
```

The current SCNM access control list (ACL) method is controlled by fcd as follows. When fcd starts, it looks for a default configuration file – called 'act_auth.conf' (activation authorization) for a list of host names or IP addresses. If this list exists, monitoring data may only be forwarded to these hosts. If no configuration file is present or the file is empty, then the ACL will be empty and data is allowed to be sent to the source host (activation host) only.

An alternative configuration file can be specified on the fcd command via the -secu flag. The configuration (authorization) can be changed dynamically. The system administrator can modify 'act_auth.conf' to add or delete a host, then send a HUP signal to the fcd daemon to reload the ACL.

3 SCNM User Guide

SCNM includes the following user-level tools for capture and analysis of packet headers:

- monitord
- apc
- fc2td
- SCNMPlot
- fc2xp/fc2xg

Quick Start Guide

The folllowing steps are used to activate SCNM captures:

- Start the data collector (`monitord`)

  ```
  monitord &
  ```

- Use the `apc` (Activation Packet Control) to send an activation packet to all SCNM hosts on the path.

  ```
  apc -dt www.abc.com 80
  ```

- Next, to look at the results you have several options:
  1. convert the SCNM data file to standard tcpdump format using `fc2td`, and then use tcptrace to generate desired data for xplot or SCNMPlot. For example:

     ```
     % fc2td < ???.dat > scnm-tcpdump.dat
     % tcptrace -Sl tcpdump.dat
     % xplot a2b...xpl
     or
     SCNMPlot file1.xpl file2.xpl ...
     ```

  2. use `fc2xp` to generate desired data for `xplot`. E.g.:

     ```
     fc2xp -sn -s 2 VR555_???.dat | xplot
     ```

  3. use `fc2xg` to generate desired data for `xgraph`. E.g.:

     ```
     fc2xg -sn -s 2 VR555_???.dat | xgraph -m -nl
     ```

3.1 monitord

To collect packet headers from the SCNM host, do the following:

First, start monitord on a host where the packet headers will be sent to:

```
monitord &
```

‘`monitord`’ writes out data to file named ‘`FC_N_SRCIP:SRCPORT_DSTIP:DSTPORT.EXT`’, where:

- N is a unique number generated by (SCNM-CFDFCD)
- SRCIP is the src ip in dotted decimal format
- SRCPORT is the src port
- DSTIP is the dst ip in dotted decimal format
- DSTPORT is the dst port
- EXT is either hdr if the file contains header info, or dat if the file contains actual SCNM-CFDFC data

An example file might be named:

```
FC0160598_104.101.32.200:43196_80.87.164.90:61497_80ce000.dat
or
VR555_13.23.200.111:0_13.23.200.244:1901_80ce000.dat
```

The file name will start with either FR (for fixed record) or VR (for variable length records).

The file name should look like this:

```
FC0160598_104.101.32.200:43196_80.87.164.90:61497.dat
```

if it looks like:

```
FC0160598_104.101.32.200:43196_80.87.164.90:61497_80ce000.dat
```

(notice the 7 extra chars near the end), it is a ’temporary’ file, and will be renamed without the 7 extra chars when the session finishes.

3.2 apc

‘apc’ sends an "activation packet" from the client host:

```
apc -dt destination port_to_watch
```

For example:

```
apc -dt www.abc.com 80
```

To stop monitoring (before it times out)

```
apc -dt destination port_to_watch -da
```

As long as ‘apc’ is running, the capture daemon will continue to capture packet header information of streams specified by ‘apc’. If ‘apc’ is stopped, the capture daemon will stop capturing in 15-20 minutes. If ‘apc’ is restarted within 15 minutes with the same option, the monitoring will not be interrupted. If another ‘apc’ is started with different destination host or port, a new monitor will be generated and a new file will be written by monitord.

3.3 fc2td: convert SCNM trace files to tcpdump format

SCNM packet trace files are slighly different than tcpdump files (SCNM traces are more compact).

Use fc2td to convert a SCNM data file to tcpdump file.

Then you can use tcptrace to look at the tcpdump data file. The "-r" option will convert tcpdump file back to SCNM formatted data.

```
% fc2td < ???.dat > scnm-tcpdump.dat
```

3.4 tcptrace

tcptrace can be used to analyze tcpdump files.

For more info, see: http://www.tcptrace.org/

3.5 SCNMPlot

SCNMPlot can be used to plot multiple tcptrace files at the same time.

For more info see: http://www-didc.lbl.gov/SCNM/SCNMPlot.html

3.6 fc2xp and fc2xg

fc2xp generates a file for xplot from the SCNM trace file

fc2xg generates a file for xgraph from the SCNM trace file

For example:

```
fc2xg [options] ???.dat | xgraph -m
```

By default, fc2xg generates packet rate data for TCP traffic, and extracts only the first stream found in the data file. It also prints out all socket information about the data file. For example:

```
% fc2xg VR555_???.dat > /dev/null
 First port 1025 1901
 1 new port 1026 1901
 2 new port 1027 1901
 3 new port 1028 1901
 4 streams: out of order seq = 28 of 3847 (6168-2321) bad ack 0
  719 ack for data stream  1 (port 1025 1901)
  670 ack for data stream  2 (port 1026 1901)
  482 ack for data stream  3 (port 1027 1901)
  450 ack for data stream  4 (port 1028 1901)
```

fc2xg shows that this data file contains 4 TCP streams from source ports 1025-1028 to destination port 1901. According to this information, user can select which stream to graph. For example:

```
% fc2xg -sn -s 3 -n 2 VR555_???.dat | xgraph -m -nl
```

This example will graph sequence numbers (-sn) of two TCP streams (-n 2) starting from stream number 3 (-s 3). That is, streams 3 and 4 (source port 1027 and 1028) will be displayed. If the captured data contains non TCP traffic, "-all" option needs to be used. Common options:

```
-all    process all type of traffics.
-agt    plot aggregated throughput (all traffics) with 10 ms window.
-bits   plot packet size.
-sn     plot sequence number (TCP only).
```

For more options, use ‘`-help`’.

4 SCNM Network Engineer Guide

TBD: Description of how SCNM might be used to track down network problems goes here

5 SCNM Security Model

The SCNM request packet specifies the characteristics of the traffic to monitor including source, destination, and port(s) of the traffic. We use a thirty-two bit magic number field to permit quick rejection of spurious packets on the activation port. The activation packet format version and the sequence number of the activation packet uniquely identify this request and how to interpret it. Traffic type provides the IP protocol number of the type of traffic to monitor (e.g. UDP or TCP). The rest of the packet contains parameters specifying the characteristics of the traffic to monitor such as the source and destination addresses and ports used by the traffic. For example, the destination address and port are specified using a parameter that is six bytes in length with the first four being address and the second two being the port. If a monitoring request is accepted, the SCNM will configure a Berkeley packet filter to capture headers from the corresponding packet stream. Each request packet has a limited life-time. The user must periodically send additional request packets to maintain the monitoring. This allows for graceful recovery from crashes of the end host requesting the monitoring. Monitoring output data is sent back to the application data source or destination host.

The core of the security model revolves around the concept that a user is allowed to monitor her own data. In order to be accepted by the SCNM host, the activation packet must be traveling between the source and destination of the traffic to be monitored. The SCNM host verifies this by comparing the request parameters with the source and destination in the IP header of the activation packet. Also, the SCNM host is only willing to send resulting data to the source or destination of the monitored traffic. Thus, although spoofing of the IP source and destination might result in an extra stream being monitored, the resulting monitoring data will not be sent to the spoofing host. Also, since the monitoring data is sent over a TCP connection to the destination, it will only be sent if the host is listening for the results. Each SCNM host also maintains a local audit log of all monitoring requests.

Access control lists can be configured at an SCNM host to allow an individual site to limit the types of requests and monitoring data destinations allowed. The current SCNM access control list (ACL) method works as follows. When the filter control daemon (`fcd`) starts, it looks for a default configuration file – called ‘`act_auth.conf`’ for a list of host names or IP addresses. If this list exists, monitoring data may only be forward to these hosts. If no configuration file is present or the file is empty, then the control list will be empty and data is allowed to be sent to the source host (activation host) only.

In the current design, we assume that all users with access to an end host are allowed to monitor traffic to or from that host. If this turns out to be an issue, one can limit the ability of users to activate the monitor by using a privileged port for the activation packets. In this case, the end user would need to have root access to request monitoring of traffic to or from that host.

6 SCNM Utilities

6.1 scnm-index

When capturing the packet headers of high bandwidth flows, the trace files quickly get very large and hard to analyze or visualize. One may want to try to only capture small files, or use tcpslice to break large files into smaller, more managable pieces. Another solution is to use scnm-index, which generates an index file for speeding the data search.

A SCNM index file contains following information:

```
% scnm-index FC01V555:???.dat
This command will create an index file called FC01V555:???.dat.idx

FC01V555:???.dat.idx is constructed as:
Header -- 16 bytes
     magic number
     record type
     unused field
     time interval [default to 10 sec.]
Index data array[] -- 24 bytes each
     time stamp
     offset
     starting record #
     data in previous block
```

scnm-index can be also used to display the index file. For example:

```
        % scnm-index SCNM.dat -ti 0 400000 ; scnm-index SCNM.dat.idx -pi
built 13 indices for 30409 records in SCNM.dat
index:   offset  start-at records  (us)    date < interval = 0.400000 sec.>
    1:        32       1       1 (807525) Tue May 28 15:17:30 2002
    2:    190425    2307    2306 (207608) Tue May 28 15:17:31 2002
    3:    380846    4610    2303 (607611) Tue May 28 15:17:31 2002
    4:    565053    6837    2227 (  7624) Tue May 28 15:17:32 2002
    5:    781423    9455    2618 (407860) Tue May 28 15:17:32 2002
    6:    972701   11770    2315 (808266) Tue May 28 15:17:32 2002
    7:   1149483   13912    2142 (208548) Tue May 28 15:17:33 2002
    8:   1334713   16156    2244 (608885) Tue May 28 15:17:33 2002
    9:   1513789   18323    2167 (  8959) Tue May 28 15:17:34 2002
   10:   1724334   20876    2553 (409147) Tue May 28 15:17:34 2002
   11:   1959483   23724    2848 (809154) Tue May 28 15:17:34 2002
   12:   2183563   26439    2715 (209395) Tue May 28 15:17:35 2002
   13:   2404331   29118    2679 (609802) Tue May 28 15:17:35 2002
```

6.2 bpfbond

`bpfbond` is a user level program to bond network interfaces together to capture packets for traffic analysis. Note that this command only works of the kernel patch ‘`bond/kernel.patch`’ has been applied.

SYNOPSIS

```
bpfbond  [-bs] [-h] [-v] [-V]
bpfbond   interface pair(s)
bpfbond  -un  [interface(s)]
```

Examples

```
bpfbond  [-h] shows all options and descriptions.

bpfbond  [-V] show release date/version

bpfbond  [-v] verbose

bpfbond  [-bs]     shows current bonding status (bonded streams)

bpfbond interface pair(s)    bonds given interface pair or pairs. Any
interface that is in use will cause bpfbond failure and exit with EBUSY.
bpfbond stream pair(s)  will bond streams together.  For example:
  % bpfbond   sk0 sk1
  % bpfbond   sk2 sk3
  % bpfbond   -bs
bonded stream0: sk3 (idle) -> sk2
bonded stream2: sk1 (idle) -> sk0

% bpfbond   sk1 sk3
% bpfbond   -bs
bonded stream0: sk1 (idle) -> sk0 -> sk3 -> sk2

bpfbond -un   will unbond all bonded streams
bpfbond -un stream(s)/pair(s)     will unbond given stream(s)

For example:
  % bpfbond  -un sk0 sk3
  unbonded stream0: sk1 (idle) -> sk0 -> sk3 -> sk2


  unbonding interfaces
  % bpfbond -un sk4

  unbonded stream0: sk5 (idle) -> sk4

  % bpfbond -bs
       bonded stream0: sk1 (busy) -> sk0 -> sk3 -> sk2

  % bpfbond sk0 sk5
      bonded stream0: sk1 (busy) -> sk0 -> sk3 -> sk2 -> sk5

  In this case, a single interface -- sk5 -- was bonded to a two-pair
  stream to result in all five (5) interfaces being bonded together.
```

7 SCNM Troubleshooting

TBD: SCNM troubleshooting advice will be added here.

8 SCNM Programmer Documentation

The Selfconflib library contains the methods that create and send activation packets. For information on using this library, see: http://www-didc.lbl.gov/SCNM/activation_lib.html

Index