SAN Lessons Learned

Andy Loftus and Chad Kerner

National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, Illinois

Introduction

This document is written with the intent to provide insight into real world issues of building and managing Storage Area Networks (SAN). It does not discuss industry accepted best practices; these can be found in any good SAN book, and it is assumed that this material has already been covered. Examples and suggestions provided here are built upon the afore mentioned best practices combined with experiences learned from administering the SAN environment at NCSA over the past five years.

Our SAN started with 60TB of fibre channel (FC) disk from Data Direct Networks (DDN) connected to a Brocade 12000 switch. The hosts were a myriad of Linux, Sun, Windows, and Irix machines using mostly QLogic (QLA) HBAs. We now have 1.6 petabytes of disk from DDN and Engenio (formerly LSI Logic), eighteen Brocade switches (of varying sizes) and two Brocade routers. These are split up into six separate SAN fabrics, based on logical usage. Three SAN fabrics are each on their own supercomputer, one is used by the mass storage archive system, one is a general purpose storage pool and the last is a test SAN. Each of these SAN fabrics are isolated to improve performance, as well as to facilitate maintenance and potential troubleshooting.

Setup & Design

General

The goals of a SAN usually fall into one of the following categories: Storage Consolidation (SC), High Performance Computing (HPC), Clustering, and LAN free backup.

Consolidating storage will most likely result in many different host OS types, each accessing a common disk pool and each with different disk usage patterns. The storage infrastructure SAN at NCSA has Linux, Sun, Windows and Irix hosts performing very different disk operations ranging from random, small block I/O applications such as database and web servers to sequential, large block I/O applications including parallel file systems and data archive projects. Important issues for storage consolidation are dual host paths and host isolation (from each other). Dual host paths provide insulation from failed hardware and disk upgrades. Host isolation refers to ensuring only one initiator (host) per SAN zone, which prevents a misbehaving HBA or host driver from interfering with any of the other hosts in the SAN.

Since speed is the primary concern of High Performance Computing, oversubscription is the primary focus and SAN islands should also be investigated. A real world example is the Mercury cluster with 268 host ports and 96 disk ports and a requirement of 7200MB/s through the SAN. The final design was two fabrics each with four 64-port switches. Splitting the SAN into two separate fabrics allowed each to be connected in a full mesh thus providing maximum throughput on the ISLs and minimizing hop count. All the other designs had problems with ISL oversubscription, ISL bandwidth, hop count or simply too many ports used for ISLs.

Clustering is basically HPC without the performance requirements but will typically involve a parallel file system or some other sort of shared access by all hosts. Primary concerns will be port count and oversubscription. Looking at the Mercury cluster again, the large number of hosts all attempting to access a small number of disk ports can cause a lot of congestion at the disk port. This was addressed by spreading the disk connections evenly across all the ASICs and utilizing host to disk locality as much as possible.

LAN free backup will require a focus on storage space and zoning.

Dual Paths

Another major design issue is the amount of redundancy to build into the SAN. Building two fully redundant SANs is the only guaranteed failover solution, but is not always feasible in practice due to cost. However, having two logical paths to disk is still extremely important. Aside from equipment failure, most disk paths will have to be taken down periodically for firmware upgrades on the disk. In fact, firmware upgrades are the primary reason a disk path will be unavailable. Most SAN switches are built to be able to upgrade firmware without ever interrupting traffic. However, most disks cannot do this. Even though a disk unit has dual controllers, each controller must be rebooted in turn during the upgrade. If a host has access to only one of those controllers, it will lose access to its disk when the controller reboots. Setting up a host with dual paths requires zoning on the disk, zoning on the SAN and multipath I/O (MPIO) support on the host. Additional setup may be required on the disks and/or the host depending on the disk vendor/make/model and host OS. For example, DDN must have new WWNs assigned to the host ports on controller two of the controller pair in order for failover to work properly with the QLogic failover driver.

Historically, MPIO support on the host has been provided by the storage vendor via proprietary software and drivers. The problem with this solution is that the customer is locked into the storage vendor and there is little recourse for unanswered questions and unsolved problems. Luckily, this is changing and many recent OSs now have MPIO support built in. Red Hat Linux (and others as well) has device mapper multipath, Solaris 10 has XPATH and IRIX has XVM. Microsoft has built an API into their Server2003 release, but MPIO is not handled natively. A working Microsoft solution still requires a custom driver from the storage vendor that will interact with Microsoft's MPIO API. Setting up dual path support on Red Hat Linux is still somewhat painful because there are only a few tools already written for dealing with the different storage arrays and the multipath configuration options are not documented in great detail. Information can be gleaned from various sources on the web including the device-mapper multipath home page,[1] Red Hat Knowledge Base[2] and other individual websites found from a web search. For a walkthrough of multipath setup on Linux and other references, see the NCSA Linux multipath tutorial[3].

Testing failover paths is just as important as setting them up. If not tested, it may be discovered during an upgrade that dual paths are improperly configured when a host suddenly loses access its disk. The most obvious way to test failover is to remove access (either physically by pulling a cable or logically by changing zoning on the switch or disk) and then watching the traffic flow on the involved switch ports. The vendor provided switch monitoring tools do not tend to work well for this purpose. However, almost all switches support SNMP access to pull information about the ports. The *multi-port-mon*[4] tool does just that. Written in Perl, it uses the freely available Net-SNMP[5] Perl module and should work on just about any platform that Perl is ported to. The tool displays transmit and receive rates for each port and other identifying information about each port. Transmit and receive rates are calculated from the difference in total number of frames transmitted/received over each polling interval. It was developed to provide a real-time view of the changes in port activity when a hardware failure occurs. When part of the data path fails, traffic will start using a new path; there will be decreased throughput and the new ports will show an increase.

SAN Islands vs. Monolithic SAN

A monolithic SAN refers to one fabric with all switches connected. SAN islands, on the other hand, are smaller isolated fabrics. The primary reasons to separate SANs is isolation from SAN wide events, such as RSCNs and user management error. However, virtual SANs (V-SANs) provide these same isolation benefits without the physical separation of fabrics. With virtual SANs becoming a standard feature in switches today, the primary reasons for building multiple fabrics would be performance, oversubscription, or port count (such as when designing a solution for a specific HPC cluster or supercomputer). As was discussed earlier, the Mercury cluster SAN was designed to reduce potential performance bottlenecks, therefore two full-mesh fabrics worked better than other designs, like core-edge. When designing with switches that don't have the V-SAN capability, a design rule of thumb is to make the SAN as small as possible to meet the goals while taking into account considerations for scalability and growth. With proper

design, there should not be much need to share access across different SANs. In the few cases that a link between SAN islands is needed, a router can provide the needed visibility without merging the individual fabrics.

Cable Management

Cable management is an important issue and necessary to address during setup. Neatness is of utmost importance. If a cable cannot be traced or new cables can not be run in a timely manner, it is nearly impossible to replace failed equipment without affecting other users and can make troubleshooting very difficult. A label with start and end points at both ends of the cable may seem redundant, but it is helpful when running multiple cables at once as well as when replacing failed hardware to keep the cables on the same port. Keeping cables on the same port is important for troubleshooting and maintenance and obviously necessary if port-based zoning is in place. Coiling excess cable at the host, where there is more room will reduce clutter at the switch. Even in a full size rack, there will be only forty host connections (eighty with dual HBAs) compared to 256+ in a switch rack.

Interoperability

All equipment should always be tested. Vendors, external testers, and other users may have tested the products intended for the SAN, but each environment is unique. What may have worked in a lab might not work the same for every implementation. As technology moves forward, it generally improves, but changes are not always backwards compatible (sometimes by design and sometimes by accident) and upgrades can cause existing implementations to fail. Additionally, vendor marketing material is not always accurate. The bottom line: always test.

Testing

A test SAN that mimics (as closely as possible) the production SAN is an invaluable tool. Test SANs are useful for testing new equipment, changes to firmware, software or configuration changes or any other evaluation or testing purpose. They can be used to troubleshoot problems by attempting to recreate them in the test SAN without affecting the production SAN. They can also be used to test upgraded or new hardware before it goes into production. Cost is usually the major barrier to creating a test SAN, but if the time and money are available to build one, it is a worthwhile venture.

Administration

New Switches

The very first task performed on a new switch should be to "wipe" it. The wipe procedure is a list of commands to reset all configuration settings to the factory default value(s). This will be specific to each vendor, but should be identical on all switches made by the same company. The wipe procedure at NCSA for Brocade switches is:

Table 1. Sample switch wiping procedure for Brocade switches

1. Login as root.
2. `switchdisable`
3. `cfgdisable`
4. `cfgclear`
5. `passwddefault`
6. `portstatsclear`
7. `portlogclear`
8. `reboot`
9. `configupload`

Brocade API

The SAN fabrics at NCSA are built exclusively of Brocade switches, and the Brocade API is used in several tools. The API provides functions to interact with the switch and get just about any information about the SAN. To simplify coding, the NCSA::Brocade[6] Perl module was developed to provide frequently used functions and data structures. This module is the basis of many of the SAN management, monitoring and backup tools at NCSA. Current functions are shown in Table 2.

Table 2. Current functions provided by NCSA::Brocade Perl module

`format_wwn`	Remove or add colons to WWN.

`decode_port_state`
`decode_port_status`
`decode_port_physstate`
`decode_port_type`
	Convert integer status, as returned by the switch, to a descriptive text string.

`get_isl_info`	Return hash with ISL details:
	`isl_info = {`
	`  'REMOTE_WWN' = string,`
	`  'REMOTE_SWITCH' = hash ref,`
	`  'REMOTE_SWITCH_NAME' = string,`
	`  'REMOTE_BLADE' = hash ref,`
	`  'REMOTE_BLADE_NUM' = int,`
	`  'REMOTE_PORT' = hash ref,`
	`  'REMOTE_PORT_NUM' = int,`
	`  'TRUNK_MASTER_PORT' = int`
	`}`

`get_san_objects`	Return hierarchy of fabric with major entities:
	`'OID' => { #hash indexed by OID, lookup any object by OID`
	`  <oid> => <hash ref, fabric object returned by BrocadeSTK>`
	`},`
	`'FABRIC' => { #hierarchical layout mirrors structure of SAN`
	`  <oid> => {`
	`  'OBJ' => <hash ref, Fabric Object> ,`
	`  'SWITCH' => {`
	`  <oid> => {`
	`  'OBJ' => <hash ref, Switch Object> ,`
	`  'PORT_MODULE' => {`
	`  <oid> => {`
	`  'OBJ' => <hash ref, PortModule Object> ,`
	`  'PORT' => {`
	`  <oid> => {`
	`  'OBJ' => <hash ref, Port Object> ,`
	`  'N_PORT' => {`
	`  <oid> => {`
	` 'OBJ' => <hash ref, NPort Object>

Configuration Backups

Backups are useful for recovering from errors and mistakes. They store a history of changes and also provide a base configuration to use when introducing new switches to an existing SAN fabric or copying a config to the test fabric for debugging.

The NCSA switch backup script, *cfg_upload.pl*[7], is a modified version of a sample backup script from Brocade Connect.[8] The backup script is invoked weekly from a cron job that automatically saves all the switch configurations. The cron job actually runs a ksh wrapper script, *get_san_backup*[9], that calls the backup program for each switch. The wrapper also takes care of renaming old configurations and moving them to archive storage. The most recent backups are stored in a folder named with the actual date-time of the backup.

SAN Profile

Probably the most used and invaluable information to track about a SAN is a **current** list of devices on the SAN and what switch port each device connects to. In a dynamic environment where multiple personnel perform admin tasks on the SAN, it is nearly impossible to maintain accurate documentation of this information. But using the API, it is relatively easy to automate this task and have it run alongside the weekly configuration backups. The automated tool, *san_profile.pl*[10], generates a profile of the SAN. The generated profile shows the current state of each switch port and what, if anything, is connected on each port. The interesting part of the profile creation is done by correlating the WWN of the remote HBA or disk port to its alias stored in the zoning config. The output is similar to that of Brocade's `switchshow` command, but with an added column showing the name of the remote device and an added column showing the switch model. Table 3 shows sample output from a SAN containing a Brocade 3800 and a Brocade 3900.

Table 3. Sample output from *san_profile.pl*. Some lines have wrapped in this table, typical output shows 1 port per line. Fields are (from left to right) switchmodel, switchname, switchport, blade/port, remote WWN, alias or ISL info or local port status (if port not in use)

```
BRCD3800 silk16_5  0 00/00 20140060699069A7 silk32_1 blade=0 port=20
local_trunk_master=0
BRCD3800 silk16_5  1 00/01                  (PORT_STATE_OFFLINE PORT_STATUS_UNUSED
PORT_PHYSSTATE_NO_LIGHT PORT_TYPE_G_PORT)
BRCD3800 silk16_5    2 00/02 10000000C92EC1EE cu12_fscsi3
BRCD3800 silk16_5    3 00/03 10000000C92F45B1 cu11_fscsi3
BRCD3800 silk16_5    4 00/04 10000000C9440A6A cu10_fscsi3
BRCD3800 silk16_5    5 00/05 10000000C92F50E0 cu09_fscsi3
BRCD3800 silk16_5    6 00/06 10000000C92F467F cu08_fscsi3
BRCD3800 silk16_5    7 00/07 10000000C93D72F4 cu07_fscsi3
BRCD3800 silk16_5    8 00/08 10000000C92EB14D cu06_fscsi3
BRCD3800 silk16_5    9 00/09 10000000C92F4EB2 cu05_fscsi3
BRCD3800 silk16_5   10 00/10 10000000C92EC320 cu04_fscsi3
BRCD3800 silk16_5   11 00/11 10000000C92D10F6 cu03_fscsi3
BRCD3800 silk16_5   12 00/12 10000000C92EB90D cu02_fscsi3
BRCD3800 silk16_5   13 00/13 10000000C92F6744 cu01_fscsi3
BRCD3800 silk16_5   14 00/14                  (PORT_STATE_OFFLINE PORT_STATUS_UNUSED
PORT_PHYSSTATE_NO_LIGHT PORT_TYPE_G_PORT)
BRCD3800 silk16_5   15 00/15 201E006069906993 silk32_2 blade=0 port=30
local_trunk_master=15

BRCD3900 silk32_1    0 00/00 2000006069510840 silk16_2 blade=0 port=0
local_trunk_master=0
BRCD3900 silk32_1    1 00/01 2001006069510840 silk16_2 blade=0 port=1
local_trunk_master=0
BRCD3900 silk32_1    2 00/02 200200A0B80F4447 fastt6g_a_1
BRCD3900 silk32_1    3 00/03 200300A0B80F4447 fastt6g_b_1
```

A caveat about *san_profile.pl* is the requirement of a strict one to one mapping of WWN to alias. If an alias refers to more than one WWN (or other object) then the behavior of the tool is undefined. This is possibly a future goal to handle, but was not an initial requirement of the tool. A command line option can be used to change the field separator for easy parsing by other scripts, such as *multi-port-mon*, for testing dual paths, and the data input file for generating a gif image of the SAN layout using DOT.

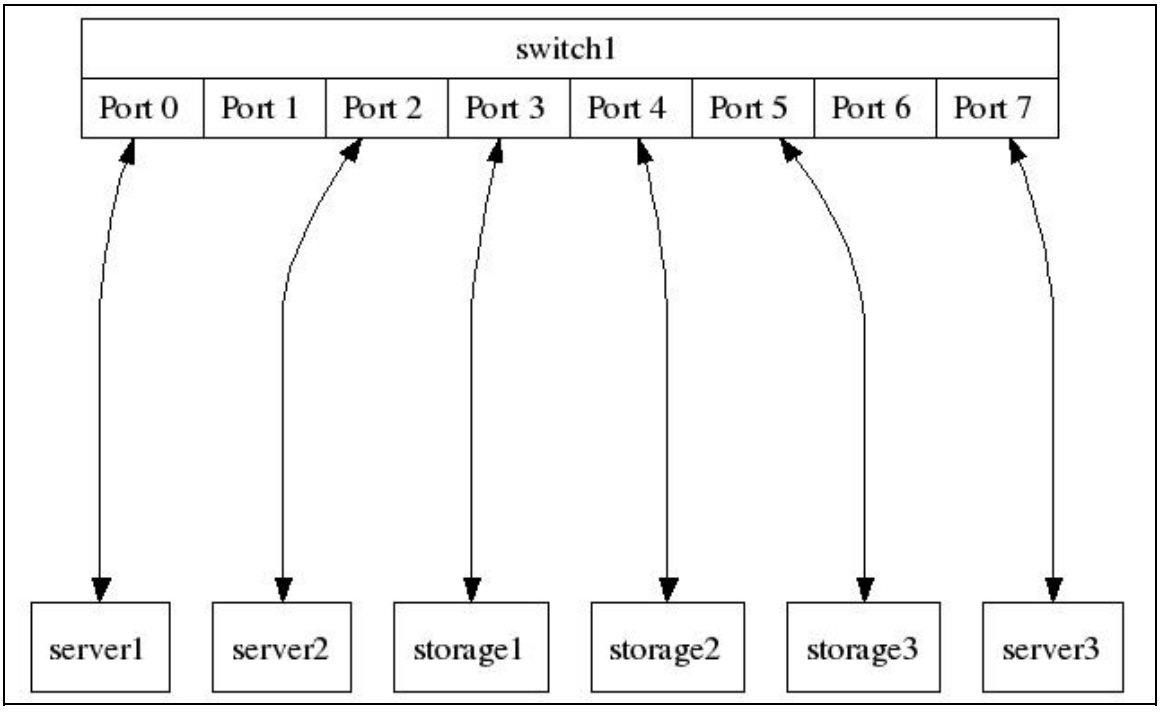
DOT Graphs

Dot is part of the Graph Visualization Software package, also known as graphviz[11]. Currently, coding is underway to take the output from the *san_profile.pl* script discussed previously and generate an input file that can be used by dot to create a gif image showing the layout of the SAN. Table 4 shows a sample dot input file and Figure 1 shows the image generated from that input file. These are very simple examples, but they show how useful a graph is for viewing connections and relationships.

Table 4. Sample dot imput file

```
digraph TestSAN {

K=3;
overlap=false;

node [shape=record];

    server1 [shape=record,label="{ server1 }"];
    server2 [shape=record,label="{ server2 }"];
    server3 [shape=record,label="{ server3 }"];

    storage1 [shape=record,label="{ storage1}"];
    storage2 [shape=record,label="{ storage2}"];
    storage3 [shape=record,label="{ storage3 }"];

    switch1 [shape=record,label="{ switch1 |{ \
               <0000> Port 0 | <0001> Port 1 | <0002> Port 2 | <0003> Port 3 | \
               <0004> Port 4 | <0005> Port 5 | <0006> Port 6 | <0007> Port 7 }}"];


    switch1:0000 -> server1 [dir=both minlen=5];
    switch1:0002 -> server2 [dir=both minlen=5];
    switch1:0007 -> server3 [dir=both minlen=5];
    switch1:0003 -> storage1 [dir=both minlen=5];
    switch1:0004 -> storage2 [dir=both minlen=5];
    switch1:0005 -> storage3 [dir=both minlen=5];

}
```

Fig. 1. Sample gif output generated from DOT.



While the idea of programmatically creating a graph of the SAN layout seems pretty simple, there are some small details that hinder the process. For instance, dot provides minimal control of the layout. While this level of control is intentional, it causes problems when layout details are required for a readable graph, as is the case in SANs with more than two or three switches. Another issue is the lines representing connections will often be drawn too close together to distinguish the separate links. Likewise, the

endpoints of the lines will often overlap so that it is impossible to see the exact port the line is pointing to. Further testing and research is needed before a determination can be made to either continue working with dot or to investigate other graphics packages.

Troubleshooting

Troubleshooting SAN problems covers a very large range of topics and levels of technical expertise. The typical troubleshooting path will start with checking hardware connections, then move to reviewing software settings and finally result in including the involved vendors and possible other third parties. However, two topics warrant discussion here: common issues and FC protocol analysis.

99% of problems are human driven

It has been shown time and time again that the majority of errors relating to computing are human generated and able to be corrected quickly, if not avoided altogether. Managing SAN fabrics is no different. The most common problem encountered in managing a SAN is lack of access from a host to a disk. In almost 99% of the cases, the problem results from either human error or simple hardware issues, usually a failure of some sort. The following list covers these issues in roughly the order of frequency:

1. SAN zoning problems cause the majority of issues. Common problems are missing targets from the host zone, host zone configured to see the wrong targets, incorrect WWN alias(es) resulting from new or replaced hardware, and new zone not added to the active configuration. Switch zoning modifications are the most common change that occurs in a SAN, which explains the increased chance for mistakes. Also, there is also no way to automate zoning since it requires human decisions to determine initiator and target accessibility.
2. Host HBA issues occur almost as frequently as SAN zoning problems. By far the most common problem is driver configuration, usually relating to lun failover setup. Until recently, lun failover was provided by the HBA or disk vendor so there was little or no similarity between different setups.. Additionally, drivers provided by the OS vendor might behave differently than those obtained directly from the HBA vendor or the native OS drivers might not provide the same configuration settings. An example of this scenario is QLogic HBAs on Red Hat Linux. Drivers that ship with Red Hat do not support lun failover. For lun failover support, drivers must be downloaded from the QLogic website and compiled manually. Another common problem is replacing a failed HBA, which will have a different WWN, but forgetting to update the host configuration or the SAN zoning configuration with the new WWN.
3. Disk zoning / lun masking provide another layer of manual configuration that can often get messed up. Disk zoning happens on the disk controller and refers to assigning only specific luns to a host. Lun masking is setup on the host and refers to hiding luns from the OS even though the HBA has access to them. This might happen in a parallel file system where multiple hosts have access to the same luns for host failover reasons, but only one host has control of the lun at any given time. Again, because it requires human decision making, disk zoning is not easily automated. In the case of a parallel file system, the lun masking and failover mechanism is fully automated in production, but the initial setup is still a manual process.
4. FC cabling is the most volatile hardware component in the SAN because it is handled the most and also spans many areas of the data center. Common issues are dirty cable ends, poorly seated cables, and cable tension and binding. Dirty cable ends should be cleaned with a lint free alcohol cloth. If ferrule caps are always replaced when cables are not in use, there will be very few instances of dirty cable ends. Poorly seated cables are fixed simply by disconnecting and reconnecting the cable. This is seen more commonly on the director class switches that have 128+ ports simply because there is less room for human fingers to fit. Tension is caused by cables getting pulled or moved once they are in place and can cause poorly seated cables to come loose and can also cause binding. Binding occurs when a cable gets bent beyond the acceptable 3 inch radius, which will adversely affect performance due to excessive light loss at the bind location.

Finisar Fibre Channel Analyzer

Problems presenting disk to a host are usually resolved using the above steps. Once the host has access to disk, troubleshooting performance or other problems quickly becomes very technical. The typical next step would be to open a ticket with the vendor and/or maintenance provider. However, once the host has access to disk, the most effective troubleshooting tool is a trace of the actual fibre channel packets.

A trace is taken by a fibre channel analyzer, which records every frame and FC control data. Analysis tools use the raw data in the trace to provide a detailed analysis of the individual frames and control data, exact timing of events, and details about the encapsulated storage protocol. Technical experts can quickly identify the cause and location of a problem. In addition to sending the trace to any involved vendors and/or maintenance providers, it can also be sent to Finisar for free expert analysis. This is very important because Finisar is external to any of the SAN hardware involved in the investigation. The significance os the trace itself, and the external analysis is that is will usually preclude finger pointing by vendors since the analysis of trace data is irrefutable. One caveat to the analyzer is that it must be in the data path to record the frames. This is accomplished by inserting a FC tap anywhere along the data path, typically between the disk port and the switch. The tap passively copies every frame and sends the copy to the analyzer. The problem here is that the connection must be broken to insert the tap. This has been addressed in the most recent switches by providing a "finisar port", which refers to the ability of the switch to copy frames from any single port to a different port that is outside of the real data path. With this capability, the analyzer can be attached directly to a free switch port and the problem connection does not have to be broken. Overall, the Finisar analyzer is by far the best and most efficient troubleshooting tool available.

Health Monitoring

Log Monitoring

Monitoring logs is probably the most basic form of tracking the health of any system. Therein lies the flexibility; it is applicable to any system that records logs. The goal for log monitoring is to be notified of problems when they occur as opposed to manually checking for any issues.

Syslog

Syslog is a standard for sending log messages over the network to another host. Developed in the 1980s and formally ratified in 2001, it is supported by most equipment, yet some SAN capable systems do not support it, namely, Engenio disks. Fortunately, Engenio provides a CLI interface to the disks, which prompted the development of the *fast_monitor.pl*[12] script to download the logs, parse them and send a one line summary to syslog. The program runs on a 15 minute cycle, so errors are reported no later than 15 minutes after they happen. In addition to providing syslog support, the actual logs are stored on the host machine and get backed up and archived along with all the other logs sent to syslog.

Initial log monitoring attempt using WOTS

WOTS is a log monitor that looks at each log line and determines what action (if any) to take. Each log line is considered individually, no state is saved between reading lines. The down side to this is that a single event might generate anywhere from 5-20 lines in the log file, so 5-20 emails will be sent about this one event. Additionally, WOTS requires a separate script to filter relevant lines from the main syslog file and redirect them to individual log files that WOTS will monitor.

Improved log monitoring using Logsurfer

The major difference provided by Logsurfer is the ability to save state, called contexts. A context is essentially a regular expression and matching lines are saved in the context. A context is created when a log line matches the regular expression and will expire when either a set amount of time has passed since the most recent matching line or a maximum number of lines are found. Upon expiration of a context, an action is performed. Typically, that action is to send an email with an attachment containing the log lines from the context. This solves the WOTS problem of multiple emails since all the relevant lines are contained in the context. The action isn't limited to just sending email, it can be anything including an

external program, in which case the log lines from the context will be sent via stdin. Incidentally, this is exactly how reports are sent from Logsurfer to Nagios (see below).

Logsurfer has other advanced features to do things such as create and delete rules dynamically. An example of how dynamic rules can be used is tracking a reboot. A single rule looks for any message that indicates a reboot and creates two new dynamic rules. The first dynamic rule creates a context to collect all logs from the specified host and is set to expire after enough time for the reboot to complete. The second dynamic rule looks for messages indicating the reboot is complete. If one is found, it sends an email that the reboot completed successfully, includes the logs from the first dynamic context and then deletes both the dynamic rules. On the other hand, if the first dynamic rule expires, it will perform its action (send email that a reboot appears to have failed) and then delete both the dynamic rules. The important point to note is that only one of the dynamic rules will expire and be allowed to run its action.

Interactive log monitoring using Nagios

Nagios is a host monitoring tool that supports both active and passive monitoring and provides a web display for status and management. While it is intended to be a full monitoring solution, its main use at NCSA is a web display of SAN health. The help desk monitors the "Problem Summary" page 24 hours a day and when a problem occurs, they can analyze it, determine what action to take and mark the problem acknowledged, all from the Nagios web interface. The Nagios implementation at NCSA is setup to actively monitor all hosts and SAN equipment for network connectivity by running a ping test every ten minutes. Nagios also actively monitors the Logsurfer and *fast_monitor.pl* processes on the log host using custom checks and actions.

The passive monitoring feature of Nagios allows external programs to send reports that will be displayed on the Nagios webpage. Logsurfer provides the passive input to Nagios by sending reports for certain events that match specific services in Nagios. For example, on a disk failure, Logsurfer sends a "Failed Drive" report to Nagios and the problem will be displayed on the Nagios webpage. When the failed disk is replaced, Logsurfer will see the replaced event in the logs and send an "OK" report to Nagios and the failure will be removed from the web page.

Automating Nagios configuration with Fruity

One of the hurdles to using Nagios is the amount of configuration. Each host (disk, switch, server) monitored must have a host record and each service on each host must have a service record. At NCSA, this amounts to 227 host definitions and 538 service definitions. All configuration settings are read from plain text files. To keep things organized, all the systems and services are separated into logical groups. Each group has it's own subdirectory and it's own script to control generation of the config files from generic templates. This makes it manageable but inter-file dependencies are still present. A future goal is to store all the configuration information in a database and auto generate the configuration settings from the database. This is possible today using the third party Nagios configuration tool, Fruity. Fruity claims to handle both version one and version two Nagios configs as well as the ability to convert from one to the other, so can also simplify upgrading to a newer Nagios version.

Future goals for log monitoring

The reason for collecting logs is to provide the availability to extract information about events should the need arise. The uses are limitless, such as looking at sequences of events just prior to a failure or tracking the frequency of one or more events or even collecting performance data. A problem arises when attempting this on logs saved in text files. Most SAN equipment has one log file (those systems that support syslog) but some equipment has multiple log files (the Engenio disks each have 2 files as a result of the *fast_monitor.pl* collection script; one file of logs in native Engenio format and another with one-line summaries of each event). Searching through these logs for post mortem information can be cumbersome and generally requires custom tools or scripts to search for useful information (related to the specific incident in question) and filtering out useless and unrelated information. As with the Nagios configuration files, storing the log data in a database would allow advanced searching capabilities using a standard interface (SQL). Splunk[13] is a freely available solution that provides this functionality. "Splunk is software that indexes and securely manages all your logs and IT data." [http://www.splunk.com]. The logs are stored in a database and searched from a webpage. The solution promotes ease of use and powerful

search capabilities providing useful results quickly. The ease of finding useful information is the driving factor.

Performance Monitoring

Performance monitoring is essentially another form of monitoring SAN health, but in a more passive way. The data sets from performance monitoring are mostly used for reference as records of historic usage patterns and to show changes over time. Monitoring performance of the SAN is important because changes in performance will usually be the first indicator of a problem, either hardware or software. The first step is to develop a baseline to define what is normal. The baseline should definitely include throughput rates and error counters for all switch ports. Other useful points of data are throughput and errors on disk ports and host ports. Errors that occur on either of these may or may not show up on the switch. Finally, if possible, end to end scsi throughput can be useful to monitor on the host as well.

Aside from impending failures, performance monitoring can also show problems with SAN design based on current usage patterns. For instance, an ISL port performing at 80% capacity could indicate possible oversubscription. SAN design can only plan around potential oversubscription based on expected usage patterns available at design time. If usage patterns change and some hosts start utilizing storage more, performance monitoring will show this change and action is necessary to resolve it. The action may be to add another ISL or perhaps relocate the storage or the host to another switch port so the traffic doesn't go over an ISL (or the traffic goes over a different ISL). Similarly, monitoring disk ports would show discrepancies between heavily used ports and less used ports. Simply distributing the hosts among the available disk ports is not sufficient if many high activity hosts share the same port while many low activity hosts are on another port. Knowing actual usage from each host will provide the necessary information to re-distribute access to the disk ports so that the high usage hosts are evenly distributed across the available disk ports. Additionally, performance monitoring on the host will show usage discrepancies between multiple HBAs, in which case lun access paths can be adjusted to better utilize the available host ports. Finally, each of these data points are useful in troubleshooting. When problems occur or performance changes, statistics for each port along the data path can be observed to help isolate the source of the problem. If the problem source is at the host, the disk ports and ISL ports on that data path will not necessarily have large reductions in throughput due to other traffic continuing at normal rates on those ports. Likewise, if the problem is on the disk, all hosts accessing that same disk port would be affected.

Interactive Tools

Although performance monitoring data is used mostly to show historic patterns, interactive tools are still useful to watch data traffic rates in real-time. These are used mostly during initial setup of a host to monitor traffic rates and/or patterns.

Switch Command Line Interface

Regardless of switch vendor, there is usually some sort of monitoring available from the command line interface to the switch. While most switches also have a GUI, the command line interface is useful because scripts can interface with the switch this way to automate tasks. Brocade switches provide the commands `portperfshow` and `porterrshow`. The `portperfshow` command displays total throughput rates (combined transmit and receive values) for all ports on the switch and a total for the entire switch (or blade if a bladed switch).

Table 5. `portperfshow` from a Brocade 12000 (dual 64-port switches, 4 blades each)

```
silk128_12_sw0:admin> portperfshow

           0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15  Total
=============================================================================================
slot 1:    0    0  517k 2.0m 2.0m 2.0m 687k   0    0    0  6.2k 1.0m 1.0m 2.0m 1.5m 518k  13m

           0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15  Total
=============================================================================================
slot 2:  9.7k 2.5m   0    0    0    0  2.0m 533k 1.0m 1.0m 513k 1.5m   0    0  518k 1.5m  11m

           0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15  Total
=============================================================================================
slot 3:    0    0    0    0  3.1m 4.1m 1.0m 1.6m   0    0  1.0m 524k 3.1m 1.1m 2.5m  54k  18m

           0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15  Total
=============================================================================================
slot 4:    0    0    0    0    0    0    0  2.5m 1.5m 1.1m  24k 1.0m   0    0  2.6m 1.0m  10m
```

The `porterrshow` command displays error counters for all ports on the switch. An example use of this data is to look for large differences (greater than an order of magnitude) between error counters from the port(s) in question and other ports on the switch or values from historical data.

Table 6. `porterrshow` from a Brocade 12000 (output truncated after 16 ports). From this output, ports 11, 14 and 15 show very high values for „enc out“ (encoding errors outside of frames) compared to the other ports. With no other errors, possible hardware issues should be investigated.

```
silk128_12_sw0:admin> porterrshow
        frames  enc  crc  too  too  bad  enc disc link loss loss frjt fbsy
       tx   rx   in  err shrt long  eof  out   c3 fail sync  sig
     ========================================================================
  0:  1.3g 2.1g   0    0    0    0    0    0    0    2    0    0    0    0
  1:  1.4g 2.1g   0    0    0    0    0    1    0    2    0    0    0    0
  2:  3.8g 377m   0    0    0    0    0    3    0    4    1    1    0    0
  3:  3.8g 374m   0    0    0    0    0    4    0    4    1    1    0    0
  4:  2.0g 1.1g   0    0    0    0    0    0    0    2    0    0    0    0
  5:  2.0g 1.1g   0    0    0    0    0    1    0    2    0    0    0    0
  6:  3.8g 343m   0    0    0    0    0    4    0    3    1    1    0    0
  7:  3.8g 294m   0    0    0    0    0    3    0    4    1    1    0    0
  8:  1.3g 2.1g   0    0    0    0    0    1    0    2    0    0    0    0
  9:  1.3g 2.1g   0    0    0    0    0    1    0    2    0    0    0    0
 10:  3.7g 378m   0    0    0    0    0    2    0    4    1    1    0    0
 11:  3.8g 368m   3    1    0    0    1   21k   0    5    7    2    0    0
 12:  2.1g 1.1g   0    0    0    0    0    0    0    2    0    0    0    0
 13:  2.1g 1.1g   0    0    0    0    0    0    0    2    0    0    0    0
 14:  3.8g 296m   4    1    0    0    1   25k   0    4    7    2    0    0
 15:  3.8g 436m   2    1    0    0    0   10k   0    5   12    2    0    0
 16:  3.4g 2.7g   0    0    0    0    0  178    0    3    0    0    0    0
```

In general the CLI commands are minimally useful for performance monitoring because it is difficult to see patterns in the data and the results do not show changes over time. Additionally, all the data points are accessible via SNMP, which provides a simpler interface for writing automated scripts since an interactive login does not have to be simulated.

Switch Web Interface

The same performance information can also be obtained from switch GUI tools. The Brocade switches come with a built in web interface, called Webtools. The Webtools interface is accessed by pointing a web browser to the switch IP address. Clicking on the "performance" button opens a window with a graph showing throughput of all the ports (Figure 2). The graph allows a very quick comparison of port performance relative to all the other ports. Additional graphs are available from a pull down menu, such as the “Port Throughput” graph (Figure 3) that shows transmit and receive rates for an individual port over time. The main advantage of the GUI tools is the ability to show history, which their CLI counterparts cannot do. Unfortunately the GUI does not provide any way to save the data nor is it able to be automated.

Fig. 2. Brocade Webtools performance graph showing total throughput of all switch ports. Placing the mouse over the graph will display a popup of the top 5 busiest ports

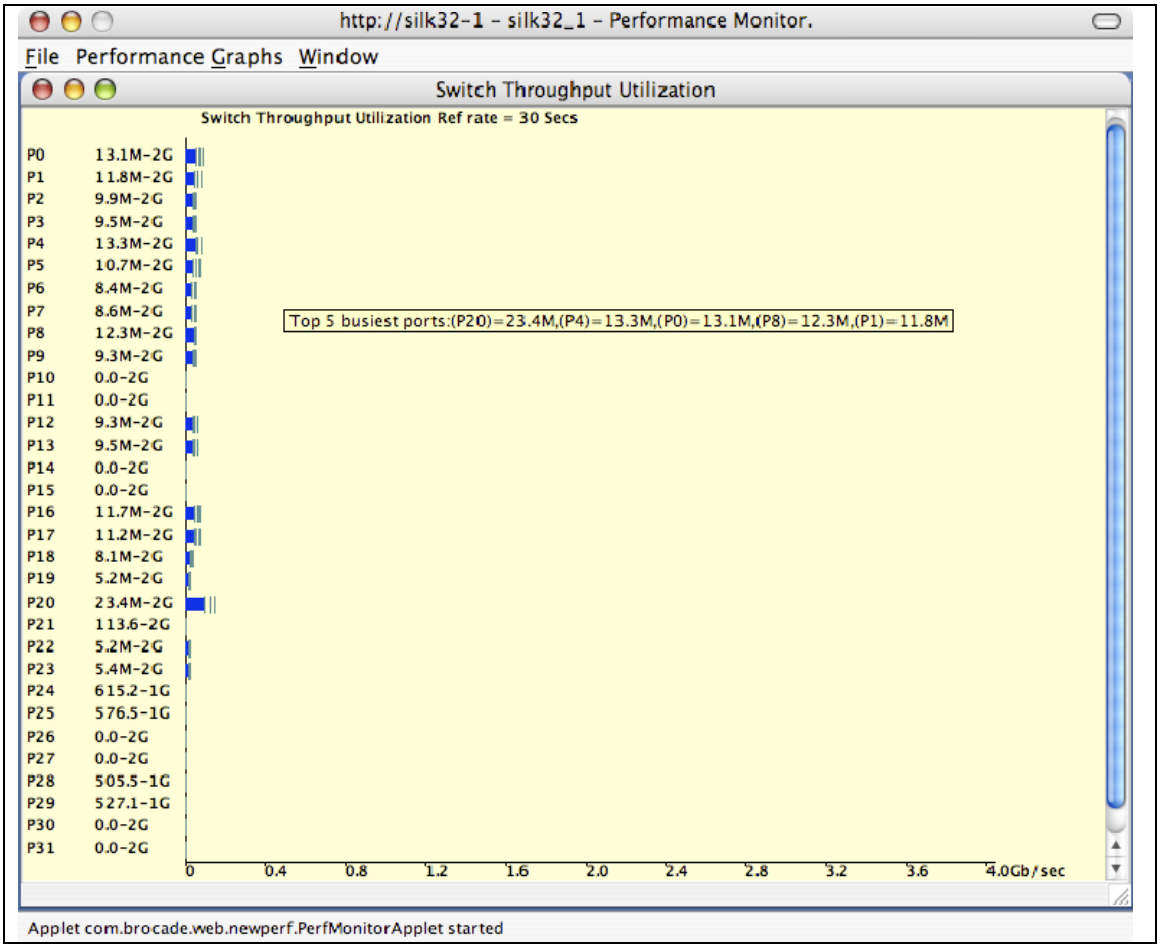
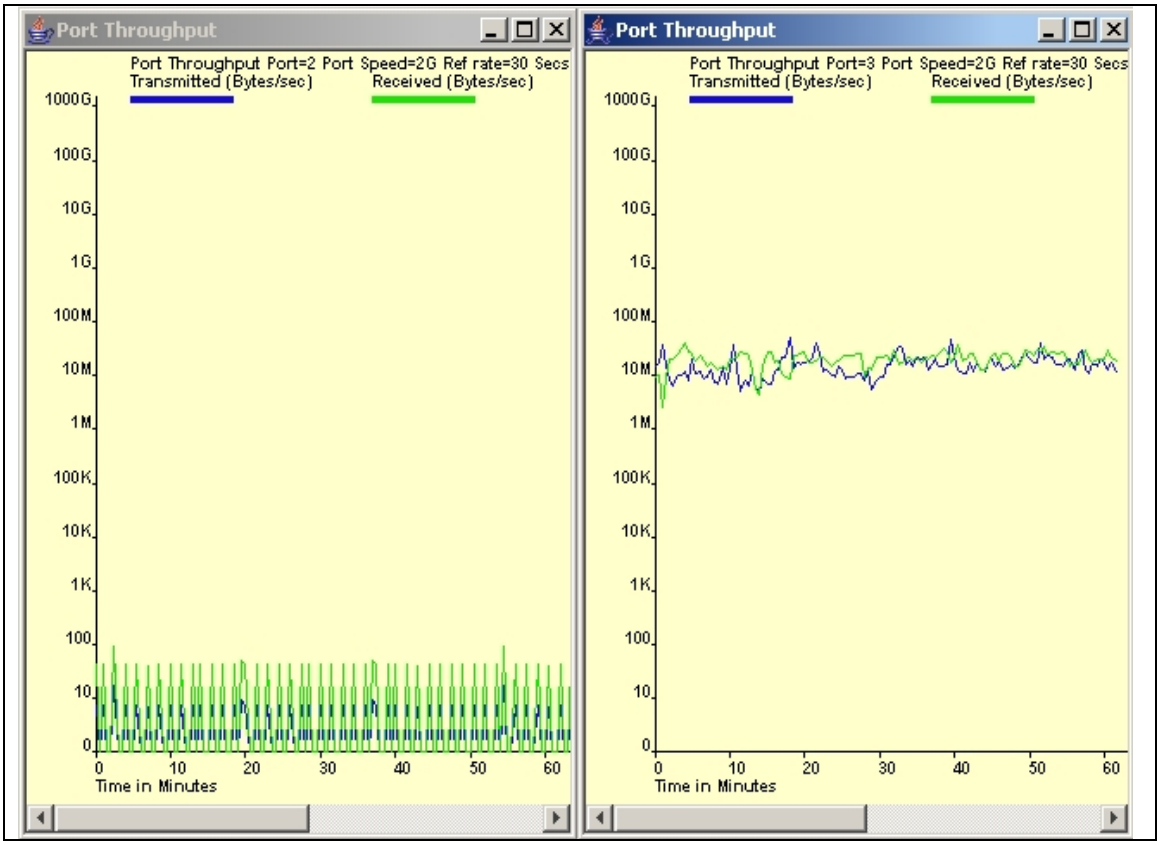


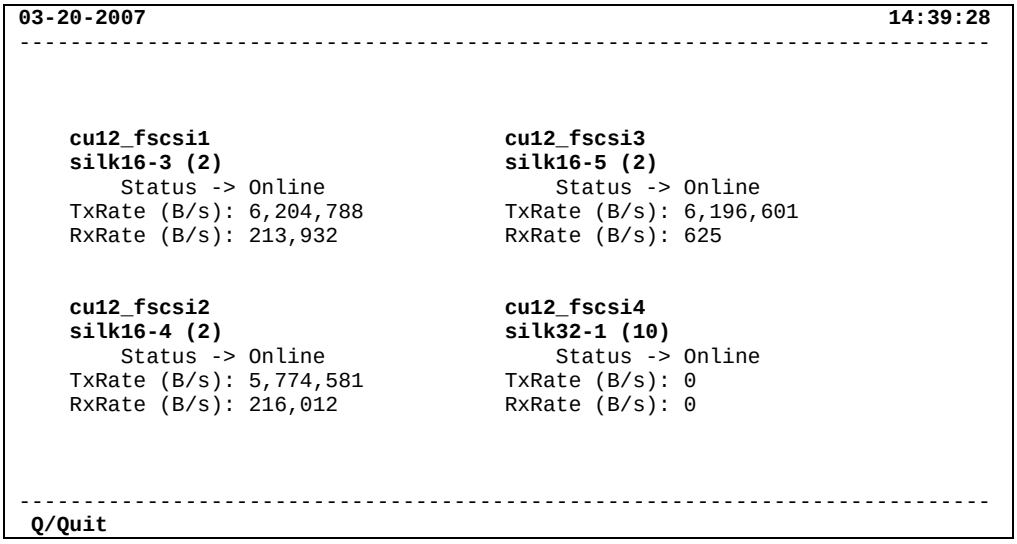
Fig. 3. The Brocade Webtools "Port Throughput" graph shows port usage over time



Customized port monitoring using *multi-port-mon*

The *multi-port-mon* tool shows status and throughput for multiple ports anywhere in the SAN regardless of what switch they are on. The usefulness of this is the ability to watch traffic on all ports in the data path. Although its intended use was for failover testing, it is very useful in any situation requiring real-time statistics of multiple switch ports, such as locating data path bottlenecks, and balancing usage across multiple host HBAs. Since the data is retrieved using SNMP, it will work with any vendor's switch that supports SNMP retrieval of port statistics.

Fig. 4. NCSAs *multi-port-mon* tool uses SNMP to display real-time throughput information about individual ports from different switches



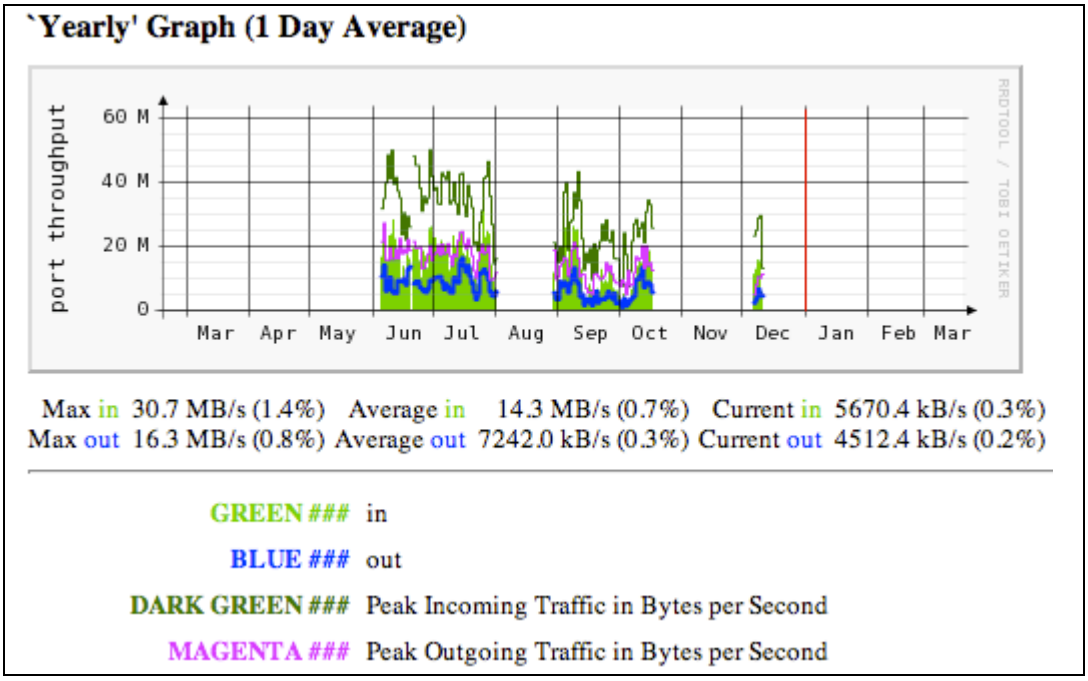
Automated Tools

Using MRTG to create a baseline

The requirements for creating a performance baseline for a SAN are automation and ability to save performance statistics over time. MRTG[14] meets both of these requirements. MRTG gathers data from the SAN switches via SNMP and saves the data either in local files or in an RRDtool[15] database. RRDtool is a custom database format designed specifically for use with MRTG that uses a round robin data storage pattern to enforce a static file size. It also provides faster access to the data than the plain files used natively by MRTG. The data points are used to create graphs, which can show just about any kind of data, but most commonly show performance over time. The graphs are viewed via a web interface and are generated on the fly the first time they are accessed, after which, they are cached to reduce CPU load and updated only when needed. MRTG can be customized to collect and graph any data, making it extremely useful for tracking other information such as error counters and environment statistics (power, temperature, etc...). MRTG will work with any switch that supports SNMP and it's free.

MRTG is a great tool and a good starting point for collecting and displaying baseline performance data. However, there are some issues that limit the functionality. First and foremost, each graph must contain exactly two inputs, such as 'frames transmitted' and 'frames received'. Values calculated from or referencing the two inputs can be graphed, but there is no support to graph only one input or add more inputs to a graph. Also, one reason for using RRDtool is to limit file size, but MRTG creates one RRD data file for each port that is monitored, so space saved in each file is essentially lost by creating so many files. Lastly, MRTG configuration is somewhat primitive in that each port monitored requires it own configuration section. There is no template ability to define common items that can be reused by each port, even though all port configurations are 99% identical.

Fig. 5. A sample MRTG graph showing throughput on a switch port over the previous year. The breaks in service are from MRTG service outages



NetWisdom Express

NetWisdom Express[16] is essentially MRTG on steroids. Similar to MRTG, it collects SAN fabric performance data and faults every five minutes, which are saved in a local MySQL database. The standout feature of the software is its impressive, in-depth report generation capability. It comes with an extensive list of canned reports including baselines, error reporting and many others. The reports can be scheduled or generated manually in order to specify the time period to report on. Scheduled reports can be emailed or saved locally for manual retrieval or web viewing. The included web interface provides access to generate and view reports from anywhere. Aside from the excellent reports, the support provided by Finisar is equally important. They will help analyze data and reports to resolve problems or provide insight on recorded metrics.

References

[1] Linux multipath tools home <http://christophe.varoqui.free.fr/>
[2] Red Hat Knowledge Base <http://kbase.redhat.com/faq/>
[3] NCSA Linux multipath tutorial <http://dims.ncsa.uiuc.edu/set/san/src/linux_mpio_tutorial.zip>
[4] NCSA multi-port-mon tool <http://dims.ncsa.uiuc.edu/set/san/src/multi-port-mon.zip>
[5] Net-SNMP Perl module < http://net-snmp.sourceforge.net/>
[6] NCSA::Brocade Perl module < http://dims.ncsa.uiuc.edu/set/san/src/NCSA-Brocade.tar.gz>
[7] NCSA switch configuration backup script < http://dims.ncsa.uiuc.edu/set/san/src/san_backup.tgz>
[8] Brocade Connect website < http://www.brocadeconnect.com/>
[9] NCSA ksh backup script < http://dims.ncsa.uiuc.edu/set/san/src/san_backup.tgz>
[10] NCSA SAN profiling tool < http://dims.ncsa.uiuc.edu/set/san/src/san_backup.tgz>
[11] Graphviz <http://www.graphviz.org/>
[12] NCSA log harvesting tool for Engenio products <http://dims.ncsa.uiuc.edu/set/san/src/Fastt_Monitor.tgz >
[13] Splunk <http://www.splunk.com/>

[14] MRTG < http://oss.oetiker.ch/mrtg/>
[15] RRDtool database < http://oss.oetiker.ch/rrdtool/>
[16] Finisar NetWisdom Express <http://www.finisar.com/>