Sick But Not Dead Failures -
Adaptive Testing, Evaluation and Design Methodologies

Tara Astigarraga[1], Michael Browne[2], Lou Dickens[3], and Ian MacQuarrie[4]
Systems and Technology Group
IBM
[1] Rochester, NY 14626
[2] Poughkeepsie, NY 12601
[3] Tucson, AZ 85744
[4] San Jose, CA 95134
{asti, browne, dickens, imacq}@us.ibm.com

***Abstract*- Enterprise data center implementations make significant investments in high availability configurations, redundant hardware, software and Input / Output (I/O) paths that are in many failure scenarios quite successful. However, in spite of all that investment clients are still facing unexpected outages and performance impacts related to a phenomenon referred to as Sick but not Dead (SBND) errors. SBND errors are sometimes lumped together in a category with other related errors including transient errors, partial failure scenarios and soft errors. While SBND errors do have many common characteristics with the errors described above, there are key differences and environment impacts which we will explore further in this paper. We will also present new proactive techniques, inject scenarios and methods to identify, characterize and address SBND failures including cross-component impacts and failures.**

Keywords- Software Testing, Sick but not Dead, Software Engineering, Partial Failure, Transient Error, Soft Failure, SAN Test, Storage Area Network Test, System Test.

I. INTRODUCTION AND MOTIVATION

Despite high availability (HA) configurations, failures are still occurring that are impacting customer environments. Impacts range from varying degrees of performance degradation to complete multi-system outages that often cause enterprise level business outages for extended periods of time. When these types of failures escalate to enterprise level loss of data access events the data verification time for these events can be lengthy and involve numerous business and information technology staff long after the error condition is resolved [1].

The type of failure leading to these impacts is one that exhibits a temporary and reoccurring behavior. Meaning errors are being recovered at various points within the system, however, the errors continue to occur at varying rates. We classify these errors as Sick but not Dead (SBND) failures. These errors are often the hardest failures to identify and can have sporadic but lasting impacts on the environment as a whole. SBND failures currently represent 80% of business impact, but only about 20% of the problems [2].

SBND errors are sometimes lumped together in a category with other related errors including transient errors, partial failure scenarios and soft errors. While SBND errors do have many common characteristics with the errors described above, there are key differences as well. SBND errors by definition derive from a component within the I/O path that is 'sick' meaning behaving in an unorthodox or partially failed fashion but not completely 'dead' or hard failed. Depending on the component exhibiting the SBND characteristics, the symptoms can vary, come and go at different intervals and persist for an extended period before the component eventually reaches a hard fail state or would otherwise persist indefinitely if not for manual intervention. It is this in-between time when the component is defined as SBND.

While there have been examples of the industry trying to address this problem proactively with technologies like IBM's Predictive Failure Analysis and S.M.A.R.T monitoring which has been incorporated in ANSI INCITS T13 Technical Committee, the problems still persist at both the device and system level [3]. Many of the predictive technologies in place today have some obvious constraints. Inherent in a new technology is the lack of experience in being able to correlate performance and calibration data with a reasonable expectation of a failure. It takes technology providers a fair amount of time and maturing of a technology to be able to make reasonable correlations. This increases the opportunity for SBND failures during this maturing time frame. The S.M.A.R.T monitoring standard has a requirement to reset all counters to zero after a firmware modification which may or may not address a component that has or is about to exhibit SBND behavior. The prediction of a failure is in opposition to the economic needs of vendors to not replace components too early under vendor warranty periods. Vendor warranty costs can be increased if the prediction is too opportunistic. These and similar factors and constraints at the component level make it very difficult to design away the SBND failures. At a system level, the holistic environment needs to be able to encounter these conditions with reasonable robustness such that the environment does not degrade to an outage situation. To adequately system test a complex environment SBND errors and scenarios need to be designed and injected into complex environment testing and holistic test observations and evaluations need to be made to determine if the complex environment is robust enough for its intended use.

Complex customer solutions and environments utilizing mixed vendor products and technologies create textbook scenarios for SBND failures to occur. Many products are intolerant of errors from other devices, and although most products respond promptly to hard failure conditions they are much slower to respond to SBND conditions and often do not deploy logic necessary to even detect SBND conditions. With current field solutions, problem determination related to SBND failure scenarios is complex, time consuming and often requires special problem determination lab trace tools and a team of cross-vendor product and solution experts. Current resolutions to SBND failure scenarios are almost always reactive vs. proactive.

One of the common areas for this reactive approach to SBND errors in the I/O path are in the area of multi-pathing function inside or in the I/O code stack for devices. It is typical for a situation to arise in the field that was ultimately caused by a particular SBND failure resulting in an undesirable system level behavior. Almost all multipath software providers have provided documented fixes for these types of failures. An example of a very common multipath driver scenario involving a SBND scenario is having multiple paths across two Storage Area Network (SAN) fabrics where a SBND failure is occurring in one fabric and the multipath software detects a timeout in that fabric. The driver then resends the transaction on another path. The driver then tests the first path finds it available and sends the next transaction down that path which then results in a delayed I/O response which then times out and the multipath software resends the transaction down the good path. When this happens repeatedly the upper level applications like a database start showing severe performance degradations. Fixes were made to the specific Novell SUSE-2011:7794 recommended update with a fix description of 679309: repeated use of flaky paths in multipath module causing performance issue. While this specific example is on a particular product it is the experience of the authors that most multipath software products or functions have had similar fixes generated due to problems encountered in the field. Vendors are working at design time to try an address these situations. For example some vendors have added policy parameters to their multipath software that use a MRU (most recently used) policy to prevent port flapping but eventually have to put the offending path back online if the most recently used path hard fails and then the bad path becomes the most recently used path and the situation can accelerate to a data loss of access event. If the real problem is downstream in an inter-switch link (ISL) the problem will appear to move around making it more difficult to detect correctly and for a multipath software function to behave as needed.

In our system test and SAN labs we have been developing new proactive techniques, protocol inject scenarios and methods to identify, characterize and address SBND failures including cross-component impacts and failures across the I/O path.

A thoughtful and holistic approach is needed in designing these tests. A test engineer can easily create failure scenarios that will never happen or that nothing will be reasonably able to recover from. If the test design does not take these factors into account the test engineer will create scenarios that no development group would agree to develop fixes for. It is therefore critical that test engineers have a very good understanding of multiple components and when things like speed changes occur in the hardware or firmware that may alter the amount of time in either direction for a SBND scenario to occur. Most of the time, these changes and their potential impacts are not explicitly called out in a typical design document. An innocent statement like changed scanning frequency in the firmware to reduce latency could significantly change an error detection rate or behavior somewhere else in the stack of software and firmware.

Our current research related to SBND defects reported shows that the highest number of SBND problems exists along the I/O path. While related problems do occasionally exist within specific internal sever paths they are significantly less frequent, easier to debug and typically contained to a single server and handled via embedded HA mechanism.

Systems generally behave properly when failures are solid or hard failures. It is when components act SBND that system availability is often at risk. In these scenarios failover or recovery mechanisms often do not behave as we should expect them to. Often times the problems are corner cases where they are not easily reproducible and hard to trouble shoot, but continue to plague customer environments. It should also be noted that SBND problems are not something that occur in a particular vendor or product set, but rather a system level event that occurs when one (or more) component(s) in the environment does not always behave consistently. Since the problem does not relate to a particular vendor or component issue it is not a simple fix but rather a system level event that must be fully understood, tested and addressed by all vendors in a distributed systems SAN environment.

The focus of this paper will be on SBND failures related to the I/O path in distributed systems Fibre Channel (FC) SAN and Fibre Channel over Ethernet (FCoE) environments. In this paper we will better define and characterize SBND failures, explain the impacts they can have on complex customer environments and introduce new testing techniques and injections we have deployed in our system test labs. We will also explain the methods used to evaluate the effectiveness of error recovery related to SBND conditions.

II. FAILURE TYPES AND CLASSIFICATIONS [4]

Traditionally, network path failures are viewed as falling into one of two categories, “permanent” and “temporary”.

Perhaps the most well understood and easiest to manage are the permanent failures which result from a catastrophic failure of a network component. These failures are typically persistent failures where all commands routed to the failing path(s) will fail. Commands are recovered through retry down alternate paths and the failing paths are permanently removed from service.

The second category of failure is temporary and transient in nature. These failures can arise from numerous sources including bit flips in electronics due to alpha particle or cosmic rays, as well as electrical noise from intermittent contacts and code defects to name a few. These can produce temporary command failures which are recovered through a single retry operation. These tend to be isolated events, handles via low level recovery and thus most often go completely undetected.

Both of these traditional failure categories are handled well by existing multipath drivers available in the industry today and in fact rarely result in any adverse effects on the operation of the system.

Unfortunately, as the speed and complexity of high speed networks has and continues to increase over time a third category of SBND failures has emerged. These failures are also temporary like the second category; however, in addition to being temporary they are also recurring at varying rates. These SBND failures can arise from marginal components or components and network routes that are insufficiently sized or over subscribed for the volume of network traffic present. Often times these failures are provoked by secondary conditions such as an instantaneous increase in network traffic or a convergence of network traffic. These types of conditions can reduce the quality of a network path(s) resulting in a propensity for them to produce temporary failures. SBND failures are very difficult for the server's multipath driver to detect and typically require an independent monitoring system, therefore, for the driver is impeded from taking action to eradicate the fault and, therefore, the condition to persist for an extended period of time. In many cases the fault persists indefinitely or until manual intervention is performed. SBND failure conditions often drive recursive error recovery by the attached servers which leads to symptoms ranging anywhere from moderate performance degradation to complete system outage.

A further complication of this third category of failure is its difficulty to isolate and resolve. Since commands from servers to storage traverse a large number of switches and links the precise component or conditions responsible for the failures are difficult to identify. Additionally, the underlying problem cannot be contained within the network itself and in most cases the network is not capable producing actionable fault indications that would enable prompt response and resolution from the network administrator.

The existing multipath drivers in use today are not capable of adequately handling this intermittent/recurring failure condition. For the most part all multipath drivers behave in a similar fashion in that they detect and take action on what is seen as individual and disparate events. Path management functions to remove and return paths to service are determined based on the outcome of these individual events.

For example, when a command failure is encountered the recovery action involves a retry operation on the same or alternate path. If one or more subsequent command operations fail on the same path, depending on the thresholds in place, the path will be determined to have failed and the path will be removed from service (first failure category described above is assumed). If subsequent commands are successful the error will often be considered temporary (second failure category described above is assumed) and the path will remain in service. Most multipath software also includes a path reclamation function that periodically tests the availability of each path through the network. If the path test is successful on a path that had previously been removed from service that path will be placed back in service. In response to the intermittent/recurring failure the multipath driver will either leave the failing path in service or remove the failing path from service only to return it to service a short time later following a successful completion of the path test performed by the path reclamation function. A behavior often observed as a result is continuous cycling of paths between offline and online states. It can be seen that based on the application of such logic for both removing and returning paths to service that the implementation of the current multipath drivers are not capable of responding appropriately to SBND failure conditions and, therefore, will not be effective at isolating servers from the negative effects of this condition.

Because specific components and/or conditions associated with these types of SBND failures in the network are often difficult to isolate, the ability to automatically detect and respond to these failures from within the multipath driver is critically important and in fact essential to maintaining a high quality of service.

III. COMMON CHARACTERISTICS OF SBND FAILURES

Most SBND failures are not obvious product failures. Often when problem determination begins all individual products in the environment appear 'healthy' and existing internal diagnostics are not reporting any serviceable events. Even error log reviews often come up clean, making problem determination very difficult. SBND problems by definition are transient errors, meaning network component or a product is temporarily misbehaving, making the side-effects or symptoms in an environment often appear and disappear.

SBND failures are frequently first noticed at the application and/or database layer and are most often initially reported by the customer. The tables below lists the most common impact symptoms and characteristics displayed when SBND failures are encountered.

TABLE I. COMMON SBND IMPACT SYMPTOMS

Moderate to severe performance degradation occurring at sporadic intervals or sustained
Transaction queuing and timeouts
Application abends
HA node failovers
Jobs running longer than usual
Mirror or replication times exceeding Service Level Agreements

TABLE II. COMMON SBND CHARACTERISTICS

Not an obvious product failure, individual products in the environment appear 'healthy' even after detailed internal dump analysis at highest levels of product support
Fault tolerance mechanisms not seeing errors and don't react.
Hard for software and monitoring products to detect, internal diagnostics often do not find anything
Symptoms often appear and disappear
Symptoms often amplify over time

Note: the two tables above were compiled using defect data from problems that were encountered in the IBM system test labs and the IBM field support group from 2010 through 2013.

One might fail to realize the size and/or scope of a SBND failure, by examining the symptoms alone. This is because SBND failures commonly create a sympathy sickness throughout the entire network. Sympathy sickness is when a single device or condition in one part of a network impairs the performance of other devices or other parts of the network. The list below details the most common contributors to SBND failures:

A. *Flaky adapter cards and interface modules*

Adapter cards and interface modules often do not hard fail. Instead they degrade over an extended period of time producing 1000's of bit errors in the process.

Thus a single bad SFP or adapter card in an E-port, can affect the performance of 100's or 1000's of initiators that have their frames transported over the inter-switch link (ISL). A recent study by researchers at the University of Illinois at Urbana Champaign and NetApp Inc. suggests that the majority of failures in data storage system/sub-system are not caused by disk failures, but are being caused by link errors [5]. The study asserts that up to 80% of the storage system failures are not in the disks at all. The authors surveyed approximately 39,000 storage systems with 155,000 enclosures containing roughly 1.8 million disks over a period of 44 months. During that period only between 20% and 50% of the disk system failures were due to the disk drives. The rest came from other causes, most notably SAN component interconnection problems [5].

B. *Dirty connections and cables*

Contaminated connectors and/or interface modules introduce bit errors as traffic rates increase. A link may operate with an acceptable bit error rate until such time as it is loaded down. This anomaly makes dirty connections/cables especially difficult to isolate and identify. In an article written by Steve Lytle from JDSU entitled "Fiber Connector Cleanliness Overcoming a 'Dirty Little Secret'," Steve says, "More than 75 percent of troubleshooting in optical networks results from dirty fiber connectors, a stunning fact first learned by high data rate equipment manufacturers, and later by transport installation teams in the telecom sector. Many in the cable industry may find this surprising, but the problem exists and is quickly becoming intolerable as fiber networks expand" [6].

In 1990 equipment manufacturers were experiencing a plague of dirty fiber connectors, which lead to the establishment of a team of industry experts who performed practical research within a group called iNEMI. This research is now one pillar of a pending international standard that prescribes inspection procedures and pass/fail criteria for manufacturers and operators of fiber-optic networks (IEC-61300-3-35) [7].

Figures 1 and 2 show the light loss and back reflections that occur when there is contamination in a connector. These two figures were created by iNEMI as part of their investigation.

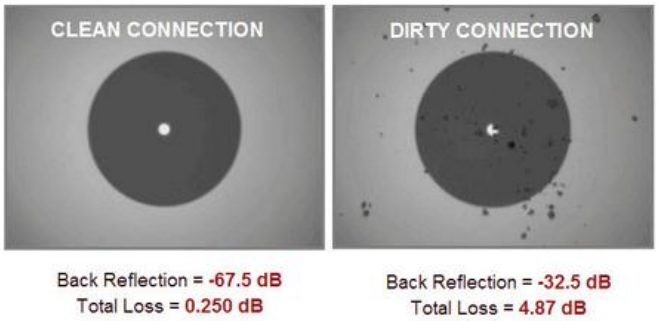

Figure 1. How contamination affect light loss [7]

When contamination is present light levels can be dramatically reduced, as seen in Figure 1. Contamination produces two undesirable side effects, 1) loss of light, 2) reflections. The loss of light reduces the distance that two ports can reliable communicate. Back reflections cause optical resonance in the laser, which creates optical noise further reducing the distance for reliable communication.

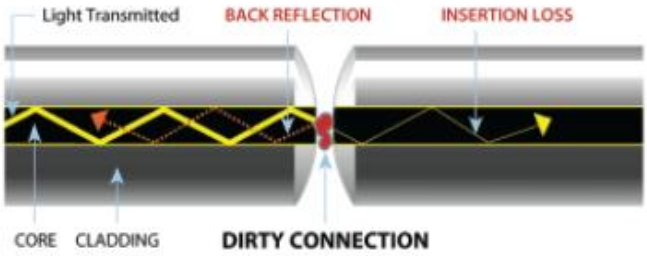

Figure 2. How contamination affects light signals [7]

Cisco, a leading network equipment manufacture, also recognizes that contamination is a problem; they created an Inspection and Cleaning Procedures for Fiber-Optic Connections document in which they state "Any contamination in the fiber connection can cause failure of the component or failure of the whole system. Even microscopic

dust particles can cause a variety of problems for optical connections. A particle that partially or completely blocks the core generates strong back reflections, which can cause instability in the laser system. Dust particles trapped between two fiber faces can scratch the glass surfaces. Even if a particle is only situated on the cladding or the edge of the endface, it can cause an air gap or misalignment between the fiber cores which significantly degrades the optical signal.

- A 1-micrometer dust particle on a single-mode core can block up to 1% of the light (a 0.05dB loss).
- A 9-micrometer speck is still too small to see without a microscope, but it can completely block the fiber core. These contaminants can be more difficult to remove than dust particles" [8].

Figure 3 shows JDSU's recommendation for acceptable levels of contamination, which is based on iNMEMI investigation and years of experience in the test and measurement world.

ZONE NAME (diameter)	SCRATCHES	DEFECTS	
A. Core Zone	(0 – 25 µm)	None	None
B. CLADDING Zone	(25 – 120 µm)	No Limit	No Limit < 2 µm 5 from 2 – 5 µm None > 5 µm
C. ADHESIVE Zone	(120 – 130 µm)	No Limit	No Limit
D. CONTACT Zone	(130 – 250 µm)	No Limit	None > 10 µm

Figure 3. JDSU's Recommendation for Contamination [6]

C. Temporally exceeding the capacity limits of a port/device

Part of a SAN administrator's job is to insure that network capacity is never exceeded. Normally this is not a problem; ports and/or devices that are communally operating at or near their capacity are easily identified. Transient loads however are different, they can only be detected while they are occurring, which makes them very elusive.

When a port/device reaches its capacity one of two things occur, 1) frames are dropped or, 2) frames are buffered until such time as they can be delivered, which can produce undesirable side effects including latency and bottlenecks.

When frames are dropped or discarded in the network it often results in damaged SCSI exchanges which need to be timed out, and subsequently aborted and re-driven by the host device driver. This type of error recovery is very costly given the read/write timeout interval for SCSI retry is in the 20 to 60 second range depending on operating system. Error recovery for command timeouts is generally tolerated at the application and database layers as long as they are transient events; however, recursive recovery for command time-outs is rarely sustainable even when occurring at what seem to be relatively low rates.

Latency and bottlenecks both create back pressure on other devices throughout the SAN. These devices are forced to wait until the condition is resolved before they can resume sending and/or receiving frames. If the wait is long enough (typically 500ms) the switch will begin discarding frames in an attempt to limit the scope of the impact and command time-out recovery will be required on all damaged SCSI exchanges.

D. Buffer to buffer credit problems

A port must have buffer-to-buffer credit in order to originate a frame. Ports without buffer-to-buffer credit are forced to wait until such time as they receive a credit, which creates latency, bottlenecks, and/or opportunity for time-out conditions. There are a number of different circumstances that can lead to buffer-to-buffer credit shortages, these circumstances are listed below:

1) Long Links or High Latency Links:

All ports start out with an initial buffer-to-buffer credit that is established during the login process. The default initial buffer-to-buffer credit value is a median value that is adequate under normal conditions. However, if the links are long or the link has a high latency then the default initial buffer-to-buffer credit values will not be adequate for this environment.

2) Lossy Links:

When a link is experiencing errors it corrupts the traffic flowing over it (frames and primitives). When a Receiver Ready (R_Rdy) or a Virtual Circuit Ready (VC_Rdy) is corrupted in flight they are discarded by the recipient, which result in a loss of credit. Over time these credit losses can slow a link to a crawl and severely impact its performance.

3) Low Speed Device in a Critical position:

When higher speed devices are communicating with lower speed devices buffer-to-buffer credit is used to throttle the frame origination of the high speed device. Normally, networks are configured such that low speed devices are not in critical paths. However, in the event of a link failure a device could be placed in a critical path by a routing protocol such as Fabric Shortest Path First (FPFS).

E. Compatibility issues

All Fibre Channel equipment vendors have a support matrix where they document tested and supported configurations. Most vendors do a good job of keeping this information current within the first several years of a products life cycle, however, in time the information becomes incomplete and even misleading, which can result in interoperability issues that can negatively impact performance and/or availability. Additionally, vendor provided migration paths from aging legacy hardware to newer offerings can also introduce some risk. Many of the methods and functions used to provide a migration path such as "switch interoperability mode" do not receive the same level of testing across the full range of configurations and conditions as best of bread environments and therefore are at higher risk of encountering defects. Moreover, configurations intended primarily to be

used to facilitate migrations often end up being a permanent part of the environment which can further compromise availability.

IV. EMERGING FACTORS

The incident rate for the SBND category of conditions has been on the rise since SANs were initially introduced in the early 1990s. This has occurred as a result of a multitude of influencing factors: (1) Incremental increases in the speed of network ports and devices have placed a higher demand on the quality of the transport layer. Higher speed networks have a higher sensitivity to degradations in link and device quality which can result in transmission failures. (2) Incremental increases in server capacity to drive IOPs made possible by advancements in processor and bus speeds increases the potential to drive port and device utilization beyond their reliable limits. (3) Increases in port and device utilization have also occurred due to increases in I/O density brought about by the increasing use of virtualization of both servers and storage. (4) Added complexity of SAN topologies associated to speed matching required for maintaining legacy hardware, multi-site replication architectures, as well as the sheer scale of networks resulting from growth.

Additionally, the impact rate associated with SBND conditions has also been on the rise due in large part to the increased demand on transaction based workloads. In the past, degradation in performance caused by SBND could often be tolerated to some extent while the problem was being diagnosed and resolved. This is rarely the case any longer where any degradation in performance impeding the system's ability to sustain transaction volume and quality of service are likely have negative consequences for the business and will therefore be considered an outage.

As we move to new computing models such as integrated infrastructure, software defined, and cloud it will become even more imperative that these SBND conditions are dealt with quickly and autonomously. Enabling system design and development to meet these objectives requires a testing approach and methodology that allows these systems to be measured and evaluated on how they perform when subjected to SBND conditions.

V. TEST APPROACH

In a proactive attempt to better address and improve test and design around SBND customer failures, IBM introduced an internal quality improvement effort to better define, categorize and test SBND failures. As part of this ongoing effort, the IBM Systems and Technology Group labs have started introducing a variety of SBND symptoms into complex system test environments using a four-pronged approach. 1. Build a center of competency around identifying, characterizing and debugging SBND failures in the I/O path. 2. Target modified reliability, availability and serviceability (RAS) microcode to better identify and flag SBND failures for troubled areas. 3. Targeted test case coverage related to SBND failures, symptoms and characteristics. 4. Redefine the criteria for success and failure of the new test cases to better reflect the symptoms impacting data centers today. It is no longer sufficient to measure success of error recovery based solely on permanent vs. recoverable error conditions. It is common with SBND conditions for service interruptions to occur yet all I/O be recovered at various layers of the I/O stack, therefore, it is clear that additional criteria that incorporates performance attributes is required to effectively test for this condition. Recursive error recovery while it may be successful has a significant impact on throughput. Although error recovery will always have a measurable impact on performance, error recovery needs to be able to not only recover I/O but also determine and execute actions necessary to remove failing resources/paths from the I/O path. The speed and accuracy at which error recovery is able to accomplish this is the critical measurement of success in handing SBND conditions.

This paper will focus on the 3rd and 4th prongs described above as they relate to increased SBND testing and early results.

A. *Targeted test case coverage related to SBND failures, symptoms and characteristics*

In late 2010 the SAN test labs within IBM began technical analysis on SBND errors and targeted ways to not only inject SBND failures, but to proactively monitor the environment as a whole for related defects and outages. This was a detailed and controlled approach consisting of injects in three primary locations within the I/O path, as outlined in Figure 4 below.

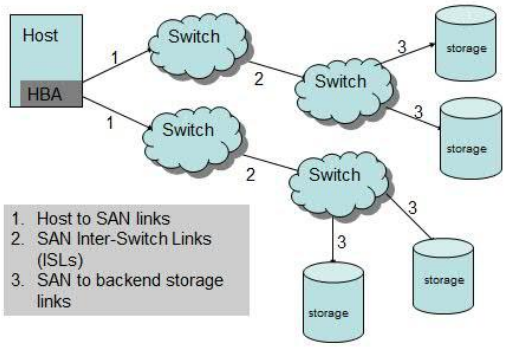

Figure 4. SAN Inject Points

Once the inject areas were established and test tools in place we began targeted testing covering the most frequent SBND symptoms and characteristics described in Tables I and II. Table III below outlines some of the test injects symptoms and test case examples that were created to inject SBND symptoms into our SAN environments to monitor for proper handling and unintended side effects across the environment.

TABLE III. SBND TEST SCENARIO INJECTS

Symptom:	*Types of Injects Used:*	*Test Case Examples:*

Severe Performance Degradation	1. Credit starvation 2. Inject Delay	1. Replace R_Rdy primitives with IDLE/ARB (FC), inject PFCs for Class 3 traffic (FCoE). 2. Hold all frames for *x* microseconds
Mirror or Replication times exceed Service Level Agreement	1. port flaps 2. drop frames 3. jitter	1. Port shut/no shut activity (FC,VFC,Eth) 2. Drop every *xth* frame in each direction 3. Corrupt sof, eof, crc and other header data
I/O redrives or near redrives	1. drop, corrupt or re-order data frames 2. short holds of frames	1. Target data frames and drop or re-order 2. Hold all data frames and/or transfer ready frames for *x* seconds.
Application sensitivity to Recoverable I/O Events	1. virtual link jams 2. link resets 3. corrupt frames	1. FDISC drops, VFC jitter, VSAN jams 2. Inject NOS, OLS, LR and/or LRR onto link 3. Corrupt bits in the FC or FCoE header and recalculate CRC
Product behaviors related to unforeseen external trigger events	1. protocol violations 2. unexpected data returns 3. partial recovery scenarios	1. Inject protocol deviations from standard and monitor destination handling 2. Return Check Cond to write exchange 3. Drop data frame, then drop subsequent ABTS, allow re-driven ABTS to flow through un-jammed.

B. *Redefine the criteria for success and failure of SBND test cases to better reflect the symptoms impacting data centers today*

For years the industry has measured the success/failure of error recovery test cases using *permanent I/O error* as the measurement. In today's client environments the majority of SAN impacts events are occurring as a result of the performance degradation associated with the error recovery, they are not typically caused by permanent I/O errors. Accordingly, verifying that SCSI and related error inject scenarios recovered and did not result in a permanent I/O error is no longer sufficient.

In today's high speed environments the time it took to fully recover and the impact on other I/O (often times non-related I/O) is essential. As Server consolidation and virtualization trends continue, the impacts of non-permanent I/O errors will continue to plague environments and cause severe performance impacts until we change the way we test, develop and measure recovery.

In order to evaluate the effectiveness of error recovery for SBND conditions we had to redefine the test criteria around performance attributes. The critical attributes measured and quantified are 1) the amount of degradation that occurs during the recovery and 2) the length of the recovery defined by the time it takes for performance to return to nominal.

VI. RESULTS

Overall, we established a test suite consisting of over 100 unique SBND test cases, which are run in a controlled SAN environment allowing us the capabilities to inject a single error (or bucket combinations of errors) and monitor the environment as a whole. The majority of the problems we have identified are defects that would have been near impossible to detect and correlate in a customer environment. The ability to understand which variables are being injected at which time and location in the SAN and watching all associated host, switch and storage logs provides the ability to correlate and connect events that otherwise would have appeared to be non-related. Further, having packet level traces at each point in the SAN allows the ability to deep-dive into the traces. Figure 5 below illustrates one SBND inject example where every 5 minutes the Not Operational primitive sequence (NOS) was injected to simulate a bouncing or partially failed port in the SAN. Figure 5 below shows the subsequent behaviors following one of the NOS injects which resulted in failed link initialization. For link initialization to complete successfully following our NOS injects the primitive sequences OLS/LR/LRR/IDLE/IDLE have to be traded sequentially. In Figure 5, we can see one SAN vendor sent extra R_RDY primitives and LRRs prior to sending the final IDLE packets required to complete link initialization. These extra packets prevented proper link initialization and resulted in a substantial delay, impacting link recovery by 20 seconds. Delays of this magnitude produce excessive service times as seen by the application and may result in transaction timeouts, as well as potentially expose application layer sensitivities which could lead to an outage. After the SBND defect was fixed and verified the recovery time dropped from 20 seconds to just milliseconds.

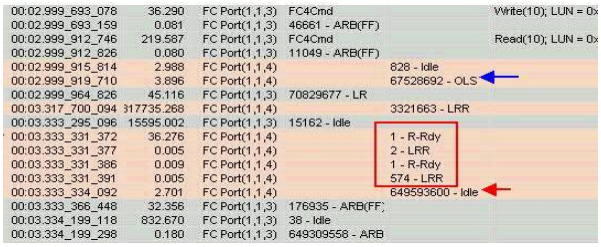

Figure 5. Protocol Trace Review

The protocol trace analysis and frame level debug functionality provides enhanced problem determination capabilities, that when combined with associated host, switch and storage logs present a clear picture of the problem and greatly assists with cross-vendor problem determination.

Typical product system test environments and test plans are designed to analyze and validate recovery capabilities in a product or system offering along with potential implementation architectures and then inject hard errors to determine if products under test were behaving according to specification and customer requirements. A high level example would be a system test environment that had been

designed and implemented with full redundancy of all components in order to minimize Service Level Agreement (SLA) violations [10]. The test engineer would then introduce failures of the components at injectable points in the configuration to validate and verify the system offering would meet SLA requirements. What this technique misses is the "almost errors" that are not specified or articulated as customer requirements. Additionally, there is some level of subjectivity to a SBND event actually occurring and convincing the designers that such a situation would or could exist in the real world. A test engineer also has to use reasonable judgment in designing the injection as any SBND injection can be pushed to unrealistic limits and then the test can be declared invalid. For example, when testing credit starvation one must be cautious in the rate of R_Rdy (frame buffer credit) drops that are injected as too many will cause link resets, replenishing credits back to the agreed upon limit during login. For SBND scenarios, the tester would want to identify the buffer credits allotted during login and drop R_Rdys at a rate which slowly impacts the environment without causing an immediate link reset. It is this careful balance that must be pursued in the test design and execution. Having a test engineering center of competency for SBND problems that can provide real world patterns of these injections is critical to wining the subjective discussions between test engineers and designers.

Since starting this work in 2010 we have seen a dramatic spike in internally found SBND related defects being identified and fixed in system test. In 2010 when we started this testing only 5% of the defects found in SAN system test were related to SBND error handling. In 2012-2013 SBND related defects represent 52% of the overall defects opened by the SAN system test teams. The defects opened are spread across multiple vendors and I/O path components including operating systems, host HBA/CNA firmware and drivers, multipath drivers, SAN and FCoE switch code and storage firmware and drivers. Although defect signatures are often unique, there are common trends that emerge; we will examine those trends in more detail below.

A. Common SBND defect trends

1) Bit errors from a simulated SBND device often lead to error recovery escalation

Bit errors from a simulated SBND device often lead to device driver error recovery escalation and have unintended impacts on non-related I/O streams. Fibre Channel standards allow for a single bit error to occur only once in a million bits (1 in 10^{12}) [10]. In a real world example at 16Gbs that would allow for 56 bit errors per hour. On healthy links bit error occurrences are typically much less frequent, but in any environment occasional bit errors will occur and SBND testing helps to confirm proper handling and recovery. SBND testing related to bit errors typically consists of corrupting or flipping bits at a given location in the environment at a predetermined rate. Inject rates range from multiple errors per second to one error per hour. The frequencies at which the errors are injected greatly vary the results and handling practices. When SBND errors are hit by the same host within a given period of time (varies by host adapter vendor and multipathing option) the host device often reacts by escalating recovery methods and selecting more aggressive task management and error recovery options. The SCSI protocol has a number of task management function defined, some of which impact single initiators tasks and others that affect all commands in the task set or even all commands running within a SCSI target [11]

For example, in one scenario two errors encountered on the same path within 10 seconds led the host device to issue a SCSI Logical Unit (LUN) Reset to the associated storage target LUN. A LUN reset will abort all in-flight commands for the associated LUN and take the LUN back to power on state.

SCSI Target Reset is another task management option which is even more severe than the LUN Reset given its broader scope. A Target Reset will abort all in-flight commands for all LUNs within a given storage target and then take them all back to power on state. It is typically used after other forms of error recovery failed or during frequent SBND events. Target reset was obsoleted by the standard in 2002 due to the harsh impacts it has on environments and non-related I/O streams, however, it is still in active use by multiple adapter vendors today [11].

In our SBND test environments we give special attention to target resets and review each scenario in which they were issued. The goal is to reduce the use of target reset when a different form of task management could be used. In a SBND environment if one host side link is plagued with errors and that host aggressively uses task management commands including Target Reset all other host devices zoned and setup on the shared storage target will be continually impacted. This will result in a large scale performance impact and a difficult to isolate SBND environment impact condition. In the field these problems are challenging to debug as often symptoms first show up on non-related hosts and it takes technical experts and combing through host, switch and storage logs to ultimately identify a single rogue host device impacting the overall environment.

Ideally, in the case of a host device that has a link plagued with SBND errors the multipath driver would identify the problem and remove the paths from their active selection lists, thereby preventing further errors and potential impact to non-related hosts and devices.

In the past year advancements have been made in multipath handling of SBND errors as a result of SBND testing, design reviews and client impact events related to SBND errors. Several multipath drivers are becoming more aggressive at failing paths and more cautious to continually return those same paths to service.

2) Error injects on ISLs are harder to detect by multipath drivers, typically paths are quickly taken offline then put back online within a few seconds

Many of the classic SBND symptoms are caused by a partially failed or flaky ISL. Intermittent problems are difficult to assess, ultimately the fabric must determine if a problem is critical enough to disable the ISL port [12]. Buffer credit

depletion, link flapping or marginal cable or SFP components are most often the cause of SBND symptoms on ISLs. These symptoms are often detected by frame discards, offline/online events or cyclic redundancy check (CRC) errors on ISL switch ports. SBND ISLs can cause intermittent I/O failures, application layer timeouts and severe performance degradation.

To understand the impacts on the environment lets first explore how multipath health checkers typically work. Host multipath software characteristically uses Test Unit Ready commands to check health status of a path. After one or two successful Test Unit Ready commands with good status the path is assumed to be healthy and brought back online if previously in an offline state. In the case of SBND ISLs the errors are intermittent and multipath has no means of correlating the errors across time and detecting the path as SBND.

The end result of SBND ISLs combined with a multipath driver not equipped to handle SBND failures is we see I/O exchanges fail and the associated paths are marked as failed, however, shortly after (typically around 2 seconds) the paths are brought back online. The disproportionately high number of SCSI I/O commands versus Test Unit Ready commands makes it likely that an I/O command will fail and a link maintenance command will succeed. This leads to the offline/online cycling of paths which could continue until manual intervention or the path degrades to a hard fail.

3) Virtualization Components have higher defect rates related to SBND.

Virtualization technologies provide many efficiencies and capabilities and continue to advance and mature. However, virtualization also creates new challenges and debug scenarios for network administrators and technology vendors. The consolidation of virtual servers and virtual links adds complexity and can create bottlenecks, sporadic load patterns and makes it harder to troubleshoot and identify SBND components in a complex virtualized environment.

Links shared across multiple initiators are harder to debug and the sporadic nature of SBND events coupled with virtualized environments makes these errors difficult for multipath drivers to detect and respond appropriately to. Additionally, some server virtualization technologies separate link layer and SCSI layer management responsibilities into entities that reside on different virtual images. This separation adds additional complexity and makes it more difficult for the firmware, driver stack(s) and multipathing software to stay in-sync and respond properly to SBND events.

Host side virtualization related SBND defects often come down to the host adapter firmware detecting SBND events but not passing detailed information up the stack to the SCSI emulation layer so that the device drivers and multipath drivers can make informed pathing decisions. The same concept also holds true for the storage virtualization layer, SBND defects often result from communication breakdowns and faulty state management synchronization between the virtualization layer and backend storage controller(s).

4) Buffer to Buffer Credit Recovery

Buffer to buffer credit recovery is basic and key functionality in a SAN environment. SBND Credit related defects typically fell into two categories; devices who did not handle loss of credit(s) properly and devices that could not perform credit recovery properly. For devices that did not handle the loss of credit(s) properly typical symptoms included large performance impacts or flat lines following the loss of a single or a few credits. For devices that could not perform credit recovery properly the typical behavior was often a credit reset attempt that did not complete properly and would lead to a link reset. Fig. 6 provides one example of a target device that could not properly perform credit recovery. Essentially, anytime the target device port or the connected switch port ran out of credit it ultimately resulted in a temporary loss of access event simultaneously for every host in the fabric zoned to that target port. Due to the devices inability to properly handle credit resets the port would fail, loose-sync, then re-initialize and log back in with the fabric. From the initiator side, paths to that storage port would fail, multipathing would fail over and all open exchanges would have to be recovered and redriven. Once the target device was back online and logged in with the switch the hosts would re-login with the target device and lost paths would recover. Had the initial credit recovery attempt worked the credit reset would have been seamless and unnoticeable to the related hosts.

Figure 6: Credit Reset Causes Link Reset Trace

B. Closed Loop Process for Field Problems

IBM Systems test engineers are often brought in to help debug, recreate, analyze or test fixes for client field problems. Many test engineers are also assigned as customer advocates to client accounts that match their industry and technical areas of expertise. These client interactions help the test organizations form tighter client relationships and provide key field data to be shared with IBM test and development labs via closed-loop processes designed to improve test coverage and client experiences. These models help us to improve test coverage, analyze test escapes and perform coverage analysis across the numerous IBM System Test labs worldwide.

As SBND errors are gaining recognition and focus our client relations help us to better understand the impacts these events have on various network design layouts.

Understanding the implications of these errors on client environments help us to test and architect future solutions that will better handle and coordinate SBND events, ultimately reducing the impacts they will have on future environments.

Outlined below is one example of SBND handling enhancements that were submitted as a result of the closed loop process analysis after client impact events.

1) Multipath Maintenance Improvements

A large client experienced a storm of command timeouts which escalated to LUN and Target Resets failure conditions resulting in serious performance degradation and application failures. The symptoms stemmed from a SBND ISL that was toggling up/down. The AIX MPIO and SDDPCM multipath software would fail when the ISL path went down and recover after health check commands succeeded on these paths, thus continually failing and recovering the problematic paths. After debug involvement and technical review AIX MPIO and SDDPCM multipath enhancements were made to coordinate failures across time to better detect and handle SBND errors and prevent the perpetual failure and rapid recovery storms of SBND related paths. Starting with SDDPCM v2.6.3.0 a new feature and timeout_policy device attribute of disable_path was introduced. The disable_path attribute will permanently disable a path (not the last path) if it experiences timeouts above the set threshold within a given period of time. The path will stay in the disabled state, until the user manually recovers it [13].

C. SBND Implications on Design Review Process

Traditional design reviews focused on network path failures and error recovery based on two traditional category of events; permanent and temporary. Permanent failures are typically the easiest failures to design for and have solid coverage in design reviews. Temporary or transient errors historically have good design review coverage as well but the number of possible transient errors is much greater than permanent failures, making full coverage more challenging to review than permanent failures.

SBND errors have only recently been added to design reviews. The nature and unpredictability of SBND failures makes design reviews for SBND a constant challenge. As we continue to further understand SBND errors and their implications on complex environments we often find that a fix for one situation sometimes further aggravates another. There is a definite balance to be considered when determining how aggressive we can be when addressing SBND conditions and still ensure implemented solutions benefit all environments.

Recent SBND errors have also uncovered additional temporary or transient error coverage review requirements. Design and code review processes have been updated to review for any single try events that may exist in a code path. A single try event mixed with SBND errors or even a single transient error impacting the frame could result in an unexpected failure.

For example, in our SBND testing we have corrupted fabric login (FLOGI) and fabric discovery (FDISC) extended link services frames, many devices were able to handle these corruptions and re-drive login processes, however, we also discovered initiator and storage targets that were unable to recover from these injects. The implications of this testing was if SBND errors impacted extended link service login frames the devices would not be able to recover and would not complete fabric login. These SBND defects were opened with the product vendors and SBND impacts on extended link services were also added to the closed loop design review process.

VII. CONCLUSION AND FUTURE WORK

As complexity, virtualization, business criticality and mixed vendor solutions continue to grow in the IT industry and customer solutions, the need for highly-skilled SBND low-level testing will also continue to increase. In an industry where quality is expected and customer defects can cause costly outages it is no longer sufficient to test products for correct recovery only in hard failure scenarios. We need to continue to put increased focus on solution testing, and further on solution injects and handling of hard failures and SBND failures on any component within the environment. We also need to reevaluate our test pass criteria and design points for complex SAN environments and update them to not only evaluate and design for recovery, but to also evaluate the impact SBND error(s) have on the environment and performance. We need to continue to focus on solutions that can rapidly detect, isolate and address SBND components in an environment. Good progress has been made since we first started this focus in 2010; however, there is still considerable work to be done.

As we continue to expand SBND testing scope described in this paper, we are concurrently pursuing plans to continue this effort with a second phase targeting new inject methods and focus on spreading SBND test capabilities and awareness across IBM test and partner test labs worldwide. Given the economic costs of the tools to inject SBND scenarios and the skill required we are also innovating in economically scalable methods to do this type of testing in more diverse testing and test skill environments. We also continue to drive a close-loop feedback process between IBM test, development and support teams and across OEM partners, ensuring that the SBND defects that have been found are fixed and lessons learned are applied to future product development and monitoring capabilities.

It is our hope and vision that impacts of SBND failures be understood across the industry and that more SBND testing and proactive measures are taken to help minimize the impacts these failures have on the environments of the future.

VIII. ACKNOWLEDGMENTS

The authors would like to thank their employer, International Business Machines (IBM) for supporting their

efforts to produce educational content. We would also like to thank those parties who provided quotations and background art for use in this paper.

References

[1] Tara Astigarraga, Michael Browne, and Lou Dickens "Sick But Not Dead Testing – A New Approach to System Test", VALID 2012, ISBN: 978-1-61208-233-2 http://www.thinkmind.org/index.php?view=article&articleid=valid_2012_1_30_40098 (last accessed 12/09/2013)

[2] A. Hanemann, D. Schmitz, and M. Sailer "A framework for failure impact analysis and recovery with respect to service level agreements", Services Computing, 2005 IEEE International Conference, Content resides on, vol. 2, no., pp. 49- 56 vol. 2, 11-15 July 2005 doi: 10.1109/SCC.2005.10 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1524423&isnumber=32587 (last accessed 12/09/2013)

[3] ATA8-ACS, ANSI Draft Standard T13 Project 1699D, 2004. http://www.t13.org/documents/UploadedDocuments/docs2007/D1699r4a-ATA8-ACS.pdf (last accessed 12/09/2013)

[4] Ian MacQuarrie, William Carlson, Jim O'Connor, Limei Shaw and Shawn Wright, "Configuring SDDPCM for High Availability", IBM Redpaper, October 2012. http://www.redbooks.ibm.com/redpapers/pdfs/redp4928.pdf (last accessed 12/09/2013)

[5] Weihang Jiang, Chongfeng Hu, Yuanyuan Zhou, and Arkady Kanevsky "Are Disks the Dominant Contributor for Storage Failures – A Comprehensive Study of Storage Subsystem Failure Characteristics", USENIX Conference, Storage Technologies 2008 (FAST'08), https://www.usenix.org/legacy/event/fast08/tech/full_papers/jiang/jiang_html/paper.html (last accessed 12/09/2013)

[6] Steve Lytle, "Fiber Connector Cleanliness – Overcoming a Dirty Little Secret", October 2008. http://www.cablefax.com/tech/operations/bestpractices/31826.html (last accessed 12/09/2013).

[7] IEC-61300-3-35, IEC Standard, 2009, avalable for purchase at http://kandk-fi-bin.directo.fi/@Bin/f1c7665a67b79ef92a25406cc614ff18/1386673907/application/pdf/101256/IEC_61300-3-35_Standard.pdf (last accessed 12/09/2013)

[8] Cisco Corporation, "Inspection and Cleaning Procedures for Fiber-Optic Connections", September, 2006. http://www.cisco.com/en/US/tech/tk482/tk876/technologies_white_paper09186a0080254eba.shtml (last accessed 12/09/2013),

[9] B. Rogers, "z/OS 1.11 Sysprog Goody Bag", SHARE Session 2228, March 2010. http://mobile.share.org/client_files/SHARE_in_Seattle/S2228RR092920.pdf (last accessed 12/09/2013).

[10] Jon Tate, Pall Beck, Hector Hugo Ibarra, Shanmuganathan Kumaravel and Libor Miklas "Introduction to Storage Area Networks and System Networking", November 2012, ISBN 0738437131. http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf (last accessed 12/09/2013).

[11] Lou Dickens, "Fibre Channel for the 21st Century" February, 2013, ISBN 978-1-939883-00-1 pp. 219-223 www.yahweheducation.com] (last accessed 12/09/2013).

[12] Brocade Corporation, "SAN Fabric Resiliency Best Practice v2.0", July 2013. http://www.brocade.com/downloads/documents/best_practice_guides/san-fabric-resiliency-bp.pdf (last accessed 12/09/2013).

[13] IBM Multipath Subsystem Device Driver Path Control Module,(PCM) Version 2.6.3.0 Readme for AIX, April 2013. ftp://ftp.software.ibm.com/storage/subsystem/aix/2.6.3.0/sddpcm.readme.2.6.3.0.txt (last accessed 12/09/2013).

[14] FC-FS-3, ANSI Standard 5.2.4-5.2.5, 2008. http://www.t11.org/t11/stat.nsf/1158203694fa939f852566dc0049e810/a7bf7ee8c25bd7b9852572e50079eed4?OpenDocument (last accessed 12/09/2013).

[15] FC-MI, ANSI Standard 3.2.14-3.2.34, 2001. http://www.t11.org/t11/stat.nsf/1158203694fa939f852566dc0049e810/86a95105bd279d148525772000567d1d?OpenDocument (last accessed 12/09/2013).