Sick But Not Dead Testing -
A New Approach to System Test

Tara Astigarraga[1], Michael Browne[2], Lou Dickens[3],
Systems and Technology Group
IBM
[1] Rochester, NY 14626
[2] Poughkeepsie, NY 12601
[3] Tucson, AZ 85744
{asti, browne, dickens}@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; System Test.

I. INTRODUCTION AND MOTIVATION

Despite high availability (HA) configurations, customers are still experiencing outages and severe performance declines in their environments. These outages typically show no signs of hard component failures for which the HA infrastructure would react to and provide recovery. 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 it can take anywhere from seconds to months for the component to finally reach a hard fail state. It is this in-between time when the component is defined as SBND.

Complex customer solutions and environments utilizing mixed vendor products and technologies create textbook scenarios for SBND failures to occur. Many products are intolerant of misbehavior of other devices and most failure paths deal promptly with hard failure scenarios, but are slower and more cautious to react to partially failed, misbehaving, or SBND components in a Storage Area Network (SAN). 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. 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.

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

 ISBN: 978-1-61208-233-2

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.

II. COMMON CHARACTERISTICS OF SBND FAILURES

Most SBND failures are not obvious product failures. Often when problem determination begins all individual products in the environment may appear 'healthy' and existing internal diagnostics often do not flag anything. Even error log reviews may come up relatively clean, making problem determination very difficult. SBND problems by definition are transient errors, meaning 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 host or application layer. The tables below outline the most frequent symptoms and characteristics displayed when SBND failures were encountered.

TABLE I. MOST FREQUENT SBND SYMPTOMS

Severe performance degradation at sporadic intervals
Mirror or replication times exceeding Service Level Agreements
I/O redrives
I/O near redrives
Application sensitivity to Recoverable I/O Events
Product interaction behaviors related to unforeseen external trigger events

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
HA Mechanisms see no error and don't react
Hard for software and monitoring products to detect, internal diagnostics often do not find anything
Problems often appear and disappear
Start slowly and often amplify with time

Note; the 2 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 2012.

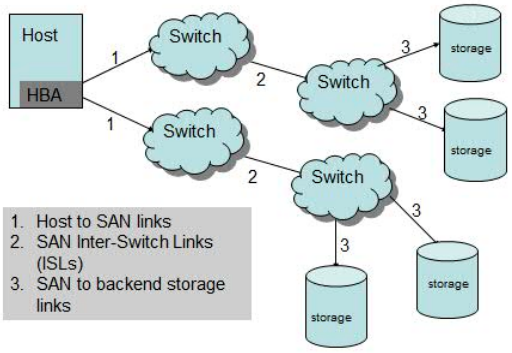
One might fail to realize the size and/or scope of a SBND failure, by examining the symptoms alone. This is because SBND failures commonly create a sympathy sickness throughout the entire network. Sympathy sickness is when a single device or condition in one part of a network impairs the performance of other devices or other parts of the network. For example, a single bad small form-factor pluggable (SFP) in one of the E-ports of an inter-switch link (ISL) can intermittently corrupt frames that are being transported through the ISL [3]. The other switches in the SAN or the end devices will discard these corrupted frames. This will result in the initiators having to perform error recover, and re-drive the corrupted I/O exchanges. Thus one bad SFP in an E-port, can affect the performance of 100's or 1000's of initiators that have their frames transported over the ISL.

III. 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 begun introducing a variety of SBND symptoms into complex system test environments using a three-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.

For this paper we will cover the 3rd prong described above related to increased SBND testing and early results. In late 2010 our 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 1 below.

Figure 1. Example of Typical SAN



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 1 and 2. Table 3 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.

TABLE III. SBND TEST SCENARIO INJECTS

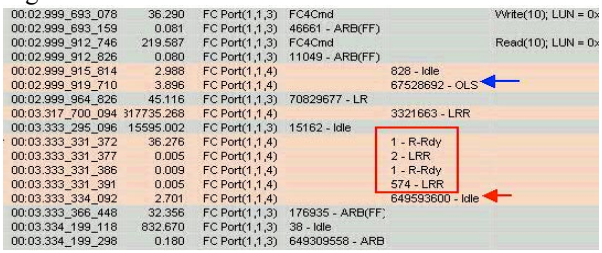
Symptom:	*Types of Injects Used:*	*Test Case Examples: [4]*
Severe Performance	1. Credit starvation 2. Inject Delay	1. Replace R_Rdy primitives with IDLE/ARB (FC), inject

Degradation		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.

IV. EARLY 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 combination 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 2 below illustrates one SBND inject example where every 5 min the Not_Operational primitive sequence (NOS) was injected to simulate a bouncing or partially failed port in the SAN. Figure 2 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 2 you can see one SAN vendor sent extra R_RDY primitives and LRRs prior to sending the final IDLE packets required to complete link initialization.

Figure 2: Protocol trace review



The protocol trace analysis results and frame level debug capabilities, provide enhanced problem determination capabilities and when combined with associated host, switch and storage logs and traces present a clear picture of the problem and greatly aid in cross-vendor problem determination.

Typical product system test environments and test plans will analyze 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 [1]. 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 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 May of 2012, these defects accounted for 48% 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

 ISBN: 978-1-61208-233-2

drivers, SAN and FCoE switch code and storage firmware and drivers.

V. CONCLUSION AND FURTHER DEVELOPMENT

As complexity, virtualization 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 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.

As we continue to implement deeper SBND testing described in this paper, we are pursuing plans to continue this effort with a second phase targeting new inject methods and focus on spreading these testing capabilities and awareness across IBM 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.

VI. 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 for use in this paper.

REFERENCES

[1] 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 on* , vol.2, no., pp. 49- 56 vol.2, 11-15 July 2005 doi: 10.1109/SCC.2005.10
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1524423&isnumber=32587 [retrieved: July, 2012]

[2] B. Rogers, "z/OS 1.11 Sysprog Goody Bag" Presented at SHARE Session 2228," March 2010. [Online]. Available: http://mobile.share.org/client_files/SHARE_in__Seattle/S2228RR092920.pdf [retrieved: July, 2012].

[3] FC-MI, ANSI Standard 3.2.14-3.2.34, 2001.

[4] FC-FS-3, ANSI Standard 5.2.4-5.2.5, 2008.

 ISBN: 978-1-61208-233-2