iSCSI SANs Don't Have To Suck

Derek J. Balling, Answers.com <derekb@answers.com>

Summary: We have created a iSCSI SAN architecture which permits maintenance of network components without any downtime, thus improving our ability to maintain the SAN beyond iSCSI's mediocre reputation. We currently use this system to provide web-based services for all of Answers.com. During development of this architecture we learned three important lessons: Packet loss/congestion is absolutely fatal. Network design must be smart about Spanning Tree events, and a separate network, not separate VLANs, is required. Forethought in design creates a system that is highly maintainable, permitting an "isolate and swap" methodology for upgrades and maintenance.

Answers.com rolled out a cost effective iSCSI SAN solution from LeftHand Networks (now a part of HP's StorageWorks division) in 2008 as part of our first data-center deployment. It was designed for modular growth, scalability, and all the usual checkbox features one would expect. However, iSCSI does not have a reputation for being easy to administer or highly available. We sought to overcome those issues.

A network consultant created our initial configuration with the SAN running on the same network hardware as our regular data-network, separated only by being on a different VLAN. We already had a number of VLANs used to separate data such as front-end servers, databases, and so on, so this seemed to make a lot of sense. At the time, we were using a pair of core switches, for our "A" and "B" side networks. The core switches were connected to each other, but also to each cabinet switch (so "Core-A" might be connected to both "Core-B" as well as "Cabinet1-A", and "Core-B" would be connected to "Core-A" and "Cabinet1-B"). For maximum redundancy, the cabinet-switches were also interconnected, with that interconnect set to a lower priority than their uplink to their respective core switch, so that Spanning Tree Protocol would disable it unless the link to the upstream core switch had failed. Often, "Cabinet" switches would actually be Blade Switches in an HP blade chassis instead of a standard 1U access switch, but the wiring principles were the same (with the "A" and "B" side blade switches connecting internally via some blade-chassis black-magic). Finally, every host and every SAN module in the network had links to both the "A" and "B" networks, either in an active/passive configuration (for Linux)[1], or active/active (for VMware).

---

[1] After extensive testing we saw an order of magnitude better performance having the Linux bonding driver use Active/Passive and only get one NIC's worth of bandwidth than we did when we used Active/Active and tried to get two NICs' worth of bandwidth. It was completely counterintuitive, but since we almost never come close to flooding a NIC itself, Active/Passive was totally fine for our needs.

The only place where we did not use the same network hardware was in the blade-chassis systems, because each NIC on a blade was exposed via a different switch, so NIC1 was mapped to a switch in Interconnect Bay 1, NIC2 to Bay 2, etc. In this way we ended up, effectively, having two data-side switches (in Bays 1 and 2) and two SAN-side switches (in Bays 3 and 4). However, in keeping with our original design docs, those switches were configured identically, other than which VLAN the blade-side ports were configured to be on. Bays 1 and 2 were "A" and "B" networks respectively, as were Bays 3 and 4.

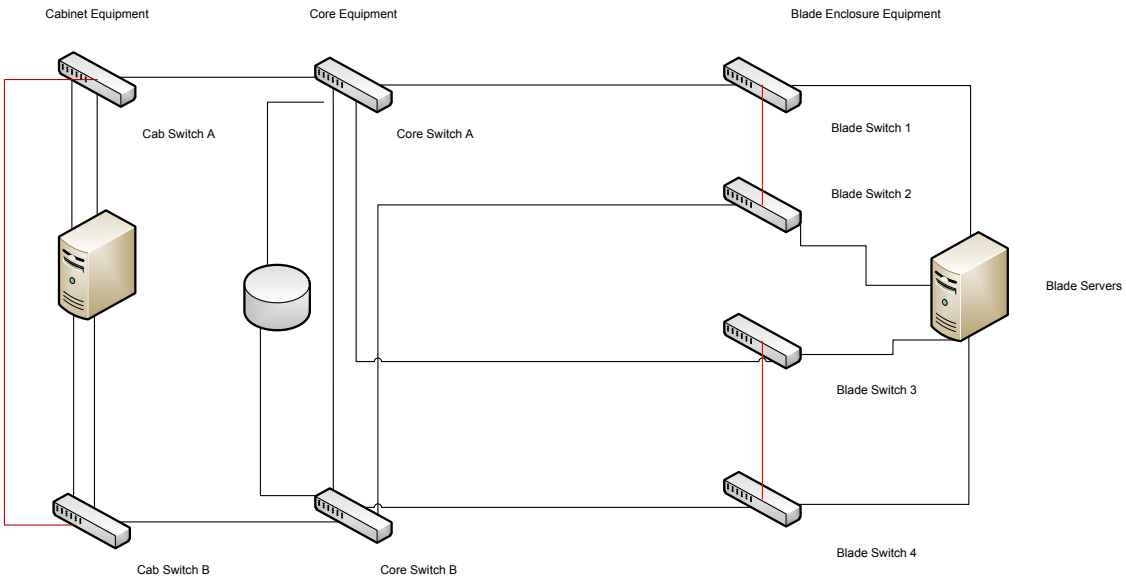

Figure 1 : Unified Network Diagram As Designed

Immediately we noticed, while we were still in the build-out stage of the data-center, that iSCSI was extremely sensitive to even the slightest hiccup in its ability to get packets from Point-A to Point-B. Now, this may sound silly, but we had no idea going into this exactly how sensitive it would be to network issues. If we connected a new set of blade switches, for instance as we built out a new cabinet of blades, the resulting Spanning Tree Protocol (STP) reconvergence – which might last only a small fraction of a second – would often cause iSCSI to give up and put all of the LUNs into read-only mode, effectively crashing their OSes.

Every time we added a new blade chassis (which, during build-out was happening quite often), we would see STP events and more often than not, those STP events would cause a hiccup to

one or more iSCSI LUNs, requiring cleanup and administrator attention. We realized that above all else we would need to minimize the number of STP events in the network to hopefully approach zero if we wanted to trust any portion of our production environment to iSCSI. The first, low-hanging fruit, was that we enabled a feature in the HP Blade Switches called "Uplink Failure Detection". This feature had the switch monitor the upstream link to the core and, if the trunk went down for whatever reason, it would simply turn off link on all the internal ports connected to the blades themselves. So, instead of the blades continuing to send their data to the BladeSwitch-"A", and having that switch send it to BladeSwitch-"B" to make its way to the Core, the blades would automatically start using BladeSwitch-"B" directly, since their "A" sides had gone dark. This allowed us to disable the A/B interconnect in the blade switches, and disable STP on their uplink ports in the core, removing good quantities of our STP events.

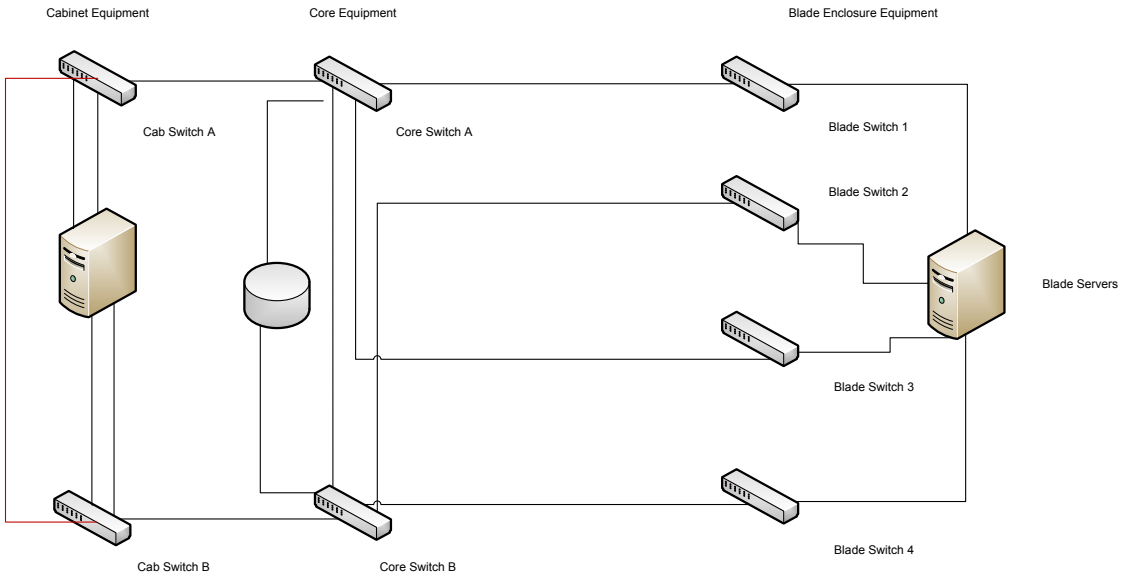

Figure 2 : Network Diagram After Enabling Uplink Failure Detection

But, we discovered it wasn't enough. We'd still occasionally deploy "standard density" cabinets, which for us meant cabinets equipped with 1U access switches, interconnected for redundancy, and building out a cabinet in that configuration, or rebooting one of those switches, would still cause an STP event across the entire network. One thing we'd noticed in our deployment, though, was that our SAN usage, because of the various conveniences inherent to blade systems, had moved

itself almost entirely to the blade infrastructure, so having the SAN VLAN present on the cabinet access switches (and subject to their STP events) was largely counterproductive. We decided to do what we should have done in the first place, and create a separate, electrically isolated, network for the SAN. We would install a new pair of "SANCore" switches, bringing all the Bay 3 and Bay 4 blade-switches into those SANCore switches. For the few non-blade SAN clients, we would backhaul their connections directly into the SANCores.

The practical upshot of this would be that there would only be one "A/B" connection in the entire SAN network, the one between the SANCores. This meant that STP could be completely disabled, as there would never be a potential loop in the environment again (short of someone accidentally creating one, but in our environment of mostly-blade-switches, that was unlikely).

Our network infrastructure's robust design lent itself to a "isolate and swap" process that has worked well for us a number of times now, but got its first real test during the move to the new SANCore switches. The process evolved organically through repeated scribblings on an office whiteboard, and while it seems simple and intuitively obvious in hindsight, every time we discuss it with our peers, the level of redundancy it provides always seems to come as a surprise to them.

First, we disconnected the SAN modules from the B-side core switch, forcing them to use the A-side core. Then we disabled the B-side network completely on all blade and cabinet switches, causing downstream devices to only use their A-side NICs. This isolated the B-side Core, allowing us to move all the physical connections to the SAN from the "Core-B" switch to the "SANCore-B" switch (which was preconfigured to also have all its ports in the disabled state, so no devices would try to talk through it until we were ready for them to do so). The "SANCore-B" switch was then temporarily connected to the "Core-A" switch. Once the connections were all complete, we turned the B-side ports back up, and everything was normal, with no interruptions. After taking a break (this is an excellent place for what we referred to as the "pre-programmed 30 minute hold"), we repeated the process for the other side, to install the SANCore-A switch, and then finally simply removed the temporary cabling that connected SANCore-B and Core-A (since there was no longer anything SAN related connected to either of the Core-A/B switches).

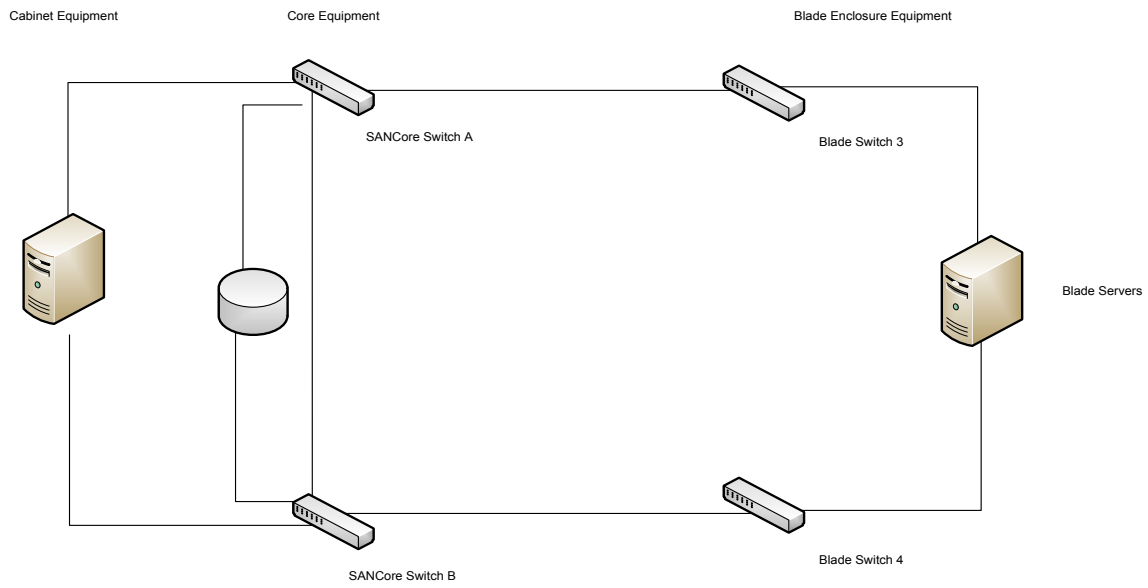

Figure 3 : (SAN) Network Diagram After Upgrade

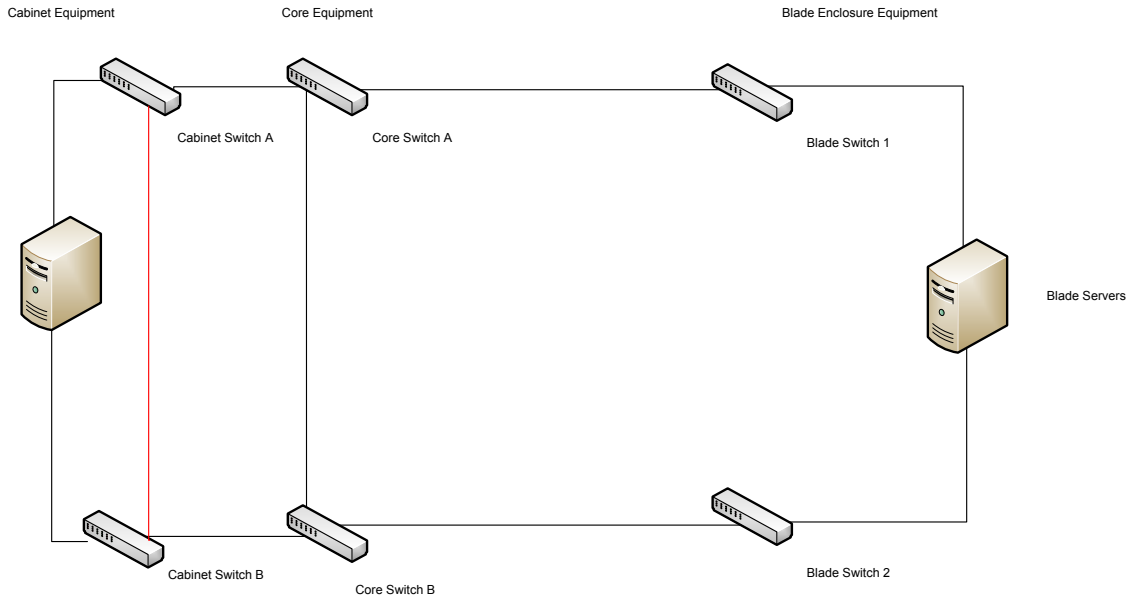

Figure 4 : (Data) Network Diagram After Upgrade

We used this same “isolate and swap” process a couple other times as we have grown to replace and upgrade our SANCore switching infrastructure, while we settled on what hardware we would be using. I knew we were onto something when we would tell people we had hot-swapped out our core-switching infrastructure for our iSCSI SAN without causing a single hiccup in the iSCSI LUNs. We’ve even used similar processes to replace our LAN-side core switches, although that environment

did trigger occasional STP reconvergences, but this was not an issue as standard TCP/IP traffic was more forgiving of half-second or so of packet-loss.

Does it always work? It can, *if* you've prepared your change procedure exhaustively beforehand and follow it religiously. A clear set of step-by-step procedures, which you've designed while standing in front of a whiteboard with one or more of your peers, is essential to success. At every step in the process, you should determine both "Are there any sort of events that trigger when I make this change" (uplink failure detection triggers, STP events, etc.) as well as, "Trace the path that every device will use to try and talk to the network", and ensure that after each step you haven't accidentally disconnected a consumer from its provider. Most importantly, *follow those procedures!* Don't race through your maintenance window assuming that you remember the process, or that you remember the repercussions of various actions. Inevitably, you'll misremember and the results are disastrous. You invested the time in the room with the whiteboard for a reason.

In conclusion, we were able to construct a fully fault-tolerant iSCSI SAN. The key to this redundancy is to have a flat, separate network with redundant hardware. The solution has scaled, for us, to several dozen connected devices, and should scale to significantly larger, but that has not been tested. The design works very well for us, and should be useable by other sites of similar size, as well as larger sites potentially. The big win for us, obviously, is the ability to remove key network hardware for maintenance, upgrades, or even forklift-upgrade, without causing any sort of outage. This "isolate and swap" process was a significant win for ease of maintenance.