PROVIDING SERVICE REDUNDANCY THROUGH VIRTUALIZATION ON A CAMPUS BASED NETWORK

Dustin Rogers and Chris Schroeder
Information Systems (IS)
Information Assurance (IA)
St. Cloud State University
720 4th Ave South
St. Cloud, MN 56301-4498
rodu0601@stcloudstate.edu
scch0305@gmail.com

Abstract

The first goal of any institution is to provide a service to their clients. We find that the security of the service becomes a close second. The goal of this project is to show that using modern, operating system virtualization techniques a variety of services can be made redundant in a cost-effective manner. We will define virtualization and redundancy for the purposes of the examples in this report. Example one will verify the importance of application-level service virtualization redundancy, whereas example two is a virtualized hardware-level redundancy system.

Introduction

Redundancy: The first goal of any institution is to provide a service to their clients. Applying the rules of 'Maslow's hierarchy of human needs' to institutional operations, we find that the security of the service becomes a close second. This is inherently due to the fact that the clients depend on the service, and if the service is made unavailable, even temporarily, the client might find another entity to provide the service. A mainstay of any organization today is the electronic information systems that are utilized by both employee and customer, as they take on the role of a client. For example, an employee may need to access their scheduling information by using their home computer and internet service to access a company web server which provides such information. A customer may access their bank account information from a bank's online service in a similar manner. Another "outgoing" example would be a student that is using the web browser from a lab computer to access the internet for their research.

Various mechanisms affect the quality of these services to the end user (the client). For instance, the student that is accessing the internet from his lab computer relies on the following services, just to name few; internet service from the campus, name resolution service to map a requested alpha-numeric 'URL' to a numeric IP address, and a network firewall service to prevent malicious activity. If administering the dozens of services that are usually required, (in a variety that is usually unique to the organization,) wasn't difficult enough, providing redundancy becomes extremely complicated. This holds especially true when the organization requires that the systems scale to provide a larger client base. This level of complexity can prove to be quite costly in terms of extra computer hardware, network media, and personnel. Although some of these costs cannot be overcome, the goal of this project is to show that using modern, operating system virtualization (full virtualization [6]) techniques a variety of services can be made redundant in a cost-effective manner. Using virtualization, we will first show that a collection of application-level services can be made to be redundant, such as a global file-system. As a second example we will show how a traditionally hardware based redundant firewall/internet gateway system can be virtualized as well. Cost effectiveness is explored as three variables; administrative overhead, approximate power savings, and a rough hardware costs comparison. Scalability logic of the traditional methods will also be compared to those that have been 'virtualized'.

Background Information

What is Virtualization?

Just as redundancy can be a relative term, 'virtualization' has also found a home amongst a slew of contexts in today's IT environment. This is just as well, because we are finding out that nearly every aspect of computing can be virtualized on some form or another. In his report entitled "Virtualization in Software Engineering," Dennis Kalinowski of the University of Wisconsin, Platteville quotes Wikipedia's definition of virtualization. "Virtualization is the application of abstraction to hide the technical detail of a computer's

resources through encapsulation." He continues, "These resources can be software (servers, operating systems, applications, etc.) or hardware (storage devices, processors, etc.)."[6] One good hardware example is found in almost every household computer today, which is 'virtual memory'. Virtual memory can be simply described as reserved space on the hard disk that is used similar to physical memory, or random access memory (RAM). For the purposes of this report, we will be discussing full virtualization of operating systems which, as stated, is a form of software virtualization.

Virtualization of operating systems is essentially logically dividing the total system resources of a single computer among many different logical computers. There are many benefits to logical division of physical resources in this fashion. One reason is that it allows a system administrator to make use of expensive hardware as efficiently as possible. For instance, one particular organization may have a physical web server that uses twelve percent of its system resources, and an external DNS (name resolution) server that use only nine percent of its system resources. By creating two virtual machines, (which will provide the two services,) on one of the physical hosts, they would still effectively be utilizing only twenty-one percent of the total resources. Of course, the physical system (host or hypervisor) that organizes the virtual machines (guests) requires resources of its own; so a direct linear resource savings is not quite the effect. The more guests we can have on a host, without 'bogging' down the host or the guests, the more resourceful we are being. The outcome in this circumstance is that we have freed the resources of an entire computer, which can be used to serve another purpose.

It is important to remember that when we use fewer computers, which is what is essentially being done with virtualization, we use less electricity, which also saves money. From a security and reliability point-of-view, if a particular machine is compromised for any reason, the other guests, as well as the host are logically separated. For example, in the case of the host that supports a web server guest and a DNS server guest; if one of the guests were to be compromised by an attacker, it is safe to assume that the other would be directly unharmed.

It is for the aforementioned reasons that virtualization is invaluable to system and network administrators today. Creatively, administrators can extend the concept of virtualization resourcefulness to many realms. Virtual machines make great test beds. In fact, it is safe to say that without virtualization as a test-bed creating tool, many simulation experiments could not have been achieved. Kalinowski quotes the proceedings entitled "Virtualization of Linux based computers: the Linux-Vserver project" by B. Ligneris at the 19th International Symposium on High Performance Computing Systems and Applications,

> "When designing software for platforms or environments different than the developer's machine, virtualization can be used as a tool to build or test the software. Alternatively, if a developer does not want to contaminate his own system, builds and tests can be done within an isolated virtualization environment. These approaches vastly improve portability of software and allows ease of development." [8]

Perhaps the most beneficial side effect, created by the tool of resourcefulness that is virtualization, is redundancy. Freeing up those extra system resources allows system

administrators the benefit of creating secondary systems in case the primary server fails to provide service for any reason. The purpose of the following section is to provide a background on system redundancy.

What is Redundancy?

As stated in the introduction to this report, any given network is going to require a variety of services to provide communication functionality…even at its minimum. Providing redundancy becomes an immediate secondary objective for most network administrators. However redundancy is a relative term, as it can be used in a variety of contexts. In this section we will define redundancy for the purposes of the examples in this report.

The first system of redundancy that most people think of is a 'backup' of the data, which can be restored in case of an emergency. Other people may feel that a redundant system is one in which a secondary server is present, as well as, ready to perform in case the primary system fails. We consider this 'fail over' redundant system. For the sake of this paper, we will refer to fail-over systems in which the second system is operationally running, (therefore allowing real-time transfer of the service from the dead primary server to the backup.) We mention this because some methods involve 'waking' a secondary server, which in turn a starts the backup service automatically. Finally, the optimum form of a redundant system is one in which various other valuable 'features' are made available through the service, on top of providing fail-over functionality. The main feature in which most system's administrators are primarily interested in is 'load balancing', which occurs when the secondary and primary servers are able to share the load. Since the latter system makes use of the hardware in the most efficient manner, we consider this system to be the most effective from a total cost of ownership (TCO) perspective.

Of course, referring to redundant systems as merely 'primary' and 'secondary' can be considered outdated due to the fact that many organizations need services that are so robust that even a two computer, load balancing system is not nearly enough. Many organizations today require that a 'cluster' of computers are configured to provide massive levels of load balancing, allowing many machines to cooperate to complete a difficult task. Adding, (and sometimes subtracting) from the cluster is known as 'scaling'. When a fail-over, redundant system with load balancing easily allows more hardware to be added for scalability, in a reliable fashion, then that is considered an 'enterprise-level' system. In the case of our example for this report, we will be showing that using virtualization, a system administrator can provide fail-over redundancy, both physically and logically, for a variety of services.

Example 1 – Redundant Network Services using Virtualization

The importance of one particular service that is easy to understand is a network file system. A network file system (NFS) is used by organizations as a centralized location for storing user's files. It provides easy an access point for network clients, as well as, promotes effective and efficient file management. The importance of a stable/reliable NFS or any other

network service becomes abundantly clear. If the NFS is for some reason unreachable, employees are unable to access the files needed to do their job; and productivity suffers. In another scenario, students may not be unable to access critical files for an upcoming lecture. The need for a redundant, fail-over system is obvious.

NFS can be configured on almost any Linux or Unix distribution available. Once NFS is installed, it is possible to configure a Distributed Replicated Block Device (DRBD) service to synchronize the files between the primary and secondary servers. This would be in conjunction with a Heartbeat service, which is used as the control instance of the high availability (HA) service to properly support our replication. DRBD mirrors block-data, storage devices (such as hard disks, partitions, redundant array of inexpensive disks (RAID) devices, and logical volumes,) between multiple hosts. The replication is transparent to other applications and services running on the host systems. In simplest terms, the solution uses DRBD to create a exact mirror (technically known as RAID level one,) over a network between multiple hosts. In a non-virtualized world these two machines would be separate physical entities

We have solved this same problem using virtualization by creating two virtual machines on two separate physical machines. One might ask where the cost savings would be if two physical machines still needed to be purchased. Considerations need to be made with the idea that an NFS is just one of many services needed to support an organization. In fact, since many virtual machines can exist on one physical host, two 'off-the-shelf' computers can host, and replicate, many services.

This set of software coupled with virtualization creates a unique combination that allows the system administrator to pool server resources and provide file storage that is more efficient and effective in servicing its clients. It relies on this software configuration to support a redundant NFS which provides HA services with minimal increases to cost. Another advantage that needs to be addressed is the added value of using a Virtual Disk Image (VDI). This allows the system administrator to virtualize hard disk drives, where by 'snapshots' can be taken for easy system recovery. Snapshots are essentially operating system 'restore' points. If the server is having problems, the virtual machine can be shut down, at which point the VDI file can be replaced quickly and easily by the administrator. The virtual machine can then be started with little to no additional configuration needed; saving time and money in a disaster recovery plan.

Example 2 – Redundant Firewall Strategy using Virtualization

Once software type network services have been made redundant using virtualization, it is time to turn the attention to systems that are traditionally hardware based. One of the most common shortcomings of small organization's information systems is failure to provide network connectivity redundancy. Many network administrators will have their organizations services made redundant on two separate servers, each with; hard-disk arrays, dual network cards, and dual power supplies (each leading to a separate uninterruptible power source (UPS), which are plugged into separate circuits.) See Figure 1. The problem is that the firewall becomes a single point of failure. (We also see that the switch and the network

connections themselves become points of failure. This will be addressed at the end of the example.)

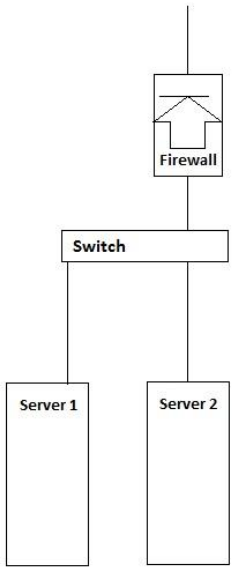

Figure 1: Server 1 is primary and Server 2 is secondary. Both servers would have dual power supplies, each leading to a separate battery.

Network level firewalls are crucial for any organization's production system. Although software firewall solutions have been available for some time, most system administrators consider them unreliable, or unable to perform as well as their hardware counterparts. This is why most organizations today use hardware firewalls, such as a Cisco Systems Inc. PIX Series firewall.

The problem for smaller organizations, hardware solutions for every baseline network device can become costly. Another issue is that small organizations must either employ a systems administrator that is familiar with all of the various proprietary operating systems that come installed on the hardware devices, or they will need to outsource that service. Both of these solutions can be costly.

In order to provide an effective firewall redundancy strategy in a traditional, hardware or software based environment, a system administrator would have to configure two hardware firewalls separately, which can be quite extensive. Then they would need to configure 'static routes' and 'protocols' that designate traffic paths in the event of a failure. This could look as simple as Figure 2, or as complex as Figure 3, depending on the level of connectivity redundancy we are trying to achieve.

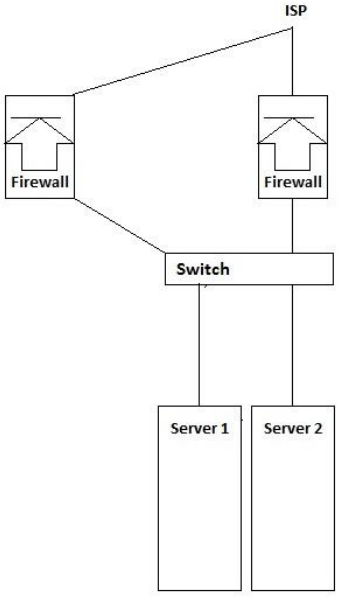

Figure 2: – Simple redundant firewall and server configuration.

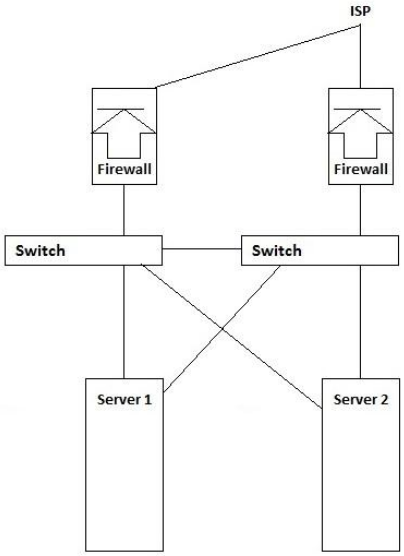

Figure 3: – System that provides more connection redundancy.

However, software based firewalls have become excitingly more powerful in recent years. Smoothwall software is a hardened Linux kernel that an administrator can install on a computer with two, or more, network cards. Once the Smoothwall operating system is installed it can easily be configured to filter specific traffic from one subnet to another. This works exceptionally well for system administrators that wish to separate a small lab or office environment from a larger network. See [5]

Just as in Figure 2, the fact remains that most of the user computer terminals, in any environment, will have one network interface, and therefore, the network card, network cable, or connected switch becomes a single point of failure. Using the logic of Figure 3, we can virtualize many guest servers on one host, which makes it fairly easy to configure two network interfaces per guest, since they guests would/could all share. Then we would plug those into two, separate crossover-linked switches to provide full redundancy. Figure 4 shows what this would conceptually look like.

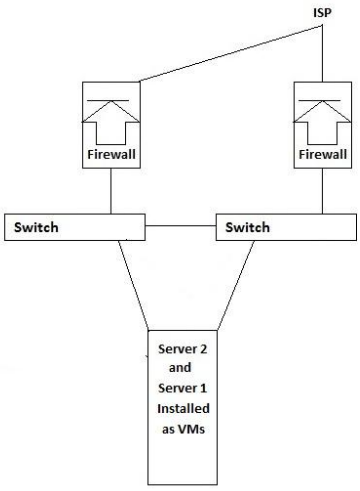

Figure 4: Simple network with virtualized servers and two firewalls for redundancy.

Implementation

Here is where we begin describing our virtualized, redundant firewall strategy. Inherently, the research lab was a dual internet service provider production system. We had pushed the network mask down by one bit, from /24 to /25, effectively dividing our 255 address network into two halves. Each ISP was assigned to handle the traffic from one half. The clients were configured to use one firewall/gateway, or the other, depending on which half of the network their IP address resided. (We used the software-based, kernel-level iptables firewall package.) See Figure 5

The problem occurred when anything stopped network connectivity on either ISP. This could have been that the ISP was having problems or some part of the network medium on our side was having difficulties. Basically, which ever clients were configured to use that particular ISP would lose connectivity until the matter was resolved.

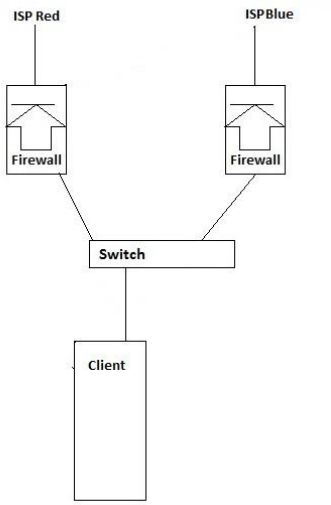

Figure 5: Virtualized servers on a redundant network.

We decided to look at the various options for making our entire system redundant, at least down to the switch. Our first option involved a hardware solution. Figure 6 shows how a dual-ISP system can provide full network connectivity redundancy using many hardware devices, and sophisticated routing protocols such as border gateway protocol (BGP) and open shortest path first (OSPF).

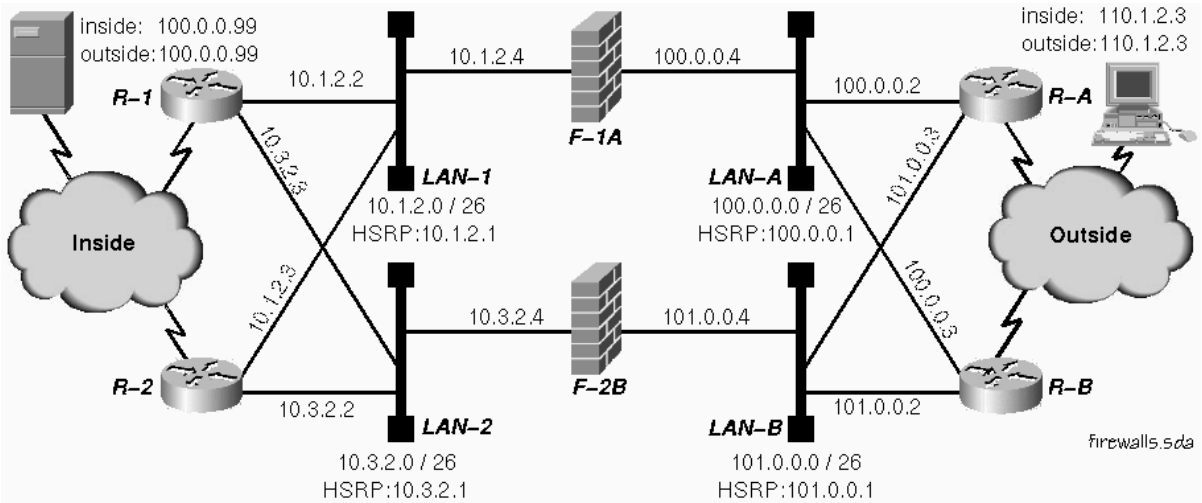

Figure 6: Example of a redundant firewall system exclusively using hardware devices.

However, we did not have the hardware, and did not have the funds to obtain it. So we thought to approach the design with the 'iptables' software package. Since 'iptables' is a service, we figured we could use the same 'heartbeat' approach we had used to provide global file system redundancy. The only modification that we would add for a deidicated heartbeat would be a direct crossover link between the two firewalls. After doing some research, we found a nice software package called pfSense that seemed to provide the firewall functionality we planned on using iptables for. Moreover, it had a built-in fail-over service known as common address redundancy protocol (CARP). We then devised a 'divide and conquer' plan; which meant we would first provide full redundancy for the 'ISP Blue' side, and then duplicate the strategy and create 'ISP Red' side redundancy.

pfSense is another software firewall/routing package that is based on the Free BSD distribution of Linux. pfSense is extremely sophisticated, and rivals even the most expensive hardware firewalls in terms of functionality and features. Of course, the relative speed of any software firewall is going to be based on the hardware invested in the machine on which it is being operated. We installed pfSense software on two machines, and then, using the built in CARP fail-over functionality built into the software, provided redundancy. We forced the fail-over traffic through a dedicated third interface on each of the firewalls. Figure 7 depicts the test environment we created.

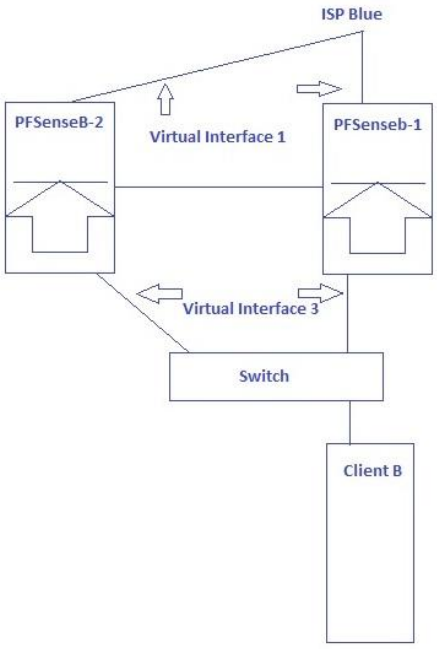

Figure 7: A simple network using two pfSense software router systems for redundancy with dual ISPs

As you can see, virtual interfaces are created that the two firewalls share. This is yet another example of virtualization. Clients then refer to that interface, for instance, as their internet gateway. This is the same logic that 'Heartbeat' uses, in that, if one firewall goes down, the

other firewall continues to serve the client by taking control of the virtual interface. ‘pfSense Blue 1’ is the primary firewall, and ‘pfSense Blue 2’ is the backup firewall.

The other side of the network ends up looking like a mirror image of the ‘ISP Blue’ side, and this is shown in Figure 8.

Putting the ‘Blue’ and ‘Red’ networks side by side; we have configured a system in which both firewalls have been made to be redundant. Using four physical hosts, each with three interface cards, this system would unfortunately fail to provide connectivity to one side if the other went down. In fact, the two sides are obviously mutually exclusive.

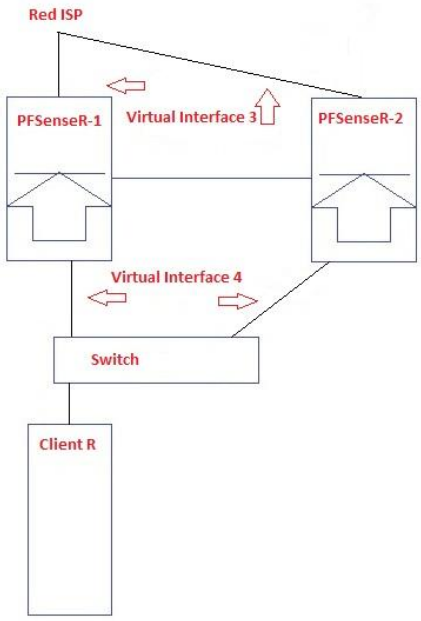

Figure 8: Second half of a redundant, dual-ISP firewall system.

The beauty of virtualization is that we can physically ‘combine’ our two networks. That is, by virtualizing our ‘pfSense’ installations and then putting two VM’s on one machine. Now each of the two physical hosts will need five physical interfaces. Figure 9 depicts the collaboration of the two networks in a virtualized environment.

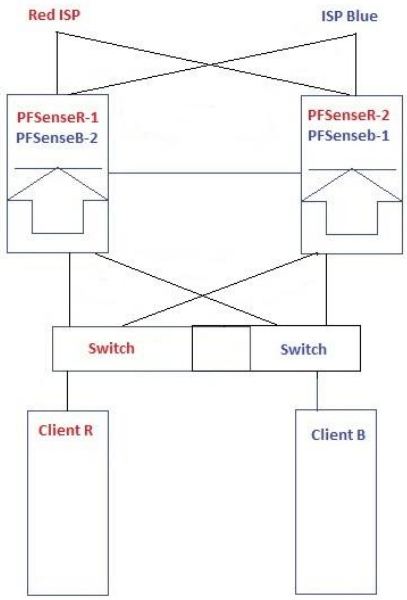

Figure 9: A completed. full redundant fail-over system using virtualized firewalls.

We left the separators between the switches to show that they can be combined into one switch, or left separate. Combining the switches is fine, since most clients really only connect to one switch, it becomes a single point of failure. (Installing dual network interface cards in the client, and connecting those to separate switches would make entire system redundant. For critical systems this may be the best choice.) However, we must keep all four interfaces from the firewalls, because each pair is forming a virtual interface. Therefore, by combining the two networks, we have actually made the ISP connections redundant as well.

In the end, we had a fully redundant firewall system for a dual-ISP fueled network. The final product, in our case, is using two 'off-the-shelf' computers with ten network interface cards, combined. This is opposed to the two hardware firewalls, four routers, and (most likely) four switches used in the hardware solution. Our software solution before virtualization (Blue and Red networks, side-by-side) uses four 'off-the-shelf' computers, with a total of twelve interface cards, combined.

Conclusion

With the need to be ever more responsible in an overtaxed environment the concept of virtualization offers enormous benefits in regard to saving resources in IT operations. In 'Example 1', the number of computers required to run a production domain are reduced considerably. For example in our organization, we were able to reduce physical resources from nine servers to one. Assuming the average power consumption per box at a conservative 100 watts per hour, this would be a savings of 800 watts per hour or 19.2 kilowatts per day. At 10 cents a kilowatt hour that is over $700 per year. Further, this value

could be, (at least,) doubled when one considers the heat that does not need to be dissipated by air conditioning.

Also, less hardware needs to be purchased to perform the same tasks, which means less money is needed up front to upgrade existing equipment. Specifically, [7] found that in a case study involving a six host virtualization disaster recovery case study that the virtual architecture could save $39,000 in personnel costs and over $100,000 in hardware cost when compared to a traditional hot disaster recovery model.

As far as scalability is concerned; in 'Example 1', we can see that since we are effectively utilizing our physical resources (with a nine-to-one ratio.) Organizations implementing virtualization to provide service redundancy can increase their business volume and minimize hardware costs, and provide full redundancy. Organizations interested in 'Example 2' can see that by replacing our redundancy crossover cable with a switch (Figure 9) we can add more five-interface, physical hosts, and in doing so, add additional firewalls in the rare case the others fail to provide a service. Of course, this scalability logic has its limits as well, but not nearly that of traditional physical systems.

'Example 2' shows us that we can creatively apply virtualization in a 'network overlay' fashion, which also creates a collaboration of physical resources. Virtualization can be applied in a 'test-bed' setting; and then easily transferred to production systems when the time is right. All of these factors lower costs from both a hardware perspective, and through personnel, due to ease-of-administration.

References

[1] Calzori, Federico. (2006) "High Availability Using Vitualization." Diss. UNIVERSITÀ DI PISA, *ArXiv.org E-Print Library*. Cornell University Library. Web. 23 Jan. 2010. <http://arxiv.org/pdf/0910.1719>.

[2] Chiueh, Tzi-cker. *Resource Virtualization Techniques for Wide-Area Overlay Networks*. *CiteSeerX - Scientific Literature Library and Digital Search Engine*. CiteSeerX. Web. 27 Jan. 2010. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.3560&rep=rep1&type=pdf>.

[3] *DRBD:What Is DRBD*. LINBIT. Web. 4 Mar. 2010. <http://www.drbd.org/>.

[4] Foster, Ian, and Carl Kesselman. (2004) *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, Calif.: Morgan Kaufmann, Print.

[5] Guster, Dennis C. and Charles Hall. (October 2001) "A Firewall Configuration Strategy for the Protection of Computer Networked Labs in a College Setting". Journal of Computing in Small Colleges 17:1 181-187,

[6] Kalinowski, Dennis. (Spring 2007) *Virtualization in Software Engineering*. University of Wisconsin,. Web. 23 Jan. 2010. <http://www.uwplatt.edu/csse/courses/prev/csse411-materials/s07/Dennis%20Kalinowski-Virtualization.doc>.

[7] Lee, Olivia F. Dennis C. Guster, Mark B. Schmidt, and Brandon McCann. (2009) 'Applying the scale-free degree distribution algorithm to assess communication complexity and failure points in disaster recovery models', *Journal of Information Technology Management,* In press.

[8] Ligneris, B. (2005, May). Virtualization of Linux based computers: the Linux-VServer project. *High Performance Computing Systems and Applications, 2005. Proceedings. 19th International Symposium*. 340-346

[9] *Linux-HA*. Web. 4 Mar. 2010. <http://www.linux-ha.org/wiki/Main_Page>.

[10] Mochizuki, Yojiro, Hiroshi Maeda, and Masafumi Sadakari. *Virtualization-based Operation Support Systems:Improved Service Availability and Dynamic Resource Management*. Tech. NTT Comware Corporation, Japan. Web. 23 Jan. 2010. <http://mnet.skku.ac.kr/data/2007data/APNOMS2007/Papers/InnovationSessions/I2-3.pdf>.

[11] *PfSense Open Source Firewall Distribution - Home*. BSD Perimeter, LLC. Web. 4 Mar. 2010. <http://www.pfsense.org/>.

[12] *SmoothWall Express*. SmoothWall, LTD. Web. 4 Mar. 2010. <http://www.smoothwall.org/>.

[13] *Ubuntu Home Page | Ubuntu*. Canonical, LTD. Web. 4 Mar. 2010. <http://www.ubuntu.com/>.

[14] Virtualization. The Wikimedia Foundation. Web. April 2, 2007 http://en.wikipedia.org/wiki/Virtualization

[15] *White Paper: Configuration for Transparently Redundant Firewalls*. Networking Unlimited,Inc. Web. 10 Sept. 2009. <http://www.networkingunlimited.com/white001.html>.