SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 5, NO. 5
ISSN: 2054 -7420

An Achievement of High Availability and Low Cost on Data Center Infrastructure

Yen-Jen Chen and Po-I Lee
Dept. of Electronic Engineering, Ming Chi University of Technology, Taiwan, ROC
yjchen@mail.mcut.edu.tw, M04158002@mail2.mcut.edu.tw

ABSTRACT

This study is to provide a design of low-cost and high-availability data center infrastructure for small/medium-scale businesses. The basic concept is to establish an infrastructure consisting of primary and backup sides through the clustering technology. By default, the primary side is active and responsible for data switching while the backup side is standby for the failure of the primary side. The design achieves the features of 1) providing full-level high availability (HA) at network, server, application, and management levels, 2) controlling the routing among network-level clusters to solve the "PPPOE connection racing" and "winding path" problems, 3) monitoring and recovering the objects in each level with an economic and effective way, and 4) handling events resulting from changing of object states in an event center. Finally, the experiment results are exhibited with five testing scenarios for verification and elaboration of the effectiveness of the HA design. The system can recover the failed objects and solve the routing problems of PPPOE connection racing and winding path among HA clusters automatically.

Keywords: Cloud; Data Center; High Availability (HA); Virtualization Clustering; Failover; Load Balance.

1 Introduction

With the development of Information Technology (IT) in recent years, the concepts of Cloud [1, 2, 19] have been utilized in the data centers of enterprises. Cloud concepts emphasizing on a High Availability (HA) [3] of a system, are usually realized by virtualization and clustering technologies [4]. In a large-scale enterprise, it can hire a lot of IT professionals and purchase an expensive HA solution to secure its data. However, for small/medium-scale companies, where the number of employees below 200, they cannot provide the resources as those of a large-scale one. There are many small/medium-scale businesses facing an IT security challenge as same as a large-scale one. The difference between them is that the large-scale one has larger traffic volume than those of the small/medium-scale ones. Therefore, a low-cost and high-availability design of a data center for small/medium-scale business is very essential [5].

This paper proposes a design of an economic, effective, and high-availability solution for a data center. The basic concept is to establish an infrastructure consisting of primary and backup sides through the clustering technology. By default, the primary side is active and responsible for data switching while the backup side is standby for the failure of the primary side. This can be implemented with clustering technology. However, there are several clusters in a data center and the routing between them is not

DOI: 10.14738/tnc.55.3777
Publication Date: 26th October 2017
URL: http://dx.doi.org/10.14738/tnc.55.3777

considered in the current clustering technology. The primary and backup devices in a cluster determine whether the other one is alive by issuing "heartbeat packets" [6] to each other. The devices also determine that each connected network segment works properly by checking whether the individual monitoring point, created at each segment, is alive. If an active device detects one segment failed, it will become standby and notify its backup device to be active. However, it is a huge effort to arrange a monitoring point in each segment and maintain it always up and running. An unstable monitoring point will make the devices in a cluster changing between active and standby frequently. A simple approach to detect the states of a device in network segments is to check the states of the device's interfaces connected to the network segments. In fact, this is an effective way since a network segment failed will not be recovered even if a standby device becomes active and take over the control. Although the technologies of Spanning Tree Protocol or EtherChannel are the failure recovery solutions for a network segment between two Layer 3 devices, these two techniques are not being emphasized in the current design.

To solve the problems mentioned above and consider a full-level of High Availability (HA) on a data center, the proposed design applies clustering technology to introduce HA in the network and server levels individually and Load Balancing (LB) [7] in the application level. In addition, it utilizes a monitor server to monitor the network-level devices for controlling the routing among clusters. The monitor server recovers the objects in network, server, and application levels by issuing reboot or restart commands to the objects. The monitor server also acts as an event center and uses Syslog protocol to record the events of state changes or transitions. The event center provides a web page to indicate the states of objects at the individual levels and other pages to display the details of events describing the state changes. In order to achieve management-level HA, dual monitor servers are designed and deployed to support the resilience of event center functionalities.

The proposed design for the data center infrastructure has the features of 1) providing full-level HA at network, server, application, and management levels, 2) controlling the routing among network-level clusters, 3) monitoring and recovering the objects in each level with an economic and effective way, and 4) handling events resulting from changing of object states in an event center. It can achieve a high-availability and low-cost infrastructure since the HA functions and monitor mechanisms are designed in these four levels as mentioned previously and implemented by free, open source software, such as Linux Virtual Server (LVS) [8], Vyatta Router and Firewall [9], and NMap Scanner [10]. The "low-cost" is also included in the design that events are only generated on state changes. This saves the space and time to store and de-duplicate events. The rest of the paper is organized as follows: Section 2 presents the background knowledge and related work of the proposed design. Section 3 illustrates and presents the details of the data center infrastructure design and its implementation. Section 4 shows the experiment results for verifying and elaborating of the approaches. The final section concludes the paper and describes the future work.

2 Background and Related Work

2.1 Background knowledge

High Availability is a commonly seen cluster mechanism. The goal of cluster is usually to maintain services and allow them to operate at high stability. Database servers in corporate environments often implement this mechanism by setting two database servers as a high availability cluster. If one database server is damaged due to uncontrollable reasons, the other database server can automatically takeover in a short period of time and the user will not notice any service interruption.

Load Balancing is another cluster mechanism allowing for a large number of service requests. The Front-end application server in a corporate environment often plays a role by allocating multiple application servers to simultaneously service a lot of requests from the client side. This mechanism works by accepting user requests in the load balancer, then using load-balancing rules [11] to allocate requests to application servers for processing. The structure is as shown in Figure1, where the real server refers to the application server.

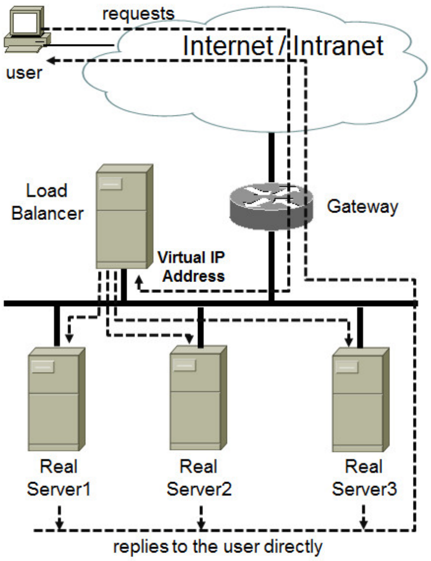


Figure1. Network and system architecture of Load Balancing

Heartbeat is as the word suggests. It is a mechanism responsible for monitoring cluster services. The heartbeat mechanism in a cluster node framework as shown in Figure2 will ping to detect if the opposing machine is alive. The heartbeat packet, e.g. one detection every 2 sec, is sent and if the opposing machine cannot be detected after a period of time, e.g. 30 sec, then the mechanism will determine operation failure for that cluster node and trigger a preset action such as taking over the services of the broken node. Heartbeat suite currently supports the following network signaling types:

- Unicast UDP over IPv4
- Broadcast UDP over IPv4
- Multicast UDP over IPv4
- Serial Link (Console Port)

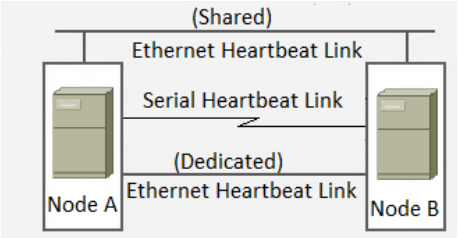


Figure2. Network architecture of Heartbeat mechanism

Failover mechanisms usually refer to the cluster services that are protected, such as server, operating system, network, application, etc. When an error or abnormality causes service interruption, the mechanism will automatically repair online services without human intervention. When using high availability heartbeat suite, the failover technology is achieved using “IP address takeover”; therefore, the active node of a cluster will have a virtual IP address to provide client-side services. When the client side sends a service request destined to the virtual IP address, the active node will reply the request. When problems occur in the active node, heartbeat functions will automatically toggle the service to standby node and give it the virtual IP address. Simultaneously, ARP packets are sent to local networks. The refreshed MAC address will perform smooth and error-free takeover of client-side related services.

Vyatta provides software-based virtual router, virtual firewall and VPN products for Internet Protocol networks (IPv4 and IPv6). A free download of Vyatta has been available since March 2006. The system is a specialized Debian-based Linux distribution with networking applications such as Quagga, OpenVPN, and many others. A standardized management console, similar to Juniper JUNOS or Cisco IOS, in addition to a web-based GUI and traditional Linux system commands, provides configuration of the system and applications. In recent versions of Vyatta, web-based management interface is supplied only in the subscription edition. However, all functionality is available through the connections of serial console or SSH/telnet protocols. The software runs on standard x86-64 servers.

2.2 Related work

In information technology, high availability refers to a system or component that is continuously operational for a desirably long length of time [12, 13]. In other words, the high availability is described through service level agreements and achieved through an architecture focusing on constant availability even in the face of failures at any level of the system [14]. In a data center, normally a redundant design at all levels ensures that no single component failure which impacts the overall system availability. While maintaining capability through load balancing, backup and replication, a mirrored facility, or a modular architecture replacing the monolithic one, it requires a significant investment and full monitoring [15].

On the other hand, in order to deploy high availabilities of a data center to protect its mission-critical services, the intelligent techniques need to be introduced or monitoring in the three main resource pools, i.e., computer, storage, and network, involving multi-layered approaches. The layered approach is the basic foundation of a data center design that seeks to improve scalability, performance, flexibility, resiliency and maintenance [16]. For example, in a Cisco approach, the layers of a data center design are the core, aggregation, and access layers. The core layer provides connectivity to multiple aggregation modules and provides a resilient Layer 3 (i.e., network layer) routing. The aggregation layer modules provide functions, such as firewall, load balancing to optimize and secure applications, intrusion detection, network analyses and more. Access layer provides both Layer 2 (data link layer) and Layer 3 topologies, fulfilling the various applications of servers. In addition, the primary data center design models are either multi-tier model or server cluster model for an enterprise or an academic and scientific community respectively. The models, at the aggregation and access layers, include “passive” redundancy built into data centers to overcome power or internet provider failures, as well as “active” redundancy that leverages sophisticated monitoring to detect issues and initiate failover procedures [14, 16]. Furthermore, additional log shipping can be adapted to achieve data storage protection [17].

As mentioned above, a lot of researches focus on the high availability and the relative performance in a single cluster. However, IBM Corp. published a patent [18] to solve the problem that the systems comprising cluster of identical servers are unable to provide a high availability processing environment with respect to web services for entire applications which are processed by such systems. In fact, such a system at best provides high availability for no more than localized portions of the application rather for the entire application. Usually, a web service system includes a web server cluster coupled to an application server cluster and the application server cluster coupled to a database server cluster. The system can function continuously while a local fault occurs in each cluster, but may not while a connection fails between two coupled clusters. IBM's solution is to add into the system a control server, which is linked to each server in each cluster with a connection path. Therefore, the control server can monitor the state of the link (or say, connection path) between the two servers respectively from two coupled clusters. If a link between two coupled clusters is down, the control server will inform the clusters to adjust their server status, either active or inactive for achieving high availability.

This paper also focuses on the high availability affected by the relationship between clusters. It further emphasizes the performance of routing path between clusters.

3 Economic Design of Full-level HA

The proposed design for a data center infrastructure has the features: full-level HA, system monitoring, recovering, and controlling, and event handling, which are presented in the following subsections.

3.1 HA model of data center infrastructure

The proposed HA design is based on a general model of data center infrastructure, as shown in Figure3, suitable for a small/medium-scale company (SMSC).

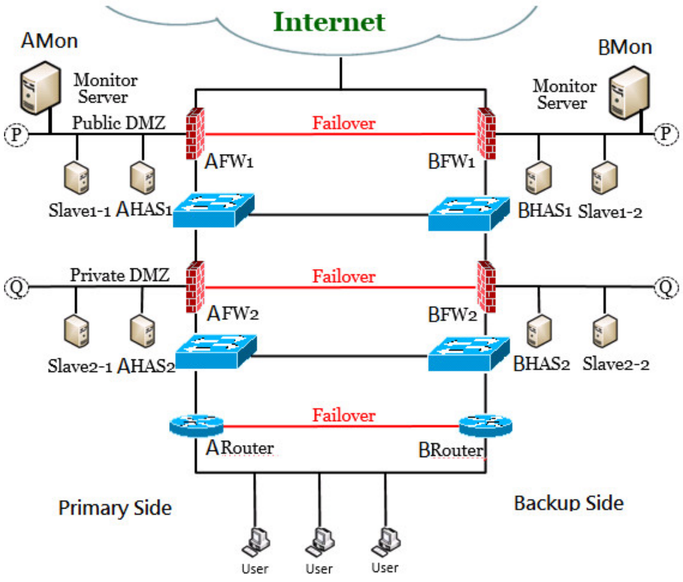


Figure3. HA model of data center infrastructure for SMSC

The design uses two tiers of firewalls to create the Public and Private DMZ network segments for accommodate the Internet servers and intranet servers, respectively. The Internet servers are utilized to settle the public-domain service applications such official Web, Email, FTP, etc. while the intranet servers to settle the private-domain ones such as HR, Finical, Asset systems, etc. In addition, one router tier is designed to create one or multiple internal network segments for the internal departments of company. The HA designs in network, server and application levels based on this model are shown as below.

3.1.1 Network-level HA

The network system in this model consists of the network devices of two-tier firewalls and one-tier routers. For network-level HA, each tier is grouped as a network cluster and divided into two sides: Primary and Backup. However, the public and private DMZ segments in Primary and Backup sides are connected in point Ⓟ and Ⓠ, respectively. By default, the Primary-side firewalls and router are active in their network-level clusters while the Backup-side ones are standby. The internal user can issue a connection to the Internet through the active router (ARouter), then the active firewall 2 (AFW2), and finally the active firewall (AFW1). On the other side, the firewall BFW1 and BFW2 and the router BRouter are standby for backup. There is a failover network cable, between the active and standby devices of each tier, used as a passage for mutual monitoring. Once the active device fails and is unable to provide service, the virtual IP (VIP) address representing the cluster will be taken over by the backup device to realize uninterruptible network services. This is shown as Figure4. The network devices are implemented with general host computers running Vyatta system image. A Vyatta machine can act as a firewall or router and support HA functions.

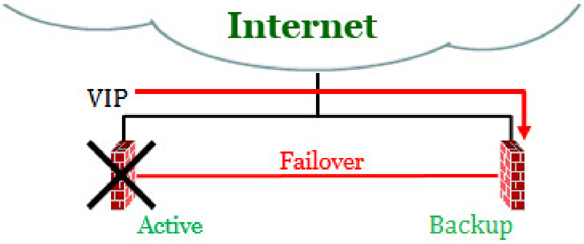


Figure 4. The association of VIP address changes on failover

There are two routing problems among the network-level clusters. First, many small/medium-scale companies use PPPOE connections in ADSL links to connect to the Internet. In the proposed model as shown in Figure3 , when firewalls AFW1 and BFW1, implemented as Vyatta machines, are in powered-on state and if their PPPOE connections are not controlled, both firewalls will fight for the PPPOE connections through ADSL and stall the network. In order to avoid this situation, a PPPOE connection coordination program, as shown in Figure5, must be set in AFW1 and BFW1 individually. The PPPOE program in the active firewall sets PPPOE commands to issue the PPPOE connection while the one in the standby firewall removes all PPPOE commands to close the PPPOE connection. The program thus avoids simultaneous PPPOE connections through ADSL. The PPPOE setting is never saved in configuration file. Thus, whenever the firewalls reboot, no PPPOE connection will be issued.

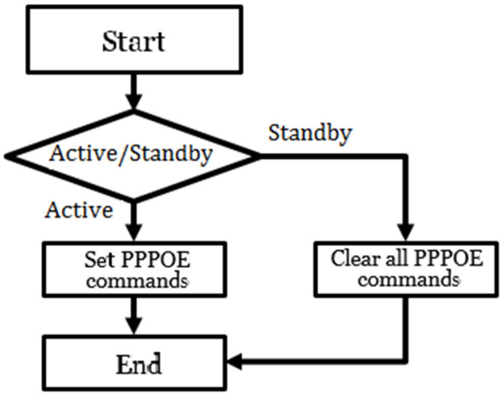


Figure 5. The PPPOE program in AFW1 and BFW1.

The second problem is that a winding path occurs while the routing among clusters is not controlled. Consider the scenario that AFW2 in a non-preempt HA reboots due to version upgrade. The BFW2 becomes active while AFW2 becomes standby. A winding path for an internal user’s connection to the Internet occurs from ARouter, then through BFW2, and finally to AFW1. As shown in Figure3 , there are four switches in this path. If AFW2 finishes the reboot phase and works well, the better path is from ARouter, then through AFW2, and finally to AFW1 or from BRouter, then through BFW2, and finally to BFW1. There are only two switches in the individual paths. This routing can be adjusted by controlling the active/standby state of a network device. The monitor and control processes are designed in the monitor servers and describe in Section 3.2.

3.1.2 Server-level and application-level HA

In the server level, Linux Ubuntu is used to implement the server systems of the model. There are 8 Ubuntu servers created in the public DMZ and private DMZ network segments, named AHAS-1~2, BHAS-1~2, Slave1-1~2, and Slave2-1~2 (Figure3). In the public DMZ segment, AHAS-1 and BHAS-1 are grouped in the public server cluster and placed into the Primary and Backup sides, respectively. For server-level HA, AHAS-1 is active by default while BHAS-1 is standby for backup. This is implemented with Linux LVS and Keepalived utilities, which provide the Heartbeat mechanism to determine if the other side server is alive. Once the active server fails and is unable to provide service, the virtual IP (VIP) address representing the cluster will be taken over by the backup server. In the private DMZ segment, AHAS-2 and BHAS-1 is grouped in the private server cluster, whose HA operates as same as that of the public server cluster.

For application-level HA, an application service is distributed into the servers of DMZ segment, which dynamically share the service requests and thus achieve the HA of application service with the way of service Load Balancing (LB). In the public DMZ segment, by default, AHAS-1 acts as an active Master server while BHAS-1 acts as a standby Master, but they do not offer any application service. The active Master plays a role of load balancer to dispatch service requests to the Slave servers actually hosting the application service. Slave1-1 and Slave1-2 are the Slave servers in the public DMZ segment. The design of the LB adopts the Direct Routing (Figure1) architecture for reducing the traffic load of the Master servers. It is implemented with Linux LVS and Keepalived utilities, which are installed in the Master servers to provide the health checks of application services in the Slave servers and thus know how to dispatch service requests. For the private DMZ segment, the LB operation is implemented with the same way as that in the public DMZ.

3.2 Monitoring, Recovering, and Controlling HA

In the proposed design, the primary and backup devices in a cluster determine if the other one is alive by the Heartbeat mechanism. However, for network-level HA, the network devices additionally determine if each connected network segment works well by checking if the individual monitoring point, created in each segment, is alive. However, it is a large effort to arrange a monitoring point in each segment and maintain it always up. Instead, the design detects the states of network segments of a device by checking the states of its interfaces connected to network segments. A monitor cluster is created in the Public DMZ and consists of two Monitor servers, in the Primary and Backup sides, named AMon and BMon, respectively. By default, AMon is active to monitor the objects in network, server, and application levels and then recover them, if failures (also including HA failures) are detected, through reboot or restart commands in the Vyatta or Ubuntu systems. Simultaneously, BMon is standby for backup of AMon to achieve the management-level HA. The active Monitor server uses the Ping utility to monitor the objects in network and server levels while using the NMap utility to monitor the services in application level.

After the monitoring and recovering phases, the active Monitor server starts the HA control process. First, it checks the route to the Internet to prevent AFW1 and BFW1 from issuing the PPPOE connections simultaneously. The control flow is as shown in Figure6. Initially, it notifies the two firewalls to clear their PPPOE setting and then checks which firewall is active in HA. It notifies the active firewall to run the PPPOE program, as shown in Figure5, to issue the PPPOE connection. Once the connection establishes, the IP address is allocated to the PPPOE interface of active firewall. Then, the control process Pings the IP address to determine if the route to the Internet is successful. If the Ping detection fails, the process tries to reboot the firewall failing on PPPOE.

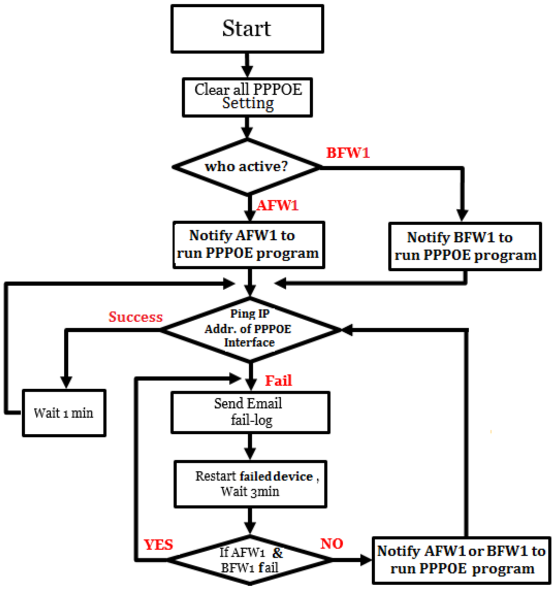

Start
Clear all PPPOE Setting
who active?
BFW1
AFW1
Notify AFW1 to run PPPOE program
Notify BFW1 to run PPPOE program
Success
Ping IP Addr. of PPPOE Interface
Wait 1 min
Fail
Send Email fail-log
Restart failed device , Wait 3min
YES
If AFW1 & BFW1 fail
NO
Notify AFW1 or BFW1 to run PPPOE program

Figure 6. The control of PPPOE by Monitor server.

Secondly, the active Monitor server controls the routing among the network-level clusters to avoid a winding path through the clusters. The way is to control all active network devices in one side, either the

Primary or Backup side (Figure3), and all standby ones in the other side. The control process is called One-side-active process, as shown in Figure7. It tries to control the firewall successfully issuing PPPOE connection, either AFW1 or BFW1, together with other active network devices in the same side. Consider the following scenario. Assume that AFW1 has successfully established a PPPOE connection to the Internet. Obviously, AFW1 is active in its cluster. If AFW2 is standby in its cluster, the One-side-active process will reboot BFW2 to make AFW2 active and BFW2 thus standby.

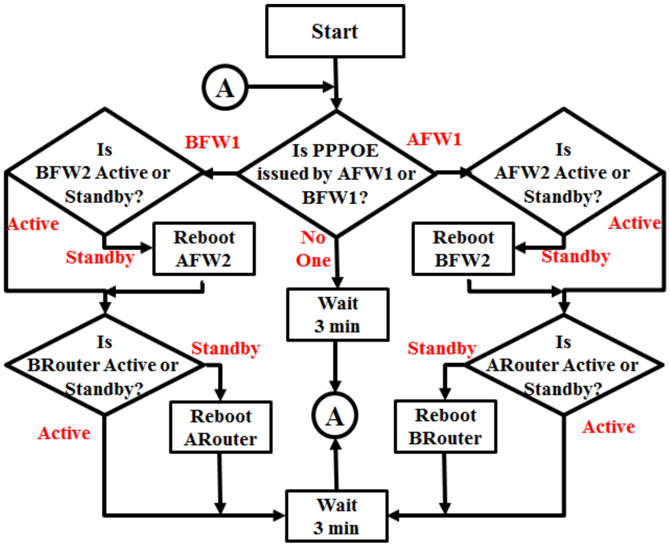


Figure 7. The control of One-side-active by Monitor server.

3.3 Event Handling

In the HA model of data center infrastructure, in order to coordinate the operations among the HA clusters of all levels, the concept of Event Center (EC) is introduced into and implemented in the Monitor server. As an EC, the Monitor server monitors and controls the whole data center and records and analyzes events to alert the administrator while a critical event occurs. Its system architecture is as shown in Figure8. The Monitor & Controller module undertakes the monitoring, recovering, and controlling of the objects in all levels for achieving full HA. The operation states of each object can be simply defined as Normal and Abnormal. An event from an object is defined as the state change of the object.

The monitor server is implemented with Linux Ubuntu system and utilizes the mechanisms of Syslog, Database, and Rotated Files to systematically record events. Whenever the monitor and control module detects a state change of an object, it creates an event to describe the state change. The event is sent to the local Syslog server with the Linux Logger utility or encapsulated as an email and then sent to a mail server. When the Syslog server receives the event, it can transfer the event to a remote Syslog server by the local Syslog client, save into the Rotated Files by the Logrotate utility, or store into a Database such as PostgreSQL. The local Syslog server and client module is implemented with the Rsyslog utility. The design uses state change to trigger the creation of an event and thus reduces the quantity of events. This is a low-cost design, compared to an event design which continuously generate the same-reason events from periodical detection of an Abnormal object until the object becomes Normal.

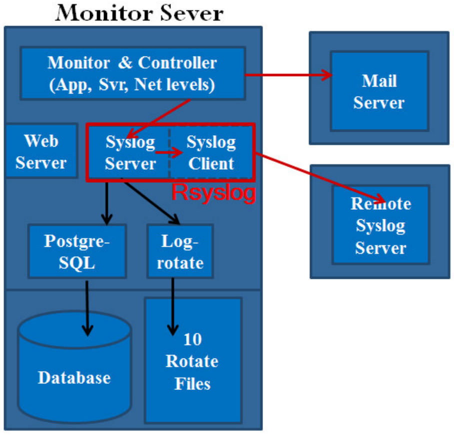
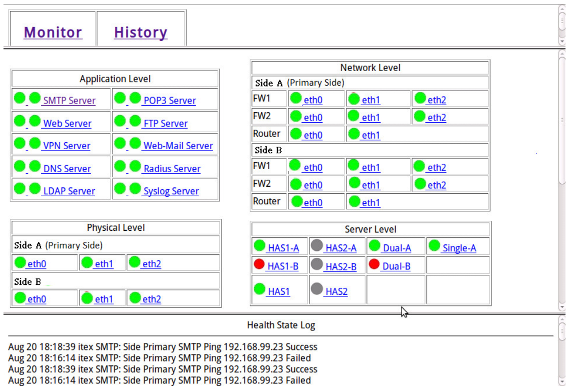


Figure 8. System architecture of Monitor Server.



Application Level	
🟢 🟢 SMTP Server	🟢 🟢 POP3 Server
🟢 🟢 Web Server	🟢 🟢 FTP Server
🟢 🟢 VPN Server	🟢 🟢 Web-Mail Server
🟢 🟢 DNS Server	🟢 🟢 Radius Server
🟢 🟢 LDAP Server	🟢 🟢 Syslog Server

Network Level			
Side A (Primary Side)			
FW1	🟢 eth0	🟢 eth1	🟢 eth2
FW2	🟢 eth0	🟢 eth1	🟢 eth2
Router	🟢 eth0	🟢 eth1	
Side B			
FW1	🟢 eth0	🟢 eth1	🟢 eth2
FW2	🟢 eth0	🟢 eth1	🟢 eth2
Router	🟢 eth0	🟢 eth1	

Physical Level		
Side A (Primary Side)		
🟢 eth0	🟢 eth1	🟢 eth2
Side B		
🟢 eth0	🟢 eth1	🟢 eth2

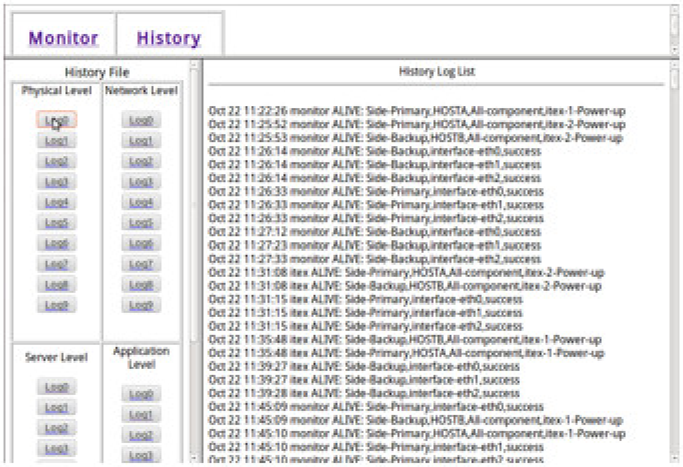
Server Level			
🟢 HAS1-A	⚫ HAS2-A	🟢 Dual-A	🟢 Single-A
🔴 HAS1-B	⚫ HAS2-B	🔴 Dual-B	
🟢 HAS1	⚫ HAS2		

Health State Log

```
Aug 20 18:18:39 itex SMTP: Side Primary SMTP Ping 192.168.99.23 Success
Aug 20 18:16:14 itex SMTP: Side Primary SMTP Ping 192.168.99.23 Failed
Aug 20 18:18:39 itex SMTP: Side Primary SMTP Ping 192.168.99.23 Success
Aug 20 18:16:14 itex SMTP: Side Primary SMTP Ping 192.168.99.23 Failed
```

Figure 9. Web page shows the states of monitored objects.

The states of objects in all level are exhibited with the signs drawn in a web page, as shown in Figure9. The Normal and Abnormal states are presented as Green and Red signs, respectively. The logs record the stage changes of objects in different levels are exhibited in individual web pages, as shown in Figure10. These pages are formed by a web server as shown in Figure8, which is implemented with the Tomcat utility.



History Log List

```
Oct 22 11:22:26 monitor ALIVE: Side-Primary,HOSTA,All-component,itex-1-Power-up
Oct 22 11:25:52 monitor ALIVE: Side-Primary,HOSTA,All-component,itex-2-Power-up
Oct 22 11:25:53 monitor ALIVE: Side-Backup,HOSTB,All-component,itex-2-Power-up
Oct 22 11:26:14 monitor ALIVE: Side-Backup,interface-eth0,success
Oct 22 11:26:14 monitor ALIVE: Side-Backup,interface-eth1,success
Oct 22 11:26:14 monitor ALIVE: Side-Backup,interface-eth2,success
Oct 22 11:26:33 monitor ALIVE: Side-Primary,interface-eth0,success
Oct 22 11:26:33 monitor ALIVE: Side-Primary,interface-eth1,success
Oct 22 11:26:33 monitor ALIVE: Side-Primary,interface-eth2,success
Oct 22 11:27:12 monitor ALIVE: Side-Backup,interface-eth0,success
Oct 22 11:27:23 monitor ALIVE: Side-Backup,interface-eth1,success
Oct 22 11:27:33 monitor ALIVE: Side-Backup,interface-eth2,success
Oct 22 11:31:08 itex ALIVE: Side-Primary,HOSTA,All-component,itex-2-Power-up
Oct 22 11:31:08 itex ALIVE: Side-Backup,HOSTB,All-component,itex-2-Power-up
Oct 22 11:31:15 itex ALIVE: Side-Primary,interface-eth0,success
Oct 22 11:31:15 itex ALIVE: Side-Primary,interface-eth1,success
Oct 22 11:31:15 itex ALIVE: Side-Primary,interface-eth2,success
Oct 22 11:35:48 itex ALIVE: Side-Backup,HOSTB,All-component,itex-1-Power-up
Oct 22 11:35:48 itex ALIVE: Side-Primary,HOSTA,All-component,itex-1-Power-up
Oct 22 11:39:27 itex ALIVE: Side-Backup,interface-eth0,success
Oct 22 11:39:27 itex ALIVE: Side-Backup,interface-eth1,success
Oct 22 11:39:28 itex ALIVE: Side-Backup,interface-eth2,success
Oct 22 11:45:09 monitor ALIVE: Side-Primary,interface-eth0,success
Oct 22 11:45:09 monitor ALIVE: Side-Backup,HOSTB,All-component,itex-1-Power-up
Oct 22 11:45:10 monitor ALIVE: Side-Primary,HOSTA,All-component,itex-1-Power-up
Oct 22 11:45:10 monitor ALIVE: Side-Primary,interface-eth1,success
Oct 22 11:45:10 monitor ALIVE: Side-Primary,interface-eth2,success
```

Figure10. Web page shows the historical events in each level.

4 Experiment results

In order to verify the proposed design works well, the data center infrastructure is implemented mainly with the Linux Ubuntu and Vyatta system software. The cluster HA functions in the two operating systems are used to provide individual-level cluster HA. The Monitor & Controller module designed in the Monitor server is to achieve a full HA by monitoring, recovering, and controlling the objects of all levels. It solves the routing problems of "PPPOE connection racing" and "winding path" among the network-level clusters. The experiment results exhibited in the following subsections explain how to verify the approaches.

4.1 Monitoring and recovering test

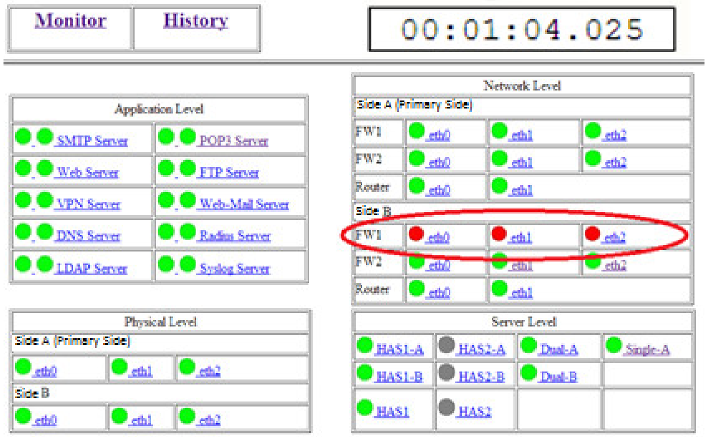


Figure 11. BFW1 is detected to be failed.

The testing scenario is as follows: The active side is the Primary side; that is, all the network devices in the Primary side are active while all the ones in the Backup side are standby. Close the standby firewall BFW1. The test is to check if the failure of BFW1 will be detected and then BFW1 will be recovered.

The time is reset to zero when BFW1 is closed. As shown in Figure11, BFW1 is detected to be failed at 1 min 4.025 sec. Then, the monitor server starts to recover BFW1 through rebooting. BFW1 is detected to be successful at 2 min 45.606 sec (Figure12).

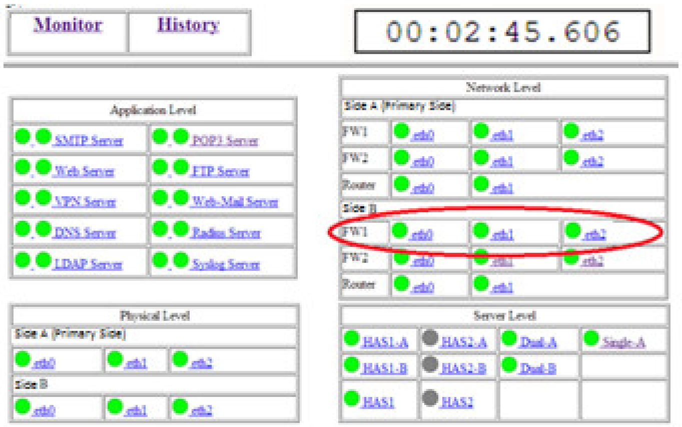
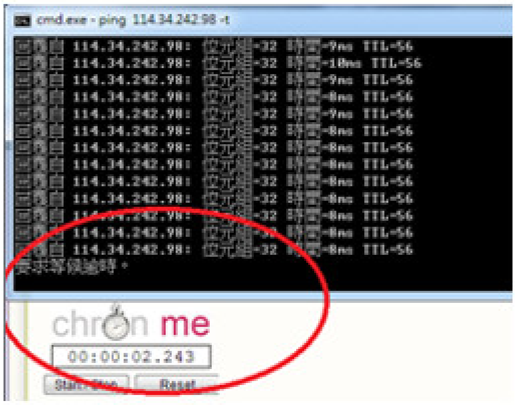


Figure12. BFW1 is recovered and detected to be successful

4.2 PPPOE connection test

The testing scenario is as follows: The active side is the Primary side. Close AFW1, the active firewall in the first tier. The test is to check if the Monitor server can detect the failure of AFW1, then recover it, and next notify BFW1 to establish the PPPOE connection since BFW1 has become active.



```
回覆自 114.34.242.98: 位元組=32 時間=9ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=10ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=9ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=9ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
要求等候逾時。
```

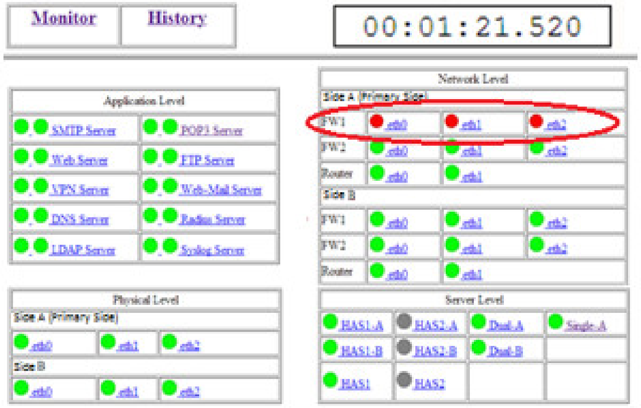
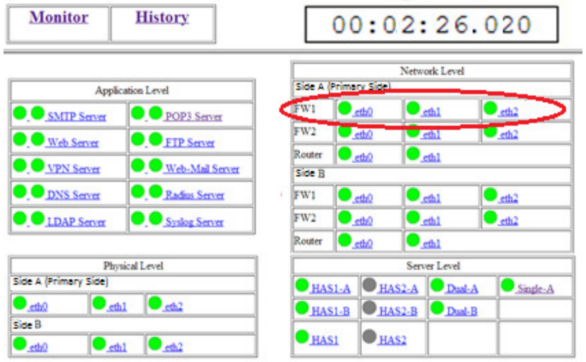
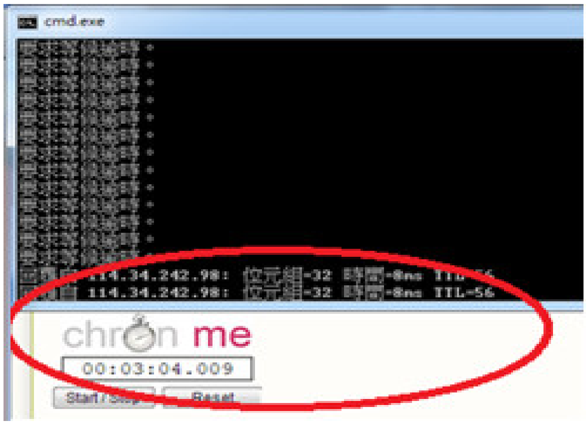


Figure13. The route to the Internet is failed

Figure14. AFW1 is detected to be failed.

The time is reset to zero when AFW1 is closed. As shown in Figure13, the PPPOE connection is failed at 2.243 sec. However, the Monitor server detects the failure of AFW1 after 1 min 21.520 sec (Figure14). It then recovers AFW1 through rebooting. AFW1 is detected to be successful after 2 min 26.020 sec (Figure15). Finally, the PPPOE connection is recovered at 3 min 4.009 sec (Figure16).





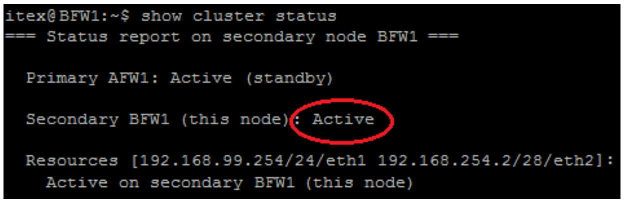
```
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
要求等候逾時。
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
回覆自 114.34.242.98: 位元組=32 時間=8ms TTL=56
```

Figure15. AFW1 is recovered and detected to be successful

Figure16. The route to the Internet is recovered

4.3 One-side-active process test

This test scenario continues that of PPPOE connection test. When the PPPOE connection is recovered as shown in Figure16, One-side-active process will enforce BFW2 and BRouter to be active since BFW1 is active (Figure17), and issues the PPPOE connection.



```
itex@BFW1:~$ show cluster status
=== Status report on secondary node BFW1 ===

  Primary AFW1: Active (standby)

  Secondary BFW1 (this node): Active

  Resources [192.168.99.254/24/eth1 192.168.254.2/28/eth2]:
    Active on secondary BFW1 (this node)
```

Figure 17. BFW1 is active while AFW1 is standby

Following the timing of PPPOE connection test, in which the time is reset to zero when AFW1 is closed, the scenario shows AFW2 and ARouter are rebooted and their failures are detected at 4 min 47.361 sec (Figure18). Then, they are detected to be successful at 5 min 58.760 sec (Figure19).

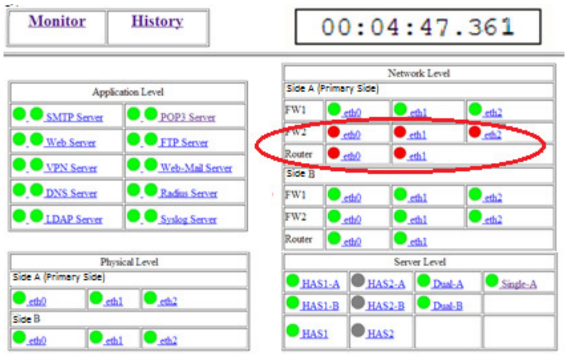


Figure18.One-side-active process reboots AFW2 and ARouter.

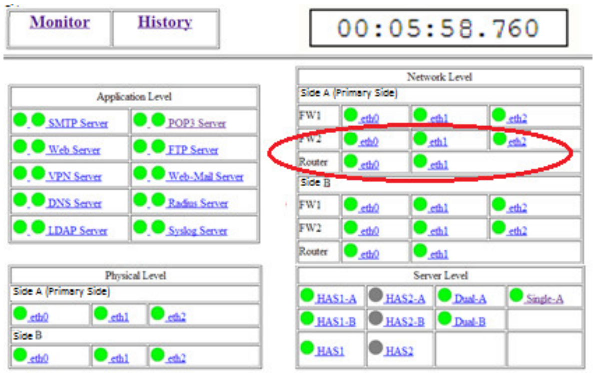
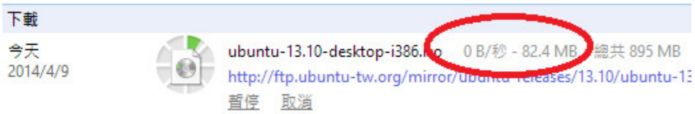


Figure19. AFW2 and ARouter reboot successfully.

4.4 Data continuity test

Vyatta devices can support stateful failover function to prevent a session from being broken while it encounters a HA failover. The proposed design enabling this function is to protect sessions from loss in the regulation enforced by One-side-active process.

The testing scenario is as follows: A user in the LAN segment between ARouter and BRouter gets data from the Ubuntu FTP site. While data is being transferred, ARouter is closed since it is active in its cluster. Due to stateful failover, BRouter undertakes the rest data transfer of the FTP connection. As shown in Figure 20 and 21, data continuity works well.

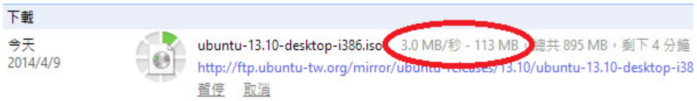


下載

今天
2014/4/9

ubuntu-13.10-desktop-i386.iso 0 B/秒 - 82.4 MB，總共 895 MB
http://ftp.ubuntu-tw.org/mirror/ubuntu-releases/13.10/ubuntu-13
暫停 取消

Figure 20. Data transfer is paused during failover.



下載

今天
2014/4/9

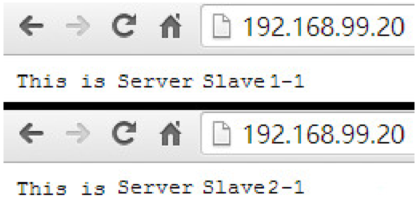
ubuntu-13.10-desktop-i386.iso 3.0 MB/秒 - 113 MB，總共 895 MB，剩下 4 分鐘
http://ftp.ubuntu-tw.org/mirror/ubuntu-releases/13.10/ubuntu-13.10-desktop-i38
暫停 取消

Figure 21. Data transfer is continued after failover.

URL:http://dx.doi.org/10.14738/tnc.55.3777

4.5 Load Balancing test

Application-level HA is achieved with the Load Balancing (LB) mechanism applied on the servers hosting application service. In the testing scenario, an IE browser is used to issue connections to the VIP address, 192.168.99.20, of a web server cluster having two slave servers Slave1-1~2 in the public DMZ segment. Each connection is dispatched to one of the slave servers in a LB scheduling. The testing web pages are as shown in Figure22. If server Server1-2 is off, the connection dispatching will skip it and always select Slave1-1, as shown in Figure23.



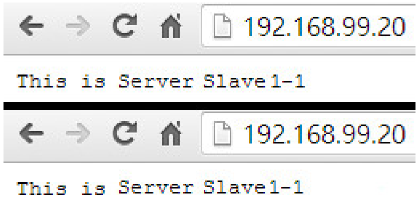
192.168.99.20

This is Server Slave1-1

192.168.99.20

This is Server Slave2-1

Figure 22. Connections are dispatched to two slave servers



192.168.99.20

This is Server Slave1-1

192.168.99.20

This is Server Slave1-1

Figure 23. Connection dispatching skips an "off" server

5 Conclusions

The motivation is to provide a design of low-cost and high-availability data center infrastructure for small/medium-scale businesses. The proposed design has the features as follows: 1) providing full-level HA on network, server, application, and management levels, 2) controlling the routing among network-level clusters to solve the "PPPOE connection racing" and "winding path" problems, 3) monitoring and recovering the objects in each level with an economic and effective way, and 4) handling events resulting from object state changes with an event center. The design is low-cost since it is implemented with free software, such as Linux Virtual Server (LVS), Vyatta Router and Firewall, and NMap Scanner. In addition, the event center generates events only on state changes. This saves the space and time to store and de-duplicate events, caused by the same-reason source. The experiment results show the administrator can see the current and historical state changes of the objects in the data center infrastructure through the Event Center in the Monitor server. Its Monitor & Controller module can recover the failed objects and solve the routing problems of PPPOE connection racing and winding path among HA clusters automatically. The results also exhibits that the data continuity realized with stateful failover is still workable with the proposed design, especially at its One-side-active process.

The future work is to analyze and enhance the performance of the Monitor & Control module. This includes: 1) what is the proper lower and upper bounds of the monitoring interval to the objects of all levels, 2) how to shorten the time for silencing the rebooted or restarted objects for recovery, 3) finding better method of recovery instead of reboot.

REFERENCES

[1] Cegielski, C. G., Bourrie, D. M. & Hazen, B. T., Evaluating Adoption of Emerging IT for Corporate IT Strategy: Developing a Model Using a Qualitative Method. Information Systems Management, 2013. 30 (3): p. 235–249.

[2] Duan, Q., Yan, Y. & Vasilakos, A. V., A Survey on Service-Oriented Network Virtualization toward Convergence of Networking and Cloud Computing. IEEE Transactions on Network and Service Management, 2012. 9 (4): p. 373–392.

[3] Araujo, J. A., Lazaro, J., Astarloa, A., Zuloaga, A. & Garcia, A., High Availability Automation Networks: PRP and HSR ring implementations. IEEE International Symposium on Industrial Electronics (ISIE) 2012: p. 1197–1202.

[4] Sharkh, M. A., Jammal, M., Shami, A., & Ouda, A., Resource Allocation in a Network-Based Cloud Computing Environment: Design Challenges. IEEE Communications Magazine, 2013. 51 (11): p. 46–52.

[5] Bitar, N., Gringeri, S. & Xia, T. J., Technologies and protocols for data center and cloud networking. IEEE Communications Magazine, 2013. 51 (9): p. 24–31.

[6] Liao, C. F., Chang, H. C. & Fu, L. C., Message-Efficient Service Management Schemes for MOM-Based UPnP Networks. IEEE Transactions on Services Computing, 2013. 6 (2): p. 214–226.

[7] Lin, C. C., Chin, H. H. & Deng, D. J., Dynamic Multiservice Load Balancing in Cloud-Based Multimedia System. IEEE Systems Journal, 2014. 8 (1): p. 225–234.

[8] LVS. http://www.linuxvirtualserver.org as of October 11, 2015.

[9] Vyatta. Vyatta Community Documentation. http://www.vyatta.org/ as of August 15, 2014.

[10] NMap. http://nmap.org/ as of October 11, 2015.

[11] Membrey, P., Plugge, E. & Hows, D., Practical Load Balancing: Ride the Performance Tiger, 2012. 1st Ed. Apress.

[12] Rusu, L. & Smeu, A., Managing the Reliable Design of an Enterprise IT Network Infrastructure. Information Systems Management, 2010. 27 (3): p. 238–246.

[13] Radhakrishnan, R., Mark, K., & Powell, B., IT Service Management for High Availability. IBM Systems Journal, 2008. 47 (4): p. 549–561.

[14] Mahmood, A. & Rashid, I., Comparison of load balancing algorithms for clustered web servers. International Conference on Information Technology and Multimedia (ICIM) 2011: p. 1–6.

[15] IT Today. Is High-Performance Computing For You? IBM Systems Magazine, Power Systems, January. 2013.

[16] Cisco. Cisco Data Center Infrastructure 2.5 Design Guide. Cisco Validated Design. 2013.

[17] HP. Designing Disaster Tolerant High Availability Clusters. Manufacturing Part Number: B7660-90006. HP Document. 2014.

[18] Ahmed, I., Auvenshine, J. J., & Blackburn, J., System for Autonomic Monitoring for Web High Availability. IBM Corp. US 7996529 B2; Patent. August 9, 2011.

[19] Sim, K. M., Agent-based Approaches for Intelligent InterCloud Resource Allocation, IEEE Transactions on Cloud Computing, 2016: p. 1-14.