PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Eléctrica
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

**Héctor Joshua Andrés Valdés Piña**

# Diseño de cluster basado en tecnología NUC de Intel

**Informe Proyecto de Título de Ingeniero Electrónico**


**Escuela de Ingeniería Eléctrica**
**Facultad de Ingeniería**

**Valparaíso, 26 de Septiembre de 2018**


# Diseño de cluster basado en tecnología NUC de Intel

Héctor Joshua Andrés Valdés Piña

Informe Final para optar al título de Ingeniero Electrónico,
aprobada por la comisión de la
Escuela de Ingeniería Eléctrica de la
Facultad de Ingeniería de la
Pontificia Universidad Católica de Valparaíso
conformada por

Sr. David Velásco López
Profesor Guía

Sr. Francisco Alonso Villalobos
Segundo Revisor

Sr. Sebastían Fingerhuth Massmann
Secretario Académico

Valparaíso, 26 de septiembre de 2018

# Agradecimientos

Gracias a Dios por la oportunidad de volver a estudiar. A mi madre Jessica quién me dio el amor y el apoyo de enfrentar este nuevo reto. A mis suegros Juan Carlos y Yuly, quienes constantemente estuvieron presentes en cada una de las etapas y de quienes siempre estaré agradecido. A mi novia Vanessa Gallardo y a mi hija Emma Valdés, quienes son la motivación más grande que puedo tener y cuyo amor iluminó mi camino. A mis amigos Freddy López y Pablo Álvarez, quienes me acompañaron en el día a día de la universidad y cuyas historias recordaré hasta viejo. A mis profesores de título, que me permitieron crear y aprender mucho de este maravilloso proyecto. Finalmente, a todos los que hacen la universidad, ya que es gracias a ellos, de cualquier manera, quienes me brindaron la simpatía y el conocimiento necesario para finalizar esta etapa.

*Valparaíso, 26 de septiembre de 2018*

Héctor Valdés Piña

# Resumen

Este trabajo dará a conocer la construcción de un clúster de alta disponibilidad utilizando la tecnología NUC de Intel. En el cuál se abordará el funcionamiento de las redes y sus protocolos desde la perspectiva de los modelos TCP/IP y OSI. Aquí, además, encontraremos los softwares utilizados en la construcción del clúster de alta disponibilidad, añadiendo también la construcción de un segundo clúster replicador de datos – DRBD - el cuál será adjuntado para mejorar las condiciones del clúster principal. El objetivo que tiene este proyecto es el de mantener una aplicación web funcionando 24/7 presentando una alta disponibilidad en dos partes del sistema. Primero, una alta disponibilidad en el balanceo de carga de los servidores proxys, trabajo realizado por dos computadoras NUC de Intel y los cuales serán responsables de recibir las peticiones entrantes y enviarlas a los servidores. Segundo, una alta disponibilidad en el balanceo de carga de los servidores web, mediante la replicación de datos del disco con un sistema DRBD, sistema tal que permitirá la copia de los datos almacenados, tales como página web, base de datos, etc. Y las posibles caídas en el servidor principal por medio de la participación de un segundo servidor. La topología final del sistema demostrará una estructura redundante la que podrá sostener aproximadamente 20.000 conexiones simultáneas considerando que este clúster será *semi* homogéneo (de diferente hardware pero de mismo sistema operativo).

Palabras clave: Redes de computadoras, programación, alta disponibilidad, protocolo, sistema operativo.

# Abstract

This work will reveal the construction of a high availability cluster using Intel NUC technology. In which the functioning of the networks and their protocols will be approached from the perspective of the TCP / IP and OSI models. Here, in addition, we will find the software used in the construction of the high availability cluster, also adding the construction of a second data replicator cluster - DRBD - which will be attached to improve the conditions of the main cluster. The objective of this project is to maintain a web application running 24/7, presenting high availability in two parts of the system. First, high availability in the load balancing of the proxy servers, work done by two Intel NUC computers and which will be responsible for receiving the incoming requests and sending them to the servers. Second, high availability in the load balancing of web servers, by replicating data from the disk with a DRBD system, such system that will allow the copying of stored data, such as web page, database, etc. And the possible falls in the main server through the participation of a second server. The final topology of the system will demonstrate a redundant structure which will be able to sustain approximately 20,000 simultaneous connections, considering that this cluster will be semi-homogeneous (of different hardware but of the same operating system).

Key words: Computer network, programming, high availability, protocol, operating system.

# Índice general

Introducción ..... 1
Objetivo general ..... 3
Objetivos especificos ..... 3

1 Conceptos generales ..... 4
1.1 Definición. ..... 4
1.1.1 Alto rendimiento ..... 4
1.1.2 Alta disponibilidad ..... 5
1.1.3 Balanceo de carga ..... 5
1.1.4 Escalabilidad ..... 6
1.2 Configuraciones y clasificaciones ..... 6
1.3 Estado de arte ..... 7
1.4 Modelos OSI y TCP/IP ..... 8
1.4.1 Capa Física ..... 10
1.4.2 Capa Enlace de datos ..... 11
1.4.3 Capa de Red ..... 12
1.4.4 Capa de Transporte ..... 12
1.4.5 Capa de sesión ..... 14
1.4.6 Capa de presentación ..... 14
1.4.7 Capa de aplicación ..... 15
1.4.8 Comparación OSI -TCP/IP ..... 15
1.5 Balance de carga ..... 16
1.5.1 Balance de carga en Servidores web ..... 16
1.5.2 DHCP ..... 18

2 Solución al problema ..... 21
2.1 Topología de red ..... 21
2.2 Trabajo previo ..... 24
2.2.1 Asignando una IP estática ..... 24
2.2.2 Instalando y configurando SSH ..... 25
2.3 Características de los nodos ..... 27
2.3.1 Nodos balanceadores de carga ..... 28

2.3.2 Nodos servidores web ..... 28

3 Desarrollo ..... 30
3.1 Softwares asociados ..... 30
3.1.1 Softwares en cluster de balanceo de carga ..... 30
3.1.2 Softwares en cluster de replicación de datos ..... 40
3.2 Firewall UFW ..... 45
3.2.1 Firewall en cluster de balance de carga ..... 45
3.2.2 Firewall en cluster de replicación de datos ..... 46

4 Pruebas y resultados obtenidos ..... 48
4.1 IPerf ..... 48
4.2 Apache Bench ..... 50
4.3 Failover ..... 54
4.3.1 Failover con Pacemaker en balanceadores de carga ..... 54
4.3.2 Failover con DRBD en cluster de alta disponibilidad ..... 56

Discusión y conclusiones ..... 60

Bibliografía ..... 63

# Introducción

La alta demanda de servicios por Internet ha aumentado rápidamente estos últimos años, en la cual Chile no se encuentra atrás implementando servicios de alta disponibilidad en servidores que se utilizan para mantener aplicaciones web, por ejemplo, o cualquier sistema de administración que necesite almacenamiento en una nube. Debido a su rápido crecimiento, la necesidad de crear más sistemas de servidores se ha vuelto un potencial trabajo para aquellos Ingenieros que se interesan en los medios de comunicación digital o telecomunicaciones. Para poder implementar un sistema de este tipo no es necesario contar con mucho dinero, por esta razón se mostrará en el presente informe la construcción un clúster de alta disponibilidad utilizando dos computadoras o nodos de bajo consumo como son las computadoras NUC de Intel. NUC, por sus siglas en Inglés Next Unit of Computing – Siguiente Unidad de Cómputo – refiriéndose a la siguiente generación de computadoras son, por diseño, el nuevo modelo económico y potente de Intel.

Dentro de las especificaciones que se encuentran en el modelo DN2820FYYKH utilizados en este proyecto, se encuentran: Puerto USB 3.0, salida HDMI y conexión de WiFi/Ehternet, todo esto acompañado con un procesador Intel Celeron N2820 de 2.41 GHz y 4 GB de memoria RAM, lo que los convierte en unas excelentes máquinas para realizar un clúster para balanceo de carga. Este balanceo se refiere al proceso de dividir las tareas para prevenir los cuellos de botella en el sistema. Gracias a la implementación de los NUC de Intel al sistema no solo logrará volverlo más económico, sino también mejorará la ergonomía del diseño del clúster al ser más simples que un computador convencional.

Por medio de la construcción de un clúster de balanceo de carga se logrará proteger la integridad de la información que los servidores puedan poseer, esto se logra configurando en una primera instancia a los nodos NUC como servidores Proxys los cuales se encargarán de crear una barrera de protección a la entrada ante peticiones entrantes al servicio web, peticiones las cuales serán distribuidas al servidor disponible dentro de otro arreglo de clúster llamado DRBD - Distribuited Replicated Block Device – .

El resultado de lo anterior, desencadenará un clúster de alta disponibilidad para prevenir posibles caídas del sistema, con clonación de datos y protección ante posibles ataques al servicio.

El objetivo de este proyecto tiene como principio el mantener una alta disponibilidad constante, es por eso que el desarrollo de este constará de tres partes fundamentales. Primero, la construcción del clúster de balanceo de carga por medio de los NUC de Intel, trabajo el cual constituye en encontrar la manera de comunicar las dos máquinas NUC utilizando el sistema operativo Ubuntu 16.03 de Linux y una pequeña red LAN. Para esto último es necesario conocer el funcionamiento y protocolos requeridos en la comunicación a través de la red. En este punto se encuentran protocolos tales como: TCP/IP para conocer el proceso de modulación/demodulación de la información a través de las distintas capas que el Internet posee. Además, ayudará a demostrar conceptos tales como *IP* y *Puerto de red* los cuales estarán en todo momento ligados a la construcción de este proyecto. El servicio asociado que dará utilidad al clúster será una plataforma web, servicio tal que deberá manejar una alta demanda de personas en línea. Este es el primer paso para iniciar la comunicación por medio de direcciones IP, a cada uno de los nodos participantes y en los cuales se identificarán como 192.168.1.3 - 192.168.1.4 a los NUC 1 y NUC 2. Otro protocolo importante en este punto es el protocolo HTTP, quien mostrará el camino que la información debe recorrer para poder mostrarse a través de Internet, comunicándose principalmente por el reconocido puerto 80. Para diferenciar el trabajo realizado en el sistema de clúster y el trabajo de página web, se tomarán como referencia los conceptos de *back-end* y *front-end* propiamente tal.

El segundo punto en el desarrollo de este proyecto, es la instalación de los softwares asociados a la construcción del clúster de alta disponibilidad. Como información inicial al proyecto, se establece la regla de utilizar tres softwares que realizarán el trabajo de alta disponibilidad, como lo son los softwares: Pacemaker; como administrador de recursos (Utilizado para organizar los recursos asociados al cluster), Heartbeat; como recurso de mensajería (Utilizado para conocer el estado de uno o varios nodos dentro de un clúster). Y finalmente: HA-Proxy; Recurso que convierte un nodo en servidor proxy (Utilizado para crear una barrera de entrada, y como tal, protección al sistema de servidores) y que además realiza el trabajo del balanceo de carga, es decir, reconocerá a qué servidor enviar la petición entrante.

El tercer punto es la construcción de un clúster de servidores de datos replicados DRBD. Para lograr una alta disponibilidad, es necesario también contar con un sistema de servidores que almacenen los datos que el servicio necesite, es por esto que se añaden a la topología inicial dos nuevos nodos los cuales tendrán la misión de resguardar la información y de emitirla en caso que una petición entrante lo requiera. A estos nodos, se les identificará como 192.168.1.5 y 192.168.1.6 para el nodo 3 y nodo 4, nombrándolos además como ‘Server1’ y ‘Server2’. Estos servidores utilizarán dos softwares que darán vida a este clúster, los cuales son: 1) LAMP (Linux, Apache, MySQL/MariaDB, Pearl/PHP/Python); Como servicio web del cual se utilizará Apache, MySQL y PHP. Y, 2) DRBD (Distribuited Replicated Block Device) propiamente tal, quién realizará el trabajo de clonar los datos recopilados en un segundo servidor.

Las proyecciones que se esperan de este proyecto son poder crear un servicio web el cual pueda responder a una alta demanda de peticiones, considerando dentro de estas un rango de entre 15.000 y 20.000 de conexiones simultáneas realizadas desde cualquier dispositivo, respuesta que estará asociada a los límites de hardware que cada nodo participante del clúster tendrá según las

mediciones por el software IPerf. También, se asocia a esta construcción la redundancia que el servicio presenta, ya que la alta disponibilidad demostrará no solo la confiabilidad del servicio, sino también, otorgará una solución a los problemas de protección de la integridad de los datos que los servidores alojen. La información en detalle de lo mencionado en esta introducción quedará expuesta en el siguiente informe.

## Objetivo general

- Lograr la construcción de un cluster de alta disponibilidad utilizando la tecnología NUC de Intel®.

## Objetivos especificos

- Conocer el funcionamiento de Internet mediante los modelos OSI y TCP/IP.
- Realizar la construcción de un clúster de balanceo de carga.
- Realizar la construcción de un clúster de replicación de datos DRBD.
- Configurar la seguridad del sistema.

# 1 Conceptos generales

Antes de comenzar con la construcción del cluster es necesario conocer el funcionamiento que las redes poseen y sus protocolos para acceder a Internet. En este capítulo, se mostrarán los conceptos más importantes y los cuales serán utilizados como fundamentos en la creación del proyecto y que estarán presentes en todo momento.

## 1.1 Definición.

El término *cluster* (del Inglés *Cluster*, racimo) hace referencia a un grupo de computadoras unidas por una red de alta velocidad y los cuales se comportan como un solo sistema. Los usos que tienen los *cluster* son varios, y van desde aplicaciones de súper cómputo, así como también usos como servidores web y hasta el uso en comercio electrónico. La idea en utilizar este tipo de sistema es la de poder combinar la disponibilidad de microprocesadores y las redes de alta velocidad, ya sea para procesar múltiples tareas o para mantener aplicaciones de alta demanda. Los *cluster* son, por lo tanto, herramientas utilizadas para el desarrollo de sistemas y de los cuales se espera obtener.

- Alto rendimiento
- Alta disponibilidad
- Balanceo de carga
- Escalabilidad

### 1.1.1 Alto rendimiento

Un Cluster de alto rendimiento, es un conjunto de computadoras diseñadas para dar alta prestación de un servicio dada su capacidad de cálculo. La necesidad de utilizar este tipo de sistema dependerá del tamaño del problema. Una característica importante que debe cumplir el problema es que este debe poder dividirse por partes, ya que el proceso de cálculo dependerá de qué problema puedan ir resolviendo los nodos. Ya que si el problema a resolver no cumple con esta característica, el problema no podrá ser *paralelizable*. El alto rendimiento, en un *Cluster*, está diseñado para garantizar el funcionamiento ininterrumpido de ciertas aplicaciones.

### 1.1.2 Alta disponibilidad

Un Cluster de alta disponibilidad, es un sistema que permite mantener servicios o aplicaciones disponibles en todo momento ante cualquier petición entrante, por ejemplo: peticiones en un servidor. Además, la integridad de los nodos participantes estarán con monitoreo constante para prevenir cualquier fallo que afecte la salud del sistema. Para lograr una alta disponibilidad es necesario que las máquinas asociadas tengan grandes capacidades de cálculo en donde la velocidad juega un papel fundamental, ya que depende de esta característica la capacidad de poder recuperar al sistema de un error. Un ejemplo de *clúster* es el del tipo *Beowulf*, como lo muestra la Figura 1-1.

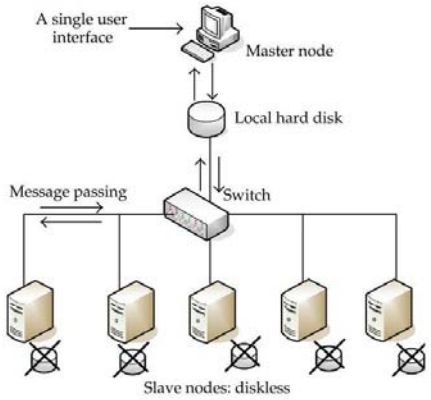


Figura 1-1: Cluster Beowulf (Fuente: http://www.wikiwand.com/)

La alta disponibilidad en este tipo de *clúster* se puede medir en dos partes:

- Alta disponibilidad en infraestructura: Si existe algún error de Hardware de uno de los nodos participantes el software de alta disponibilidad intentará arrancar los servicios en cualquier otra máquina dentro del *cluster*. A este proceso se le llama *Failover*. Cuando la máquina que falló se recupera, los servicios son nuevamente migrados a la máquina original, a este proceso se le llama *Failback*.

- Alta disponibilidad de aplicación: Al existir una alta disponibilidad de la infraestructura el sistema puede responder a una recuperación automática de los servicios, esto nos garantiza que la aplicación utilizada en un *cluster* de este tipo podrá estar disponible en todo momento, minimizando la percepción de fallos por los usuarios ante posibles errores en el sistema.

### 1.1.3 Balanceo de carga

El balanceo de carga, corresponde a una técnica usada para poder compartir el trabajo entre varios procesos, computadoras, discos u otros recursos. Debido a la gran demanda de los servicios de Internet y a la transferencia de información, los sistemas deben funcionar los 365 días del año sin errores. La gran problemática de realizar un balanceo de carga es tratar de gestionar las

solicitudes de un gran número de usuarios, como por ejemplo: una página web, servidores de correo electrónico, almacenamiento en la nube, etc. Su funcionamiento consiste en que todas las solicitudes que reciba el sistema se repartirán entre los diversos servidores, y con ello, aumentar la cantidad de usuarios que concurran a dicho servicio.

Para implementar un balanceo de carga es necesario conocer la cantidad de solicitudes que se esperan recibir en un servicio. Esto quiere decir, que las solicitudes podrán ser atendidas de manera limitada según la cantidad de peticiones entrantes, aumentando su capacidad a medida que mejoran las condiciones del Hardware. Así mismo, este último dependerá del software asociado al balanceo de carga, el cual también posee un límite. Algunos de los softwares que se utilizan en un balanceo de carga son: Apache, IIS y MySQL, entre otros. Mediante la regla de dividir el problema se puede balancear y configurar diversos servidores los cuales son capaces de funcionar en conjunto, mejorando el rendimiento individual.

### 1.1.4 Escalabilidad

Esta es una propiedad deseable de cada sistema de red o proceso, ya que indica la habilidad para reaccionar y adaptarse sin perder calidad. Así mismo, para también manejar el crecimiento continuo de trabajo de manera fluida, o bien, para hacerse más grande sin perder calidad en los servicios ofrecidos. La definición de escalabilidad proviene de la capacidad de un sistema informático de cambiar su tamaño o poder adaptarse a los constantes cambios.

## 1.2 Configuraciones y clasificaciones

Los componentes elementales de un cluster son el Hardware y Software, los cuales serán utilizados en el proceso de comunicación y resolución de problemas. Dada la variación de estos componentes es posible construir distintos tipos de configuraciones, entre los cuales podemos encontrar:

- Cluster homogéneo: Estos poseen el mismo hardware y software. Utilizados según el rendimiento esperado en un servicio.

- Cluster semihomogéneo: Estos poseen diferente rendimiento por su hardware, pero se componen de un mismo sistema operativo.

- Cluster heterogéneo: Este tipo de clúster posee distinto hardware y software como sistema operativo. Se utiliza generalmente para resolver la necesidad de recopilar archivos.

Las clasificaciones, por otro lado, se definen según el uso que se les dará y de los servicios que estos ofrecerán los cuales se pueden clasificar en:

- HPPC: *High performance Computing Cluster*, es un cluster de alto rendimiento. Este tipo requiere una gran capacidad de hardware tanto en almacenamiento como en memoria

RAM. Son muy requeridos en proyectos los cuales la tarea pueda comprometer los recursos del sistema por largos periodos.

- HACC: *High Avaibility Computing Cluster*, es un cluster de alta disponibilidad. El objetivo de este tipo es proveer disponibilidad y confiabilidad. La confiabilidad se demuestra a través de un Software que permita detectar fallos y recuperarse del mismo.

- HTCC: *High Throughput Computing Cluster*, es un cluster de alta eficiencia. Su objetivo es poder ejecutar la mayor cantidad de tareas en el menor tiempo posible.

## 1.3 Estado de arte

Existen diversas empresas las cuales están ligadas a las comunicaciones y que trabajan con sistemas de *cluster* para fines como: Almacenamiento y base de datos, Big Data, estadísticas, música, videos y fotos, etc. Como por ejemplo: Facebook, IBM, Google y Microsoft, entre otros. La Pontificia Universidad de Valparaíso utiliza un sistema de redes de *cluster* para su base de datos, logrando almacenar y resguardar la información de los alumnos y trabajadores en un servicio interno.

Un ejemplo de lo anterior es el súper computador de Intel, el cuál provee a los usuarios de Latinoamérica la capacidad de probar y ejecutar códigos con un alto nivel de cómputo utilizando la última generación de procesadores XEON de Intel. Esta es capaz de resolver investigaciones, simulaciones y *Benchmarking*, utilizada en las ciencias administrativas de las empresas. En la Figura 1-2 podemos observar un cluster conformado de computadoras Intel XEON E3-1240 con un procesador de 8 núcleos y 16 GB de RAM para cada una.

Figura 1-2: Cluster Intel XEON (Fuente: http://www.isp21.com/)

Dentro del estado de arte es posible encontrar no sólo las capacidades que el hardware en las máquinas posean, sino también los softwares asociado a estas, quiénes tienen una gran responsabilidad al momento de construir un sistema confiable. Los Softwares permitirán conocer el límite programable de configuraciones las cuales permitirán obtener el margen real de conexiones que el sistema podrá abordar, y dentro de los cuales podemos encontrar los más utilizados, como se observa en la Figura 1-3.

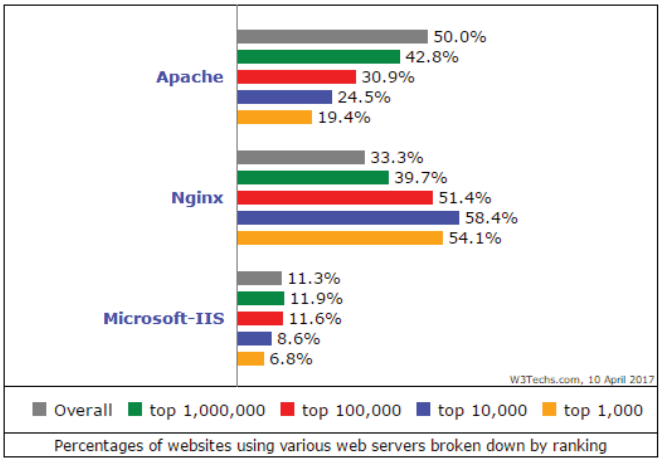


Figura 1-3: Softwares de uso frecuente (Fuente: https://w3techs.com/)

Los Softwares más utilizados en servidores son Apache y Nginx, los cuales lideran el mercado dada su confiabilidad en las tareas a realizar. En el proyecto de alta disponibilidad será utilizado Apache, por la cantidad de información y contenido disponible por Internet para su uso.

Dentro de la adopción a nivel mundial que Apache posee, es posible reconocer los países que más lo utilizan dentro de los cuales podemos encontrar:

- España: 65.31% del total de dominios.
- Chile: 62.71% del total de dominios.
- Argentina: 54.58% del total de dominios.
- México: 53.52% del total de dominios.
- Colombia: 27.07% del total de dominios.

Fuente: https://es.wikipedia.org/wiki/Servidor_HTTP_Apache

## 1.4 Modelos OSI y TCP/IP

Ante el problema de realizar un clúster de alta disponibilidad con los dispositivos NUC de Intel, se propusieron los siguientes desafíos:

- Comunicar a los nodos que participarán del clúster
- Conocer el funcionamiento de las redes asociadas
- Conocer el funcionamiento de puertos y protocolos

Para poder enfrentar estos desafíos es necesario conocer el concepto utilizado en las comunicaciones de red, concepto tal expresado en el modelo OSI como una división de niveles los cuales realizan distintos trabajos para encapsular la información y transmitirla por Internet. Creada en 1984, el modelo OSI hace referencia de Interconexión de Sistemas Abiertos – *Open System Interconecction* – Este es un marco descriptivo creado por ISO, cuya descripción estándar muestra la definición de arquitecturas de interconexión en sistemas de comunicaciones. El sistema OSI cuenta con 7 capas definidas como se muestra en la Figura 1-4.


Figura 1-4: Descripción del modelo OSI (Fuente: http://www.seaccna.com)

Cada una de estas capas posee una característica en el encapsulamiento y desencapsulamiento de la información. La Tabla 1-1 muestra una pequeña descripción de cada una de estas capas.

Tabla 1-1: Capas del modelo OSI

| Modelo OSI | | | |
|---|---|---|---|
| | Capas (Layer) | Protocolo de datos por unidad (PDU) | Función |
| Host Layers | 7.- Aplicación | Datos | APIs de alto nivel, incluyendo recursos para compartir archivos de acceso remoto. |
| | 6.- Presentación | Datos | Traslación de datos entre una red de servicio y una aplicación; incluyendo codificación de caracteres, compresión de datos y enciptado/desencriptado. |

| | | | |
|---|---|---|---|
| | 5.- Sesión | Datos | Manejo de las sesiones de comunicación, por ejemplo: El continuo cambio de información en la forma de múltiples *back-and-forth* (de aquí para allá) transmisiones entre dos nodos. |
| | 4.- Transporte | Segmento (TCP) /Datagram (UDP) | Transmisiones confiables de segmentos de datos entre puntos de una red, incluyendo la segmentación, el *ACK (Reconocimiento de un mensaje)* y multiplexación. |
| Media Layers | 3.- Red | Paquetes | Estructura y manejo de múltiples redes de nodos, incluyendo *direcciones, routing y control de tráfico.* |
| | 2.- Unión de datos | Frames | Transmisión confiable de *data frames* entre dos nodos conectados por una *capa física.* |
| | 1.- Física | Bit | Transmisión y recepción de un sin número de bits entre medio físico |

Procedimiento:

1. Los datos a ser transmitidos están compuestos en la capa más alta del aparato trasmisor hacía el protocolo de unidad de datos (Protocol data unit) o PDU.

2. EL PDU es pasado a la capa siguiente, la cual se conoce como la unidad de servicio de datos (Service Data Unit) o SDU.

3. En este punto, el SDU es concatenado o unido con un Header, un Footer, o ambos, produciendo una capa N-1 PDU. Que pasará a la siguiente capa N-2.

4. El proceso continua hasta que se llega hasta el nivel más bajo (Ver en orden decreciente), desde que los datos son transmitidos hacía el aparato receptor.

5. En el aparato receptor los datos son pasados desde el nivel más bajo al nivel más alto en capas como una serie de SDUs mientras van eliminando sucesivamente el encabezado – Header - o el pie de página – Footer- que cada capa le había otorgado, hasta llegar a la capa superior, donde se consumen los últimos datos.

### 1.4.1 Capa Física

La capa física determinará las especificaciones eléctricas y físicas de los datos de conexión. Este define la relación entre un aparato y medio de transmisión (por ejemplo, un cable eléctrico, un cable de fibra óptica, o una unión de radio frecuencias). Esto incluye el diseño de

los pines, el voltaje, la línea de impedancia, las especificaciones del cable, el tiempo de la señal y las características similares que compartan los aparatos, por ejemplo en frecuencia para aparatos inalámbricos (5 GHz o 2.4 GHz etc.). Este punto es importante ya que es responsable de la transmisión y recepción de un sinfín de datos estructurados. El control del *Bitrate* se realiza en esta capa. Los modos en los cuales se define la transmisión son: Modo simplex, half duplex, y full dúplex. Este punto, además, es importante ya que define la topología de red como los tipos bus, malla (mesh) o el anillo (ring), siendo estos los más comunes.

La capa física - Figura 1-5 - es la capa que funciona con los equipos de red de bajo nivel, tales como los Hubs, los cables (red), y los repetidores. La capa física es la que no tiene protocolos ya que estos suelen verse en capas superiores.

Figura 1-5: Módem AMR con hardware emulado HSP de transmisión telefónica

### 1.4.2 Capa Enlace de datos

Esta capa provee la transferencia nodo a nodo, es la unión entre dos nodos conectados de manera directa. Esto detecta y posibilita corregir los errores que puedan ocurrir en la capa física. La cual define el protocolo para establecer y determinar la conexión entre dos aparatos conectados de manera física. También este define el protocolo de control de ritmo (Flow control) entre ellos.

Según la IEEE en su insignia IEEE 802, divide la capa de unión de datos en dos subcapas como se observa en la Figura 1-6.

- Capa MAC (Medium Access Control): Responsable de controlar tantos aparatos se vayan incorporando a la red ganando acceso al medio y permiso a los datos a transmitir.

- Capa LLC (Logical Link Control): Responsable de identificar y encapsular los protocolos de la capa de red, además de controlar los errores revisando y sincronizando los frames transmitidos.

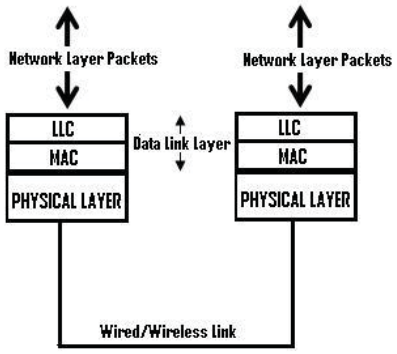


Figura 1-6: Subcapas dentro de la capa enlace de datos

Ejemplos: Ethernet 802.3, 802.11 Wi-Fi, y 802.15.4 ZigBee - Figura 1-7 - operara en la capa de enlace de datos.


Figura 1-7: ZigBee 802.15.4 (Fuente: http://www.microsolution.com.pk)

### 1.4.3 Capa de Red

Esta capa provee el procedimiento funcional de transferir un dato de largo variable, llamados datagramas, desde un nodo a otro, conectados en redes diferentes. La red es un medio el cual muchos nodos pueden ser conectados, y en el cual cada uno de los nodos tiene una dirección especifica que lo identifica. De esta manera, los nodos de una red pueden comunicarse con otros nodos en otras redes, compartiendo el contenido de un mensaje según la dirección IP del destinatario, recorriendo una o muchas redes para llegar a su destino.

### 1.4.4 Capa de Transporte

La capa de transporte aporta la parte funcional y el significado de transferir una secuencia de largo variable de datos desde una fuente a su destinatario a través de una o más redes, mientras se mantiene la calidad en las funciones del servicio.

En la Figura 1-8 se puede observar la composición de un datagrama en base a la comunicación IPV4.

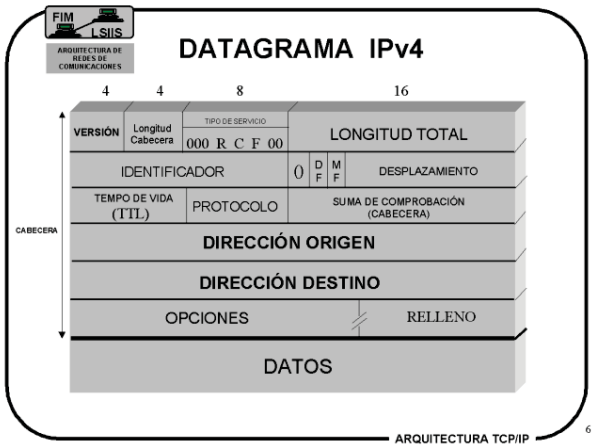


Figura 1-8: Composición de un datagrama IPV4 (Fuente: http://www.masdeipv6.blogspot.com/)

La capa de transporte controla la confiabilidad de un *link* a través el control de ritmo - *Flow control* – además de la segmentación/desegmentación, y el control de errores. Algunos protocolos son de conexiones orientadas según un estado y conexiones. Esto quiere decir, que la capa de transporte puede mantener el rastro de los segmentos y retrasmitir aquellos que han fallado. La capa de transporte, además, provee un *reconocimiento* de todas las trasmisiones exitosas y enviará el siguiente dato si ningún error ha ocurrido. La capa de transporte es la que crea paquetes del mensaje que se recibió en la capa de aplicación. Los paquetes corresponden a un proceso de dividir el largo del mensaje en pequeños mensajes.

El sistema OSI define cinco clases de modos de conexión para el transporte de protocolos dentro de un rango desde la *clase cero* (TP0: Quien provee menos características) hacía la clase cuatro (TP4: Diseñada para redes menos confiables, similares a internet). La clase cero no tiene recuperación de errores, y fue diseñada para usarse en una red de capas sin errores.

Las características de cada una de las cinco clases de modos de transporte pueden observarse en la Tabla 1-2.

Tabla 1-2: Clases de transporte existentes

| Función | TP0 | TP1 | TP2 | TP3 | TP4 |
|---|---|---|---|---|---|
| Orientado a una red de conexiones | Sí | Sí | Sí | Sí | Sí |
| Red sin conexión | No | No | No | No | Sí |
| Concatenar/descontanetar (Unir/separar) | No | Sí | Sí | Sí | Sí |
| Segmentar y re ensamblar datos | Sí | Sí | Sí | Sí | Sí |
| Recuperación de errores en caso de existir muchas PDUs sin *reconocerse*. | No | Sí | Sí | Sí | Sí |

| | | | | | |
|---|---|---|---|---|---|
| ¿Reinicia la conexión? | No | Sí | No | Sí | No |
| Permite Multiplexar/Desmultiplexar dentro de un simple circuito virtual. | No | No | Sí | Sí | Sí |
| Controla explícitamente el control de ritmo (*Flow control*) | No | No | Sí | Sí | Sí |
| ¿Retransmite si se cumple el *timeout*? | No | No | No | No | Sí |
| ¿Es un transporte de servicio confiable? | No | Sí | No | Sí | Sí |

### 1.4.5 Capa de sesión

La capa de sesión controla los diálogos (Conexiones) ente computadoras. Este establece, maneja y termina las conexiones entre una aplicación local y remota. Provee el tipo de conexiones full-duplex, half-duplex, u operación simplex, y establece un punto de control checkpoint, de interrupción, término y reinicio. El modelo OSI hace que esta capa sea la responsable del cierre de las sesiones, que es una propiedad del protocolo de transmisión de control (TCP) y la recuperación de la sesión, que generalmente no se utiliza en la Internet Protocol Suite.

La capa de sesión es implementada comúnmente de forma explícita en los entornos de aplicaciones que utilizan las llamadas a procedimiento remotos.

### 1.4.6 Capa de presentación

Esta capa de presentación establece el contexto entre las aplicaciones y las capas, en cuales la capa de aplicación puede usar una sintaxis y una semántica diferente si la presentación del servicio provee un mapeo – Mapping - entre ambas. Si el Mapping está permitido, las unidades de servicio que presenta los datos son encapsulados en una sesión del protocolo.

Esta capa provee independencia de los datos representados por traducciones entre aplicaciones y formatos de red. La capa de presentación transforma los datos en la forma que la aplicación la acepte. Es esta capa quien da el formato a los datos que se enviarán a través de la red, también llamada como la 'capa de sintaxis' – Sintax Layer -. La capa de presentación puede incluir funciones comprimidas. Además, es quien negocia la transferencia de sintaxis.

Ejemplos: Una estructura original usada en la presentación usa las Reglas básicas de codificación - Basic Encoding Rules -, cuya capacidad es la de convertir un EBCDIC-codificado texto en un archivo ASCII, o la serialización de objetos y otras estructuras de datos desde un XML.

### 1.4.7 Capa de aplicación

Esta capa de aplicación es, según el modelo OSI, la capa más cercana al usuario, lo que significa que ambos, el modelo OSI y el usuario, interactúan directamente a través de la aplicación de software. Esta capa además interactúa con los softwares que implementan un componente en la comunicación. Existen, además, algunas aplicaciones que están fuera del modelo OSI. La mayoría de las aplicaciones de capa que este modelo acepta son los que incluyen identificadores, los que determinan la disponibilidad de recursos y la sincronización en la comunicación. Cuando se identifican a los identificadores o 'Socios de Comunicación' - Communication Partners - la capa de aplicación determina la identidad y la disponibilidad de la comunicación para que una aplicación con datos pueda transmitir. La parte más importante en la distinción de la capa de aplicación es la diferencia entre la entidad de la aplicación y la misma aplicación.

Por ejemplo: Una página web de reservas puede usar dos entidades para su funcionamiento: Una, usando HTTP para comunicarse con los usuarios, y otra por un protocolo de base de datos (Ej: PHP y MySQL) para grabar las reservaciones. Ninguno de estos protocolos tiene que ver con las reservaciones, ya que esa lógica está en la misma aplicación, es decir, la capa de aplicación no tiene la capacidad de determinar la disponibilidad de recursos que se transmiten por la red.

### 1.4.8 Comparación OSI -TCP/IP

El diseño de protocolos en el modelo TCP/IP del internet no considera encapsulaciones jerárquicas. TCP/IP reconoce cuatro amplias capas de funcionalidad que se derivan del alcance operativo de sus protocolos contenidos, así como se muestra en la Figura 1-9:

- Para el Internet la capa de aplicación incluye en el modelo OSI la capa de aplicación, presentación y sesión.

- Es el fin de la capa de aplicación para el modelo TCP/IP y para el modelo OSI el que da comienzo a la capa de transporte para ambos casos.

- La capa de internet - o también Interworking Layer – es un sub arreglo de la capa de red del modelo OSI.

- La capa de enlace de datos incluye en el modelo OSI hasta la capa física lo que para el modelo TCP/IP es la capa de acceso a la red.

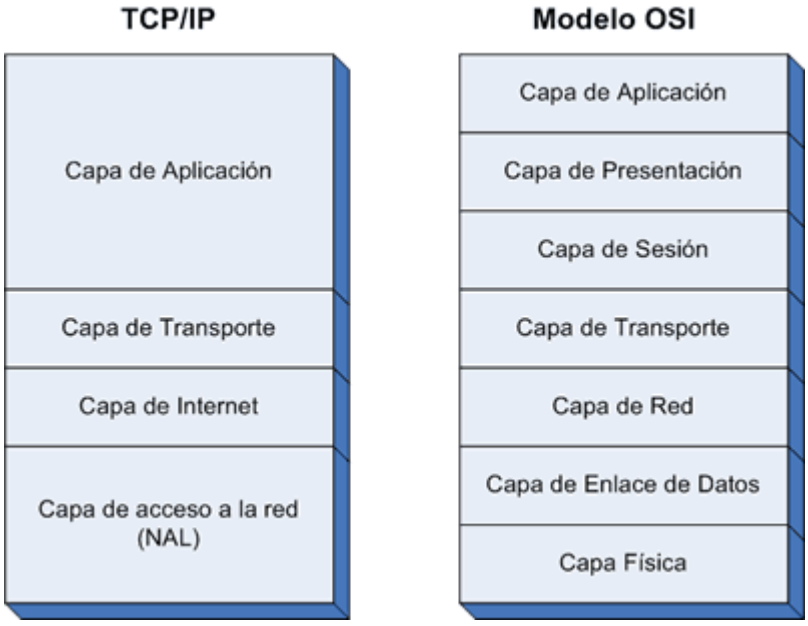


Figura 1-9: Diferencias entre modelo TCP/IP y OSI (Fuente: http://www.javiergarzas.com)

El 1 de enero de 1983, ARPA obligó a los miembros de su red a adoptar TCP/IP de esta manera pasa a ser el modelo más utilizado para la compresión del funcionamiento de la red. Además, a esta fecha se le conoce como el día que nació el Internet.

## 1.5 Balance de carga

Tal como lo planteado en el comienzo de este documento, el balance de carga es un punto importante en la resolución de problemas y cuellos de botellas para un sistema de alta disponibilidad. Son por lo tanto, sus dos comunes versiones 'Layer 4' y 'Layer 7' las cuales contribuyen al mejor manejo de datos, pero no así tienen el mismo desempeño.

Este concepto utilizado en la informática se refiere a la técnica que se utiliza para compartir el trabajo entre varios procesos, computadoras, discos u otros recursos. Esto proviene al similar sistema de multiprocesamiento. El balance de carga se mantiene gracias a un algoritmo que divide de manera más equitativa el trabajo para así eliminar ciertas limitantes como los atochamientos en un servicio de alta disponibilidad. A continuación se explicarán los dos tipos de balanceo de carga.

### 1.5.1 Balance de carga en Servidores web

Este tipo de balance de carga se utiliza para resolver los principales problemas que la mayoría de los sitios web tienen, y es el cómo gestionar un gran número de usuarios. Esto es importante para sistema de alto rendimiento o de alta disponibilidad, ya que permite la escalabilidad que surge con el continuo crecimiento del número de usuarios activos en el sistema.

El balance de carga, en un cluster, está compuesto por múltiples computadoras o nodos que actúan como *frontend* del cluster, y los cuales tienen la misión de repartir las peticiones entrantes al servicio re direccionando la petición a otras computadoras del arreglo que alojen, por ejemplo, un servicio web. A estos últimos se les reconoce como el *Backend* del cluster.

Las características más importantes de este tipo de balanceo en un cluster, son:

- Escalabilidad: al poder ampliar su capacidad al añadir más computadoras al cluster.

- Robustez: Ante la posible caída de uno de los computadores del cluster el servicio puede mermarse, pero mientras existan nodos funcionando el servicio seguirá también funcionando.

En ambos casos, (Balance de carga en servidores web y clúster de balance de carga) el balanceo puede realizarse utilizando los *algoritmos de configuración*, los cuales algunas industrias tienen como estándar:

- Round Robin: Es un método de balanceo de carga en servidores que provee una tolerancia simple a los fallos. Múltiples servidores son configurados para entregar exactamente el mismo servicio, todos estos configurados para utilizar el mismo dominio de internet, pero cada uno tendrá una dirección IP única. El servidor DNS tendrá una lista de todas las direcciones IP únicas que serán asociadas con el nombre del dominio de Internet. Cuando una petición entrante asociada al nombre del dominio de Internet sea recibida, las direcciones de los nodos se devuelven de forma secuencial rotativa.

- Weighted Round Robin: Esta utiliza como base el método simple Round Robin. En la versión por peso – o Weighted – cada servidor en el arreglo recibe una numeración según su peso o desempeño. Los servidores con altos ratings serán los que recibirán la mayor cantidad de solicitudes.

- Least Connections: Ni Round Robin ni Weighted Round Robin toman en consideración la carga actual que un servidor posee cuando se distribuyen las solicitudes. A diferencia de los puntos anteriores, el método de conexión mínima – Least Connections – si considera la carga actual que el servidor tiene, es decir, la última petición o solicitud irá al servidor que esté atendiendo el menor número de sesiones activas en el momento actual.

- Weighted Least Connections: Se construye con la misma estructura del método 'Least Connections'. Así como el método Weighted Round Robin, este método le entrega una numeración a cada servidor, el cual utiliza este valor cuando necesita enviar las

solicitudes a los servidores. Si dos servidores tienen el mismo número de conexiones activas, el servidor con el mayor rating será quien reciba la nueva solicitud.

### 1.5.2 DHCP

DHCP – Dinamic Host Configuration Protocol -. Es el protocolo de configuración de *host*, también conocido como anfitrión dinámico, cuya función es permitir que un equipo conectado a una red pueda obtener su configuración de red de forma automática, es decir, sin una intervención especial. Sólo es necesario especificarle al equipo mediante el DHCP, que entrará una dirección IP de manera independiente. Su objetivo principal es simplificar la administración de la red.

Este protocolo sirve principalmente para distribuir direcciones IP en una red, pero desde sus inicios se diseñó como un complemento del protocolo BOOTP (Protocolo Bootstrap) que se utiliza, por ejemplo, en la instalación de un equipo en una red. Básicamente, el protocolo DHCP se dedica a proporcionar automáticamente un *host* de protocolo de internet (IP) con su dirección IP, además de añadir otra información de configuración relacionada, por ejemplo: la puerta de enlace predeterminada y la máscara de subred.

Dentro del protocolo DHCP se encuentra un protocolo de errores, el cual se compone de dos modelos:

1.- Hot Standby: Dos servidores conectados, uno trabajando activamente y el otro en modo *standby*. Llamado así, ya que esta configuración permite estar atento ante un cambio en el servicio de balance de carga activo para inmediatamente accionar el que está en *standby*. Ventajas de este modelo, permite ganar redundancia al sistema para los servicios DHCP, como se observa en la Figura 1-10.

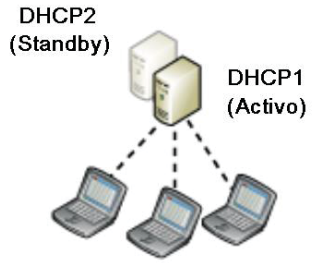


Figura 1-10: Failover Hot-Standby en DHCP

2.- Load balance mode: Este modo asigna concesiones de cliente DHCP en dos servidores. Este modo es el modo por defecto en los servicios, dos servidores DHCP sirviendo de manera simultánea direcciones IP y opciones a los clientes en una sub red (*subnet*).En este modelo el balanceo de carga puede realizarse en 50-50 o en cualquier rango desde 0 a 100% como se muestra en la Figura 1-11.

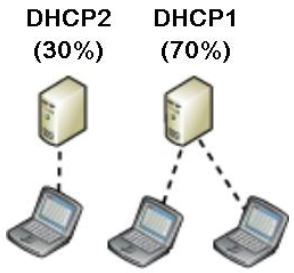


Figura 1-11: Modo Load balancer DHCP

Los sistemas de alta disponibilidad, a diferencia de los modelos propuestos para los servidores DHCP, se pueden catalogar en dos tipos de sistemas*: Warm Standby* y *Hot Standby*. Para ambos casos el servicio de alta disponibilidad se comportará de manera diferente. En la Tabla 1-3 se observan las descripciones según términos de los tipos de proyectos utilizados con DHCP.

Tabla 1-3: Clases de transporte existentes

| Antiguos términos<br><br>(Descontinuados) | Nuevo término | Tipo de proyecto | Opciones | Descripción |
|---|---|---|---|---|
| Cold failover | Warm standby | HA | Opción básica | Proyecto de simple instancia con failover activado. |
| Warm failover<br>Hot failover<br><br>Automatic failover<br><br>Failover | | Otro (No HA) | | |
| Active-Active<br><br>High availability | Hot standby | HA | Opción especializada | Dos instancias de proyecto, desarrollando una pareja de primario y secundario. |

El proyecto 'Diseño de cluster basado en tecnología NUC de Intel' será configurado con Hot Standby para dar un mejor balance de carga al servicio de alta disponibilidad. De esta manera, las solicitudes entrantes serán recibidas por dos nodos los cuales estarán en estado Activo/Hot Standby. Esta configuración permitirá solo a un nodo recibir las peticiones, mientras el otro estará atento ante cualquier cambio en la salud del primero.

# 2 Solución al problema

Para poder resolver la problemática se la construcción de un cluster, es necesario conocer la topología que se utilizará, la cual será necesaria en todo el proceso de construcción y que además servirá de guía en caso que se le necesite. En este capítulo, también, se encontrarán los softwares necesarios para cada etapa de construcción. Además, la información de este capítulo estará dividida en dos partes para poder apreciar de mejor manera las configuraciones en los scripts y las conexiones por puerto de red.

## 2.1 Topología de red

La topología de red se define como el mapa físico o lógico de una red para intercambiar datos. Es decir, es la forma en que está diseñada la red, sea en plano físico o lógico, Fuente: https://es.wikipedia.org. En esta etapa de construcción es necesario un guía que facilite, de manera visual, los contenidos faltantes y propios de cada parte del sistema. En la Figura 2-1 se puede observar la topología asociada a este proyecto, el cual da cuenta de los Softwares utilizados y las direcciones IP asignadas a cada nodo participante del clúster.

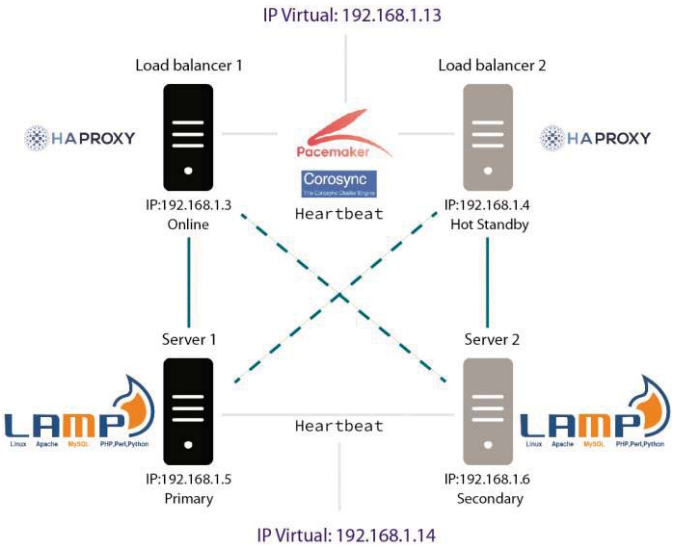


Figura 2-1: Topología del clúster de alta disponibilidad

La Figura 2-1 muestra, además, dos diferentes tipos de clúster el cual conforman el clúster de alta disponibilidad. Por un lado, los nodos Load balancer 1 y Load balancer 2 lideran la organización del sistema, siendo estos los responsables de recibir las solicitudes y los que están constituidos por las computadoras NUC de Intel. Al conjunto de nodos balanceadores, que llevan por dirección IP 192.168.1.3 y 192.168.1.4, se les reconocerá como **cluster de balanceo de carga**, los que estarán trabajando en conjunto con estados del tipo Activo/Hot Standby. Los softwares asociados al cluster de balanceo de carga, son: Pacemaker, Corosync, Heartbeat y HA-Proxy. En orden de mayor a menor relevancia. En la Tabla 2-1 se muestra la información asociada a estos softwares.

Tabla 2-1: Características de los softwares en Cluster de balanceo de carga

| Softwares | Característica |
|---|---|
| Pacemaker | Administrador de recursos del cluster |
| Corosync | Apilador de mensajes de recursos, encargado de organizar y distribuir los mensajes de los recursos usados por Pacemaker. |
| Heartbeat | Capa de mensajería, encargado de distribuir los mensajes de los recursos, al igual que Corosync. Utilizado para conocer el estado de Pacemaker en los nodos NUC. |
| HA-Proxy | Servidor Proxy. Convierte a los nodos NUC en servidores Proxy, de esta manera se crea una barrera de entrada para las solicitudes entrantes resguardando la información alojada en los servidores web. |

Por otro lado, observamos en la Figura 2-1 la participación de dos nodos posteriores al cluster de balance de carga. Estos nodos, en su conjunto, conformarán un segundo cluster el cual tendrá como función la disponibilidad de los datos y una copia constante de estos en caso de caída en uno de los servidores. Las direcciones IP asignadas a cada nodo serán: 192.168.1.5 y 192.168.1.6, para el primario y secundario respectivo. Este cluster es el corazón del servicio ya que dependerá de éste el tráfico asociado a la aplicación web. Los softwares utilizados en esta etapa serán:

Heartbeat, Lamp y DRBD. Siendo este último el responsable de la copia constante de los datos. La Tabla 2-2 muestra las características importantes de cada uno de los softwares en el **Cluster replicador de datos**.

Tabla 2-2: Características de los softwares en Cluster replicador de datos

| Softwares | Característica |
|---|---|
| Heartbeat | Capa de mensajería, encargado de distribuir los mensajes de los recursos. Utilizado para conocer el estado del software replicador de datos DRBD en los nodos servidores web. |
| LAMP | Linux, Apache, MySQL/MariaDB y PHP. Es utilizado para convertir a los nodos en servidores web, incluyendo dentro de los softwares la base de datos y el lenguaje de programación utilizado para el *frontend* del servicio. |
| DRBD | Distributed Replicated Block Device. Software utilizado en la replicación de datos. Su principal misión es la de mantener una copia exacta de los archivos, base de datos y elementos web en cada uno de los nodos participantes. |

Otra parte importante en la Figura 2-1 es la indicación de las direcciones IP virtual, las que serán utilizadas para dos fines independientes. En un principio, la dirección IP virtual 192.168.1.13, correspondiente al Cluster de balanceo de carga, será la dirección IP utilizada por el dominio para referirse al servicio web alojado en el sistema. Además, esta dirección es utilizada por el recurso Heartbeat, recurso preocupado de monitorear el estado de los dos nodos NUC. Al igual que lo anterior, la dirección IP virtual 192.168.1.14 es utilizada por Heartbeat para el monitoreo de los nodos servidores con configuración DRBD.

## 2.2 Trabajo previo

Antes de comenzar, es necesario realizar un trabajo previo para configurar un cluster de alta disponibilidad. Primero, se configurarán los nodos participantes con una IP estática, de esta manera se reconocerá la dirección a la cual se le asocia a cada nodo y no se requerirá la dirección que provee el protocolo DHCP. Segundo, se configurará una comunicación por medio de *Secure Shell*, o SSH, el cual permitirá transmitir información de manera segura encriptando cada envío con una seguridad del tipo SHA256.

### 2.2.1 Asignando una IP estática

La asignación de una IP estática a cada nodo es imprescindible en esta construcción, ya que permitirá la comunicación y configuración de todos los softwares y recursos que necesiten determinar a los participantes de cada uno. Asignando una IP estática es, además, el primer paso de seguridad para proteger la integridad del servicio, ya que esta es la dirección de acceso de cada nodo. Las configuraciones para realizar lo anterior son las de Network, hostname y hosts en la carpeta de sistema de Linux - /etc/ -.

```
1 # nano /etc/network/interfaces
```

Se copia dentro del script lo destacado en Listado 2-1 para cada uno de los nodos, colocando atención al **nombre del dispositivo de red** y la **dirección IP estática** a asignar que, en este ejemplo, se muestra para la configuración del balanceador de carga 'loadb1' cuya dirección IP es 192.168.1.3 y cuyo nombre de dispositivo de red es 'enp3s0'.

Listado 2-1: Indicando la IP estática

```
1  # This file describes the network interfaces available on your system
2  # and how to activate them.For more information, see interfaces (5).
3  Source /etc/network/interfaces.d/*
4  # The loopback network interface
5  Auto lo
6  Iface lo inet  loopback
7  # The primary network interface
8  Auto enp3s0
9  Iface enp3s0 inet static
10 Address 192.168.1.3/22
11 Netmask 255.255.255.0
12 Gateway 192.168.1.1
13 Dns-nameservers 8.8.4.4 8.8.8.8
14
15
16
```

Continuamos con la configuración de 'hostname' indicando en línea de comando la ubicación de este.

```
1 # nano /etc/hostname
```

Se copia el nombre de cada nodo en esta configuración. Esto es válido para los cuatro nodos participantes, es decir, en la configuración de 'hostname' para cada nodo debe contener el nombre de este.

```
1 GNU nano 2.5.3           File: /etc/hostname
2 Loadb1
```

Es necesario en este paso remplazar el nombre por el propio de cada nodo, esto servirá además para configurar otros softwares y recursos más adelante. Se continúa con la configuración de hosts en el cual se copia el nombre de los nodos que participarán en cada cluster, es decir, en el **cluster de balanceo de carga** y en el **cluster replicador de datos DRBD**.

Abrimos el documento de hosts en cada nodo.

```
1 # nano /etc/hosts
```

Se copia la información en el cluster de balanceo de carga.

```
1 GNU nano 2-5-4           File: /etc/hosts
2 127.0.0.1         Localhost
3 192.168.1.3       loadb1
4 192.168.1.4       loadb2
```

Luego, se copia la información en el cluster replicador de datos DRBD.

```
1 GNU nano 2-5-4           File: /etc/hosts
2 127.0.0.1         Localhost
3 192.168.1.5       server1
4 192.168.1.6       server2
```

La configuración final realizará su efecto una vez reiniciado el nodo configurado. Un método directo de reiniciar mediante terminal es con el siguiente comando en modo root.

```
1 # init 6
```

Una vez reiniciado el sistema se pueden observar los cambios con el comando.

```
1 # ifconfig -a
```

### 2.2.2 Instalando y configurando SSH

El protocolo llamado Security Shell (también denotado como SSH) es un protocolo de comunicación creado con el fin de poder comunicarse con servidores privados, de manera remota, por una compuerta trasera. El servicio que el protocolo presta funciona utilizando el puerto 22/TCP, dentro de la capa de transmisión, que permitirá el envío y recibo de paquetes con los mismos privilegios que un administrador local posea, esto último se logra configurando al servicio SSH con encriptación de comunicación SHA256 y las llaves públicas que el servidor SSH envíe a sus respectivos clientes, configuración tal que será demostrada más adelante.

El funcionamiento de este protocolo permite la comunicación entre dos computadoras creando una seguridad por 2 factores, esto quiere decir que el servicio de SSH creará dentro del archivo de configuración, en la máquina que se esté creando el servicio, dos archivos de identificación los cuales constituyen ser las llaves privadas y llaves públicas del host local. Luego, el servicio enviará de manera segura una invitación con la llave pública y la llave privada al cliente que se requiera

unirse a la red SSH. Además de poder crear un túnel de comunicación, SSH permite copiar, pegar y crear archivos y o carpetas utilizando dos servicios que por sí solos conforman una línea de protocolos independientes, los cuales son SCP y SFTP.

- **SCP**: Secure Copy Protocol, este servicio permite transferir de manera segura archivos informáticos entre un host local y otro remoto, o entre dos hosts remotos, utilizando como base el protocolo SSH. La idea principal de este protocolo consiste en enviar información entre un punto y otro utilizando el canal que el servicio SSH ya generó. La gran ventaja de este tipo de envíos es que evita que existan perdidas en captar información, también conocido como Sniffers, los que interfieren la señal transmitida para gestionar un packet sniffer, el cual se traduce como una extracción de información útil de paquetes de datos.

  La forma en la que se enviarán los datos por terminal debe seguir el siguiente comando.

```
1 # scp ruta_y_nombre_de_archivo cliente@dirección_ip_destino:/ruta_de_recibo
```

- **SFTP**: SSH File Transfer Protocol, este servicio permite de igual manera que el SCP enviar y recibir archivos de manera remota pero comportándose como un servicio más independiente que el mismo SCP. Un ejemplo de lo anterior se manifiesta en que SFTP tiene mayores privilegios en crear listados de directorios, crear directorios, borrar directorios, borrar archivos y así. Las gran ventaja de SFTP por sobre SCP, es que solamente se dedica en enviar archivos de manera segura. Ambos servicios relatados funcionan utilizando el túnel creado por el servicio SSH en el puerto 22/TCP. Para realizar un envío desde este protocolo se podrá realizar de dos formas:

  1. Sin iniciar sesión por SFTP:

```
1 # sftp cliente@dirección_ip_destino:/ruta_del_archivo_local
```

  2. Iniciando sesión por SFTP:

```
1 # sftp cliente@dirección_ip_destino
```

  Este comando requerirá de una password para poder continuar. Una vez seleccionada esta clave el sistema iniciará sesión en el otro dispositivo del cual se tomará el control total. En el caso de querer enviar un archivo desde un nodo a otro, en la sesión iniciada del otro dispositivo se copia el siguiente código.

```
1 # sftp > put local.profile
```

  Luego, se busca y envía el archivo mediante el comando.

```
1 # uploading local.profile to /tecmint/local.profile
```

Lo anterior es sólo un ejemplo, los datos deben ser remplazados con la ruta de envío y recibo de estos.

La instalación de SSH permitirá comunicarse a través de la red sin colocar en riesgo la integridad de los datos, además de volver más dinámica la configuración de los servicios al tener que crear y enviar archivos en vez de escribir en ellos por separado. A continuación se muestra la instalación de la aplicación SSH en la línea de comando con privilegios de administrador.

```
1 # apt-get install openssh-server openssh-client
```

Una vez instalado, se crea una clave en común para que los nodos puedan conectarse al nodo principal.

```
1 # sudo ssh-keygen
```

Con la clave generada se crearán dos archivos que son las llaves privada (id_rsa), y pública (id_rsa.pub) en la ruta ~/.ssh los cuales se observar con el comando de 'lista' en la terminal.

```
1 # -ls
```

Para establecer comunicación entre las direcciones IP de los equipos se requiere que estos pertenezcan a la red SSH. Para esto, se hace envío de la llave pública a través de la red con el siguiente comando.

```
1 # ssh-copy-id ~/.ssh/id_rsa.pub cliente@dirección_IP
```

Nota: La clave asignada en el SSH del clúster de alta disponibilidad será 'cluster' con minúsculas y sin comillas.

Para conocer las conexiones entrantes y salientes utilizadas por el túnel SSH se puede observar el siguiente archivo.

```
1 # nano /var/log/auth.log
```

Finalmente, para iniciar sesión en otra máquina se escribe el siguiente comando.

```
1 # ssh cliente@dirección_ip
```

El cual requerirá la clave de root de la otra máquina para acceder.

## 2.3 Características de los nodos

Conocer las características de los nodos es elemental, ya que el desempeño y rendimiento del cluster de alta disponibilidad dependerá de cada detalle que estos puedan poseer. Las características de los nodos del clúster de balanceo de carga no deben ser tan potentes, dado que dicho balanceo no utilizará más del 5% de requerimientos en el sistema. No así en el caso de los servidores web, ya que estos tendrán que retener por un tiempo limitado la conexión del usuario.

Es importante que las configuraciones de cada sección del cluster de alta disponibilidad estén correctas para prevenir posibles fallos.

### 2.3.1 Nodos balanceadores de carga

Correspondiente a las computadoras Intel NUC según el modelo DN2820FYKH, las cuales tienen un consumo de 37 W cada una, acompañados de un potente procesador Intel Celeron N2820 de 2.41 GHz, datos que convierten a éstas máquinas en un computador liviano, económico y perfecto para la tarea de balance de carga. Las especificaciones para el modelo mencionado se pueden observar en la Tabla 2-1.

Tabla 2-1: Características de Intel NUC DN2820FYKH

| Característica | Especificación |
|---|---|
| Modelo | DN2820FYKH |
| Fecha de lanzamiento | 4to trimestre del 2013 |
| Disco duro | 500 GB, 2.5” |
| Memoria RAM | 4 GB DDR3L-1066/1333 MHz |
| Procesador | Intel Celeron N2820 (1 M de caché, 2.41 GHz) |
| Tarjeta de video | Integrada |
| Salida de video | HDMI 1.4a |
| Red | Ethernet /Intel Wireless N 7260 + Bluetooth 4.0 |

### 2.3.2 Nodos servidores web

Los nodos que serán servidores, a diferencia de los anteriores, no tienen las mismas especificaciones. Esta variación en las características de cada uno desencadenará distintos comportamientos en el desempeño y rendimiento. Una de las principales diferencias entre ambos, es que uno de los nodos es un computador convencional, mientras que el otro es un servidor dedicado. En este punto encontraremos solo la descripción de cada uno de estos nodos, dejando la evaluación, desempeño y rendimiento en el capítulo 3. Las configuraciones que los servidores poseen pueden observarse en la Tabla 2-2.

Tabla 2-2: Características de los servidores

| Característica | Especificación Server 1 | Especificación Server 2 |
|---|---|---|

| | | |
|---|---|---|
| Modelo | Sin especificar | Power Edge R200 |
| Fecha de lanzamiento | Sin especificar | 2010 |
| Disco duro | 1 TB, 3.5” | (x2)160 GB / 250 GB, 3.5” |
| Memoria RAM | 16 GB | 8GB DDR2/1333 MHz |
| Procesador | Intel i5-4440 @3.1 GHz | Intel Xeon @2.83 GHz |
| Tarjeta de video | Integrada ,128 MB | Integrada, 32 MB |
| Salida de video | HDMI/VGA | VGA |
| Red | Ethernet | Ethernet |

# 3 Desarrollo

En este capítulo se abordarán las configuraciones realizadas a los softwares que dan vida al cluster de alta disponibilidad. Esto se logra tomando como referencia la topología de la Figura 2-1, que además servirá de guía al momento de entrar en la configuración de los recursos. También, en este capítulo se mostrará la configuración de Firewall necesaria para cada nodo que participa del cluster.

## 3.1 Softwares asociados

Los softwares utilizados en la construcción del cluster son responsables de que la alta disponibilidad sea constante y sin errores. Una correcta configuración de cada uno de los softwares dará no solo mayor seguridad al sistema, sino también permitirá obtener el mejor rendimiento de cada nodo que participe del clúster de alta disponibilidad. Tal como se menciona al comienzo del capítulo, esta sección se dividirá en dos partes con el fin de mejorar la comprensión de cada una de las configuraciones. Primero, haciendo mención al cluster de balanceo de carga, configuración propia de esta sección. Segundo, refiriéndose al clúster replicador de datos DRBD.

### 3.1.1 Softwares en cluster de balanceo de carga

Los softwares utilizados para el clúster de balanceo de carga son cinco, los cuales poseen distintos usos dentro de esta organización. Y dentro de los cuales podemos encontrar.

- Pacemaker (Figura 3-1): Es un administrador de recursos. Este se preocupa de llevar una máxima disponibilidad detectando y recuperando a nivel de nodos y recursos, los diversos problemas que puedan existir. Pacemaker, como tal, utiliza recursos que trabajan en la capa de mensajería para conocer el estado de los nodos. Además, utiliza, en las versiones recientes, a la aplicación Corosync como apilador de recursos. Dentro de

las posibles configuraciones de nodos que Pacemaker puede administrar se encuentran los tipos:
Activo/Pasivo, N+I (Multinodo), N-TO-N (Copia de datos) y Split-Site (Ej: Un nodo para E-mail y otro para Intranet)


Figura 3-1: Logotipo del administrador de recursos Pacemaker (Fuente: https://wiki.clusterlabs.org/wiki/Pacemaker)

La instalación de este programa se realiza por línea de comando en la terminal de Linux, línea que se especifica con el siguiente comando.

```
1 # sudo apt-get install pacemaker
```

Una vez instalado configuramos al administrador de recursos para que conozca a los nodos participantes del clúster de balanceo de carga. Lo anterior se logra con el siguiente comando.

```
1 # crm configure edit
```

El comando anterior abrirá un script de configuración donde se mencionarán los nodos y recursos necesarios para administrar. Además, de ajuntar la dirección IP virtual necesaria para conocer el estado de los nodos con Heartbeat.

Listado 3-1: Script de configuración para Pacemaker

```
node 1: loadb1 \
attributes standby=off
node 2: loadb2 \
attributes standby=off
primitive haproxy haproxy \
params conffile="/etc/haproxy/haproxy.cfg" \
op monitor interval=30s \
meta is-managed=true target-role=Started
primitive virtual-ip IPaddr \
params ip=192.168.1.13 cidr_netmask=32 \
op monitor interval=5s \
meta is-managed=true target-role=Started
group HAproxyGroup virtual-ip haproxy
property cib-bootstrap-options: \
have-watchdog=false \
dc-version=1.1.14-70404b0 \
cluster-infrastructure=corosync \
cluster-name=debian \
stonith-enabled=false \
no-quorum-policy=ignore \
default-action-timeout=240
rsc_defaults rsc-options: \
resource-stickiness=100
```

Como se observa en el Listado 3-1 se incluyen los nodos que participarán en el cluster de balanceo de carga añadiendo a estos el atributo standby=off. Lo anterior hace referencia a la operación - Activo/Hotstandby - que el balance de carga en el cluster de alta disponibilidad requiere. También, es posible reconocer al software HA-Proxy el cual se especifica como un recurso a administrar pero con configuraciones propias en la ubicación de éste. Finalmente, se puede observar la aparición de la dirección IP virtual, la cuál será utilizada como puerta de acceso al servicio y como medio de comunicación de los estados de los nodos por medio de Heartbeat. La configuración anterior, queda resguardada en un grupo de trabajo llamado 'HAproxyGroup', grupo el cual se crea con el fin de poder generar cambios de estado directamente en el grupo y no de manera independiente, previniendo posibles errores futuros en los cambios de estado.

- **Corosync** (Figura 3-2): Es un proyecto de código abierto derivado del proyecto OpenAIS. Es un sistema de comunicación en grupo con características adicionales para implementar alta disponibilidad en aplicaciones. Es usado como infraestructura de alta disponibilidad por el administrador de recursos Pacemaker. Corosync centra sus esfuerzos en desarrollar, liberar y dar solución de manera gratuita a múltiples proyectos y productos de red.

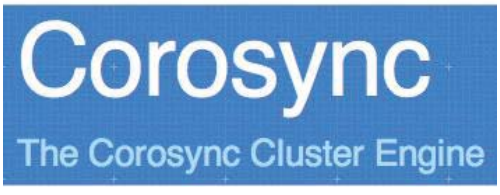


Figura 3-2: Logotipo del apilador Corosync (Fuente: http://www.brainupdaters.net)

Corosync, al igual que Heartbeat, utiliza la capa de mensajería para reconocer los recursos que viajen a través de ésta. El trabajo que realiza éste forma parte de una sección de mejoras las cuales se pueden impartir en alta disponibilidad. Corosync, en este caso particular, será utilizado como apilador de recursos el cual permitirá configurar de mejor manera a estos y con la posibilidad de añadir más nodos al conjunto. Su gran participación en el futuro será la posibilidad de trabajar en modo Multicast para la transmisión de video en tiempo real.

La instalación de este software se realiza añadiendo el siguiente texto en la línea de comandos de Linux en ambos nodos participantes.

```
1 # sudo apt-get install corosync
```

Luego, se instala una llave de autenticación la cual permitirá conectarse únicamente con otro nodo que la posea. Lo anterior se realiza con el siguiente comando.

```
1 # sudo corosync-keygen
```

Durante la creación de la llave el sistema requerirá presionar las teclas en el teclado con el fin de crear entropía en los mensajes y colocar a prueba el servicio. Una vez finalizado, es necesario otorgar los permisos al firewall de Linux para que libere el puerto de comunicación que éste software utilizará, y el cual será el 5404/TCP.

```
1 # sudo ufw allow 5404
```

Al liberar el puerto de comunicación de Corosync éste comenzará inmediatamente a trabajar, tan solo retiene su funcionamiento la llave de autenticación y la configuración que deba Corosync seguir. Lo anterior se resuelve copiando la llave de uno de los nodos mediante SSH y configurando el script de Corosync, como se demuestra en el Listado 3-2.

```
1 # scp /etc/corosync/authkey loadb2@192.168.1.4:/etc/corosync
```

Luego, se configura el script de Corosync en ambos nodos.

```
1 # nano /etc/corosync/corosync.conf
```

Listado 3-2: Script de configuración para Corosync

```
1  GNU nano 2.5.3 File: corosync.conf
2  totem {
3    version: 2
4
5    cluster_name: cluster1           #Nombre del cluster
6    transport: udpu                  #Servicio UNICAST
7    interface {                      #Datos del Sistema general
8      ringnumber: 0                  #Es el nombre del conjunto
9      bindnetaddr: 192.168.1.255     #Es el dato Baddr en ifconfig
10     broadcast: yes
11
```

```
    mcastport: 5405                #Puerto para Multicast
  }
}
quorum {
  provider: corosync_votequorum    #Quien hará el quorum
  two_node: 1
}
nodelist {                         #lista de nodos
  node {                           #Identificamos a un nodo
    ring0_addr: 192.168.1.3        #Dirección IP del 1er nodo
    name: loadb1                   #Nombre del 1er nodo
    nodeid: 1                      #Identificador de nodo
  }
  node {                           #Identificamos el 2do nodo
    ring0_addr: 192.168.1.4        #IP del 2do nodo
    name: loadb2                   #Nombre del 2do nodo
    nodeid: 2                      #Identificador de nodo
  }
}
logging {
  to_logfile: yes                  #Queremos activar el log
  logfile: /var/log/corosync/corosync.log
  to_syslog: yes                   #Adjuntamos al log del sistema
  timestamp: on
}
```

Una vez realizado el procedimiento anterior, se crea una carpeta para adjuntar el funcionamiento de Corosync con Pacemaker. Esto significa que Pacemaker, quien es el encargado de ver qué datos se transmite y a donde, se le advertirá que existe otro software, llamado 'Corosync', que estará funcionando en sus servicios.

Para realizar el paso anterior se crea un nuevo directorio con *mkdir*, el cual se realiza con el siguiente comando:

```
# sudo mkdir -p /etc/corosync/service.d
```

Siguiente, se crea un archivo de configuración del tipo *vim*, cuya misión será el comunicar Pacemaker con Corosync.

```
# sudo vim /etc/corosync/service.d/pcmk
```

Se copia en detalle el script de configuración para el servicio Pacemaker en Corosync.

```
service {
          name: pacemaker
          ver: 1
}
```

Finalmente, se abre el archivo /etc/default/corosync, donde se copia la ruta de los archivos de configuración para Corosync, como se observa en el Listado 3-3.

```
# nano /etc/default/corosync
```

Listado 3-3: Script de configuración para comunicar Pacemaker y Corosync

```
GNU nano 2.5.3    File: /etc/default/corosync

# Corosync runtime directory
#COROSYNC_RUN_DIR=/var/lib/corosync

# Path to corosync.conf
#COROSYNC_MAIN_CONFIG_FILE=/etc/corosync/corosync.conf

# Path to authfile
#COROSYNC_TOTEM_AUTHKEY_FILE=/etc/corosync/authkey

# Command line options
#OPTIONS=""
START=yes
```

Una vez terminado el proceso, es posible observar su comportamiento con el siguiente comando.

```
# sudo corosync-cmapctl | grep members
```

Es recomendable cambiar la prioridad de aparición de nuestro programa Corosync, ya que existe la posibilidad de falla al momento de iniciar los servicios en conjunto con los otros servicios del computador y de nuestra alta disponibilidad. Para adaptar esta prioridad escribimos:

```
# sudo update-rc.d pacemaker defaults 20 21
```

El correcto funcionamiento de Corosync puede observarse en la Figura 3-3, la cual muestra la conexión de ambos nodos balanceadores en el apilador.

```
● corosync.service - Corosync Cluster Engine
   Loaded: loaded (/lib/systemd/system/corosync.service; enabled; vendor preset:
   Active: active (running) since mar 2018-09-11 11:31:18 -03; 1h 56min ago
 Main PID: 1165 (corosync)
    Tasks: 2
   Memory: 58.6M
      CPU: 1min 34.595s
   CGroup: /system.slice/corosync.service
           └─1165 /usr/sbin/corosync -f

sep 11 11:31:18 loadb1 corosync[1165]:   [QB    ] server name: cfg
sep 11 11:31:18 loadb1 corosync[1165]:   [QB    ] server name: cpg
sep 11 11:31:18 loadb1 corosync[1165]:   [QB    ] server name: votequorum
sep 11 11:31:18 loadb1 corosync[1165]:   [QB    ] server name: quorum
sep 11 11:31:18 loadb1 corosync[1165]:     [TOTEM ] adding new UDPU member
{192.168.1.3}
sep 11 11:31:18 loadb1 corosync[1165]:     [TOTEM ] adding new UDPU member
{192.168.1.4}
sep 11 11:31:18 loadb1 systemd[1]: Started Corosync Cluster Engine.
sep 11 11:31:18 loadb1 corosync[1165]:       [TOTEM ] A new membership
(192.168.1.3:105484} was formed. Members joined: 1
sep 11 11:31:22 loadb1 corosync[1165]: Sep 11 11:31:22 notice  [TOTEM ] A new
membership {192.168.1.3:105488} was formed. Members joined: 2
```

```
30 sep 11 11:31:22 loadb1 corosync[1165]: [TOTEM ] A new membership
(192.168.1.3:105488} was formed. Members joined: 2
```

Figura 3-3: Correcto funcionamiento de Corosync

En la línea 3, de la Figura 3-3, se muestra el estado del servicio de Corosync en el nodo balanceador 1 (Loadb1). Además, el servicio reconoce al nodo balanceador 2 (Loadb2) que funciona dentro de su sistema, como lo muestra la línea 20.

- **Heartbeat** (Figura 3-4): Es un recurso que ofrece servicios de infraestructura de clúster (comunicación y afiliación) a sus clientes. Esto permite a los clientes saber de la presencia de procesos iguales en otras máquinas y poder intercambiar mensajes de manera fácil. Este recurso puede trabajar como *demonio*, es decir, como un proceso independiente oculto a simple vista del usuario. Este recurso puede trabajar con un administrador de recursos como Pacemaker o solo.


Figura 3-4: Logotipo del recurso de mensajería Heartbeat

Heartbeat, como se menciona al comienzo de este informe, es un software que trabaja en la capa de mensajería. Lo anterior hace mención al trabajo que Heartbeat realiza en el sistema, ya que es un mensaje que envían los nodos hacía Pacemaker para conocer el estado de los participantes. La configuración que Heartbeat necesita para funcionar consta de tres archivos propios de éste, los cuales son: ha.cf, haresources y authkeys. A continuación se muestran las configuraciones para estos archivos.

```
1 # nano /etc/heartbeat/ha.cf
```

Se edita el script de configuración para Heartbeat indicando los tiempos de conexión y las restricciones para cada nodo.

Listado 3-4: Script de configuración para Heartbeat

```
GNU nano 2.5.3                File: /etc/ha.d/ha.cf
autojoin        none            # Ningún nodo puede autoingresar solo
bcast           enp3s0          # Mencionamos la red Wlan o eth a ocupar
logfacility     local0          # Afirmamos la existencia de un archivo log
logfile /var/log/ha-log         # Colocamos la ruta del archivo log
debug           1
warntime        10              # Tiempo de espera en enviar la advertencia
deadtime        20              # Tiempo en dar por muerto un nodo
initdead        120
keepalive       2               # Tiempo que se mantiene al cliente conectado
node            loadb1                  # Añadimos al nodo loadb1
node            loadb2                  # Añadimos al nodo loadb2
mcast           enp3s0 239.0.0.1 694 1 0
crm             respawn
ucast           enp3s0 192.168.1.3            # unicast loadb1
ucast           enp3s0 192.168.1.4            # unicast loadb2
```

**Nota:** Recordar que este procedimiento se debe realizar en ambos nodos balanceadores de carga.

Luego, para añadir seguridad en la comunicación de los nodos unidos por Heartbeat, se creará una encriptación para las claves de autenticación asociadas a estos equipos. Esto se logra con el siguiente comando directamente en la terminal:

```
# ( echo -ne "auth 1\n1 sha1" ; \
> dd if = /dev/urandom bs=512 count=1 | openssl md5) \
>> /etc/ha.d/authkeys
```

Otorgamos el permiso de lectura y escritura al archivo 'authkeys'.

```
# chmod 600 /etc/ha.d/authkeys
```

Luego, se envía el archivo de autenticación al balanceador 2 (loadb2)

```
# scp /etc/ha.d/authkeys loadb2@192.168.1.4:/etc/ha.d/
```

Finalmente, para terminar la configuración de Heartbeat, se menciona el nombre y dirección IP del nodo que se está configurando en la red que transmitiremos. Esta configuración debe hacerse en ambos nodos balanceadores, desde la ruta /etc/ha.d/haresources.

```
# nano /etc/ha.d/haresources
```

Una vez dentro, se edita el script de haresources.

```
GNU nano 2.5.3          File: haresources
loadb1 192.168.1.3
```

Para conocer el estado del servicio Heartbeat, se reinicia el sistema y en la terminal se coloca:

```
1 # service heartbeat status
```

Cuya respuesta se observa en la Figura 3-5.

```
1  ● heartbeat.service - LSB: High-availability services.
2     Loaded: loaded (/etc/init.d/heartbeat; bad; vendor preset: enabled)
3     Active: active (running) since mar 2018-09-11 11:31:18 -03; 1h 56min ago
4       Docs: man:systemd-sysv-generator(8)
5    Process: 1287 ExecStart=/etc/init.d/heartbeat start (code=exited, status=0/SUC
6   Main PID: 1349 (heartbeat)
7      Tasks: 9
8     Memory: 18.4M
9        CPU: 39.976s
10    CGroup: /system.slice/heartbeat.service
11            ├─1349 heartbeat: master control proces
12            ├─1363 heartbeat: FIFO reader
13            ├─1364 heartbeat: write: bcast enp3s
14            ├─1365 heartbeat: read: bcast enp3s
15            ├─1366 heartbeat: write: mcast enp3s
16            ├─1367 heartbeat: read: mcast enp3s
17            ├─1368 heartbeat: write: ucast enp3s
18            ├─1369 heartbeat: read: ucast enp3s
19            └─1637 /usr/lib/heartbeat/ccm
20
21
22
23
24
```

Figura 3-5: Correcto funcionamiento de Heartbeat

En la línea 1 de la figura se observa el estado del servicio de Heartbeat, mostrando en color verde el modo activo de éste.

- **HA-Proxy** (Figura 3-6): Es un software que provee alta disponibilidad en balanceo de carga, además de agregar servicio proxy por medio de TCP y HTTP en las aplicaciones que se basen en dichas tecnologías que necesiten responder ante peticiones en múltiples servidores. Está programado en C, por lo que tiene la reputación de ser rápido y eficiente (En términos de reputación y uso de memoria). HA-Proxy es utilizado en muchos sitios de alto perfil, incluyendo GoDaddy, GitHub, Bitbucket, Stack Overflow, Reddit, Speedtest.net, Tumblr, Twitter y Tuenty. Además, es utilizado en OpsWorks producto de Amazon Web Service. Este software es el encargado de direccionar las peticiones en el clúster de alta disponibilidad, abriendo también la posibilidad de configurar de muchas formas para prevenir peticiones maliciosas por rutas indebidas.


Figura 3-6: Logotipo del software HA-Proxy (Fuente: https://www.haproxy.com )

HA-Proxy forma una parte importante en la configuración de DRBD para los servidores, ya que este contiene la información de quienes serán los destinatarios a recibir la información que los servidores proxys alojarán momentáneamente. Para esta configuración, se asignarán las direcciones IP de los servidores Server1: 192.168.1.5 y Server2: 192.168.1.6.

Abrimos el archivo de configuración para HA-Proxy

```
1 # nano /etc/haproxy/haproxy.conf
```

El archivo de configuración para los balanceadores de carga deberá quedar de la siguiente manera, como se muestra en el Listado 3-5.

Listado 3-5: Script de configuración para HA-Proxy

```
 1 global
 2         log 192.169.0.1 local0 notice
 3         maxconn 2000
 4         user haproxy
 5         group haproxy
 6         daemon
 7 defaults
 8         log     global
 9         mode    http
10         option  httplog
11         option  dontlognull
12         retries 3
13         option  redispatch
14         maxconn 2000
15         timeout connect 5000
16         timeout client  10000
17         timeout server  10000
18
19 frontend http_front
20         bind *:80
21         stats uri /haproxy?stats
22         default_backend http_rear
23
24 backend http_rear
25
26         balance roundrobin
27         option httpchk
28         server pageie.cl        192.168.1.5:80 check
29         server pageie.cl        192.168.1.6:80 check backup
30
```

Notamos en esta configuración, del Listado 3-5, el cómo se añaden a la lista los nodos los servidores utilizados para DRBD, en el cual asignamos a uno como primario y otro como *backup*, tal como se observa en las líneas 28 y 29. Además, se considera como algoritmo de comunicación 'Round Robin' el cual irá constantemente probando la conexión del segundo nodo a la espera de que este sea el nodo primario.

Una vez terminada la configuración para el cluster de balanceo de carga, se realiza una prueba del servicio con Pacemaker, el cual deberá indicar a cada uno de los recursos utilizados y mostrar la conexión de cada uno de los nodos participantes, como se muestra en la Figura 3-7.

```
root@loadb1:/home/loadb1# crm status
Last updated: Wed Jul 18 16:23:26 2018          Last change: Wed Jul 18 16:18:06
2018 by root via crm_attribute on loadb2
Stack: corosync
Current DC: loadb1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

Online: [ loadb1 loadb2 ]

Full list of resources:

 Resource Group: HAproxyGroup
     virtual-ip   (ocf::heartbeat:IPaddr):      Started loadb1
     haproxy      (ocf::heartbeat:haproxy):     Started loadb1
```

Figura 3-7: Correcto funcionamiento de Pacemaker y servicios

### 3.1.2 Softwares en cluster de replicación de datos

Los softwares utilizados para el clúster de replicación de datos son tres. Los cuales conforman las áreas de balanceo interno de carga, estados de los nodos y replicación de datos. Por medio de HA-Proxy, software del clúster de balanceo de carga, será posible recibir las peticiones en el servidor apuntando a solo uno de ellos por medio de su dirección IP. Los softwares que se mostrarán a continuación serán los encargados de recibir esta petición y administrarla según el requisito de la misma.

- **Heartbeat** (Figura 3-4): Al igual que en el cluster anterior, Heartbeat forma parte importante en la comunicación de los nodos. De esta manera, es posible conocer los estados que estos nodos posean y el cual servirá de guía para las acciones que el software replicador de datos DRBD realice sobre los servidores. Se instala Heartbeat en ambos servidores al igual que la vez anterior.

```
# apt-get install heartbeat
```

  La configuración de Heartbeat está situada en tres archivos importantes dentro de los nodos, estos son: ha.cf, haresources y authkeys. Archivos los cuales quedarán sus scripts de la siguiente manera.

```
# nano /etc/ha.d/ha.cf
```

Listado 3-6: Script de configuración para Heartbeat en servidores web

```
mcast enp4s0 239.0.0.10 694 1 0
warntime 4
deadtime 5
initdead 15
keepalive 2
auto_failback off
node server1
node server2
```

  Así mismo para haresources.

```
# nano /etc/ha.d/haresources
```

Para server 1

```
server1 IPaddr::192.168.1.14/24/enp4s0 drbddisk::r0
Filesystem::/dev/drbd0::/mnt::ext4 apache2
```

Para server 2

```
server2 IPaddr::192.168.1.14/24/eno1 drbddisk::r0
Filesystem::/dev/drbd0::/mnt::ext4 apache2
```

Finalmente, el archivo de autenticación authkeys

```
# nano /etc/ha.d/authkeys
```

Para server 1 y server 2

```
auth 1
1 md5 Access1cluster
```

- **LAMP** (Figura 3-8): Este es un acrónimo usado para describir un sistema de infraestructura de Internet que usa las siguientes herramientas. Linux, como sistema operativo. Apache, el servidor web. MySQL/MariaDB, como el gestor de datos o la base de datos. Finalmente, Perl/PHP/Python, como los lenguajes de programación. Este software incluye un paquete completo para volver un nodo en servidor, además que comunica inmediatamente cada servicio instalado previniendo posibles errores de configuración durante su instalación.


Figura 3-8: Logotipo del software para servidor LAMP

Una vez instalado el sistema operativo Ubuntu Desktop 16.04, se procede a instalar las aplicaciones necesarias para convertir este nodo en un servidor. Estas aplicaciones se pueden encontrar en un paquete disponible al instalar el software “Tasksel”.

```
# sudo apt-get install tasksel
```

Luego, se selecciona instalar el programa ‘Lamp-Server’, que instalará: Apache2, MySql, y PHP. También incluye otros tipos de descriptor de datos MariaDB, así como otros lenguajes de programación tales como Perl o Python.

```
1 # sudo tasksel install lamp-server
```

Una vez completada la instalación, se pedirá ingresar una clave para nuestra base de datos MySQL, la cual contendrá la estructura misma. Añadimos, según lo anterior, la clave 'SQLClusterZKP!'. Aprovechamos la misma instancia para instalar los softwares asociados a la pruebas del servicio 'Apache Bench' e IPerf, los cuales serán vistos en el capítulo 4.

Apache Bench:

```
1 # apt-get install apache2-utils
```

Iperf:

```
1 # apt-get install iperf
```

Las configuraciones del servicio web, ya sea, la base de datos y la misma información del diseño del servicio corresponden a un ítem separado a los motivos de este proyecto. Es por que en este informe se demostrarán sólo las vías de instalación.

- **DRBD** (Figura 3-9): Distributed Replicated Block Device. Es un replicador de datos de sistema para la plataforma Linux. Se implementa como un driver el cual administra el espacio utilizado en el disco. DRBD es normalmente utilizado en alta disponibilidad (HA) en clústers, pero además, en su última versión DRBD9, puede ser utilizado para crear un gran almacenamiento con foco de integración a la nube. DRBD, se refiere además, como un dispositivo el cual hace referencia a un bloque en un esquema lógico de volúmenes, es decir, afecta directamente a los discos, muy similar a RAID. Este software es gratuito ya que cuenta con licencia GNU (General Public Licence) en su versión 2. La versión utilizada en este proyecto es DRBD8, del año 2007, la cual soporta configuraciones de balanceo de carga, permitiendo a los nodos acceder al sistema DRBD en modos Lectura/Escritura con semántica de almacenamiento compartido.


Figura 3-9: Logotipo del software replicador de datos DRBD

La instalación de este software se realiza mediante el siguiente comando en terminal de Linux para ambos servidores.

```
1 # apt-get install drbd8-utils
```

En esta oportunidad se utilizará la versión 8 de DRBD ya que sólo se requiere un balanceo de carga simple en caso que el servicio principal falle. Además, ya que los nodos servidores no contienen las mismas especificaciones de hardware estaremos

construyendo un clúster semi homogéneo, el cual se conforma de distintas piezas pero mismo sistema operativo. Es importante notar que el servidor 1 sólo cuenta con un solo disco duro, en cambio el servidor 2 contiene dos, por lo que para que exista una clonación en los datos será necesario particionar el disco del servidor 1, o simplemente agregar otro disco. Para esta oportunidad se mostrará el desarrollo con el servidor 1 particionado, de esta manera de ser necesario en el futuro se pueda aprovechar al máximo las particiones en cada disco.

El disco duro de 1TB servirá para alojar la página web y el almacenamiento que los datos de esta página pueda contener, de esta manera aprovechar al máximo la disponibilidad de ese disco. Las tablas siguientes (Tabla 3-1 y 3-2) hacen referencia a las particiones señaladas.

Tabla 3-1: Particiones para el servicio web

| Servicio | Partición | Memoria | Nodo |
|---|---|---|---|
| Página Web | /dev/sda3 | 5 GB | Servidor 1 |
| Página Web | /dev/sdb1 | 5 GB | Servidor 2 |

Las particiones *sda* se reconocen como aquellas que pertenecen al disco donde se aloja el sistema operativo, es por eso que al tener un disco independiente el servidor 2 el sistema lo reconoce como *sdb1*.

Tabla 3-2: Particiones para el servicio de nube

| Servicio | Partición | Memoria | Nodo |
|---|---|---|---|
| Nube | /dev/sda4 | 230 GB | Servidor 1 |
| Nube | /dev/sdb2 | 230 GB | Servidor 2 |

Aprovechando la misma situación anterior, se particiona otra parte del disco para generar otra copia del almacenamiento que la página web pueda alojar. Es importante reconocer el tamaño que las particiones tendrán, ya que estas no se sumarán sino que se comportarán como particiones independientes.

Una vez particionado el disco del servidor 1, se procede a dar un formato *ext4* a los discos que participarán en DRBD, es decir, sda3 y sdb1.

El formato para sda3.

```
1 # dd if=/dev/zero of=/dev/sda3 bs=1M count=128
```

Así mismo, el formato para sdb1.

```
# dd if=/dev/zero of=/dev/sdb1 bs=1M count=128
```

Una vez completo el formato en los discos se procede a editar los archivos de configuración de DRBD, este procedimiento debe realizarse en ambos servidores ya que la información de configuración deberá retenerse al momento de intercambiar labores.

El archivo de configuración del programa DRBD se aloja en /etc/drbd.conf y el cual para crear la configuración básica deberá mostrar lo siguiente.

Listado 3-7: Script de configuración para Heartbeat en servidores web

```
global { usage-count no; }
common { protocol C; }
resource r0 {

        net { cram-hmac-alg sha1;
                shared-secret "clusterdrbd";
        }
        on server1 {
                device /dev/drbd0;
                disk /dev/sda3;
                address 192.168.1.5:7788;
                meta-disk internal;
        }
        on server2 {
                device /dev/drbd0;
                disk /dev/sdb1;
                address 192.168.1.6:7788;
                meta-disk internal;
        }
}
```

Una vez configurado DRBD en ambos nodos es necesario crear un nuevo recurso que esté preocupado de copiar la información entre ambos. Para esto, colocamos en terminal de ambos nodos los siguientes tres códigos.

```
# modprove drbd
# drbdadm create-md r0
# drbdadm up r0
```

Finalmente, para comenzar la escritura en los discos primero deberá partir por el nodo primario. La copia de este disco primario será automáticamente realizada en el secundario. Es por eso que el siguiente código sólo es válido para el servidor 1.

```
# drbdadm -- --overwrite-data-of-peer primary r0/0
```

Terminado este procedimiento el sistema deberá mostrar la información entre los servidores con un estado *UpToDate*/*UpToDate*. Esta visualización se logra viendo el proceso DRBD con el siguiente comando *cat*.

```
# cat /proc/drbd
```

El cual muestra como resultado.

```
root@server1:/home/server1# cat /proc/drbd
version: 8.4.7 (api:1/proto:86-101)
srcversion: 2DCC561E7F1E3D63526E90D
 0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:4 nr:8 dw:12 dr:17 al:1 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
root@server1:/home/server1#
```

## 3.2 Firewall UFW

El Firewall es el servicio de seguridad que garantiza que exista comunicación únicamente por los puertos habilitados. En esta oportunidad se utilizará el Firewall que Ubuntu Server 16.04 (Figura 3-10) provee ya que su simple uso permite modificar las tablas –IP tables- de acceso por puerto de manera dinámica. Uncomplicated Firewall, por sus siglas en Inglés – Firewall sin complicación – es un firewall o cortafuegos diseñado para ser de fácil uso. Este utiliza la línea de comandos para configurar las tablas usando un pequeño número de comandos simples. UFW está escrito en Python y es un programa para GNU/Linux.


Figura 3-10: Logo Uncomplicated Firewall de Ubuntu.
(Fuente: https://universo-digital.net)

Para agilizar la configuración de las tablas, se permitirá cualquier tipo de conexión por dirección IP que circule dentro de los puertos, quedando la responsabilidad de bloquear estas conexiones una vez terminada la topología.

Una parte importante es la seguridad por medio del bloqueo de puertos de comunicación que el sistema dispone. De esta manera, es posible restringir el acceso a agentes externos o intrusos sin privilegios que puedan afectar a la información y configuración del sistema. Además, se mejorará la seguridad determinando los caminos por los cuales la información podrá acceder. Dentro de la configuración final necesaria para permitir el funcionamiento de todos los softwares y recursos del sistema es necesario dejar las tablas de ambos clusters como se muestra a continuación.

### 3.2.1 Firewall en cluster de balance de carga

Se permite, en esta configuración, el acceso a los servicios asociados al cluster de balance de carga. Los puertos utilizados hacen referencia a los softwares y recursos vistos anteriormente, los cuales acceden a su activación y configuración con el siguiente comando.

```
1 # sudo ufw enable
```

Luego, se escribe el comando que permitirá el uso de un puerto especificando en 'software' el nombre de este, o simplemente colocando el puerto asociado.

```
# sudo ufw allow software
```

Nota: Existen pocos softwares los cuales se pueden habilitar de esta manera con UFW, ya que en general son los softwares asociados a la comunicación básica. Ejemplo: SSH.

El resultado final de las tablas de Firewall según los softwares utilizados en este cluster quedará de la siguiente manera.

```
To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
5404                       ALLOW       Anywhere
Anywhere
694/udp                    ALLOW       Anywhere
443                        ALLOW       Anywhere
5001                       ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
5404 (v6)                  ALLOW       Anywhere (v6)
694/udp (v6)               ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
5001 (v6)                 ALLOW      Anywhere (v6)
```

Figura 3-11: Estado del servicio de Firewall UFW para clúster de balanceo de carga.

Los puertos utilizados para esta configuración pertenecen a:

- Puerto 22: SSH
- Puerto 80: HTTP
- Puerto 5404: Corosync
- Puerto 694: Heartbeat
- Puerto 5001: IPerf
- Puerto 443: SSL

### 3.2.2 Firewall en cluster de replicación de datos

El mismo procedimiento anterior se lleva a cabo en los servidores, como muestra la Figura 3-12.

```
To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
694/udp                    ALLOW       Anywhere
7788                       ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
694/udp (v6)               ALLOW       Anywhere (v6)
7788 (v6)                 ALLOW      Anywhere (v6)
```

Figura 3-12: Estado del servicio de Firewall UFW para clúster de replicación de datos

Los puertos utilizados para esta configuración pertenecen a:

- Puerto 22: SSH
- Puerto 80: HTTP
- Puerto 694: Heartbeat
- Puerto 7788: DRBD

# 4 Pruebas y resultados obtenidos

Los resultados provienen de los datos obtenidos luego de realizar distintas pruebas al servicio. En este capítulo se utilizan los softwares IPerf y Apache Bench de los cuales mostrarán los límites en la línea de transmisión de datos y la cantidad de peticiones que el sistema puede abordar. Ambos softwares forjarán la idea de la reacción del servicio ante las configuraciones realizadas en los capítulos anteriores. Además, se mostrarán pruebas de recuperación – o Faiolver- que demostrarán en el terminal la correcta configuración de los softwares y recursos asociados a la topología final del clúster de alta disponibilidad.

## 4.1 IPerf

Para conocer la cantidad de máxima de datos que se pueden transmitir dado un ancho de banda en los servidores utilizaremos IPerf para que realice esta prueba. Primero, realizaremos la prueba en el servidor 1 reconociendo un ancho de banda de aproximadamente 100 Mbits/sec correspondientes al ancho de banda por canal existente en el router.

```
root@loadb1:/home/loadb1# iperf -c 192.168.1.5 -i 1

------------------------------------------------------------
Client connecting to 192.168.1.5, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.3 port 45758 connected with 192.168.1.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  1.0- 2.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  2.0- 3.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  3.0- 4.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  4.0- 5.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  5.0- 6.0 sec  11.1 MBytes  93.3 Mbits/sec
[  3]  6.0- 7.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  7.0- 8.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  8.0- 9.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  9.0-10.0 sec  11.1 MBytes  93.3 Mbits/sec
[  3]  0.0-10.0 sec   112 MBytes  94.1 Mbits/sec
root@loadb1:/home/loadb1#
```

Realizando el mismo procedimiento para el servidor dos y considerando el mismo ancho de banda se encuentran los siguientes resultados.

```
1  # iperf -c 192.168.1.6 -i 1
   ------------------------------------------------------------
   Client connecting to 192.168.1.6, TCP port 5001
   TCP window size: 85.0 KByte (default)
   ------------------------------------------------------------
   [  3] local 192.168.1.3 port 36568 connected with 192.168.1.6 port 5001
   [ ID] Interval       Transfer     Bandwidth
   [  3]  0.0- 1.0 sec  11.2 MBytes  94.4 Mbits/sec
   [  3]  1.0- 2.0 sec  11.1 MBytes  93.3 Mbits/sec
   [  3]  2.0- 3.0 sec  11.2 MBytes  94.4 Mbits/sec
   [  3]  3.0- 4.0 sec  11.2 MBytes  94.4 Mbits/sec
   [  3]  4.0- 5.0 sec  11.1 MBytes  93.3 Mbits/sec
   [  3]  5.0- 6.0 sec  11.2 MBytes  94.4 Mbits/sec
   [  3]  6.0- 7.0 sec  11.2 MBytes  94.4 Mbits/sec
   [  3]  7.0- 8.0 sec  11.2 MBytes  94.4 Mbits/sec
   [  3]  8.0- 9.0 sec  11.2 MBytes  94.4 Mbits/sec
   [  3]  9.0-10.0 sec  11.1 MBytes  93.3 Mbits/sec
   [  3]  0.0-10.0 sec   112 MBytes  94.1 Mbits/sec
   root@loadb1:/home/loadb1#
```

En conclusión, según lo observado en las respuestas de IPerf y considerando un mismo ancho de banda para ambos nodos observamos que no existe diferencia alguna entre la cantidad de datos transferidos por segundo las cuales llegan a completar 112 MBytes en 10 segundos de transferencia. Dada la similitud de los datos se asumen una respuesta igual ante un mismo ancho de banda, por lo que el detalle de las gráficas serán admitidas tanto para el Servidor 1 y al Servidor 2, como se puede observar en la Figura 4-1 y 4-2.

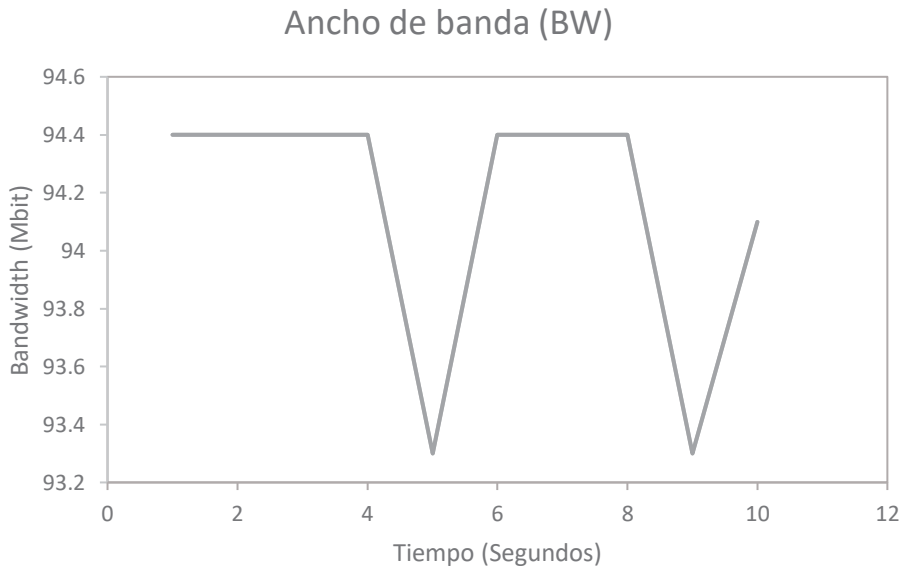


Figura 4-1: Gráfica de ancho de banda para Server 1 y Server 2

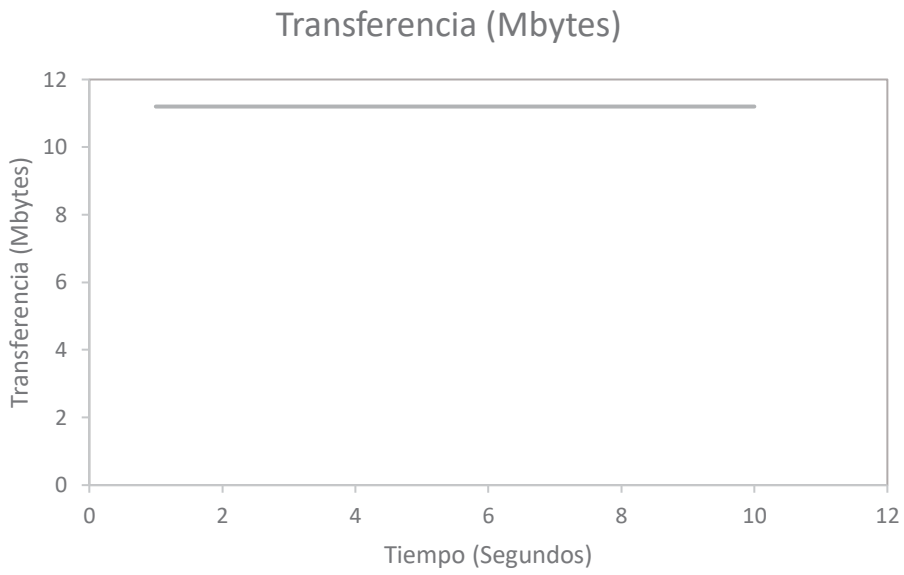


Figura 4-2: Gráfica de transferencia realizada para Server 1 y Server 2

Se puede observar mediante las gráficas que el comportamiento de la transferencia de datos no depende del poder de computo de cada nodo, sino del límite existente en el ancho de banda que la red posee, es por eso que para este caso los resultados serán iguales para cualquier cantidad de nodos existentes dentro de la red.

## 4.2 Apache Bench

Para conocer la cantidad de peticiones los servidores pueden atender en un cierto periodo se utiliza Apache Bench. Esta herramienta considerará el poder de computo que cada nodo servidor posea como hardware. Por lo tanto, el tiempo y la cantidad de peticiones son las variables a reconocer en esta medición. La Figura 4-3 muestra una prueba realizada con Apache Bench desde un balanceador a un servidor.

```
root@loadb1:/home/loadb1# al -k -c 350 -n 20000 192.168.1.5:80/posts
The program 'al' is currently not installed. You can install it by typing:
apt install mono-devel
root@loadb1:/home/loadb1# ab -k -c 350 -n 20000 192.168.1.5:80/posts
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.5 (be patient)
Completed 2000 requests
Completed 4000 requests
Completed 6000 requests
Completed 8000 requests
Completed 10000 requests
Completed 12000 requests
Completed 14000 requests
Completed 16000 requests
```

```
Completed 18000 requests
Completed 20000 requests
Finished 20000 requests


Server Software:        Apache/2.4.18
Server Hostname:        192.168.1.5
Server Port:            80

Document Path:          /posts
Document Length:        280 bytes

Concurrency Level:      350
Time taken for tests:   0.993 seconds
Complete requests:      20000
Failed requests:        19
   (Connect: 0, Receive: 0, Length: 19, Exceptions: 0)
Non-2xx responses:      19981
Keep-Alive requests:    19853
Total transferred:      9905091 bytes
HTML transferred:       5594680 bytes
Requests per second:    20138.01 [#/sec] (mean)
Time per request:       17.380 [ms] (mean)
Time per request:       0.050 [ms] (mean, across all concurrent requests)
Transfer rate:          9739.69 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   1.9      0      28
Processing:     1    6   1.9      6      37
Waiting:        0    6   1.9      6      37
Total:          1    6   3.0      6      47

Percentage of the requests served within a certain time (ms)
  50%      6
  66%      6
  75%      6
  80%      6
  90%      6
  95%      7
  98%     11
  99%     19
 100%     47 (longest request)
```

Figura 4-3: Prueba de Apache Bench en Servidor 1

Considerando el hardware que el servidor 1 posee, es posible conocer la respuesta para 20000 peticiones simultáneas, las cuales toman 47 milisegundos, considerando también que la respuesta a la mitad de estas peticiones se realizó en tan sólo 6 milisegundos.

Se realiza la misma prueba anterior en el servidor 2, considerando las especificaciones de hardware que esta máquina posee. Los resultados se demuestran en la Figura 4-4.

```
root@loadb1:/home/loadb1# ab -k -c 350 -n 20000 192.168.1.6:80/posts
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
```

```
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.6 (be patient)
Completed 2000 requests
Completed 4000 requests
Completed 6000 requests
Completed 8000 requests
Completed 10000 requests
Completed 12000 requests
Completed 14000 requests
Completed 16000 requests
Completed 18000 requests
Completed 20000 requests
Finished 20000 requests


Server Software:        Apache/2.4.18
Server Hostname:        192.168.1.6
Server Port:            80

Document Path:          /posts
Document Length:        280 bytes

Concurrency Level:      350
Time taken for tests:   9.787 seconds
Complete requests:      20000
Failed requests:        31
   (Connect: 0, Receive: 0, Length: 31, Exceptions: 0)
Non-2xx responses:      19969
Keep-Alive requests:    19777
Total transferred:      9896353 bytes
HTML transferred:       5591320 bytes
Requests per second:    2043.52 [#/sec] (mean)
Time per request:       171.273 [ms] (mean)
Time per request:       0.489 [ms] (mean, across all concurrent requests)
Transfer rate:          987.47 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    3  50.9      0    1003
Processing:     0   64 587.9      0    8775
Waiting:        0   60 566.7      0    8775
Total:          0   66 622.1      0    9775

Percentage of the requests served within a certain time (ms)
  50%      0
  66%      0
  75%      0
  80%      0
  90%      0
  95%      0
  98%     20
  99%   2116
 100%   9775 (longest request)
root@loadb1:/home/loadb1#
```

Figura 4-4: Prueba de Apache Bench en Server 1

En esta oportunidad se observa que la respuesta del sistema es un poco más lenta en comparación a la prueba anterior, dadas las mismas peticiones entrantes, ya que esta prueba promete cumplir las 20000 conexiones en 9775 milisegundos o 9,7 segundos. Dicha velocidad de respuesta permite decidir que el servidor 1 siempre será quien más rápido responda a más solicitudes haciendo cumplir la responsabilidad de estar siempre activo. La importancia de estas mediciones muestran uno de los peores casos, al estar configurado por script para cumplir con el servicio *KeepAlive* en las peticiones entrantes, esto se refiere, a mantener las conexiones entrantes ligadas al sistema sin desconectar, servicio que permitirá mantener las sesiones abiertas para el usuario en todo momento.


Figura 4-5: Gráfica de peticiones entrantes para el servidor 1


Figura 4-6: Gráfica de peticiones entrantes para el servidor 2

Se puede observar, en las Figuras 4-5 y 4-6, las peticiones donde el desempeño del servidor 2 es más instantáneo que el servidor 1, ya que logra conectar un 95% de las 20000 peticiones entrantes en un tiempo cercano a 1 ms, pero demorando más de 9 segundos en responder al 5% restante. A

diferencia de este, el servidor 1 responde al 95% de las solicitudes en 7 ms (7 veces más lento que el servidor 2) pero logrando llegar al 100% de las solicitudes en 47 ms. Lo que considera ser más apto a ser servidor principal al servidor 1 y dejando de servidor de respaldo al servidor 2.

## 4.3 Failover

Para probar el comportamiento del sistema ante posibles fallos se utilizarán algunas herramientas que el administrador de recursos de clúster Pacemaker (CRM) proporciona. Además, se realizará una prueba de transmisión mediante la desconexión del cable de Ethernet, suponiendo el fallo de hardware por parte de uno de los servidores proxys.

### 4.3.1 Failover con Pacemaker en balanceadores de carga

Al igual que la vez anterior, se realiza un failover con Pacemaker para conocer la respuesta que los servidores tienen ante una posible falla. Para comenzar con la prueba, se genera la suspensión de uno de los nodos con el siguiente comando:

```
1 # crm node standby
```

La respuesta del servicio Pacemaker muestra que el nodo loadb1 ha sido afectado por la configuración de estado, tal como se muestra en la Figura 4-7.

```
root@loadb1:/home/loadb1# crm status
Last updated: Wed Jul 18 16:08:00 2018          Last change: Wed Jul 18 16:07:56 2018 by
root via crm_attribute on loadb1
Stack: corosync
Current DC: loadb1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

Node loadb1: standby
Online: [ loadb2 ]

Full list of resources:

 Resource Group: HAproxyGroup
     virtual-ip   (ocf::heartbeat:IPaddr):      Started loadb2
     haproxy      (ocf::heartbeat:haproxy):     Started loadb2

root@loadb1:/home/loadb1#
```

Figura 4-7: Failover en loadb1 asumiendo loadb2

Una vez realizado el paso anterior, se procede a volver a levantar el sistema en el nodo loadb1 y de la misma manera se realizará la prueba en el nodo loadb2, como se muestra en la Figura 4-8.

```
root@loadb2:/home/loadb2# crm status
Last updated: Wed Jul 18 16:13:12 2018          Last change: Wed Jul 18 16:13:09 2018 by
root via crm_attribute on loadb2
Stack: corosync
Current DC: loadb1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

Node loadb2: standby
Online: [ loadb1 ]

Full list of resources:
```

```
 Resource Group: HAproxyGroup
     virtual-ip   (ocf::heartbeat:IPaddr):     Started loadb1
     haproxy      (ocf::heartbeat:haproxy):    Started loadb1

root@loadb2:/home/loadb2#
```

Figura 4-8: Failover en loadb2 asumiendo loadb1

Esta prueba de Failover nos muestra el caso que el sistema siga funcionando con uno de los nodos caído pero no necesariamente desconectado de la red. En caso de conocer el desempeño del sistema en caso de desconectar su cable de red su resultado sería el mostrado en la Figura 4-9.

```
root@loadb2:/home/loadb2# crm status
Last updated: Wed Jul 18 16:19:03 2018         Last change: Wed Jul 18 16:18:06 2018 by
root via crm_attribute on loadb2
Stack: corosync
Current DC: loadb2 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

Online: [ loadb2 ]
OFFLINE: [ loadb1 ]

Full list of resources:

 Resource Group: HAproxyGroup
     virtual-ip   (ocf::heartbeat:IPaddr):     Started loadb2
     haproxy      (ocf::heartbeat:haproxy):    Started loadb2

root@loadb2:/home/loadb2#
```

Figura 4-9: Failover con desconexión en loadb1 asumiendo loadb2

Al igual que lo visto en la Figura 4-9, se vuelve a conectar el cable de red en loadb1 y se procede a desconectar el cable del nodo loadb2, como se muestra en la Figura 4-10.

```
root@loadb1:/home/loadb1# crm status
Last updated: Wed Jul 18 16:20:54 2018         Last change: Wed Jul 18 16:18:06 2018 by
root via crm_attribute on loadb2
Stack: corosync
Current DC: loadb1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

Online: [ loadb1 ]
OFFLINE: [ loadb2 ]

Full list of resources:

 Resource Group: HAproxyGroup
     virtual-ip   (ocf::heartbeat:IPaddr):     Started loadb1
     haproxy      (ocf::heartbeat:haproxy):    Started loadb1

root@loadb1:/home/loadb1#
```

Figura 4-10: Failover con desconexión en loadb2 asumiendo loadb1

Finalmente, al volver conectar el cable de red al nodo loadb2 el sistema volverá a estar activo, como se muestra en la Figura 4-11.

```
root@loadb1:/home/loadb1# crm status
Last updated: Wed Jul 18 16:23:26 2018         Last change: Wed Jul 18 16:18:06 2018 by
root via crm_attribute on loadb2
```

```
Stack: corosync
Current DC: loadb1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

Online: [ loadb1 loadb2 ]

Full list of resources:

 Resource Group: HAproxyGroup
     virtual-ip   (ocf::heartbeat:IPaddr):      Started loadb1
     haproxy      (ocf::heartbeat:haproxy):     Started loadb1

root@loadb1:/home/loadb1#
```

Figura 4-11: Sistema activo Online/HotStandby

En caso que el servicio se viera comprometido de alguna manera, la respuesta de este sería cambiar inmediatamente al nodo de respaldo para intercambiar el control y mantener el estado constante del servicio. Lo observado en las imágenes de prueba de failover muestran el desempeño de los nodos balanceadores configurados para responder a una configuración Online/HotStandby, configuración tal que cualquiera de los dos nodos pueda asumir el mando.

### 4.3.2 Failover con DRBD en cluster de alta disponibilidad

El sistema DRBD, en su versión 8, tal como se menciona en el capítulo anterior, permite dentro de sus configuraciones el realizar un balanceo de carga en caso que uno de los nodos tenga errores. La configuración final permitirá a la topología del clúster de alta disponibilidad trabajar en cuatro posibles casos, los que se muestran en las Figura 4-12.

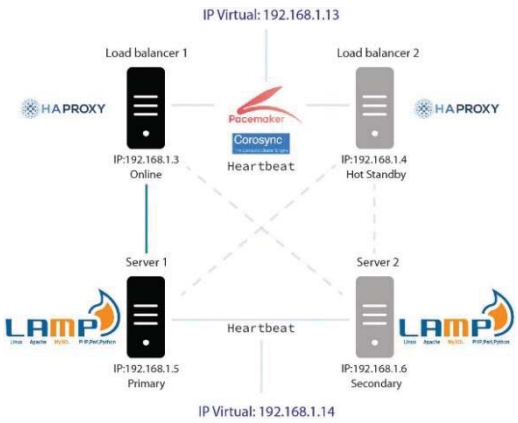


Figura 4-12: Failover caso normal

En la Figura 4-12 es posible observar el comportamiento normal del clúster de alta disponibilidad, el cual mantiene los nodos principales en ambos clúster, es decir, los nodos 192.168.1.3 para el clúster de balanceo de carga y 192.168.1.5 para el clúster de replicación de datos.

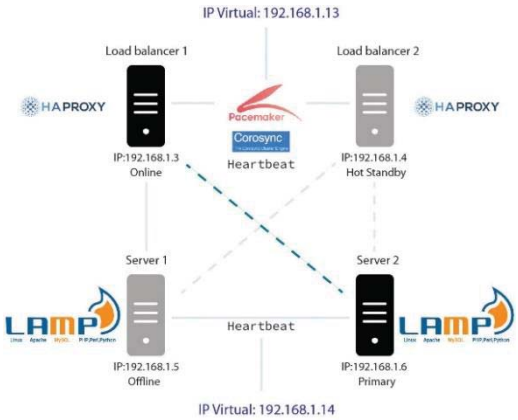


Figura 4-13: Failover falla en el servidor 1

En la Figura 4-13 se observa la falla en el Servidor 1, pero el sistema de balanceo de carga, en Loadb1, con el software HA-Proxy permite, con el algoritmo RoundRobin, que la comunicación sea retomada hacía el servidor 2 tomando milésimas de segundo en responder a las peticiones que el Servidor 1 mantenía.

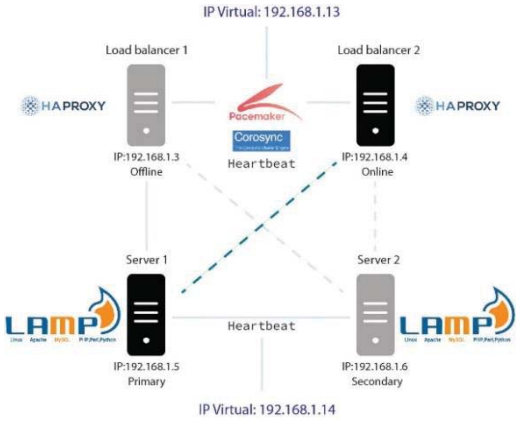


Figura 4-14: Failover en el balanceador 1

En la Figura 4-14 se observa la falla en el balanceador de carga 2 (loadb2), el cual compromete el acceso de las peticiones al servidor. El sistema de balanceo de carga de Heartbeat en los balanceadores permite que otro servidor proxy retome el control del acceso inmediatamente.

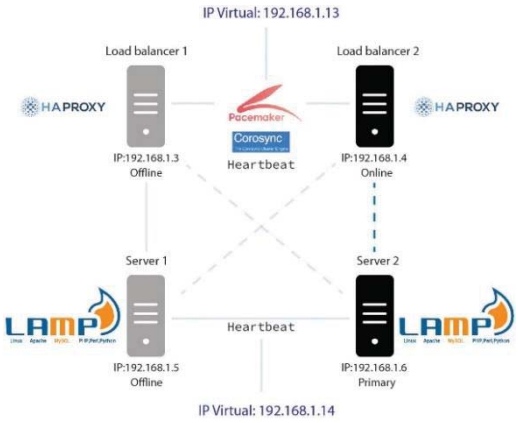


Figura 4-15: Failover en el balanceador 1 y el servidor 1

En la Figura 4-15 se observa la falla en ambos sistemas principales, comprometiendo completamente la integridad del servicio. Esta posible falla sería el peor de los casos, ya que representa una falla general pero que el clúster de alta disponibilidad está dispuesto a resolver mediante el failover configurado. En los posibles casos anteriores queda demostrado el nivel de recuperación de errores que el sistema podría resolver, funcionando como un sistema independiente en caso de que ocurra cualquier falla inesperada. La precisión de la respuesta dependerá de la cantidad de peticiones que el sistema esté manejando. En la Tabla 4-1 es posible observar el resumen de las pruebas con failover realizadas.

Tabla 4-1: Resumen de pruebas realizadas

| Formación | Loadb1 | Loadb2 | Server1 | Server2 | Detalle |
|---|---|---|---|---|---|
| Original | X | | X | | El sistema funciona de manera normal. Los nodos participantes Loadb1 y Server1 mantienen una respuesta deseada al poder responder a 20000 solicitudes en 47 ms. |
| Failover 1 | X | | | X | El sistema ha entrado en estado de error a causa del Server 1, esto ocasiona un Failover el cual es asumido por el Server 2. El sistema ahora funcionará más lento al poder responder en 9 segundos las 20000 solicitudes entrantes. |
| Failover 2 | | X | X | | El sistema ha entrado en estado de error a causa del servidor proxy Loadb1, esto ocasiona un Failover el cual es asumido por el servidor proxy Loadb2. Ya que ambos balanceadores cuentan con las mismas especificaciones de Hardware el sistema seguirá su funcionamiento normal. |
| Failover 3 | | X | | X | El sistema ha entrado en estado de error a causa del Server 1, lo cual ocasiona un Failover el cual es asumido por el Server 2. Al igual que en Failover 1, el sistema funcionará más lento al poder responder en 9 segundos las 20000 solicitudes entrantes. Por otro lado, las peticiones entrantes seguirán funcionando con el mismo rendimiento ya que ambos servidores proxys poseen el mismo Hardware, lo cual el Failover en el balanceo no ocasiona ningún cambio relativo. |

# Discusión y conclusiones

La realización de este proyecto ha concluido abarcando distintas áreas tanto como redes, protocolos, seguridad y programación. El objetivo final se ha obtenido gracias a la investigación de muchos meses de trabajo en la cual se presentó una mejora al sistema original, que sólo contaba con 3 nodos y que, al agregar un cuarto, se obtuvieron excelentes resultados. Queda también demostrado en este informe que la creación de un clúster de alta disponibilidad, en capa 4, utilizando la tecnología NUC de Intel es posible y que aporta de manera más económica y ergonómica en la creación de este tipo de proyectos. Dentro del desarrollo de este proyecto se pueden destacar varios segmentos que son de mucha utilidad, como son: La comprensión de los modelos OSI y TCP/IP, quienes contribuyen en la comunicación y protocolos que gobiernan Internet, así como la descripción de cada uno de los protocolos. Estos modelos ayudan al lector a conocer los procesos asociados a la modulación de la información que se transmitirá en un medio que, gracias a este, permite la comunicación por Internet. Además, el uso del sistema operativo Linux utilizando su terminal como línea de comando, que permite sumergirse en el mundo de las redes de manera más directa y dinámica. Esto, que además ayuda en la comprensión de los manuales de software necesarios para la construcción, como son, los softwares ligados al clúster de balanceo de carga, los que permiten conocer acerca de otras posibles configuraciones tanto en servidores proxys, como en el propio balanceo de carga. También, los softwares asociados al clúster de replicación de datos DRBD, quienes nos muestran acerca de las bases de datos, puesta en marcha de un servidor y la replicación o clonación de datos gracias al sistema distribuido. Todo lo anterior va acompañado con el material necesario para generar un balanceo de carga, el cuál es la base fundamental de todo servicio en alta disponibilidad.

Otro segmento de utilidad, son los softwares de prueba IPerf y Apache Bench, los cuales dan a conocer la respuesta del servicio dada ciertas condiciones entregadas. Estos permiten medir el servicio dentro y fuera de la red local, es decir, midiendo la información transmitida por una red LAN o WAN creando una simulación más real de las condiciones de trabajo, y cuyos límites de esta simulación provienen tanto del hardware de los nodos como la velocidad máxima que transmite la red. Gracias a este tipo de medición es posible configurar los nodos para que obtengan el máximo rendimiento dadas estas condiciones y, caso que sea necesario, aportar a la robustez del servicio añadiendo más nodos de ser necesario.

El sistema final cumple con las condiciones exigidas para este proyecto, pero puede ser mejorado aún más. Según lo anterior, es posible añadir mayor seguridad al sistema indicando, por ejemplo, un servidor DNS dentro del clúster de alta disponibilidad, permitiendo ocultar las direcciones IP asociadas a cada nodo. Así como también, la configuración de un protocolo SSL el cual permitiría reconocer el puerto proveniente de los servidores del clúster en la red. Esto daría la ventaja de poder acceder al sitio web con el prefijo HTTPS el cual le indicaría al usuario que está en un sitio seguro. Así también, es posible configurar de manera más precisa al software HA-Proxy, indicándole a este las restricciones para prevenir un posible ataque DDoS – Distributed Denial of Service- al sistema. Lo anterior podrá complementarse con una mejora en las tablas de restricciones de Firewall, creando un sistema de red más hermético potenciando la seguridad en el servicio.

Las ventajas que este proyecto presenta en la realidad son diversas. Primero, la capacidad de responder hasta en 20.000 conexiones simultáneas con KeepAlive, es decir, manteniendo cada petición como una sesión abierta en los servidores, respondiendo a estas en un tiempo de 47 ms. Segundo, la posibilidad de conocer el tráfico asociado al sitio web, ya sea por medio de archivos de registro – logs-, o por páginas de estadísticas otorgadas por Apache las cuales nos mostrarán las peticiones entrantes y salientes del servicio web. Tercero, la escalabilidad en el servicio, al contar con una aplicación la cual puede seguir creciendo y que permite añadir otros tipos de recursos y topologías de red ampliando la robustez de este. Cuarto, el bajo consumo asociado al balanceo de carga, gracias al uso de los computadores Intel NUC quienes son los actores principales del proyecto. Quinto, la seguridad provista por HA-Proxy y el firewall UFW, el que restringe o da acceso limitado a las conexiones existentes. Sexto, y final, la alta disponibilidad de un clúster, siendo esta de gran utilidad para diversos tipos de trabajo y servicios en las empresas y en la industria.

El proyecto de un cluster de alta disponibilidad con tecnología NUC de Intel puede llevarse a distintos ámbitos según sean los requerimientos del cliente. Por lo que queda dispuesto en este informe un manual que el lector podrá utilizar para la comprensión, adaptación o mejora de este mismo servicio, o para la construcción de un clúster de similares características. Promoviendo las aristas más importantes que las telecomunicaciones y redes requieren como medio para entablar comunicación, como son: La topología, los modelos de red OSI y TCP/IP y los protocolos de comunicación dentro de estos. Así también, los distintos tipos de protocolos asociados al balanceo de carga tales como DHCP, NAT (para IPv4,) y sus algoritmos. Además, de la importancia de un Firewall en la seguridad del servicio y la prueba ante errores, o Failover, que vuelve a cualquier sistema escalable en robusto. Finalmente, las definiciones de cada tipo de clúster permitiendo obtener una idea del tipo de servicio, estado de arte y aplicaciones necesarias para lograr el objetivo final.

Dadas las configuraciones mostradas en este informe, el sistema queda abierto a la posibilidad de mejorar el diseño de éste, de manera de volver al sistema más eficiente y confiable para su uso en alta disponibilidad. Además, con el aporte de los balanceadores de carga, que a su vez son servidores proxys - gracias a HA-Proxy - es posible añadir más nodos como *backend* del sistema, invitando al administrador a agregar más servicios dentro de la topología. Por ejemplo, es posible

añadir: Servidores de E-mail, servidores de video (Streaming/Multicast), servidores de fotos o servidores de foros (Blogs), entre otros. Así mismo, utilizando la configuración actual, es posible continuar este proyecto agregando una aplicación o software que requiera de una alta disponibilidad, y que además necesite un respaldo de datos en la nube. Un ejemplo de lo anterior, son: Aplicaciones web, aplicaciones para dispositivos Android/iOS, servicio de almacenamiento en la nube, etc. Sistemas que necesiten funcionar las 24 horas los 365 días del año. Esto se logra, indicando el tipo de desarrollo frontend, a nivel de aplicación, que se requiera en alta disponibilidad.

# Bibliografía

[1] Red Hat, «redhat.com,» Red Hat, 17 Noviembre 2017. [En línea]. Available: https://access.redhat.com/. [Último acceso: 28 Noviembre 2017].

[2] Blaze Meter, «blazemeter.com,» Blaze Meter, 25 Octubre 2017. [En línea]. Available: https://www.blazemeter.com/. [Último acceso: 26 Agosto 2017].

[3] Cluster Labs, «clusterlabs.org,» clusterlabs, 14 Mayo 2018. [En línea]. Available: https://wiki.clusterlabs.org/wiki/Main_Page. [Último acceso: 26 Septiembre 2018].

[4] HA Proxy, «haproxy.org,» haproxy, 26 Septiembre 2018. [En línea]. Available: http://www.haproxy.org/. [Último acceso: 2018 Septiembre 2018].

[5] Server for Hackers, «serversforhackers.com,» serverforhackers, 13 Agosto 2018. [En línea]. Available: https://serversforhackers.com/. [Último acceso: 26 Septiembre 2018].

[6] Digital Ocean, «digitalocean.com,» Digital Ocean, 13 Agosto 2018. [En línea]. Available: https://www.digitalocean.com/. [Último acceso: 26 Septiembre 2018].

[7] Linux High Availability, «linuxha.org,» Linux High Availability|, 12 Junio 2018. [En línea]. Available: www.linux-ha.org/wiki/Documentation. [Último acceso: 26 Septiembre 2018].

[8] Bull Ten, «bullten.net,» Bull Ten, 18 Abril 2018. [En línea]. Available: https://panel.bullten.net/. [Último acceso: 19 Octubre 2017].