Interacción de elementos de networking con contenedores de software (docker)

Carlos Binker[1], Hugo Tantignone[1], Guillermo Buranits[1], Rubén Darío Moreira[1], Eliseo Zurdo[1]

[1]Universidad Nacional de La Matanza, Florencio Varela 1903 (B1754JEC) -- San Justo, Buenos Aires, Argentina
{cbinker, htantignone, gburanits, rmoreira}@unlam.edu.ar; ezurdo@alumno.unlam.edu.ar

Abstract. En este trabajo se pretende mostrar la interacción de elementos de networking tales como switches de capa 2 y de capa 3 interactuando con hosts con diversos sistemas operativos, los cuales son emulados a partir de contenedores de software a través del empleo de docker. Se emularán terminales de consola linux haciendo uso del intérprete de comandos *bash*. También se empleará un servidor tomcat y un terminal de cliente gráfico Firefox construidos con docker. Se empleará la plataforma GNS3 (Graphical Simulator Network 3) Dicha plataforma albergará toda la configuración de los dispositivos enumerados anteriormente en forma virtualizada (Appliances) [1]. Después de hacer un análisis más pormenorizado del GNS3 y su máquina virtual asociada se analizará un caso de estudio en donde se simula una red corporativa que conecta varias sucursales entre sí mediante enlaces punto a punto.

Keywords: GNS3, VMware, Dynamips, Appliances, docker

1 Introducción

El enorme crecimiento registrado en el ámbito del networking, teniendo en cuenta el avance vertiginoso de Internet, ha traído aparejado un importante incremento de las estructuras de hardware. Por esta razón y dada la necesidad de reducir físicamente estas enormes estructuras es que ha nacido un nuevo paradigma denominado *Virtualización* [2]. Clásicamente si por ejemplo se requería montar una granja de servidores, era necesario contar con muchas cajas físicas en donde cada caja constituía un servidor brindando una determinada aplicación. Con la virtualización, una sola caja física podría albergar múltiples servidores *lógicos* con la consiguiente disminución de las estructuras de hardware. Pero claro, esto no resulta gratuito, se requieren procesadores con alta potencia de cómputo (varios núcleos para procesamiento paralelo) y mucha capacidad de memoria RAM, como así también gran capacidad de almacenamiento de disco rígido. De esta manera nace el concepto de *Virtual Machine*, en castellano máquina virtual (en adelante VM), es decir una estructura lógica que emula los componentes básicos del hardware (disco duro, CPU, RAM, placas de red, lector de DVD, audio,

video, puertos USB, etc.) y que ejecuta un determinado sistema operativo. La VM es en esencia software que utiliza los recursos de hardware de una máquina principal denominada *host*. Cuando se instala una VM en una máquina host, para el usuario ésta se ve como si fuera una máquina física más con las mismas características de una máquina host, en donde los recursos a asignar a la VM son totalmente configurables. La VM instalada en una máquina host se denomina comúnmente *guest,* y el sistema operativo (en adelante SO) asociado también se denomina SO *guest*.

1.1 Descripción de la tecnología docker

Los containers [3] son una abstracción en la capa de aplicación que agrupa código y dependencias juntos. Múltiples contenedores pueden ejecutarse en una misma máquina y compartir el núcleo del sistema operativo (kernel linux preferentemente aunque también puede ser Windows) con otros contenedores, además de archivos binarios y bibliotecas. Cada uno de estos contenedores se ejecuta como un proceso aislado en el espacio de usuario. Los contenedores ocupan menos espacio que las VM (las imágenes de contenedores suelen tener decenas de MB de tamaño) y comienzan su procesamiento casi de manera instantánea. El hypervisor es la interface gráfica que permite que múltiples VM se ejecuten en un solo host físico. Cada VM incluye una copia completa de un SO, una o más aplicaciones, binarios y bibliotecas necesarias, ocupando decenas de GB. Las VM suelen ser lentas para arrancar. Esto es una diferencia sustancial con los contenedores. Los contenedores y las VM utilizados conjuntamente proporcionan una gran flexibilidad en la implementación y administración de aplicaciones (ver Fig. 1).

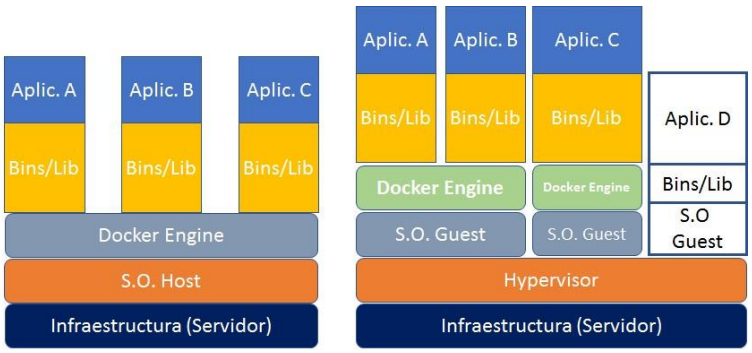

Fig. 1. Análisis comparativo entre VM y docker y combinación de VM con docker

1.2 Empleo del espacio de usuario para el lanzamiento de contenedores

Según la Fig. 2 ejecutar docker significa ejecutar dos programas en el espacio de usuario. El primero es el *docker daemon*, que se trata de un proceso que debe estar siempre ejecutándose. El segundo es el docker CLI. Este es el programa *docker engine* que interactúa con los usuarios ejecutando las aplicaciones a partir de los recursos del kernel. Si se desea iniciar, detener o instalar software, se emitirá un comando utilizando

el programa docker. La Fig. 2 también muestra tres contenedores en ejecución. Cada uno se ejecuta como un proceso hijo del demonio docker, encapsulado dentro de un contenedor, y el proceso delegado se ejecuta en su propio subespacio de memoria del espacio de usuario. Los programas que se ejecutan dentro de un contenedor pueden tener acceso solamente a su propio subespacio de memoria y acceso a los recursos según el alcance del contenedor.

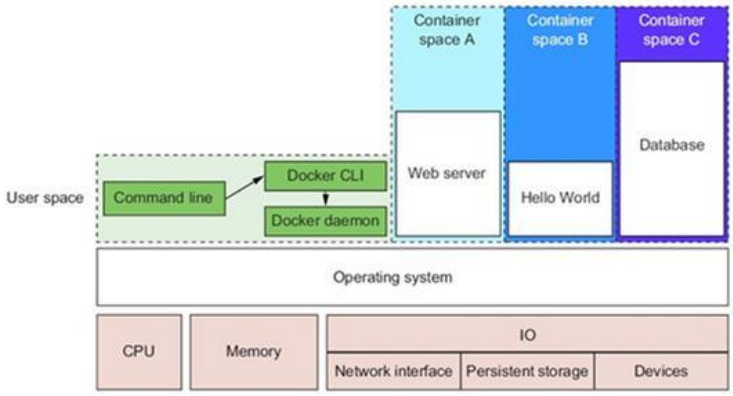

Fig. 2. Empleo del espacio de usuario para el lanzamiento de programas

1.3 Problemática de las aplicaciones con dependencias compartidas

Sin docker, una PC parece un caos en cuanto a las aplicaciones se refiere. Las aplicaciones tienen todo tipo de dependencias. Algunas dependen de bibliotecas de sistemas para aspectos tales como sonido, redes, gráficos, etc. Otras, en cambio dependen de bibliotecas estándar para el idioma en el que están escritas. Algunas dependen de otras aplicaciones, como la forma en que un programa Java depende de su versión específica. La VM o una aplicación web puede depender de una base de datos, etc. Es común que un programa requiera acceso exclusivo a algún recurso escaso tal como una red, conexión o un archivo. Sin docker, las aplicaciones se distribuyen por todo el sistema de archivos y terminan creando una confusa red de interacciones. La Fig. 3 ilustra cómo diferentes aplicaciones dependen de distintas bibliotecas comunes presentes en el sistema. Por ende, docker mantiene las cosas organizadas aislando todas estas dependencias en contenedores de software que incluyen todos los elementos necesarios, básicamente binarios y librerías, para evitar el caos de la Fig. 3. Ver Fig.4

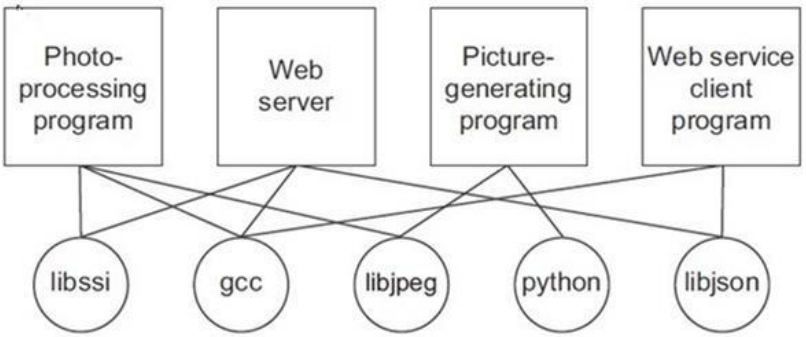

Fig. 3. Ejemplo de aplicaciones con dependencias compartidas

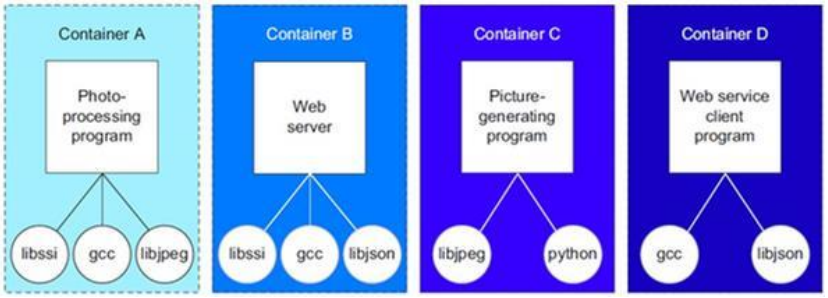

Fig. 4. Programas corriendo dentro de un contenedor con copia de sus dependencias

2 Descripción sucinta de la Plataforma de simulación GNS3

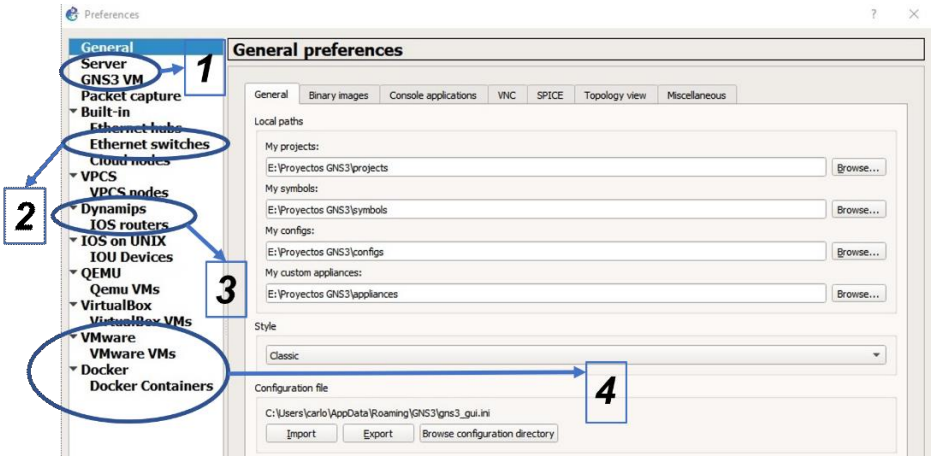

Fig. 5. Aspectos relevantes de la configuración de la plataforma GNS3

En la Fig. 5 en la configuración ***1*** (Server – GNS3 VM) se fijan la habilitación del servidor local (host donde reside el GNS3) y la habilitación de la VM que albergará

todas las configuraciones de routers, switches, VM y contenedores docker. Además se elige el tipo de Hypervisor, en este paper se empleará Vmware Workstation. En la configuración ***2*** (Ethernet Switches) se configura el switch de capa 3 (L3). Se emplea para ello un router Cisco C3745 con un módulo NM-16ESW que es un switch de 16 puertos que implementa VLAN. Este switch se emplea como switch de *core* de la topología en estudio. Como switches de acceso se emplean los Ethernet Switches incluidos en la misma plataforma. En realidad el switch L3 es un router que al agregarle el módulo mencionado lo convertimos en un switch de capa 3, para ello es necesario marcarlo como *Etherswitch router*. En la configuración ***3*** el router 3745 se simula mediante Dynamips, que permite emular el SO iOS (Internet Working Operating System) de cisco. En el C3745 se ha agregado también un módulo WIC 2T que contiene dos interfaces de WAN para establecer los enlaces punto a punto correspondientes. En consecuencia se han generado dos templates, uno como router convencional, y el otro como un switch de capa 3. Finalmente en la configuración ***4*** se configuran los containers en base a las imágenes bajadas del docker Hub [4].

3 Diseño de un caso de estudio empleando GNS3 con docker

Se hará la síntesis de la siguiente topología de red:

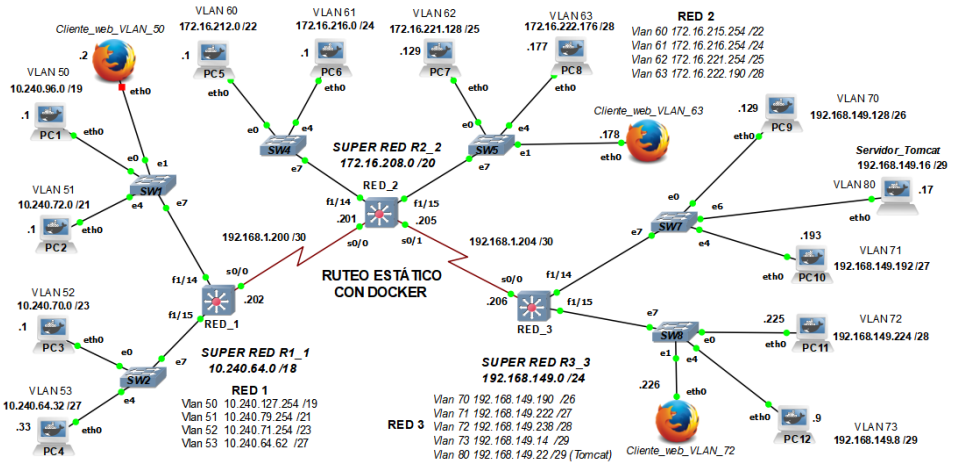

Fig. 6. Topología de red a implementar mediante GNS3

La topología consiste en tres redes principales indicadas como RED_1, RED_2 y RED_3 respectivamente. Cada una de estas redes posee cuatro vlans (a excepción de la RED_3 que incorpora la vlan 80 para el servidor Tomcat, que está implementado por docker) cuyas direcciones de red y máscaras se observan en la Fig. 6. En los switches de acceso (SW1, SW2, SW4, SW5, SW7 y SW8) se han asociado las *vlan* a los puertos del switch y se han configurado los *trunks* mediante el estándar 802.1q. Igual situación se ha hecho con los switches L3 en donde además se ha configurado el ruteo de las vlans. Además las vlans en cada red principal se han sumarizado a fin de minimizar el tamaño de las tablas de enrutamiento, resultando la superred ***10.240.64.0/18*** para la RED_1, la superred ***172.16.208.0/20*** para la red RED_2 y la superred ***192.168.149.0/24***

para la red RED_3. El ruteo global de toda la topología se lleva a cabo mediante rutas estáticas, ya que el hecho de sumarizar las vlans implica tan sólo tres sentencias de configuración en cada switch de capa 3 con la consiguiente performance que el ruteo estático otorga. A continuación se muestran las tablas correspondientes a las redes y a las superredes de la topología en cuestión.

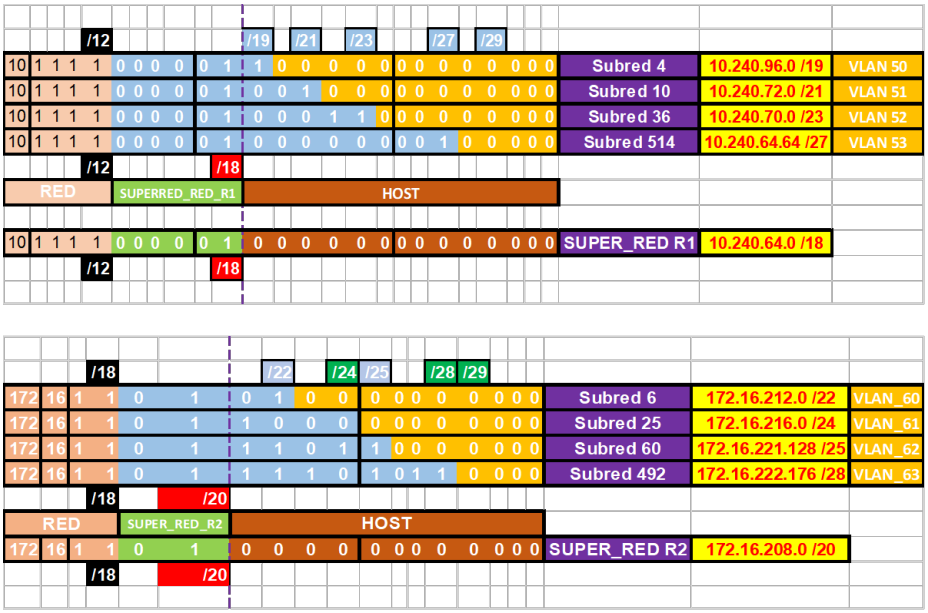

Tabla 1. Redes y superredes correspondientes a los switches RED_1 y RED_2

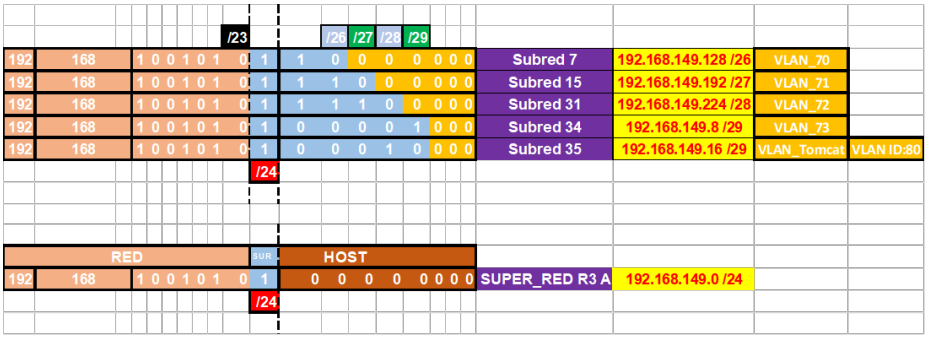

Table 2. Red y superred correspondiente al switches RED_3

4 Desarrollo de la experiencia

En este apartado se aportará la experiencia de vislumbrar la interacción del mundo del networking con las computadoras y los servidores simulados mediante *docker* . Para

ello primero haremos una síntesis preliminar de las herramientas utilizadas y finalmente presentaremos las capturas más relevantes de los resultados obtenidos.

4.1 Herramientas utilizadas

Para la simulación de la topología de red mencionada con anterioridad emplearemos los siguientes elementos de hardware y software:

- HP Notebook Pavilion AMD A12-9720P RADEON R7, 12 COMPUTE CORES 4C + 8G 2,70 GHZ con 16 GB de RAM. Windows 10 home de 64 bits
- Plataforma de software GNS3 versión 2.1.8
- VMware Workstation 14 Pro version 14.1.2 build-8497320
- 3 routers C3745 configurados como Etherswitch con una interfaz WIC-2T
- 12 containers emulando una consola bash Ubuntu server 14.04 (ubuntu:14.04)
- Servidor tomcat (imagen latest obtenida en Docker Hub https://hub.docker.com/)
- 3 containers emulando un Terminal web (browser Firefox). La imagen es gns3/webterm:latest https://hub.docker.com/r/gns3/webterm/, en Docker Hub)
- 6 switches de acceso (SW1, SW2, SW4, SW5, SW7 y SW8) embebidos en GNS3

4.2 Configuraciones de los dispositivos involucrados en la topología de red

```
gns3@gns3vm:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
tomcat              latest              df50c9d355cf        4 days ago
463 MB
ubuntu              14.04               578c3e61a98c        3 weeks ago
223 MB
gns3/webterm        latest              d504c725cf86        4 weeks ago
389 MB
gns3@gns3vm:~$ _
```

Fig. 7. Imágenes docker con sus hashes correspondientes empleadas en el caso de estudio

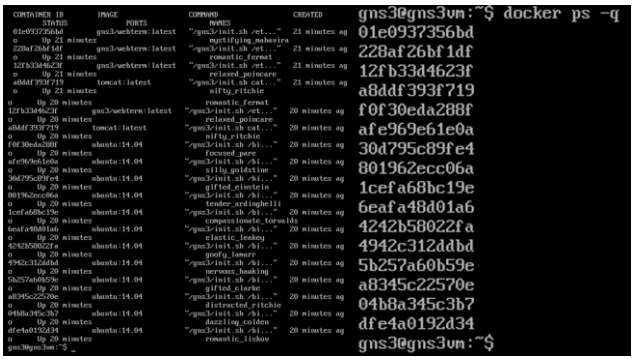

Fig. 8. Contenedores lanzados en docker. La parte izquierda se ve con ps -a y la derecha con ps -q

A fin de demostrar el correcto funcionamiento de la red, y por cuestión de espacio, ponemos la configuración del switch L3 que implementa la RED_3, tal como se ve en las Figuras 9, 10, 11 y 12 respectivamente.

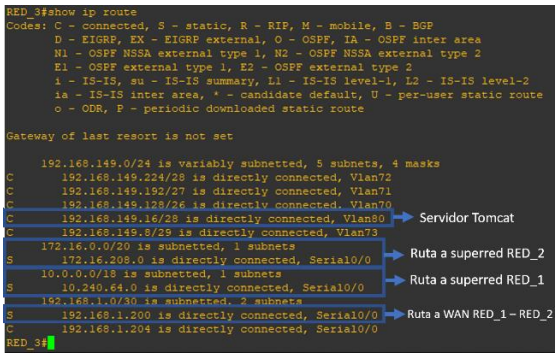

Fig. 9. Tabla de ruteo del switch L3 RED_3

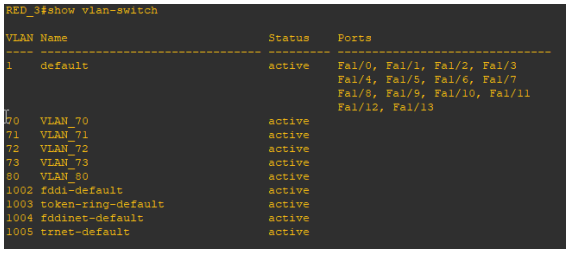

Fig. 10. Configuración de vlans en el switch L3 RED_3

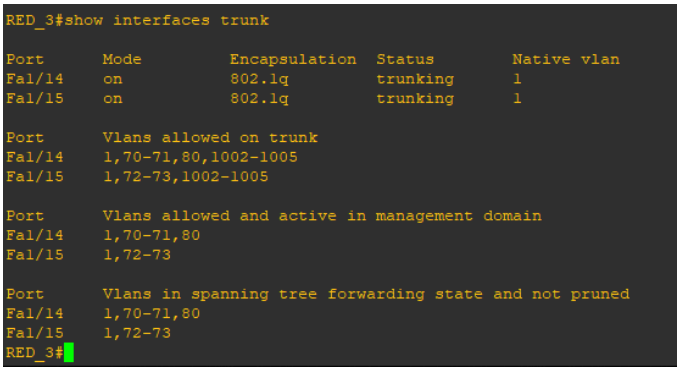

Fig. 11. Observación de los puertos de trunk en el switch L3 que implementa RED_3

```
interface FastEthernet1/14
 switchport trunk allowed vlan 1,70,71,80,1002-1005
 switchport mode trunk
!
interface FastEthernet1/15
 switchport trunk allowed vlan 1,72,73,1002-1005
 switchport mode trunk
!
interface Vlan1
 no ip address
!
interface Vlan70
 ip address 192.168.149.190 255.255.255.192
!
interface Vlan71
 ip address 192.168.149.222 255.255.255.224
!
interface Vlan72
 ip address 192.168.149.238 255.255.255.240
!
interface Vlan73
 ip address 192.168.149.14 255.255.255.248
!
interface Vlan80
 ip address 192.168.149.22 255.255.255.240
!
!
ip forward-protocol nd
ip route 10.240.64.0 255.255.192.0 Serial0/0
ip route 172.16.208.0 255.255.240.0 Serial0/0
ip route 192.168.1.200 255.255.255.252 Serial0/0
```

Fig. 12. Direcciones de red de vlans y rutas estáticas a las superredes y WAN RED_1 – RED_2

4.3 Resultados alcanzados

A continuación, mostraremos tres capturas realizadas desde cada container en cada RED enviando ping hacia los otros containers en otras redes. Finalmente mostraremos la captura desde un terminal web hacia el servidor tomcat. Con esto se corrobora el funcionamiento del laboratorio.

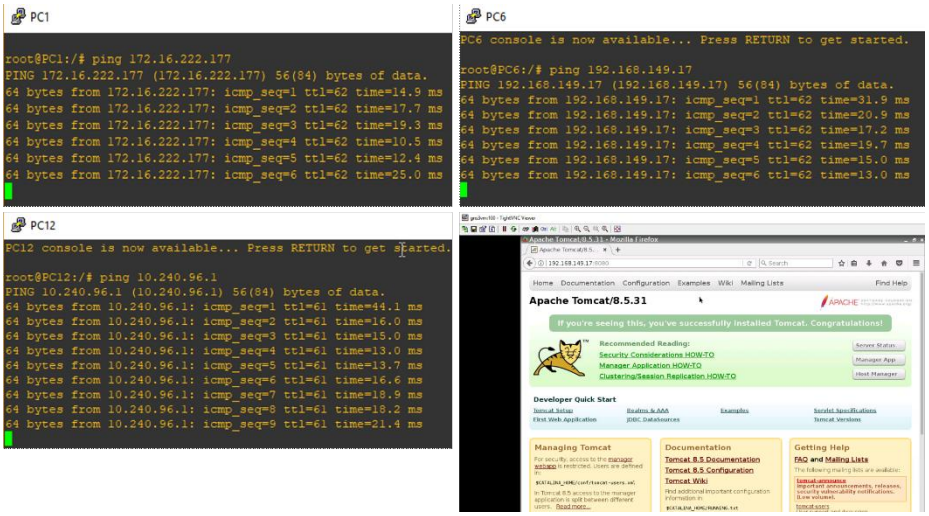

Fig. 13. Prueba de ping entre diferentes contenedores y desde un terminal web al *TOMCAT*

5 Conclusiones y trabajo futuro

1. Ahora en GNS3 es posible conectar a la infraestructura de networking VMs más contenedores de software. Los containers a diferencia de las VMs se ejecutan muy rápido y ocupan algunos MB frente a los GB de las VMs.
2. En una futura investigación se estudiará docker-machine. Esto permite instalar en una VM un conjunto de VMs, donde en cada una de ellas se instala docker y pueden ejecutarse aplicaciones distribuidas entre estas VMs además de administrarse a todas estas VMs de manera centralizada.
3. En relación con docker-machine y a GNS3 se trabajará en un futuro con redes en IPv6, dando de esta manera conectividad dual stack.
4. Los requerimientos de hardware al trabajar con contenedores de software resultan más flexibles que al trabajar con máquinas virtuales, en donde es necesario gran capacidad de RAM, de disco y de procesamiento.

6 Referencias

1. Appliances. Emulación de elementos de networking interactuando con máquinas virtuales. Carlos Binker. CACIC 2016.
2. Virtualization Essentials 2nd Edition. Matthew Portnoy. Publicación: 29 de agosto de 2016. Editorial Sybex. ISBN 10: 1119267722.
3. Virtual Machines: Versatile Platforms for Systems and Processes (The Morgan Kaufmann Series in Computer Architecture and Design) 1st Edition. Publicación:17/062005. Editorial: Morgan Kaufmann. ISBN 10: 1558609105.
4. Docker Up and Running. Karl Matthias & Sean P. Kane. Publicación: 03/07/2015. Editorial: O'Reilly Media. ISBN 10: 1491917571. Docker in Action. Jeff Nickoloff. 1st Edition. Publicación: 27/03/16. Editorial: Manning Publications. ISBN 10: 1633430235