Aplicación de técnicas de virtualización ligera para la evaluación de redes de comunicaciones

Trabajo Final de Estudios
Ingeniería Telemática

Enrique Fernández Sánchez
Universidad Politécnica de Cartagena

Revisión 28 Abril 2021

Índice

1 Introduction

2 *Espacio de nombres* en *Linux*

2.1 ¿Qué es un *espacio de nombres*?

Los *espacios de nombres*, o también llamados, *namespaces*, son una característica del kernel de Linux que permite gestionar los recursos del kernel, pudiendo limitarlos a un proceso o grupo de procesos. Suponen una base de tecnología que aparece en las técnicas de virtualización más modernas (como puede ser Docker, Kubernetes, etc). A un nivel alto, permiten aislar procesos respecto al resto del kernel.

El objetivo de cada *namespaces* es adquirir una característica global del sistema como una abstracción que haga parecer a los procesos de dentro del *namespace* que tienen su propia instancia aislada del recurso global.

2.2 ¿Cúantos *namespaces* hay?

El kernel ha estado en contaste evolución desde que 1991, cuando Linus Torvalds comenzó el proyecto, actualmente sigue muy activo y se siguen añadiendo nuevas características. El origen de los namespaces se remonta a la versión del kernel 2.4.19, lanzada en 2002. Conforme fueron pasando los años, más tipos diferentes de namespaces se fueron añadiendo a Linux. El concepto de *User namespaces*, se consideró terminado con la versión 3.9.

Actualmente, tenemos 8 tipos diferentes de namespaces, siendo el último añadido en la versión 5.8 (lanzada el 2 de Agosto de 2020).

1. UTS (hostname)
2. Mount (mnt)
3. Process ID (pid)
4. Network (net)
5. Interprocess Communication (ipc)
6. User ID (user)
7. Control group (cgroup)
8. Time

2.2.1 UTS namespace

El tipo más sencillo de todos los namespaces. La funcionalidad consiste en controlar el hostname asociado del ordenador, en este caso, del proceso o procesos asignados al namespace. Existen tres diferentes rutinas que nos permiten obtener y modifcar el hostname:

- *sethostname()*
- *setdomainname()*
- *uname()*

En una situación normal sin namespaces, se modificaría una String global, sin embargo, si estamos dentro de un namespace, los procesos asociados tienen su propia variable global asignada.

Un ejemplo muy basico de uso de este namespaces podría ser el siguiente:

Listing 1: Example usage UTS namespace

```
$ sudo su                          # super user
$ hostname                         # current hostname
> arch-linux
$ unshare -u /bin/sh               # shell with UTS namespace
$ hostname new-hostname            # set hostname
$ hostname                         # check hostname of the shell
> new-hostname
$ exit                             # exit shell and namespace
$ hostname                         # original hostname
> arch-linux
```

En el ejemplo planteado, vemos que utilizamos el comando `unshare`. Utilizando la documentación de dicho comando, `man unshare`. Podemos deducir los siguiente:

- Ejecuta un programa con algunos namespaces diferentes del host.
- En los parametros podemos especificar cual o cuales namespaces queremos desvincular.
- Tenemos que especificar la ruta del ejecutable que queremos aislar
- La sintaxis sería tal que: `unshare [options] <program>[<argument>...]`

2.2.2 Mount namespace

Un *mount namespace (mnt)* supone otro tipo de espacio de nombres, en este caso relacionado con los *mounts* de nuestro sistema. Lo primero es entender a que nos referimos cuando hablamos de *mount*. *Mount*, o montaje, hace referencia a conectar un sistema de archivos adicional que sea accesible para el sistema de archivos actual de un ordenador. Un mount, tiene asignado lo que se llama *mount point*, que corresponde con el directorio en el que está accesible el sistema de archivo que previamente hemos montado.

Por lo tanto, un namespace de tipo *mount* nos permite modificar un sistema de archivos en concreto, sin que el host pueda ver y/o acceder a dicho sistema de archivos. Un ejemplo básico de esta funcionalidad podría ser la siguiente:

Listing 2: Example of usage mount namespace

```
$ sudo su                    # run a shell in a new mount namespace
$ unshare -m /bin/sh
$ mount --bind /usr/bin/ /mnt/
$ ls /mnt/cp
> /mnt/cp
$ exit                       # exit the shell and close namespace
$ ls /mnt/cp
> ls: cannot access '/mnt/cp': No such file or directory
```

Como vemos en el ejemplo, dentro del namespaces lo que hacemos es crear un *mount* de tipo *bind*, que tiene por función que un archivo de la máquina host se monte en un directorio en específico, en este caso, un directorio unicamente del programa que hemos asignado al namespace. Otro ejemplo de uso de estos namespaces es crear un sistema de archivos temporal que solo sea visible para ese proceso.

2.2.3 Process ID namespace

Para entender en que consiste este namespace, primero tenemos que conocer la definición de *process id* dentro del Kernel. En este caso, *process id* hace referencia a un número entero que utiliza el Kernel para identificar los procesos de manera unívoca.

Concretando, aísla el namespace de la ID del proceso asignado, dando lugar a que, por ejemplo, otros namespaces puedan tener el mismo PID. Esto nos lleva a la situación de que un proceso dentro de un *PID namespace* piense que tiene asignado el ID "1", mientras que en la realidad (en la máquina host) tiene otro ID asignado.

Listing 3: Example of usage process id namespace

```
$ echo $$                  # PID de la shell
$ ls -l /proc/$$/ns        # ID espacios de nombres
$ sudo unshare -f --mount-proc -p /bin/sh
$ echo $$                  # PID de la shell dentro del ns
$ ls -l /proc/$$/ns        # nuevos ID espacio de nombres
$ ps

$ ps -ef                   # ejecutar en una shell fuera del ns. Comparar PID
$ exit
```

Si ejecutamos el ejemplo, lo que podemos comprobar es que el ID del proceso que está dentro del namespaces (`echo $$`), no coincide con el proceso que podemos ver de la máquina host (`ps -ef | grep /bin/sh`)). Más concretamente, el primer proceso creado en un PID namespace recibirá el pid número 1, y además de un tratamiento especial ya que supone un `init process` dentro de ese namespace.

2.2.4 Network namespace

8 Glosario de términos

- Namespaces. *Espacio de nombres*
- Linux. *Sistema operativo tipo UNIX, de codigo abierto, multiplataforma, multiusuario y multitarea.*
- Kernel de linux. *Núcleo del sistema operativo Linux.*
- PID. *Process Identifier*

9 Bibliografía

9.1 Enlaces y referencias

1. *Namespaces*
2. Tutorial: Espacio de nombres en Linux
3. *Time namespaces coming to linux*
4. *Container is a lie. Namespaces*
5. *Namespaces.* Uso de *cgroups.*
6. *Introduction to Network Namespaces*
7. *Build a container by hand: the mount namespace*
8. Indentificador de procesos (*process id*)
9. *Linux PID namespaces work with containers*
10. Fundamentos de Docker