



Construcción de la red Ad-hoc

Sistemas Complejos

Grupo 1

Sebastián Florido

Carlos Rojas

Luis Rodriguez

Presentado a:

Joaquín Sanchez

Junio 3 de 2024

Bogotá D.C, Colombia

Requerimientos:

1. Para poder realizar una conexión mediante red Ad-hoc, usaremos el protocolo Batman-adv, para esto vamos a usar el sistema operativo ubuntu linux, y ejecutaremos lo siguientes comandos en la terminal para instalar el protocolo:

Para instalar Batman-adv

```
sudo apt update
```

```
sudo apt upgrade -y
```

```
sudo apt install batctl bridge-utils -y
```

```
sudo modprobe batman-adv
```

```
lsmod | grep batman_adv
```

Primero debe de asegurarse de que estén actualizados todos los repositorios de ubuntu con los primeros 2 comandos, posteriormente los dos siguientes comandos es para instalar el protocolo y el último al ejecutarlo nos debe aparecer lo siguiente en la terminal:

```
luis@luis-nodo:~$ lsmod | grep batman_adv
batman_adv          266240  0
libcrc32c           12288  4 nf_conntrack,nf_nat,batman_adv,nf_tables
bridge              413696  2 batman_adv,br_netfilter
```

Al ver que batman_adv esta de color rojo, significa que hemos instalado correctamente el protocolo.

2. Una vez hecho la instalación del protocolo, procederemos a hacer un script que nos permita realizar la conexión de los computadores.

Script para la construcción de la red Ad-hoc

```
1 #!/bin/bash
2
3 # Nombre de la red ad-hoc y dirección MAC de ejemplo
4 SSID="TLON-ADHOC1"
5 MAC_AP="02:1B:55:11:22:33"
6
7 # Detener NetworkManager para evitar conflictos
8 sudo systemctl stop NetworkManager
9
10 # Cargar el módulo batman-adv
11 sudo modprobe batman-adv
12
13 # Bajar la interfaz inalámbrica, configurar en modo ad-hoc y subirla
14 sudo ifconfig wlp0s20f3 down
15 sleep 2 # Pequeña pausa para asegurar que la interfaz está completamente baja
16 sudo iwconfig wlp0s20f3 mode ad-hoc
17 sudo iwconfig wlp0s2f03 essid $SSID ap $MAC_AP
18 sudo ifconfig wlp0s2f03 mtu 1532
19 sudo ifconfig wlp0s2f03 up
20
21 # Asignar una dirección IP estática
22 sudo ifconfig wlp0s2f03 192.168.1.2/24 up
23
24
25 # Agregar la interfaz al batman-adv
26 #sudo batctl if add wlp1s0
27 #sudo ifconfig bat0 up
28
29 # Reiniciar NetworkManager si es necesario
30 #sudo systemctl start NetworkManager
31
32 echo "Configuración de red ad-hoc con Batman-adv completada."
```

El siguiente script nos realiza una serie de configuraciones que pondrá el ordenador en modo Ad-hoc, la línea 4 y 5 lo que hacen es establecer el nombre de la red y la dirección Mac de la red, después se detendrá los servicios de red, o sea que el wifi se desactiva para poner el ordenador en modo Ad-hoc.

Lo más importante a recalcar es el módulo de red, esto se mira en la terminal mediante el comando `ifconfig` o `iw dev`, se verá de la siguiente manera:

```
luis@luis-nodo:~$ iw dev
phy#0
    Interface wlo1
        ifindex 3
        wdev 0x1
        addr 40:9f:38:ce:91:e5
        ssid Familia Rodriguez_plus
        type managed
        channel 11 (2462 MHz), width: 20 MHz, center1: 2462 MHz
        txpower 20.00 dBm
        multicast TXQ:
```

Se debe de ver la parte que dice “Interface” ya que esa es la que debemos reemplazar en el script para que el módulo de batman sea cargado correctamente, en este ejemplo el script tiene la interface “wlp0s2f3” y esta debe ser cambiada por la interfaz de la máquina “wlo1” en el script.

En la asignación de direcciones ip estáticas, se debe agregar una dirección única para cada equipo, pero estas no deben ser iguales, ejemplo si un ordenador tiene la dirección ip 192.168.1.1, otro ordenador debe de tener la dirección 192.168.1.2.

Ejecución del script

Para poder ejecutar el script, se debe de otorgar permisos al archivo, entonces si el archivo se llama “network.sh”(La extensión debe ser .sh) el comando para darle permisos debe de ser “**chmod +x network.sh**” con esto se le otorga el permiso de ejecución.

```
luis@master:~/Ad Hoc$ chmod +x network_setup_script.sh
```

Para ejecutarlo se usa el siguiente comando: “**sudo ./network_setup_script.sh**”

```
luis@master:~/Ad Hoc$ sudo ./network_setup_script.sh
```

Nota: el script al ejecutar el comando “**Sudo systemctl stop NetworkManager**” la conexión a la red wifi se apagará y se pondrá en modo ad-hoc, como se ve de la siguiente manera:

```
lo          no wireless extensions.
eno1        no wireless extensions.
wlo1        IEEE 802.11  ESSID:"TLON-ADHOC1"
            Mode:Ad-Hoc  Frequency:2.412 GHz  Cell: 02:1B:55:11:22:33
            Tx-Power=20 dBm
            Retry short limit:7   RTS thr:off   Fragment thr:off
            Power Management:on
docker0     no wireless extensions.
docker_gwbridge  no wireless extensions.
```

Se aprecia que en la interfaz “wlo1” se ha cambiado a modo Ad-hoc. En caso de que necesite volver a activar el wifi de la máquina, debe ejecutar el comando comentado en el script

previamente mencionado, el “**sudo systemctl start NetworkManager**” y los parámetros volverán normales.

```
luis@master:~/Ad Hoc$ iw dev
phy#0
    Interface wlo1
        ifindex 3
        wdev 0x1
        addr 40:9f:38:ce:91:e5
        ssid iPhone de luis (3)
        type managed
        channel 6 (2437 MHz), width: 20 MHz, center1: 2437 MHz
        txpower 20.00 dBm
        multicast TXQ:
            qsz-byt qsz-pkt flows    drops    marks    overlmt hashcolt
x-bytes tx-packets
103          62          0          0        32          0          0          0          0          8
```

Si se ejecutó de manera correcta el script, y ya hay más de un nodo conectado, realice una prueba de comunicación mediante el comando ping seguido de la dirección estática que proporcionó al otro ordenador.

```
luis@master:~/Ad Hoc$ ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=3.92 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=2.03 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=1.46 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=1.73 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=0.987 ms
64 bytes from 192.168.1.2: icmp_seq=6 ttl=64 time=6.89 ms
64 bytes from 192.168.1.2: icmp_seq=7 ttl=64 time=1.29 ms
64 bytes from 192.168.1.2: icmp_seq=8 ttl=64 time=2.21 ms
64 bytes from 192.168.1.2: icmp_seq=9 ttl=64 time=0.748 ms
64 bytes from 192.168.1.2: icmp_seq=10 ttl=64 time=1.44 ms
64 bytes from 192.168.1.2: icmp_seq=11 ttl=64 time=1.97 ms
64 bytes from 192.168.1.2: icmp_seq=12 ttl=64 time=0.909 ms
64 bytes from 192.168.1.2: icmp_seq=13 ttl=64 time=0.962 ms
64 bytes from 192.168.1.2: icmp_seq=14 ttl=64 time=0.731 ms
64 bytes from 192.168.1.2: icmp_seq=15 ttl=64 time=3.62 ms
64 bytes from 192.168.1.2: icmp_seq=16 ttl=64 time=0.806 ms
64 bytes from 192.168.1.2: icmp_seq=17 ttl=64 time=0.949 ms
64 bytes from 192.168.1.2: icmp_seq=18 ttl=64 time=2.05 ms
64 bytes from 192.168.1.2: icmp_seq=19 ttl=64 time=7.86 ms
64 bytes from 192.168.1.2: icmp_seq=20 ttl=64 time=0.688 ms
64 bytes from 192.168.1.2: icmp_seq=21 ttl=64 time=1.66 ms
```

Ejecución del programa en los ordenadores

Nota: Se debe de tener instalados python y todas las librerías correspondientes para correr este ejemplo, también numpy, sockets y algún otra librería requerida.

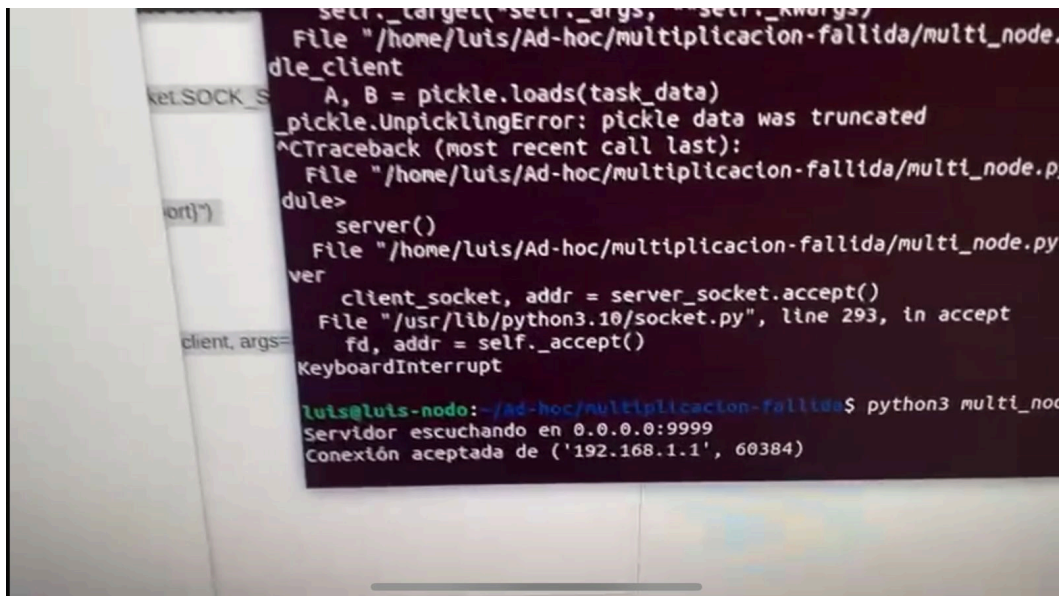
1. Debe descargar los dos scripts y repartirlos, el script de nombre “**principal_multi.py**” debe de estar en una sola máquina, este script se encargará de generar la matriz correspondiente a multiplicar y repartir el trabajo a los demás nodos. Este script solo

se ejecutará cuando se haya ejecutado el script de nombre “**multi_node.py**” en los otros nodos.

2. Al ejecutar el programa `multi_node.py` en los otros ordenadores, estos estarán en modo espera, se espera que al ejecutar el programa principal `multi.py` este genere la matriz y reparta a los otros nodos.

```
luis@master:~/Ad Hoc/Programas$ python3 multi_node.py
Servidor escuchando en 0.0.0.0:9999
```

3. Al conectarse de manera correcta, el programa empieza a hacer los cálculos correspondientes, los nodos que están en modo espera, aceptarán la conexión como se muestra a continuación.



```
setl._target(*setl._args, **setl._kwargs)
File "/home/luis/Ad-hoc/multiplicacion-fallida/multi_node.py", line 10, in <module>
    client_socket, addr = server_socket.accept()
_pickle.UnpicklingError: pickle data was truncated
^CTraceback (most recent call last):
  File "/home/luis/Ad-hoc/multiplicacion-fallida/multi_node.py", line 10, in <module>
    client_socket, addr = server_socket.accept()
KeyboardInterrupt

luis@luis-node:~/Ad-hoc/multiplicacion-fallida$ python3 multi_node.py
Servidor escuchando en 0.0.0.0:9999
Conexión aceptada de ('192.168.1.1', 60384)
```

4. Al aceptar la conexión prontamente en la máquina donde se ejecuto el principal, se verán reflejados los resultados:

```
luis@master:~/Ad Hoc$ python3 multi.py
Resultado de la multiplicación de matrices:
[[10822. 11237. 10979. ... 10505. 10529. 10499.]
 [ 9747. 10265.  9734. ...  9468.  9488.  9599.]
 [10633. 11138. 10580. ... 10069. 10194. 10254.]
 ...
 [10026. 10365.  9984. ...  9589. 10067.  9771.]
 [ 9374. 10013.  9679. ...  9166.  9210.  9338.]
 [10348. 10910. 10385. ... 10179. 10132. 10523.]]
Tiempo total de cálculo: 48.4419424533844 segundos
```

Caso de desconexión de los nodos

En dado caso uno o más nodos hayan sido desconectados, el programa automáticamente realizará una reasignación de tareas a los nodos que estén en funcionamiento, de esta manera, se garantiza la finalización de la tarea a realizar y no se detiene el servicio.

```
luis@master:~/Ad Hoc$ python3 multi.py
Error al comunicarse con 192.168.1.3: [Errno 111] Connection refused
Error al comunicarse con 192.168.1.2: [Errno 111] Connection refused
Resultado de la multiplicación de matrices:
[[ 9993.  9855.  9771. ... 10405. 10318. 10292.]
 [10048.  9657.  9869. ... 10533. 10403. 10371.]
 [ 9643.  9907.  9588. ... 10357. 10173. 10391.]
 ...
 [ 9499.  9369.  9900. ... 10032. 10078. 10454.]
 [10287.  9999.  9927. ... 10233. 10163. 10563.]
 [ 9122.  9421.  9119. ...  9550.  9702.  9444.]]
Tiempo total de cálculo: 84.76405835151672 segundos
```

Como se puede apreciar, no hubo respuesta de dos nodos, por lo tanto se reasigno todo el trabajo al principal y otro nodo que tenía disponibilidad para no detener el servicio, se tardo en tiempo un poco más pero terminó la tarea.