

FACULTAD DE INGENIERÍA  
UNIVERSIDAD DE BUENOS AIRES

DEPARTAMENTO DE COMPUTACIÓN

INTRODUCCIÓN A LOS SISTEMAS DISTRIBUIDOS - 75.43

---

**Trabajo Práctico 1 - App Layer**

---

Apellido y Nombre	Padrón	Correo electrónico
Capolupo, Mauro	90283	capolupo.mauro@gmail.com
Franco, Tomas	91013	tomasnfranco@gmailcom
Impaglione, Rocio	94178	rimpaglione@fi.uba.ar

2do Cuatrimestre 2020

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Hipótesis y supuestos</b>	<b>2</b>
<b>3. Implementación</b>	<b>2</b>
3.1. Diseño de la solución . . . . .	2
<b>4. Uso de la Aplicación</b>	<b>3</b>
4.1. General . . . . .	3
4.1.1. Sobre el servidor . . . . .	3
4.1.2. Menú de ayuda . . . . .	4
4.1.3. Nivel de logueo en cliente . . . . .	4
4.2. Ping Directo . . . . .	5
4.3. Ping Reverso . . . . .	6
4.4. Ping Proxy . . . . .	7
<b>5. Preguntas a responder</b>	<b>9</b>
<b>6. Dificultades encontradas</b>	<b>12</b>
<b>7. Conclusión</b>	<b>12</b>

# 1. Introducción

El objetivo del presente trabajo es que los estudiantes se familiaricen a nivel práctico con los contenidos teóricos presentados en la materia: Capa de aplicación y sus protocolos, la arquitectura cliente-servidor, los protocolos de transporte (TCP y UDP), cuáles son sus diferencias y en qué casos conviene utilizar cada uno. También, que aprendan el manejo de herramientas de redes que van a utilizar a lo largo del cuatrimestre.

## 2. Hipótesis y supuestos

- El tiempo que se mide al final del ping es el tiempo de ejecución total del comando.
- Cuando no se ingresa un parámetro count que controle la cantidad de pings a hacer, elegimos resolverlo del mismo modo que el ping original, es decir, haciendo *ping* de manera indefinida hasta que el usuario decida detener la ejecución.
- El usuario deberá especificar el número de puerto del servidor al que desea conectarse en el ping proxy.

## 3. Implementación

La decisión más importante a tomar y a partir de la cual pudimos comenzar a tomar acción, fue definir los protocolos a utilizar para cada una de las operaciones de ping. Sin esta elección, no podíamos comenzar a programar nuestra aplicación. Para ello, investigamos sobre las ventajas y desventajas de utilizar TCP o UDP, y elegimos un protocolo de acuerdo a las necesidades de cada operación a implementar.

Los fundamentos sobre la elección de los protocolos se encuentran desarrollados como respuesta a la pregunta 6) del cuestionario, por lo que aquí explicaremos brevemente cómo implementamos cada operación.

### 3.1. Diseño de la solución

- Ping directo: Antes de comenzar a enviar mensajes ping, el cliente envía un mensaje al servidor estableciendo el tipo de mensaje que enviará. Cuando el servidor le conteste con un mensaje de respuesta, comenzará a transmitir.

En cada iteración de ping, el cliente envía un mensaje al servidor y se guarda la hora a la que fue enviado (send\_time).

Si el servidor recibe el paquete, envía un mensaje de respuesta. El cliente recibe el mensaje y guarda la hora a la que fue recibido para realizar los cálculos de rtt.

Luego, ese rtt se almacena en una lista, que luego será utilizada para realizar los cálculos al final de la ejecución.

Si el servidor no recibe nada, no hace nada y el paquete se considera perdido cuando se alcanza el tiempo de timeout.

- Ping reverso: Para el caso del ping reverso, el cliente inicia una conexión con el servidor y envía un mensaje inicial pidiéndole que comience a mandarle mensajes ping. Este mensaje contiene el código de operación y la cantidad de veces que el usuario desea efectuar la operación.

Cuando el servidor recibe el mensaje y lo identifica con el código de operación, comenzará a realizar el envío de mensajes ping al cliente.

En cada iteración de ping, el servidor enviará un mensaje al cliente y se guardará la hora a la que fue enviado.

Luego, cuando reciba la confirmación de que el paquete llegó, guardará ese tiempo y calculará el RTT haciendo la diferencia entre la hora de envío y la de recepción de la confirmación.

Además, una vez confirmada la recepción, mandará un mensaje al cliente del tipo response”(con código de operación 11) que tendrá el siguiente formato:

1. El código de operación
2. Tamaño del paquete enviado
3. Sequence number
4. RTT

Cuando el cliente reciba este mensaje, lo parseará para tomar los datos que necesita y mostrarlos por pantalla.

Además, almacenará el RTT de la misma forma que en el ping directo para realizar los cálculos posteriores.

- Ping proxy: Para este último caso, el cliente inicia una conexión con el servidor y envía un mensaje inicial pidiéndole que comience a mandarle mensajes ping a una dirección que debe tener un servidor levantado en este puerto. Este mensaje contiene el código de operación (10), la cantidad de veces que el usuario desea efectuar la operación (count), el host destino y el puerto destino.

Cuando el servidor recibe el mensaje y lo identifica con el código de operación, comenzará a realizar el envío de mensajes ping al otro servidor.

En cada iteración de ping, el servidor enviará un mensaje al servidor y se guardará la hora a la que fue enviado.

Luego, cuando reciba la confirmación de que el paquete llegó, guardará ese tiempo y calculará el RTT haciendo la diferencia entre la hora de envío y la de recepción de la confirmación.

Además, una vez confirmada la recepción, mandará un mensaje al cliente del tipo response”(con código de operación 11) que tendrá el siguiente formato:

1. El código de operación
2. Tamaño del paquete enviado
3. Sequence number
4. RTT

Cuando el cliente reciba este mensaje, lo parseará para tomar los datos que necesita y mostrarlos por pantalla.

Además, almacenará el RTT al igual que las operaciones anteriores para poder realizar los cálculos necesarios.

## 4. Uso de la Aplicación

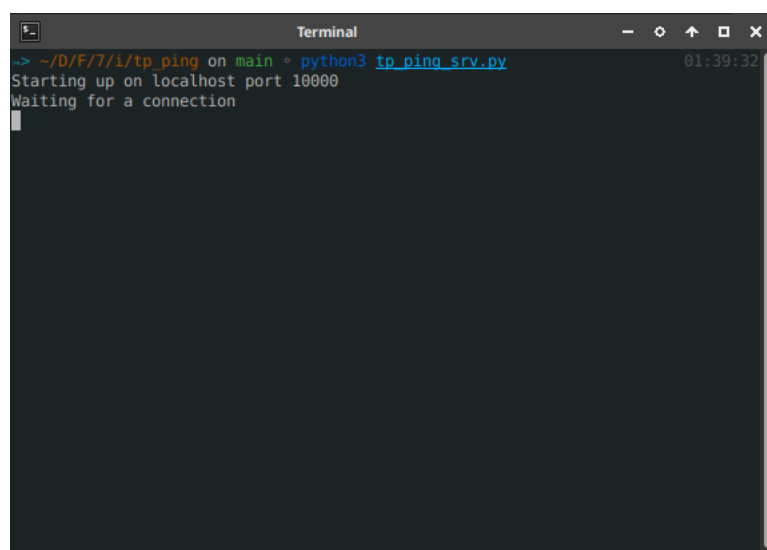
En el siguiente apartado vamos a presentar capturas de pantalla de varios casos de uso de nuestra aplicación.

### 4.1. General

#### 4.1.1. Sobre el servidor

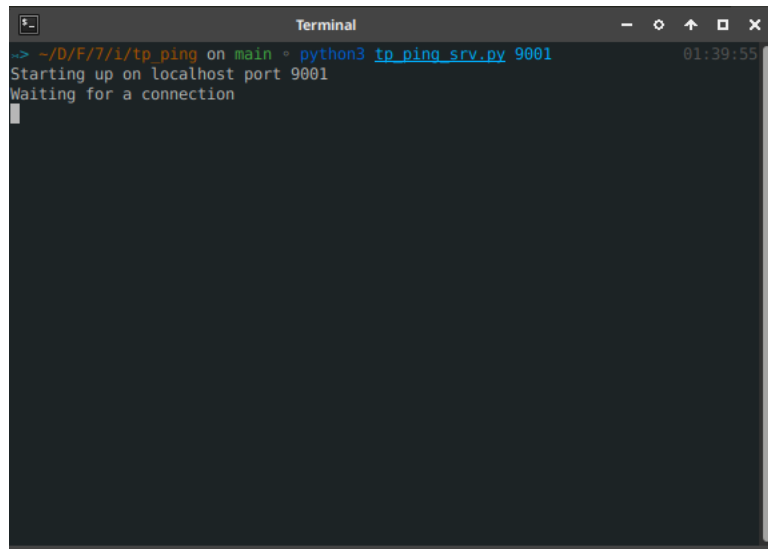
El servidor se iniciará por defecto en el puerto 10000 utilizando el comando

```
$ python tp_ping_srv.py
```



Si el usuario lo desea, también puede iniciarlo en un puerto de su elección ingresando el número de puerto como segundo parámetro:

```
$ python tp_ping_srv.py 9800
```

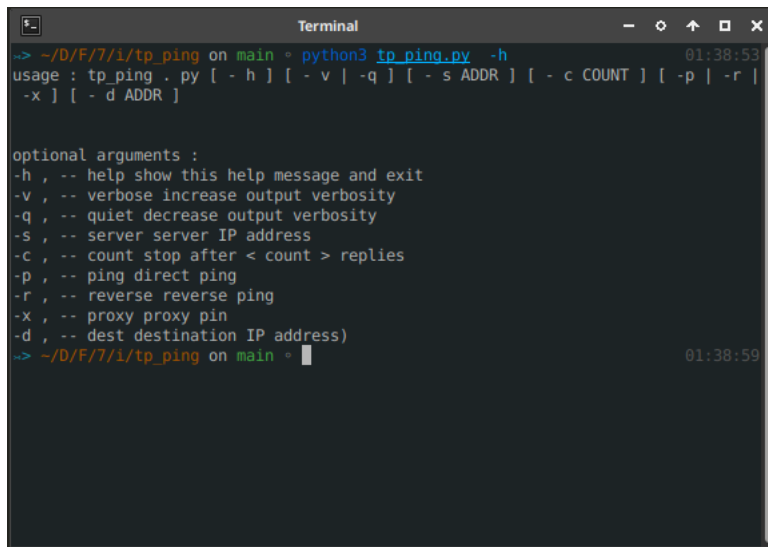


```
Terminal
-> ~/D/F/7/i/tp_ping on main = python3 tp_ping_srv.py 9801 01:39:55
Starting up on localhost port 9801
Waiting for a connection
```

#### 4.1.2. Menú de ayuda

El usuario puede consultar las opciones disponibles de la aplicación a través del comando

```
$ python tp_ping.py -h
```



```
Terminal
-> ~/D/F/7/i/tp_ping on main = python3 tp_ping.py -h 01:38:53
usage : tp_ping . py [ - h ] [ - v | -q ] [ - s ADDR ] [ - c COUNT ] [ -p | -r |
-x ] [ - d ADDR ]

optional arguments :
-h , -- help show this help message and exit
-v , -- verbose increase output verbosity
-q , -- quiet decrease output verbosity
-s , -- server server IP address
-c , -- count stop after < count > replies
-p , -- ping direct ping
-r , -- reverse reverse ping
-x , -- proxy proxy pin
-d , -- dest destination IP address
-> ~/D/F/7/i/tp_ping on main = 01:38:59
```

#### 4.1.3. Nivel de logueo en cliente

El usuario puede ajustar el nivel de logueo del cliente (quiet o verbose) utilizando los flags -q (o -quiet) o -v (o -verbose). El valor por defecto es verbose.

Los ejemplos mostrados a continuación fueron realizados con el ping directo, pero los flags son válidos para todos los tipos de ping.

```
$ python tp_ping.py -p -q
```

```
Terminal
```

```
> ~/D/F/7/1/tp_ping on main = python3 tp_ping.py -p -c 10 -q           02:18:26  
TP-PING v0.1  
Operation: Direct Ping  
Server Address: localhost  
Client Address: 127.0.0.1  
  
--- ('localhost', 10000) ping statistics ---  
10 packets transmitted, 10 received, 0.000% packet loss, time 9011.235ms  
rtt min/avg/max/mdev = 0.130/0.494/0.312/0.077 ms  
-> ~/D/F/7/1/tp_ping on main = 
```

```
Terminal
```

```
-> ~/D/F/7/1/tp_ping on main = python3 tp_ping_srv.py                 02:18:30  
Starting up on localhost port 10000  
Waiting for a connection  
Connection from ('127.0.0.1', 39384)  
Received b'00'  
Received b'00,ping'  
Received b'00,ping'  
Received b'00,ping'  
Received b'00,ping'  
Received b'00,ping'  
Received b'00,ping'  
Received b'00,ping'  
Received b'00,ping'  
Received b'00,ping'  
Received b'00,ping'  
Received b''  
No data received from ('127.0.0.1', 39384)  
Waiting for a connection  

```

```
$ python tp_ping.py -p -v
```

```
Terminal
-> ~/D/F/7/1/tp_ping on main • python3 tp_ping.py -p -c 10 -v 02:19:05
TP-PING v0.1
Operation: Direct Ping
Server Address: localhost
Client Address: 127.0.0.1
35 bytes from ('localhost', 10000): seq=1 time=0.062 ms
35 bytes from ('localhost', 10000): seq=2 time=0.343 ms
35 bytes from ('localhost', 10000): seq=3 time=0.401 ms
35 bytes from ('localhost', 10000): seq=4 time=0.348 ms
35 bytes from ('localhost', 10000): seq=5 time=0.338 ms
35 bytes from ('localhost', 10000): seq=6 time=0.323 ms
35 bytes from ('localhost', 10000): seq=7 time=0.410 ms
35 bytes from ('localhost', 10000): seq=8 time=0.295 ms
35 bytes from ('localhost', 10000): seq=9 time=0.308 ms
35 bytes from ('localhost', 10000): seq=10 time=0.365 ms

--- ('localhost', 10000) ping statistics ---
10 packets transmitted, 10 received, 0.000% packet loss, time 9011.215ms
rtt min/avg/max/mdev = 0.062/0.410/0.319/0.059 ms
-> ~/D/F/7/1/tp_ping on main • 02:19:25

Terminal
-> ~/D/F/7/1/tp_ping on main • python3 tp_ping_srv.py 02:19:09
Starting up on localhost port 10000
Waiting for a connection
Connection from ('127.0.0.1', 39390)
Received b'0'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b''
No data received from ('127.0.0.1', 39390)
Waiting for a connection
0
```

## 4.2. Ping Directo

El Ping directo medirá la latencia entre el host cliente y el host servidor, para lo cual envía un mensaje ping al servidor, el cual contesta con un mensaje de respuesta.

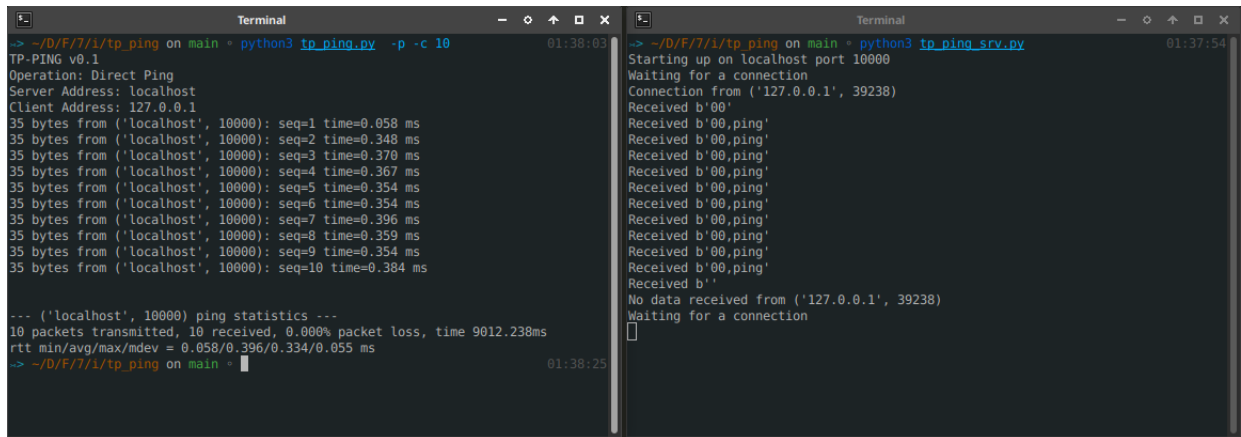
El usuario puede invocar al ping directo usando el flag -p. Por defecto, la cantidad de pings a realizar es ilimitada y sólo está limitada por el usuario. Una vez que el usuario interrumpa el proceso, el ping finalizará mostrando un resumen de la ejecución.

```
$ python tp_ping.py -p
```

```
Terminal
TP-PING v0.1
Operation: Direct Ping
Server Address: localhost
Client Address: 127.0.0.1
35 bytes from ('localhost', 10000): seq=1 time=0.076 ms
35 bytes from ('localhost', 10000): seq=2 time=0.356 ms
35 bytes from ('localhost', 10000): seq=3 time=0.351 ms
35 bytes from ('localhost', 10000): seq=4 time=0.354 ms
35 bytes from ('localhost', 10000): seq=5 time=0.334 ms
35 bytes from ('localhost', 10000): seq=6 time=0.351 ms
35 bytes from ('localhost', 10000): seq=7 time=0.358 ms
35 bytes from ('localhost', 10000): seq=8 time=0.289 ms
35 bytes from ('localhost', 10000): seq=9 time=0.356 ms
35 bytes from ('localhost', 10000): seq=10 time=0.318 ms
35 bytes from ('localhost', 10000): seq=11 time=0.346 ms
35 bytes from ('localhost', 10000): seq=12 time=0.347 ms
35 bytes from ('localhost', 10000): seq=13 time=0.297 ms
35 bytes from ('localhost', 10000): seq=14 time=0.337 ms
35 bytes from ('localhost', 10000): seq=15 time=0.299 ms
^C
--- ('localhost', 10000) ping statistics ---
16 packets transmitted, 15 received, 6.250% packet loss, time 14064.309ms
rtt min/avg/max/mdev = 0.076/0.358/0.298/0.041 ms
-> ~/D/F/7/i/tp_ping on main
Terminal
-> ~/D/F/7/i/tp_ping on main * python3 tp_ping_srv.py
Starting up on localhost port 10000
Waiting for a connection
Connection from ('127.0.0.1', 39232)
Received b''
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b''
No data received from ('127.0.0.1', 39232)
Waiting for a connection
01:36:23
01:37:15
```

El usuario también puede definir la cantidad de pings que desea ejecutar usando el flag `-c` o `--count`, y pasando un parámetro numérico al mismo.

```
$ python tp_ping.py -p -c <count>
```



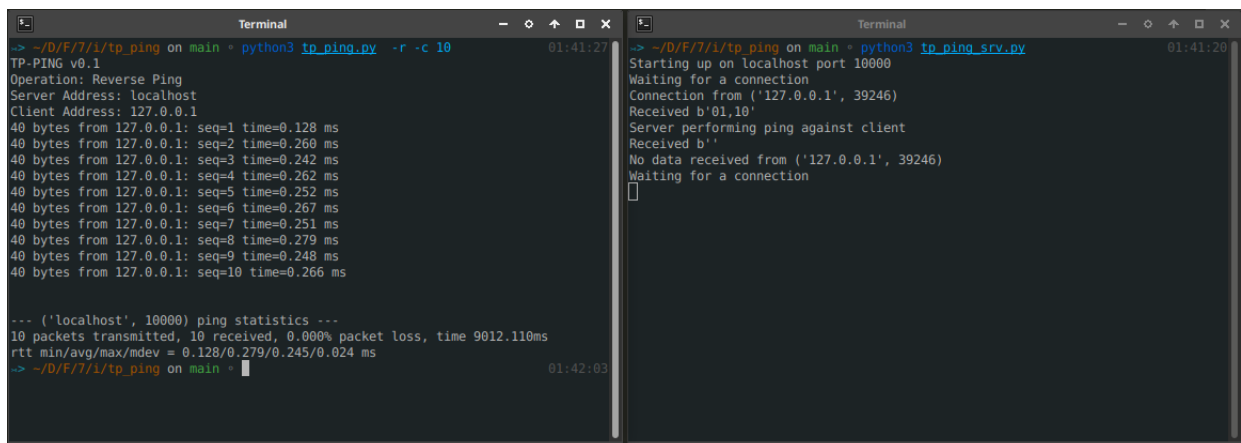
The image shows two terminal windows side-by-side. The left window is titled 'Terminal' and shows the output of `python3 tp_ping.py -p -c 10`. It displays 'TP-PING v0.1', 'Operation: Direct Ping', 'Server Address: localhost', and 'Client Address: 127.0.0.1'. It then shows 10 ping results with sequence numbers 1-10 and times ranging from 0.348 ms to 0.384 ms. Below this, it shows ping statistics: '10 packets transmitted, 10 received, 0.000% packet loss, time 9012.238ms' and 'rtt min/avg/max/mdev = 0.058/0.396/0.334/0.055 ms'. The right window is also titled 'Terminal' and shows the output of `python3 tp_ping_srv.py`. It displays 'Starting up on localhost port 10000', 'Waiting for a connection', and 'Connection from ('127.0.0.1', 39238)'. It then shows 10 received pings with sequence numbers 1-10 and times ranging from 0.348 ms to 0.384 ms. Below this, it shows ping statistics: '10 packets transmitted, 10 received, 0.000% packet loss, time 9012.238ms' and 'rtt min/avg/max/mdev = 0.058/0.396/0.334/0.055 ms'.

### 4.3. Ping Reverso

El ping reverso medirá la latencia entre el host servidor y el host cliente. Para ello, el proceso cliente debe enviar un mensaje de ping reverso al proceso servidor. Cuando reciba el mensaje, el servidor debe realizar la operación de ping al cliente.

El usuario podrá ejecutar indefinidamente el ping reverso ejecutando el siguiente comando:

```
$ python tp_ping.py -r
```



The image shows two terminal windows side-by-side. The left window is titled 'Terminal' and shows the output of `python3 tp_ping.py -r -c 10`. It displays 'TP-PING v0.1', 'Operation: Reverse Ping', 'Server Address: localhost', and 'Client Address: 127.0.0.1'. It then shows 10 ping results with sequence numbers 1-10 and times ranging from 0.242 ms to 0.266 ms. Below this, it shows ping statistics: '10 packets transmitted, 10 received, 0.000% packet loss, time 9012.110ms' and 'rtt min/avg/max/mdev = 0.128/0.279/0.245/0.024 ms'. The right window is also titled 'Terminal' and shows the output of `python3 tp_ping_srv.py`. It displays 'Starting up on localhost port 10000', 'Waiting for a connection', and 'Connection from ('127.0.0.1', 39246)'. It then shows 10 received pings with sequence numbers 1-10 and times ranging from 0.242 ms to 0.266 ms. Below this, it shows ping statistics: '10 packets transmitted, 10 received, 0.000% packet loss, time 9012.110ms' and 'rtt min/avg/max/mdev = 0.128/0.279/0.245/0.024 ms'.

Al igual que el ping directo, el usuario puede definir la cantidad de veces que ejecutará el comando ping utilizando el parámetro `-c` o `--count`

```
$ python tp_ping.py -r -c <count>
```

```

Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping.py -r
01:42:27
TP-PING v0.1
Operation: Reverse Ping
Server Address: localhost
Client Address: 127.0.0.1
40 bytes from 127.0.0.1: seq=1 time=0.053 ms
40 bytes from 127.0.0.1: seq=2 time=0.202 ms
40 bytes from 127.0.0.1: seq=3 time=0.203 ms
40 bytes from 127.0.0.1: seq=4 time=0.197 ms
40 bytes from 127.0.0.1: seq=5 time=0.197 ms
40 bytes from 127.0.0.1: seq=6 time=0.236 ms
40 bytes from 127.0.0.1: seq=7 time=0.272 ms
40 bytes from 127.0.0.1: seq=8 time=0.265 ms
40 bytes from 127.0.0.1: seq=9 time=0.297 ms
40 bytes from 127.0.0.1: seq=10 time=0.256 ms
40 bytes from 127.0.0.1: seq=11 time=0.260 ms
^C
--- ('localhost', 10000) ping statistics ---
11 packets transmitted, 11 received, 0.000% packet loss, time 10226.692ms
rtt min/avg/max/mdev = 0.053/0.297/0.222/0.047 ms
01:42:48
-> ~/D/F/7/1/tp_ping on main x

Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping_srv.py
01:42:30
Starting up on localhost port 10000
Waiting for a connection
Connection from ('127.0.0.1', 39248)
Received b'01,0'
Server performing ping against client
Received b''
No data received from ('127.0.0.1', 39248)
Waiting for a connection

```

#### 4.4. Ping Proxy

El ping proxy medirá la latencia entre el host servidor A y un servidor B (ambos servidores hostean la app servidor). El proceso cliente enviará un mensaje de ping proxy al proceso servidor A, junto con la dirección IP y el puerto del servidor B. Al recibir el mensaje, el servidor A debe efectuar la operación de ping al servidor B.

Para ello, el usuario debe ingresar el siguiente comando (teniendo levantados previamente dos servidores A y B):

```
$ python tp_ping.py -x
```

```

Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping.py -x -c 10
02:28:30
TP-PING v0.1
Operation: Proxy Ping
Server Address: localhost
Client Address: 127.0.0.1
35 bytes from ('localhost', 10000): seq=1 time=0.139 ms
35 bytes from ('localhost', 10000): seq=2 time=0.389 ms
35 bytes from ('localhost', 10000): seq=3 time=0.404 ms
35 bytes from ('localhost', 10000): seq=4 time=0.390 ms
35 bytes from ('localhost', 10000): seq=5 time=0.404 ms
35 bytes from ('localhost', 10000): seq=6 time=0.402 ms
35 bytes from ('localhost', 10000): seq=7 time=0.407 ms
35 bytes from ('localhost', 10000): seq=8 time=0.443 ms
35 bytes from ('localhost', 10000): seq=9 time=0.171 ms
35 bytes from ('localhost', 10000): seq=10 time=0.339 ms
--- localhost:10001 ping statistics ---
10 packets transmitted, 10 received, 0.000% packet loss, time 9010.699ms
rtt min/avg/max/mdev = 0.139/0.443/0.349/0.079 ms
02:29:07
-> ~/D/F/7/1/tp_ping on main x

Terminal
Starting up on localhost port 10000
Waiting for a connection
Connection from ('127.0.0.1', 39412)
Received b'10,10,localhost:10001'
Server performing ping against localhost on 10001
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Received b''
No data received from ('127.0.0.1', 39412)
Waiting for a connection

Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping_srv.py
02:28:36
Port 10000 is in use!
Starting up on localhost port 10001
Waiting for a connection
Connection from ('127.0.0.1', 44220)
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b''
No data received from ('127.0.0.1', 44220)
Waiting for a connection

```

Por defecto, el cliente se conectará al servidor que escucha en el puerto 10000 y pedirá ping al servidor que escucha en el puerto 10001, a menos que el usuario indique lo contrario a través de los flags -s (-server) y -d (-dest)

```
$ python tp_ping.py -x -d <destination_host_and_port>
```



```
Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping.py -x -d 127.0.0.1:9800 02:30:06
TP-PING v0.1
Operation: Proxy Ping
Server Address: localhost
Client Address: 127.0.0.1
35 bytes from ('localhost', 10000): seq=1 time=0.148 ms
35 bytes from ('localhost', 10000): seq=2 time=0.334 ms
35 bytes from ('localhost', 10000): seq=3 time=0.336 ms
35 bytes from ('localhost', 10000): seq=4 time=0.339 ms
35 bytes from ('localhost', 10000): seq=5 time=0.326 ms
^C

--- 127.0.0.1:9800 ping statistics ---
5 packets transmitted, 5 received, 0.000% packet loss, time 4836.329ms
rtt min/avg/max/mdev = 0.148/0.339/0.297/0.059 ms
-> ~/D/F/7/1/tp_ping on main x 02:30:41

Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping_srv.py 02:29:59
Starting up on localhost port 10000
Waiting for a connection
Connection from ('127.0.0.1', 39418)
Received b'10,0,127.0.0.1:9800'
Server performing ping against 127.0.0.1 on 9800
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Received b''
No data received from ('127.0.0.1', 39418)
Waiting for a connection

Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping_srv.py 9800 02:29:59
Starting up on localhost port 9800
Waiting for a connection
Connection from ('127.0.0.1', 38788)
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b''
No data received from ('127.0.0.1', 38788)
Waiting for a connection
```

Al igual que los otros pings, también se puede establecer un conteo:

```
$ python tp_ping.py -x -c <count> -d <destination_host_and_port>
```

```
Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping.py -x -c 10 -d 127.0.0.1:9800 02:27:33
TP-PING v0.1
Operation: Proxy Ping
Server Address: localhost
Client Address: 127.0.0.1
35 bytes from ('localhost', 10000): seq=1 time=0.182 ms
35 bytes from ('localhost', 10000): seq=2 time=0.382 ms
35 bytes from ('localhost', 10000): seq=3 time=0.345 ms
35 bytes from ('localhost', 10000): seq=4 time=0.346 ms
35 bytes from ('localhost', 10000): seq=5 time=0.354 ms
35 bytes from ('localhost', 10000): seq=6 time=0.296 ms
35 bytes from ('localhost', 10000): seq=7 time=0.302 ms
35 bytes from ('localhost', 10000): seq=8 time=0.302 ms
35 bytes from ('localhost', 10000): seq=9 time=0.295 ms
35 bytes from ('localhost', 10000): seq=10 time=0.292 ms

--- 127.0.0.1:9800 ping statistics ---
10 packets transmitted, 10 received, 0.000% packet loss, time 9012.460ms
rtt min/avg/max/mdev = 0.182/0.382/0.310/0.038 ms
-> ~/D/F/7/1/tp_ping on main x 02:27:33

Terminal
Starting up on localhost port 10000
Waiting for a connection
Connection from ('127.0.0.1', 39408)
Received b'10,10,127.0.0.1:9800'
Server performing ping against 127.0.0.1 on 9800
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Received b''
No data received from ('127.0.0.1', 39408)
Waiting for a connection

Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping_srv.py 9800 02:27:03
Starting up on localhost port 9800
Waiting for a connection
Connection from ('127.0.0.1', 38778)
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b''
No data received from ('127.0.0.1', 38778)
Waiting for a connection
```

Como mencionamos anteriormente, el usuario puede definir el servidor que debe ejecutar el ping (esto es válido para todos los tipos de ping):

```
$ python tp_ping.py -x -c <count> -s <server_host_and_port>
```

```
Terminal
-> ~/D/F/7/1/tp_ping on main x
python3 tp_ping.py -x -s 127.0.0.1:9800 -d 127.0.0.1:10000
TP-PING v0.1
Operation: Proxy Ping
Server Address: 127.0.0.1
Client Address: 127.0.0.1
35 bytes from ('127.0.0.1', 9800): seq=1 time=0.120 ms
35 bytes from ('127.0.0.1', 9800): seq=2 time=0.314 ms
35 bytes from ('127.0.0.1', 9800): seq=3 time=0.334 ms
35 bytes from ('127.0.0.1', 9800): seq=4 time=0.330 ms
35 bytes from ('127.0.0.1', 9800): seq=5 time=0.329 ms
35 bytes from ('127.0.0.1', 9800): seq=6 time=0.341 ms
35 bytes from ('127.0.0.1', 9800): seq=7 time=0.344 ms
35 bytes from ('127.0.0.1', 9800): seq=8 time=0.340 ms
^C

--- 127.0.0.1:10000 ping statistics ---
8 packets transmitted, 8 received, 0.000% packet loss, time 7976.009ms
rtt min/avg/max/mdev = 0.120/0.344/0.306/0.047 ms
-> ~/D/F/7/1/tp_ping on main x

Terminal
Starting up on localhost port 9800
Waiting for a connection
Connection from ('127.0.0.1', 38796)
Received b'10,0,127.0.0.1:10000'
Server performing ping against 127.0.0.1 on 10000
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Sending results to client
Received b''
No data received from ('127.0.0.1', 38796)
Waiting for a connection

Terminal
-> ~/D/F/7/1/tp_ping on main x python3 tp_ping_srv.py
Starting up on localhost port 10000
Waiting for a connection
Connection from ('127.0.0.1', 39430)
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b'00,ping'
Received b''
No data received from ('127.0.0.1', 39430)
Waiting for a connection
```

## 5. Preguntas a responder

### 1) Describa la arquitectura Cliente-Servidor

En una arquitectura Cliente-Servidor, la conexión es siempre entre un extremo y otro. Hay un cliente y un servidor, que se comunican entre sí, pero este último puede servir a varios clientes. Sin embargo, puede atender solamente de uno a la vez.

El cliente es quien inicia la conexión, realizando una petición al servidor.

El servidor está siempre disponible, pues no se sabe cuándo un cliente va a generar una petición. Deben tener una dirección física y conocida, llamada dirección IP, la cual es necesaria para que los clientes puedan enviar sus peticiones.

Una limitación de esta arquitectura es que los clientes no pueden comunicarse entre sí. Solamente se comunican con el servidor.

### 2) ¿Cuál es la función de un protocolo de capa de aplicación?

La función de un protocolo de capa de aplicación es estandarizar los mensajes que se envían y reciben, para que los dispositivos que se comunican entre sí puedan 'hablar el mismo idioma' y entenderse.

### 3) Detalle el protocolo de aplicación desarrollado en este trabajo

Para el presente trabajo, desarrollamos un protocolo de aplicación propio, que se detallará a continuación.

Lo principal a tener en cuenta, es que los mensajes se dividen en un header (que debe tener información obligatoriamente) y un body (que puede tener contenido o no).

Los tipos de mensajes se determinaron de acuerdo al tipo de operación, a saber:

- Código de operación OO: Ping Directo
- Código de operación O1: Ping Reverso
- Código de operación IO: Ping Proxy
- Código de operación 11: Mensaje de respuesta

A continuación, detallamos la estructura de cada mensaje que intercambiará nuestra aplicación:

- Para Ping Directo: Para el ping directo implementamos los siguientes tipos de mensaje:

### Ping Directo

Mensaje inicio conexión

header	body
00	

Respuesta inicio conexión

header	body
11	

Mensaje PING al servidor

header	body
00	'ping'

Respuesta PING al servidor

header	body
11	

- Para Ping Reverso: Para el caso del ping reverso, los mensajes se complejizaron un poco debido a que debíamos establecer la conexión, hacer el ping desde el servidor al cliente, y luego enviar al cliente los datos obtenidos como resultado. Por esta razón implementamos los siguientes tipos de mensaje:

### Ping Reverso

Mensaje inicio conexión

header	body
01	int(count)

Respuesta inicio conexión

header	body
11	

Mensaje PING al servidor

header	body
01	'ping'

Respuesta PING al servidor

header	body
11	int(packet_size), int(seq_num), float(rtt)

- Para Ping Proxy: Para este caso, el mensaje es muy similar al de ping reverso, pero agregamos los campos host destino y puerto destino al body del mensaje de inicio de conexión.

### Ping Proxy

Mensaje inicio conexión

header	body
10	int(count), string(destination_host), int(destination_port)

Respuesta inicio conexión

header	body
11	

Mensaje PING al servidor

header	body
10	'ping'

Respuesta PING al servidor

header	body
11	int(packet_size), int(seq_num), float(rtt)

4) La capa de transporte del stack TCP/IP ofrece dos protocolos: TCP y UDP. ¿Qué servicios proveen dichos protocolos? ¿Cuáles son sus características? ¿Cuándo es apropiado utilizar cada uno?

■ *TCP (Transmission Control Protocol)*: Ofrece los siguientes servicios:

- Conexión: Cliente y servidor deben iniciar la comunicación antes de intercambiar mensajes (handshake). La conexión es full-duplex, osea que cliente y servidor se pueden enviar datos el uno al otro en simultáneo.
- Garantía de entrega, integridad y orden: TCP regula el throughput de acuerdo con el estado de carga de la red. Esto preserva el bienestar de la red y el uso equitativo entre los usuarios. Esto es un bonus.
- Control de congestión

■ *UDP (User Datagram Protocol)*: UDP no ofrece nada de lo que nos da TCP: sin conexión, sin garantías de entrega ni de orden, ni control de gestión. La ventaja por sobre TCP es que es más liviano: bajo overhead y bajo impacto en CPU.

¿Para qué se utiliza cada uno? TCP se utiliza en situaciones donde necesitamos garantizar que nuestros paquetes llegarán íntegros, como por ejemplo transferencia de archivos (SSH, FTP) o email (SMTP). UDP se utiliza en aplicaciones real-time que pueden funcionar a pesar de la pérdida de paquetes, por ejemplo videollamadas.

5) *¿Cómo se identifica un socket en un host?*

Los procesos en los hosts se identifican a través del número de puerto. Es decir que si un servicio está corriendo en el puerto 8080, el número de proceso será 8080.

6) *Indicar cuál fue el protocolo de transporte utilizado para implementar las operaciones y justifique los motivos de dicha decisión.*

Para determinar qué protocolo utilizar, primero hicimos un análisis de qué era lo que queríamos hacer en cada operación, y qué atributos de cada protocolo nos resultaban útiles.

En el caso del ping directo, estuvimos de acuerdo en que no era necesario tener garantía de entrega ni de integridad, solo necesitábamos poder enviar mensajes de un host a otro. Por esa razón, pensamos en elegir el protocolo UDP

Para el caso del ping reverso y del ping proxy, nos dimos cuenta que iba a ser necesario tener conocimiento de si el mensaje era entregado y preservar la integridad del mismo, sobre todo porque necesitábamos enviar al cliente los resultados de cada ping que hiciera el servidor al cliente o el servidor al servidor destino. Además, nos dimos cuenta que era necesario establecer una conexión para que pudiera haber intercambio de mensajes entre cliente y servidor. Es por esto que optamos por implementar estas dos operaciones con el protocolo TCP.

Sin embargo, como mencionamos anteriormente, nos encontramos con una limitación técnica a la hora de programar el servidor, ya que no podíamos cambiar de protocolo de transporte en runtime. Es por este motivo que decidimos utilizar TCP para todos los tipos de ping a implementar.

7) *Al realizar las operaciones de ping y ping reverso. ¿Se puede concluir que los caminos que atravesaron los paquetes son simétricos?*

Cuando realizamos operaciones de ping y ping directo, notamos que los rtt resultantes eran similares, con lo cual podemos concluir que efectivamente, los caminos atravesados por los paquetes son simétricos.

8) *Realizar un ping al servidor usando la herramienta ping y la app desarrollada, utilizando 10 requests (c = 10) y comparar los resultados. ¿Los RTT estimados son comparables? Describir los resultados obtenidos y sacar conclusiones.*

A continuación presentamos capturas de pantalla de las corridas pedidas para la herramienta ping y para el ping implementado por nosotros, respectivamente.

Tal como se observa en las imágenes, los RTTs de la aplicación desarrollada por nosotros son mayores que los del ping original (aproximadamente 6-7 veces más grandes). No obstante, los tiempos totales son similares, por lo que podemos deducir que el tiempo total en realizar toda la operatoria es similar, pero en el caso de nuestra aplicación, tardó mucho más en recibir cada paquete, a pesar de que el tamaño de nuestros paquetes es casi la mitad del tamaño de los paquetes del ping real.

```

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.055 ms

--- 127.0.0.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 922ms
rtt min/avg/max/mdev = 0.035/0.052/0.055/0.005 ms

```

```

TP-PING v0.1
Operation: Direct Ping
Server Address: ('localhost', 8080)
Client Address: 127.0.0.1
35 bytes from ('localhost', 8080): seq=1 time=0.082 ms
35 bytes from ('localhost', 8080): seq=2 time=0.340 ms
35 bytes from ('localhost', 8080): seq=3 time=0.371 ms
35 bytes from ('localhost', 8080): seq=4 time=0.357 ms
35 bytes from ('localhost', 8080): seq=5 time=0.222 ms
35 bytes from ('localhost', 8080): seq=6 time=0.327 ms
35 bytes from ('localhost', 8080): seq=7 time=0.348 ms
35 bytes from ('localhost', 8080): seq=8 time=0.399 ms
35 bytes from ('localhost', 8080): seq=9 time=0.323 ms
35 bytes from ('localhost', 8080): seq=10 time=0.350 ms

--- ('localhost', 8080) ping statistics ---
10 packets transmitted, 10 received, 0.000% packet loss, time 901.157ms
rtt min/avg/max/mdev = 0.082/0.399/0.312/0.064 ms

```

## 6. Dificultades encontradas

Durante el proceso de desarrollo de este trabajo, no encontramos mayores dificultades que las que podría encontrar cualquier estudiante que está aprendiendo conceptos nuevos en una materia y necesita familiarizarse con ellos para poder aplicarlos a nivel práctico. La más importante de las mismas probablemente haya sido el ponerlos de acuerdo en qué protocolo utilizar, pues era una decisión clave que determinaría toda la implementación del trabajo y estaba basada en nuestro entendimiento de los temas trabajados en clase. El dilema se resolvió compartiendo en grupo qué ideas tenía cada uno, realizando pequeños ejemplos o pruebas de concepto, releando apuntes y material de clase hasta que quedamos convencidos y listos para tomar una decisión.

## 7. Conclusión

El trabajo presentado sirvió como acercamiento a los temas, y no sólo nos permitió poner en práctica lo aprendido, sino que también nos permitió profundizar y conocer otras cuestiones.

Python es un lenguaje de sintaxis sencilla pero muy poderoso, y a través de su librería de sockets nos permitió implementar rápidamente un cliente y un servidor donde pudimos hacer varias pruebas. Nos pareció muy positiva la experiencia de realizar el desarrollo en este lenguaje, no solo por su flexibilidad, sino también porque es un lenguaje sencillo para aprender.

El presente trabajo nos pareció una manera interesante de aprender cómo funcionan los protocolos de capa de aplicación (¡Y nos permitió implementar uno propio!), cuáles son los protocolos de capa de transporte, sus características y funcionamiento. Además, nos dio la libertad de tomar decisiones de implementación y justificar el por qué de las mismas con fundamentos teóricos vistos en la materia.

En síntesis, nos pareció un buen ejercicio integrador para ayudarnos a afianzar los temas tratados en las clases.