

Explicación paso a paso de cómo enviar una imagen entre un cliente y un servidor por el protocolo UDP que supere los 65535 bytes

En el siguiente ejemplo la imagen que vamos a mandar tiene un tamaño de 358 KB.

Script comentado para el cliente (envía la imagen en varias partes)

```
## Cliente
```

```
# Conexión IP con UDP
```

```
$ip = [System.Net.IPEndPoint]::new([IPAddress]::Loopback,2020)
```

```
$udp = [System.Net.Sockets.UdpClient]::new()
```

```
# Leer una imagen png y convertirla a Bytes
```

```
$Bytes = [System.IO.File]::ReadAllBytes('.\captura.png')
```

```
# Ver el tamaño de la imagen
```

```
$Bytes.Length
```

```
# Para enviar por UDP el tamaño de cada datagrama tienen que ser menor de 65507
```

```
# El tamaño máximo de un paquete IP que incluye todos los
```

```
# encabezados es de 65535 bytes. Los encabezados IP y UDP se combinan
```

```
# para al menos 28 bytes. Así que el tamaño máximo de la porción de datos
```

```
# de un datagrama UDP es 65535-28 == 65507
```

```
# Partes en la que habría que dividir la imagen y enviarla
```

```
($Bytes.Length / 65000)
```

```
# Hacemos el envío de cada parte calculando los tamaños
$udp.Send($Bytes,65000,$ip) | Out-Null
$udp.Send($Bytes[65000..(65000*2)],65000,$ip) | Out-Null
$udp.Send($Bytes[(65000*2)..(65000*3)],65000,$ip) | Out-Null
$udp.Send($Bytes[(65000*3)..(65000*4)],65000,$ip) | Out-Null
$udp.Send($Bytes[(65000*4)..(65000*5)],65000,$ip) | Out-Null
$udp.Send($Bytes[(65000*5)..$Bytes.Length],$Bytes[(65000*5)..$
Bytes.Length].Length,$ip) | Out-Null
```

```
# Cerrar conexión UDP
$udp.Close()
```

Script comentado para el servidor (recibe la imagen en varias partes y después las junta)

```
## Server
```

```
# Conexión IP con UDP
$ip = [System.Net.IPEndPoint]::new([IPAddress]::Any,0)
$udp = [System.Net.Sockets.UdpClient]::new(2020)
```

```
# Dependiendo del tamaño de la imagen que vamos recibir habrá  
que recibir
```

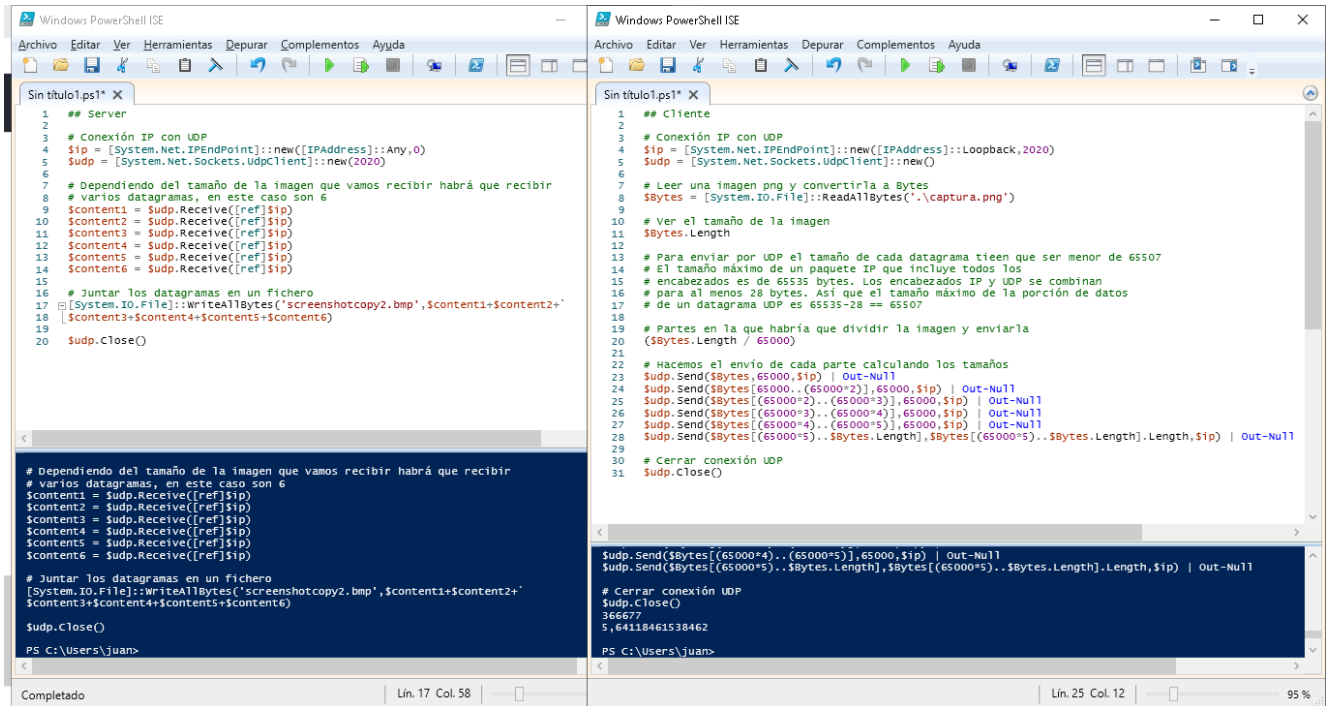
```
# varios datagramas, en este caso son 6
```

```
$content1 = $udp.Receive([ref]$ip)
$content2 = $udp.Receive([ref]$ip)
$content3 = $udp.Receive([ref]$ip)
$content4 = $udp.Receive([ref]$ip)
$content5 = $udp.Receive([ref]$ip)
$content6 = $udp.Receive([ref]$ip)
```

```
# Juntar los datagramas en un fichero
```

```
[System.IO.File]::WriteAllBytes('screenshotcopy2.bmp',$content
1+$content2+`
$content3+$content4+$content5+$content6)
```

\$udp.Close()



The image displays two side-by-side screenshots of the Windows PowerShell ISE (Integrated Scripting Environment) interface. Both windows show a script titled 'Sin título1.ps1'.

The left window shows the server script, which includes comments and code for receiving data via UDP. The script defines a server IP endpoint, creates a UDP client, and receives data in six datagrams. It then concatenates the received data and writes it to a file named 'screenshotcopy2.bmp'. The script concludes with a call to `$udp.Close()`. The status bar at the bottom indicates 'Completado' (Completed) at line 17, column 58.

The right window shows the client script, which includes comments and code for sending data via UDP. The script defines a client IP endpoint, creates a UDP client, and reads a file named 'captura.png'. It then sends the data in five datagrams, each 65,000 bytes in size. The script concludes with a call to `$udp.Close()`. The status bar at the bottom indicates '95 %' completion at line 25, column 12.

```
## Server
# Conexión IP con UDP
$ip = [System.Net.IPEndPoint]::new([IPAddress]::Any,0)
$udp = [System.Net.Sockets.UdpClient]::new(2020)

# Dependiendo del tamaño de la imagen que vamos recibir habrá que recibir
# varios datagramas, en este caso son 6
$content1 = $udp.Receive([ref]$ip)
$content2 = $udp.Receive([ref]$ip)
$content3 = $udp.Receive([ref]$ip)
$content4 = $udp.Receive([ref]$ip)
$content5 = $udp.Receive([ref]$ip)
$content6 = $udp.Receive([ref]$ip)

# Juntar los datagramas en un fichero
[System.IO.File]::WriteAllBytes('screenshotcopy2.bmp', $content1+$content2+$
$content3+$content4+$content5+$content6)

$udp.Close()

PS C:\Users\juam>
```

```
## Cliente
# Conexión IP con UDP
$ip = [System.Net.IPEndPoint]::new([IPAddress]::Loopback,2020)
$udp = [System.Net.Sockets.UdpClient]::new()

# Leer una imagen png y convertirla a Bytes
$bytes = [System.IO.File]::ReadAllBytes('.\captura.png')

# Ver el tamaño de la imagen
$bytes.Length

# Para enviar por UDP el tamaño de cada datagrama tienen que ser menor de 65507
# El tamaño máximo de un paquete IP que incluye todos los
# encabezados es de 65535 bytes. Los encabezados IP y UDP se combinan
# para al menos 28 bytes. Así que el tamaño máximo de la porción de datos
# de un datagrama UDP es 65535-28 == 65507

# Partes en la que habría que dividir la imagen y enviarla
($bytes.Length / 65000)

# Hacemos el envío de cada parte calculando los tamaños
$udp.Send($bytes, 65000, $ip) | Out-Null
$udp.Send($bytes[65000..(65000*2)], 65000, $ip) | Out-Null
$udp.Send($bytes[(65000*2)..(65000*3)], 65000, $ip) | Out-Null
$udp.Send($bytes[(65000*3)..(65000*4)], 65000, $ip) | Out-Null
$udp.Send($bytes[(65000*4)..(65000*5)], 65000, $ip) | Out-Null
$udp.Send($bytes[(65000*5)..$bytes.Length], $bytes[(65000*5)..$bytes.Length].Length, $ip) | Out-Null

# Cerrar conexión UDP
$udp.Close()

PS C:\Users\juam>
```