Porque os programas não enviam arquivos grandes como deveriam?

I- Programa 1: Utilizando timeout

Testando o programa utilizando timeout no cliente que estava configurado com dois segundos, notamos que ao enviar um arquivo superior a 2 MB, o programa cliente não conseguia receber o arquivo completamente, como pode ser visto na Figura 1 e 2. O motivo disso ocorrer é justamente por conta do tempo do timeout ser relativamente baixo fazendo com que se for enviado arquivos grandes, eles não serão enviados corretamente como na Figura 3, onde está localizado o arquivo original.

Figura 1 - Arquivo recebido pelo cliente possuindo 2.996 KB.

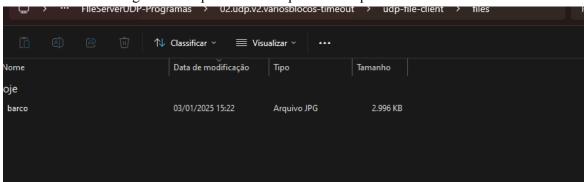


Figura 2 - Arquivo recebido pelo cliente possuindo 616 KB.

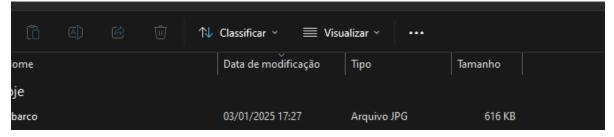
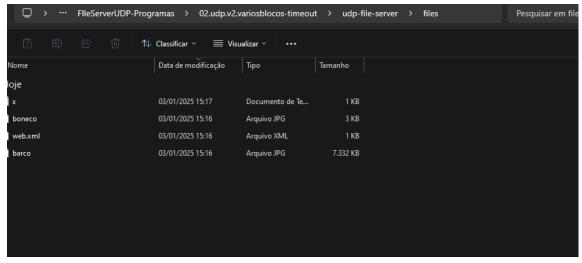


Figura 3 - Arquivo barco.jpg no servidor possuindo 7.332 KB.



Fonte: Autoria Própria, 2025.

O servidor de arquivos recebe uma solicitação do cliente para enviar o arquivo 'barco.jpg' que o envia como representado na Figura 4.

Figura 4: Resposta do servidor ao pedido do cliente.

```
C:\Users\JORDAN\Downloads\FIleServerUDP-Programas\02.udp.v2.variosblocos-timeout\udp-file-server> python udp-file-server.py
Escutando em ... ('127.0.0.1', 12345)
Recebi pedido para o arquivo barco.jpg
Enviando arquivo barco.jpg
```

Fonte: Autoria Própria, 2025.

O cliente recebe o arquivo 'barco.jpg' que é gravado na sua pasta de arquivos. Entretanto, o arquivo saiu com timeout, ou seja, o tamanho do arquivo 'barco.jpg' não foi enviado completamente, pois o timeout estabelecido interrompeu o restante do envio de dados por ter passado do tempo previamente estabelecido no programa cliente como mostrado na Figura 5.

Figura 5: Recebimento de arquivo incompleto no cliente.

```
:\Users\JORDAN\Downloads\FIleServerUDP-Programas\02.udp.v2.variosblocos-timeout\udp-file-client>python udp-file-client.py
Arquivo a pedir ao servidor: barco.jpg
Enviando pedido a ('127.0.0.1', 12345) para barco.jpg
Gravando arquivo localmente
Saiu com timeout ...
Arquivo a pedir ao servidor:|
```

II - Programa 2: Utilizando tamanho (Em blocos de 1024 e 2048)

Como solicitado, criamos um programa parecido com o primeiro, mas com a diferença de utilizar tamanho com blocos de bytes ao invés do timeout. Testamos solicitar e enviar arquivos com dois blocos de tamanhos diferentes que foram respectivamente 1024 e 2048 Bytes. Em ambos os casos, para arquivos de tamanhos inferiores a 2 MB, foram enviados e recebidos tranquilamente como se pode notar nas Figuras 6 e 7. O problema ocorreu quando tentamos enviar um arquivo muito maior representado pelo 'barco.jpg' que explicaremos a seguir.

server Downloads files ↑ Classificar ∨ ■ Visualizar ∨ Data de modificação Tamanho Nome Tipo Anteriormente nesta semana i x 30/12/2024 16:58 Documento de Te... 1 KB web.xml 30/12/2024 16:58 Arquivo XML 1 KB boneco 30/12/2024 16:57 Arquivo JPG 3 KB barco 30/12/2024 16:57 Arquivo JPG 7.332 KB

Figura 6: Tamanho original dos arquivos na pasta files no servidor.

Figura 7: Tamanho dos arquivos na pasta files do cliente.

다 > Downloads	> client	> Tiles	1	
	⑪ │ ⇅	Classificar ∨ ≣ Visua	alizar v •••	
ome		Data de modificação	Тіро	Tamanho
oje				
barco		03/01/2025 17:27	Arquivo JPG	616 KB
boneco		03/01/2025 17:08	Arquivo JPG	3 KB
web.xml		03/01/2025 17:08	Arquivo XML	1 KB
x		03/01/2025 17:07	Documento de Te	1 KB

Fonte: Autoria própria, 2025.

No teste com blocos de 1024 bytes, repare que o arquivo 'barco.jpg' na Figura 7 é o único que não teve seus dados enviados completamente ao cliente, diferente dos demais. Ao abrir o arquivo 'barco.jpg' na pasta 'files' do cliente, percebe-se que ele está incompleto como mostrado na Figura 8, diferentemente da imagem completa representada na Figura 9.

Downloads > client > files

Pesquisar em files

Q

Petalhes

Nome

Data de modificação Tipo Tamanho

Hoje

boneco

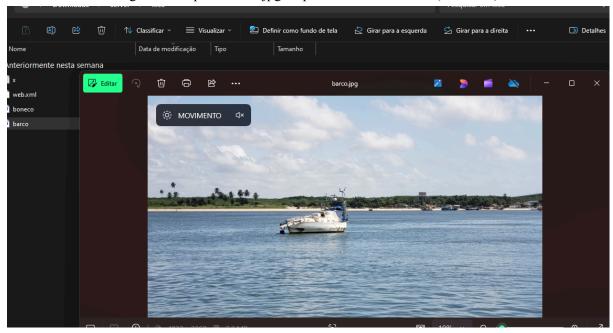
barco

Larcojpg

Larc

Figura 8: Arquivo barco.jpg na pasta files do cliente (616 KB).

Figura 9: Arquivo barco.jpg na pasta files do servidor (7.332KB)



Fonte: Autoria própria, 2025.

Figura 10: Leitura dos bytes de barco.jpg pelo cliente em blocos de 1024 bytes

```
c) Microsoft Corporation. Todos os direitos reservados.
::\Users\JORDAN\Downloads\client>python udp-file-client-size.py
Digite o nome do arquivo desejado: barco.jpg
Enviando pedido a ('127.0.0.1', 1234) para barco.jpg
D arquivo barco.jpg possui o tamanho de 7507955 Bytes.
Salvando o arquivo localmente...
idos:
          1024
idos:
          1024
idos:
          1024
idos:
           1024
idos:
           1024
idos:
           1024
idos:
           1024
idos:
           1024
idos:
           1024
 idos:
           1024
           1024
idos:
idos:
           1024
idos:
           1024
idos:
           1024
idos:
           1024
lidos:
           1024
.idos:
           1024
idos:
           1024
idos:
           1024
           1024
idos:
idos:
           1024
           1024
idos:
idos:
           1024
           1024
idos:
           1024
idos:
idos:
           1024
```

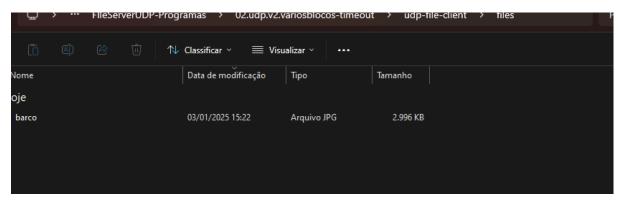
O programa cliente ao fazer a leitura dos bytes de qualquer arquivo que o servidor lhe enviar, dividirá os bytes desses arquivos em blocos de até 1024, quando esses blocos de 1024 terminarem a leitura completa do arquivo terá sido realizada. Entretanto, os 7507955 bytes presentes no arquivo 'barco.jpg' demandam uma grande quantidade desses blocos. Pelo fato dos programas estarem usando o protocolo UDP, quando o sistema operacional do cliente possui uma grande quantidade de blocos que lhe foram enviados, para receber os próximos blocos ele vai descartar aqueles que anteriormente foram recebidos e receberá os novos até que seja lido todo o arquivo. Assim, haverá a perda de pacotes, resultando na perda de bytes.

Também realizamos o teste com blocos de 2048 bytes como mostrado na Figura 11. Mas pelo fato do arquivo ser muito grande, o problema anterior continua persistindo, porém o arquivo barco.jpg no cliente terá mais bytes em decorrência do aumento de bytes nos blocos como pode ser visualizado na Figura 12. Com o aumento desses bytes, a imagem final quando se tenta visualizar, acaba tendo um pouco mais de detalhes comparada com a imagem do bloco de 1024 bytes, bem como mostra a Figura 13.

Figura 11: Leitura dos bytes de barco.jpg pelo cliente em blocos de 2048 bytes

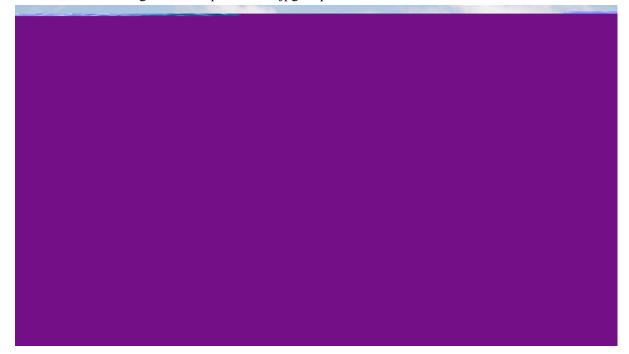
```
C:\Windows\System32\cmd.e
dicrosoft Windows [versão 10.0.22631.4602]
(c) Microsoft Corporation. Todos os direit
                                                    Todos os direitos reservados.
::\Users\JORDAN\Downloads\client>python udp-file-client-size.py
Digite o nome do arquivo desejado: barco.jpg
Inviando pedido a ('127.0.0.1', 1234) para barco.jpg
Di arquivo barco.jpg possui o tamanho de 7507955 Bytes.
Balvando o arquivo localmente...
alvando o arquivo
idos: 2048
idos:
              2048
 idos:
              2048
2048
 idos:
               2048
               2048
 idos:
 idos:
               2048
               2048
               2048
 idos:
               2048
               2048
 idos:
               2048
 idos:
               2048
               2048
               2048
               2048
              2048
2048
               2048
               2048
```

Figura 12: Tamanho do arquivo barco.jpg com blocos de 2048 bytes.



Fonte: Autoria própria, 2025.

Figura 13: Arquivo barco.jpg na pasta files do cliente 2.996 KB



Fonte: Autoria própria, 2025.

III- Conclusão:

Por fim, chegamos a conclusão de que em ambos os casos terão algum tipo de problema, seja por causa do tempo, ou por causa do tamanho dos blocos de bytes, a primeira solução que todos pensariam seria a de aumentar ambos, mas ainda assim continuaria tendo problemas, porque ou teria que esperar bastante tempo mesmo enviando um arquivo pequeno, ou por conta de existir arquivos ainda maiores, tendo que modificar o programa toda vez.