Usando como exemplo os arquivos fornecidos pelo professor e sabendo que o udp já é um protocolo de transporte que não fornece muitas garantias de entrega de pacotes ou de ordenação. Para arquivos grandes, a probabilidade de perda de pacotes é ainda maior. Ainda mais se for uma rede com latência alta.

No cliente é definido um timeout de 2 segundos

```
sock = socket.socket (socket.AF_INET, socket.SOCK_DGRAM)
sock.settimeout(2)
```

Esse tempo muitas vezes pode ser insuficiente, fazendo com que a transferência seja interrompida antes do envio. Isso ocorre mesmo que o servidor ainda esteja enviando os dados gerando arquivos truncados.

Os pacotes sendo enviados em blocos de 4096 bytes pelo servidor e o cliente não tendo nenhum mecanismo para solicitar retransmissão de dados perdidos. Fora isso sem o controle de fluxo para evitar que o cliente fique sobrecarregado com pacotes recebidos rapidamente.

```
fileData = fd.read(4096)
while fileData != b'':
    sock.sendto(fileData, source)
```

dessa forma faz parecer que todos os pacotes enviados são recebidos sem perdas, o que não é muito real.

A fragmentação de pacotes também é outra parte critica do protocolo, pois quando um pacote UDP é maior que a MTU (Unidade Maxima de Transmissão) da rede ele é fragmentado em pacotes menores aí se um único fragmento se perde, todo o pacote original é perdido piorando ainda mais os problemas de perda de dados em transferências de arquivos grandes. As soluções de timeout e envio do tamanho do arquivo não lidam com a reordenação desses pacotes que chegam fora de sequência.

De forma resumida ambas as propostas, timeout e envio do tamanho do arquivo, preenchem apenas partes superficiais do problema da transferência completa de arquivos grandes via UDP. Elas não resolvem a raiz do problema, que é a natureza não confiável do protocolo. Mesmo que os programas pareçam funcionar para arquivos pequenos, onde a probabilidade de perda de pacotes e fragmentação é menor a transferência de arquivos grandes se torna problemática devido à alta probabilidade de perda de pacotes, fragmentação e ordem incorreta de chegada. Para garantir a integridade

dos dados em transferências de arquivos grandes, é necessário implementar um protocolo de camada de aplicação que forneça mecanismos de controle de fluxo, confirmação de recebimento acks e retransmissão de pacotes perdidos, ou então utilizar um protocolo de transporte mais confiável como o TCP.