Multi-Packet Transfer Visualization

By: Anthony Sasso

March 21, 2022

Contents

Example Scenario / Setup

We Are sending a 60 GB (60 Billion Byte) transfer "***T***", using TCP/UDP or a similar method, where there is a max standard transfer of around 500 Bytes (0.005 MB). We will also say our implementation only sends packets of around 110 Bytes (0.001 MB) for reliability reasons. This then means we will need to send around $6.e + 10/110 = 545454546$ or "***P***" Packets. . .

How can we do this while having the "fastest, most reasonable methodology of data networking". I will list the way I have observed it probably works (no guarantee on accuracy).

Segmentation

First we will (while serializing the data) segment it into 100 packet chunks or "***C***", meaning there will be $P \div C$ chunks = 5454546 chunks.

Then, now that we can send somewhat more reasonably sized data we will send chunk by chunk for data consistency. This is to allow the user in the case where internet connections are unreliable to receive data but upon the connection terminating only revert to the previous chunk, not losing the entirety of their progress (as seen in clients like qBittorent).

Visualization

$T = t$	t = size of total packet in Bytes	(1)
$P = T \div p$	p = size of each packet in Bytes (also referred to as segments)	(2)
$C = P \div c$	c = number of packets in a chunk.	(3)

Leading & Trailing Chunks

Now that we have chunks to send we will send the leading "***L***" chunks and final "***F***" chunks first. After this "***L***" being the amount of chunks representative of the program header (program size, chunk size, etc) and metadata. Then finally ***F*** being the amount of chunks representative of the program tail and metadata (leading non-standard sized chunk, and CRC of the entire deserialized program).

This allows the client to begin saving data into storage / organizing more accurately, and you can think of it like a program level control packet. This will necessarily be sent ahead of time to also allow the random ordering of data between these bounds easier since they are all of a known size (the only non-standard sized chunk being the final one. . .).

Visualization