WRPC Protocol // Alon Horesh & Ilai Keinan

1. **Primary Goal**
   The main goal of the protocol is to allow for a smooth experience of controlling remote computers over the network.

2. **Messages** ([ ] = seperated, [ ] = raw, [ ] = none, | = raw null byte)

   Requests (Client -> Server):
   - `SSRQ` `[TCP]` Screenshot `[ ]`
   - `FLRQ` `[TCP]` File Content `[file_path]`
   - `LSRQ` `[TCP]` File List `[path=.]`
   - `CPRQ` `[TCP]` Copy File `[src_path, dst_path]`
   - `MVRQ` `[TCP]` Move File `[src_path, dst_path]`
   - `RMRQ` `[TCP]` Remove File `[file_path]`
   - `SCRQ` `[TCP]` Screen Control `[ ]`
   - `SWRQ` `[TCP]` Screen Watch `[ ]`
   - `CMDR` `[TCP]` Command Run `[command]`

   Actions (Client -> Server):
   - `SCIN` `[UDP/TCP]` Screen Control Input `[ ]`
   - `UPCK` `[TCP]` File Upload Chunk `[file_path | total_chunks | chunk_data]`

   Actions (Client <- Server):
   - `SCFR` `[TCP]` Screen Frame `[frame_data]`

   Actions (Client <-> Server):
   - `DNSC` `[TCP]` Disconnect Screen Control `[ ]`
   - `DNSW` `[TCP]` Disconnect Screen Watch `[ ]`
   - `CLOS` `[TCP]` Close Connection `[ ]`

   Responses (Client <- Server):
   - `SDON` `[TCP]` Screenshot Done `[ ]`
   - `DNCK` `[TCP]` File Download Chunk `[tid | i | total_chunks | chunk_data]`
   - `FOLL` `[TCP]` File List `[pickle_data]`
   - `ACSC` `[TCP]` Screen Control Ready `[width | height]`
   - `ACSW` `[TCP]` Screen Watch Ready `[width | height]`
   - `CMDO` `[TCP]` Command Ran `[output_string]`
   - `SUCC` `[TCP]` Operation Success `[ ]`
   - `ERRR` `[TCP]` Operation Failed `[ ]`

3. **Formatting**
   In WRPC, there are 2 ways to format the message - split into parts, or raw data.

   For splitting, the separator is a null byte `b”\0”`, which is purely `0000 0000`. The data will be split by the null byte, where the first part is the message code. For example, requesting to move a file from `/orig/path` to `/new/path`, the message would look like this:

   ```
   MVRQ0/orig/path0/new/path
   ```

   For raw data, the data that will be sent is gonna be split once, only between the message code and the actual data. For example, receiving a screen frame with the (very imaginary) data `05F3C9DDC0` would look like:

   ```
   SCFR005F3C9DDC0
   ```

4. **Size of Size**
   In the beginning of **ALL** messages, there will be 4 bytes which will represent the amount of bytes the message has. The receiving end should read it first, and then receive the rest of the data accordingly.

5. **Encoding/Decoding**
   There are a couple rules for encoding strings and numbers in WRPC.

   The size of size will be encoded to 4 bytes, using struct `I` (unsigned integer).

   Numbers sent through a split format, will be simply transformed into bytes:

   46 → 00101110

   And numbers sent through a raw format, will sometimes require padding, which can be easily done by explicitly specifying the amount of bytes `n.to_bytes(2)` ← 2 bytes.

   $46^{(2)}$ → 00000000 00101110

6. **Ports**
   There are 4 open ports used for communication:

34981	[TCP]	ALL	PRIMARY
34982	[TCP]	SCFR	SCREEN FRAMES
34983	[UDP]	SCIN	MOUSE INPUT
34984	[TCP]	SCIN	KEYBOARD INPUT

7. **Screen Frame**
   Encoded B-Frames, I-Frames and P-Frames, encoded with the h264 codec are sent via the `SCFR` packet.

   The implementation is highly optimized for live video transmission and was thoroughly modified to allow for a live, non-stuttery, smooth feed.

8. **Chunks**
   When sending files over the network, their content is split into chunks, in order to allow for a fast and reliable transfer. If we were to send the whole file once, and some of the data was lost or corrupted on the way, we would have to send the whole file again.

   WRPC splits files into chunks of 8192 bytes, which was tested to be the best chunk size for file transfers in TCP, within our system.

9. **Event System**
   Our system takes advantage of the asynchronous nature of internet transfers. We have created an event system which allows us to cleanly set callbacks for each type of incoming packet.

10. **Error Codes**
    `0` = Unknown Error
    `1` = File Not Found
    `2` = Bad Path