Protocols document of sending program task and files materials via TCP

Adopting asynchronous sockets to speeding up sending; there is no command answer during the file sending process, except receiving one terminal side finish answer before and after sending.

Computer side is TCP client side, controller is server side.

1. Create sockets.

2. connecting through controller IP address, default port is 3333

Antecedent command: check the file id list that pending for download 【user sending to controller】

{"\_type":"consult","proName":"programTask.json", "idList":["id1","id2",...],"proName":"program.json","zVer":"xixun1"}

User sending above commands to controller, idList contains all files id pending for download in the programTask

Controller return data to users 【controller send to user】in below:

{"\_type":"consult", "idList":["id1","id2",...]}

While idList is the files id pending for download, users will send files according to this idlist to avoid sending the same files repeatedly.

1. sending program task start command 【users send to controller】

{"\_type":"proStart","proName":"programTask.json","proSize":102828,"zVer":"xixun1"}

3，Sending all files in loop including programTask.json file and material files.

4.1 Sending file start command [ users send to controller]

{"\_type":"fileStart","id":"abc.xxx","relative\_path":"","size":3191,"zVer":"xixun1"}

4.2. Sending file data, no protocols, Raw data from beginning to end of file

4.3. Sending file end command [ users send to controller]

{"\_type":"fileEnd","id":"abc.xxx","zVer":"xixun1"}

1. Sending program task complete command [users send to controller]

{"\_type":"proEnd","proName":"programTask.json","zVer":"xixun1"}

6. {"\_type":"AckSuccess"}Receiving terminal side complete notification and then to exit thread, in this way, will avoid the thread being exited and closed sockets before server side not getting completely and caused server side get incomplete data. [controller send to users]

7.Close sockets and exit thread

**Controller as TCP server**

Adopting asynchronous sockets, receiving thread cache and then analysis the buf data that read, the processing function needs to solve the problems of sticking package, subcontracting package and breaking package.

please check the procedure in below:

1.Create service sockets, build up monitor, default port is 3333

2. Receiving the connection and create receiver sockets

1. Receiving the programtask start command, “proName” key value is the current program task name

{"\_type":"proStart","proName":"programTask.json","proSize":102828,"zVer":"xixun1"}

4.接收所有文件 receiving all files

4.1接收文件开始命令，创建文件（按控制卡app的需求存放文件的根路径，注意"relative\_path"相对于根路径而言，“”表示在根路径下，"/234234344"表示根路径下的234234344文件夹下）

{"\_type":"fileStart","id":"abc.xxx","relative\_path":"","size":3191,"zVer":"xixun1"}

4.2.接收文件数据,无协议，文件开始到结束的裸数据

4.3.发送文件结束命令

{"\_type":"fileEnd","id":"abc.xxx","zVer":"xixun1"}

5.接收节目任务完成命令，控制卡可以根据该协议中的"proName"的键值，去寻找节目任务文件

{"\_type":"proEnd","proName":"programTask.json","zVer":"xixun1"}

6.发送设备端完成通知,方便计算机端退出线程,

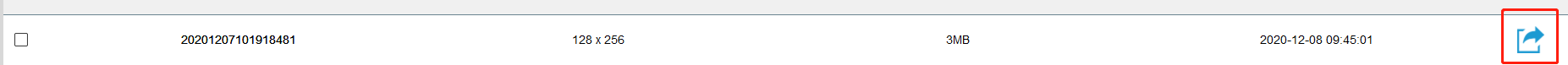
{"\_type":"AckSuccess"}

7.发送完毕后服务器端线程会关闭套接字，设备端被动关闭套接字，做好套接字关闭操作，比如退出读线程等.

8.节目json数据格式参考《xixunPlayer节目JSON说明文档.docx》

**必须先下载官网全的，再安装文件里边定制ledok Express。**

**在ledok中制作节目，点击U盘更新，导出节目的zip包（密码请默认为空或使用888 否则需要给卡设置相同的密码）**



**zip包节目下发流程：**

1，发送一个准备发送zip包的通知，告知zip文件名和文件大小

{"\_type":"fileStart","id":"111.zip","relative\_path":"","size":141349,"zVer":"xixun1"}

2，tcp数据流传输文件正文，不用封包，直接传二进制数据即可

3，发送文件传输结束通知

{"\_type":"fileEnd","id":"111.zip","zVer":"xixun1"}

4，发送播放zip文件的通知，播放器开始解压后播放zip包中的内容

{"\_type":"playZipTask","proName":"111.zip","zVer":"xixun1"}

5，删除节目

{"\_type":"DelPrograms","\_id": (如: 98e8d3cd47fad6ce8e3f7b8d42cb4d9b)","zVer":"xixun1"}

测试流程：

测试所需通信软件为NetAssist.exe

节目压缩包为111.zip

第一步



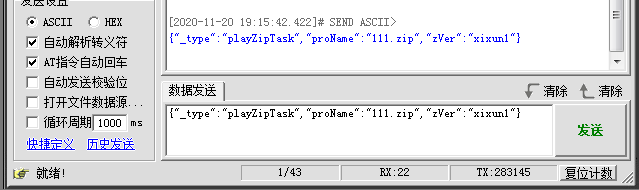
第二步，选择打开测试文件111.zip（其中包含两个节目，各显示一张图片）



第三步 发送文件传输结束命令



第四步：发送开始解压播放命令



完成。