Protocol Specification Document

Introduction:

The purpose of the protocol is to provide for quick and reliable communication between the server, renderer, and controller of a media consumption service. This protocol will allow the controller to request a list of media files from the server, then the controller can request the renderer to render the chosen file. Upon receiving this request from the controller, the renderer sends a request to the server so that the server can stream the chosen media file to the renderer for rendering.

General Considerations:

1. The renderer does not have the capability to buffer so it just renders what it receives from the server.
2. During the streaming session, controller can request the renderer to pause/resume/start-from -the-beginning the streaming.
3. The controller, renderer, and server must run on different hosts.
4. For media file types, at minimum text files must be supported.

Request Media File List:

The controller creates a payload message and encodes it into JSON. This message is then sent to the server. The server, which is listening on the port 1234 for messages, receives the message over the socket it has established, and decodes it. The server then checks the type field of the JSON message to differentiate the message from other messages. Once the server has checked the message type, it will print the list of the files convert the list into JSON and send the JSON over the socket. The controller then will receive the data, parse it into JSON format and print it.

Request Renderer to render the chosen file:

When requesting a file, the controller first asks the user to input the name of the file to be requested. The controller then creates a JSON message of type request with the content being the filename the user requested. This JSON message is then sent across a TCP Socket to the renderer’s address and port 1024. Upon receiving the message on its socket, the renderer decodes the message and checks the message type in order to determine what action should be taken. After verifying that the message type is a request, the renderer encodes the message again and establishes a new socket for transmission to the server. The renderer then connects the socket to the server’s address and port 1024 and sends the payload to the server and waits for a response. The server, upon receiving the message on port 1024, once again checks the message type to determine what response to execute. After verifying that the message type is a request, the server establishes the file to be returned from the content of the message. The server then will go to its data folder and open the file. The server will then create a JSON message of type response containing the file name and the file’s contents. This will then be sent over the connection socket. The renderer then receives this information over port 2048. Upon checking that an error message was not returned, the renderer will extract the file name and content from the JSON message and prints it.