Kunal Mukherjee

10/12/17

Networks CS 475

Mr. Mark Randall

Application Protocol

The application protocol has been named “The Kunal Awesome Protocol”, or sometimes referred to as TKA protocol. TKA protocol has two major parts: client side and the server side. The overview is that client connects to the server asking for the weather for a particular city. The server looks at what city it wants and it connects to the wunderground server requesting the data for the city. The server receives the data, parses through the data and sends it back to client for the user.

The client side is a straight forward. The program starts with ./main and the port number of the server, where it will try to make a connection. The client side connects to the server and sends data regarding its choice of which city it wants to request the data of. For example, sending “1” to get the weather and 5 day forecast for Evansville. The client then sends this data to the server and waits for the response. Once, it gets the response, the client continues to read though it until the connection is closed from the other side. After the reply is completely read, the client closes the socket and asks the user what to do next. If the user chooses to exit, the client application closes cleanly. If the user chooses to get weather condition data for a different city, a new socket is again open and the data containing the choice is sent to server. The client then keeps on waiting for the reply from the server and the process continues until the user commands the exit. The TKA protocol takes into effect that both the server and client run on TKA protocol, so the buffer that is received by the client has “$” in place of “\n”. Therefore, this conversion is done in the client side.

The TKA server protocol has two aspects to it. One regarding connecting to wundergound server and the other dealing with the tradition connection to the client relationship. Once the server is started it binds its socket to port number “60100” and keeps on listening for active connection from clients. Once the connection is made it looks for a response from the client. It looks in the receive buffer to determine what the client wants to do. After it determine what the client wants to do, it decided which choice to select. For example, if the client sends “1”, meaning the client is inquiring about the weather of Evansville. The server goes into its loop and now opens a new socket and connects it with the wunderground server and requests the xml file containing the data for this city. It keeps on receiving the data and opens a file to write the xml data. It keeps on writing it into a file. Once the whole data is received, the server terminates its socket with the wundergound server. Then the server used a library called MarkUp to help parse through the xml file and decipher the relevant data. It opens and writes the data to another parsed data file. The penultimate part is, the server now sends each line in the parsed data file and appending “$” after each line to the client. After the whole document is sent, the client socket is the terminated and all the files are closed. The server then again starts to actively listen for another connection and do the process over again.