Flask (Header parsing)

General Information & Licensing

Code Repository	https://github.com/miguelgrinberg/Flask-SocketIO, https://github.com/eventlet/eventlet
License Type	BSD 3-Clause License, MIT License
License Description	- Allows commercial and private use. - Permits modification and redistribution - Projects containing work under this license can be distributed under other licenses
License Restrictions	- Copyright notice stating original creator must be included - No warranty is given to users of the project - Creator assumes no liability

Magic ★★˚·˙˚ ☽ ⌒ ᨀ˚·★ ≋✦ ≈

Headers are metadata for http connections which do not get sent until the TCP connection has already been established. Header parsing is used extensively throughout our project and is accessed through flask. Although we used flask to build our web app, flask is a framework built upon other libraries to make development easier. Our server is run through flask-socketio, a library for easy use of the socketio library within flask web applications, which then calls eventlet (A library for concurrent networking) to start a WSGI (Web Server Gateway Interface) server. WSGI is a set of standards for python web applications that allows web apps and webservers to communicate.

This takes place in our code when we call "socketio.run" in our main function which leads to line 553 of flask-socketio.
(https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L553)

Flask-socketio then starts the server on line 679 by calling "eventlet.wsgi.server"
(https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L679)

This call creates the server which creates an object from class HttpProtocol in its function arguments. Upon initialization, this HttpProtocol object handles incoming connections via "self.handle"
(https://github.com/eventlet/eventlet/blob/85290a11dd5c5f511a45f4010b512eedfef87000/eventlet/wsgi.py#L350)

"self.handle" begins an loop where it calls "self.handle_one_request" until the server is closed.
(https://github.com/eventlet/eventlet/blob/85290a11dd5c5f511a45f4010b512eedfef87000/eventlet/wsgi.py#L383)

"handle_one_request" then tries to read the headers, by creating a "FileObjectForHeaders" from self.rfile which is the buffer of incoming data.
(https://github.com/eventlet/eventlet/blob/85290a11dd5c5f511a45f4010b512eedfef87000/eventlet/wsgi.py#L423)

"self.rfile" is defined as "conn.makefile" which is created elsewhere in a eventlet custom socket class
(https://github.com/eventlet/eventlet/blob/85290a11dd5c5f511a45f4010b512eedfef87000/eventlet/wsgi.py#L367)

The custom socket class calls "makefile" from the original python socket class.
(https://github.com/eventlet/eventlet/blob/85290a11dd5c5f511a45f4010b512eedfef87000/eventlet/greenio/base.py#L312)

"socket.makefile" creates SocketIO object called raw
(https://github.com/python/cpython/blob/7a0f3c1d92ef0768e082ace19d970b0ef12e7346/Lib/socket.py#L323)

"SocketIO" inherits from "io.RawIOBase" (https://github.com/python/cpython/blob/7a0f3c1d92ef0768e082ace19d970b0ef12e7346/Lib/socket.py#L664)

This inheritance means it defines it's own "readinto" method which calls "recv_into" (https://github.com/python/cpython/blob/7a0f3c1d92ef0768e082ace19d970b0ef12e7346/Lib/socket.py#L707)

recv_into is a similar method to recv, but it stores the bytes into a buffer rather than giving a new object. This is reading directly from the TCP socket between the client and server and is the lowest level of communication given by the python socket library.