**Python Websocket Libraries Comparison Results**

For this case study, Python libraries Websockets and AioHttp have been compared in terms of performance. Websockets was chosen because it is one of the most used libraries for socket implementations in Python and there are many examples on how to implement client-server architecture with this library. On the other hand AioHttp, promises higher performance according to benchmark tests and reseach, and there exists through documentation about how to use the library, making it easier to lear compared to other choices.

A server that sends a specific file, an excel file in this case, a number of times specified by the client has been implemented with both libraries. After the connection is established between the client and the server, the client sends a message indicating how many time it wants to receive the file. Then the file is sent exactly that many times from the server to the client. The client side records how long the full connection lasts as well as how long the file transmission takes. The file sent during the testing process is and excel file of size 273 KB.

The device used to test this client-server implementation has the following specifications:

* Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
* 16.0 GB RAM
* Windows 10

For the server implementation a single thread has been utilized. After implementing the server, two test scenarios have been created, first aiming to measure the latency and throughput of the file transfer, and the second aiming to measure the scalability of each implementation.

**Test 1: Latency and Throughput**

For this test, the clients request to receive the file 100, 200, 300, 400 and 500 times from the server. The transmission time including and excluding the connection establishment is recorded.

A graph of a chart

Description automatically generated with medium confidence A graph of a diagram

Description automatically generated with medium confidence

When 100 files are requested by the client, AioHttp server takes 3.45 seconds and Websockets server takes 5.67 seconds to transmit the files. AioHttp server reaches throughput of 7893 KBps and the Websockets server reaches throughput of 4813 KBps. The results can be seen in above graphs.

A graph of a number of files

Description automatically generated

The above graph shows the latency vs number of files transmitted for both implementations including the initial handshake and the file request time. From this graph we can clearly see that AioHttp is a lot faster than Websockets library, taking almost half the time. Additionally, the growth rate of latency with AioHttp seems to be smaller than Websockets, promising much better performance as the data size increases.

A graph of a number of files

Description automatically generated A graph with red and blue lines

Description automatically generated

The graph on the left shows only the file transmission time, excluding the initial handshake and the request sent to specify the number of times the file should be sent. AioHttp again performs better than WebSockets. When we subtract the values from previous latency graphs, we get the graph on the right showing the time it takes for the connection to be established and the file request message to be received. As the values shown are significantly small compared to the file transaction, the fluctuations in this graph do not affect the results of the experiment.

**Test 2: Scalability**

In order to measure the scalability of both implementations, the servers have been tested with a python script that performs a Distributed Denial of Service (DDOS) attack. For this test, multiple threads have been used in the client side. AioHttp server implementation could hold up to 850 threads, whereas Websockets server implementation could hold up to 175 threads. The thresholds from this experiment were determined by using trial and error techniques. When higher number of threads were utilized, both clients and servers reported timeout errors and connection establishment errors.

**Conlusion**

According to the results of the experiments, AioHttp generally performs better than Websockets in terms of throughput, latency and scalability. In perspective, after using AioHttp library instead of Websockets for socket implementation, throughput incresed by %63, latency decreased by %40 and the scalability increased by %385 resulting in an overall better performance.

**Resources:**

1. <https://deliverypdf.ssrn.com/delivery.php?ID=820106101027027099082075100123077122037089089042053026107098122081072127064017023108060016059063040047117002112066014089098075007082044064086003005081014120065102092030015033096119071020083085118081024120074123078071124124114122105125075082110124123120&EXT=pdf&INDEX=TRUE>
2. <https://ville-karkkainen.medium.com/python-is-slow-wait-its-actually-fast-6d2e49621b1>
3. <https://github.com/ftobia/aiohttp-websockets-example/tree/master>
4. <https://docs.aiohttp.org/en/stable/web.html>
5. <https://blog.stackademic.com/websockets-in-python-e8f845d52640>
6. <https://www.neuralnine.com/code-a-ddos-script-in-python/>