Assignment 2: Socket Programming

Arpit Prasad and Akshat Bhasin
2022EE11837 and 2022EE31996
COL334: Computer Network

September 7, 2025

1 Part 1: Word Counting Client

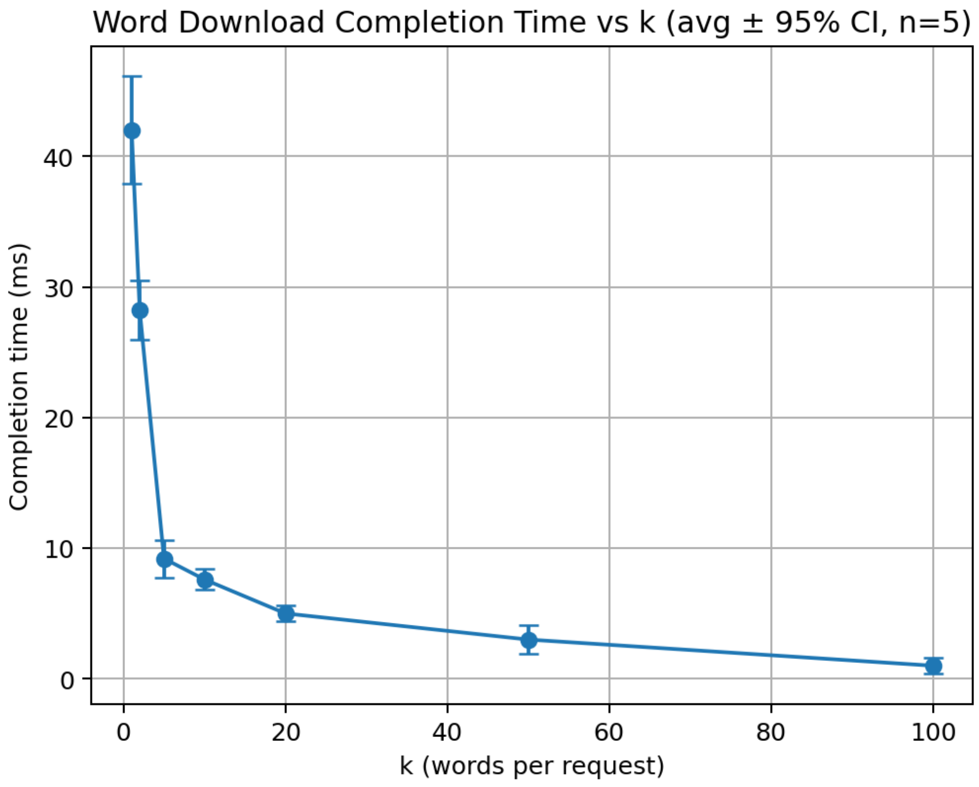

Figure 1: Completion Time vs Varying Size of chunks for file download

1.1 Implementation Details

1. **Client-Server Model**: The implementation follows a client-server architecture where the server listens for incoming connections and the client requests file downloads.

2. **Client Requests**: The client sends a request to the server specifying the file to be downloaded and the chunk size. The client downloads the entire file by continuously varying the offset and requesting chunks of the specified size until the entire file is received.

3. **Server Response**: The server reads the requested file in chunks of the specified size and sends each chunk back to the client.

1.2 Observations

1. **Trend in Graph**
   (a) **Decay**: As the chunk size increases, the completion time decreases.
   (b) **Reason**: This is because larger chunks reduce the overhead of multiple read and write operations, leading to more efficient data transfer.

2. **Trend in Variance**
   (a) **Decay in Variance**: The variance in completion time also decreases with increasing chunk size.
   (b) **Reason**: This is because larger chunks lead to more consistent transfer times, reducing the impact of network fluctuations and other transient factors.

3. **Optimal Chunk Size**
   (a) **Optimal Chunk Size**: There is a point of diminishing returns where increasing the chunk size further does not significantly reduce the completion time.
   (b) **Reason**: This is due to factors such as network latency and server processing time becoming the dominant contributors to total transfer time.

2 Part 2: Concurrent Word Counting Server

2.1 Implementation Details

1. **Concurrency Model**: The server uses a multi-threaded approach to handle multiple client requests simultaneously. Each client connection is handled in a separate thread.

2. **Client Requests**: Clients send a request to the server specifying the file to be processed and the chunk size. The client reads the entire file in chunks of the specified size and sends each chunk to the server for word counting.

3. **Server Response**: The server processes each chunk received from the client, counts the words, and sends the count back to the client.

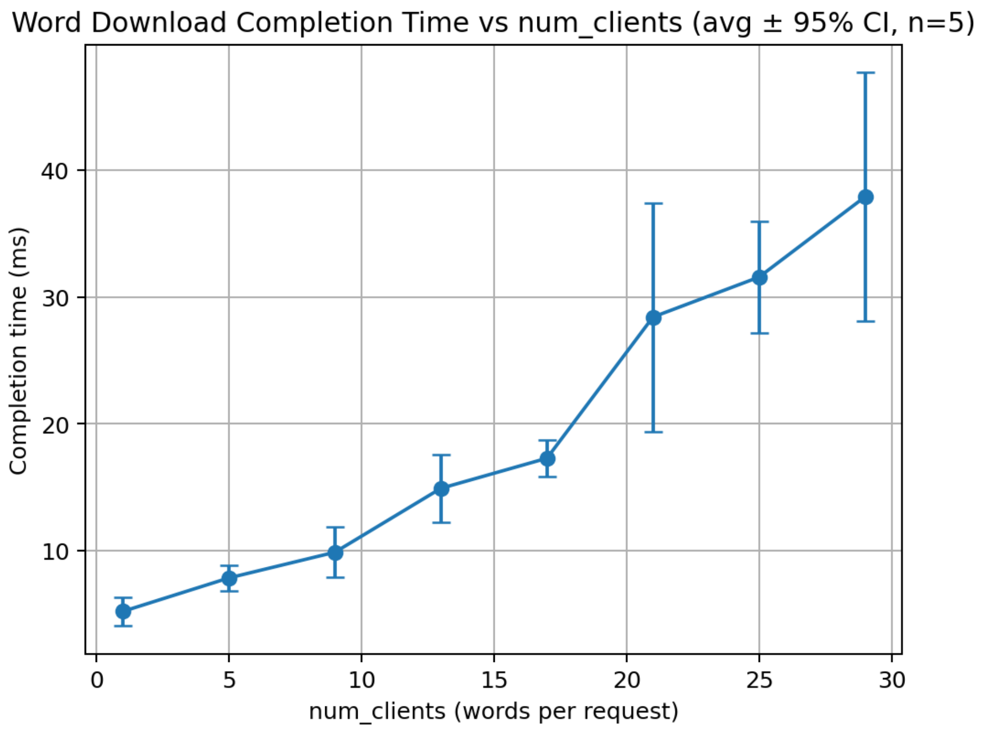

Figure 2: Average Completion Time vs Varying Number of Clients

2.2 Observations

1. **Trend in Graph**
   (a) **Increase**: As the number of clients increases, the average completion time also increases.
   (b) **Reason**: This is because the server has to handle more requests simultaneously, leading to increased contention for server resources.
2. **Trend in Variance**
   (a) **Increase in Variance**: The variance in completion time also increases with the number of clients.
   (b) **Reason**: This is due to the increased contention for server resources, leading to more variability in processing times.
3. **Scalability**
   (a) **Scalability Limitations**: There is a point where adding more clients leads to a significant increase in completion time.
   (b) **Reason**: This is due to the server reaching its maximum capacity for handling concurrent connections, leading to increased queuing delays.

3 Part 3: When Client Gets Greedy

3.1 Implementation Details

1. **Client Behavior**: One of the clients sends multiple requests to the server in back to back without waiting for the server to respond to previous requests.

2. **Server Handling**: The server processes each request as it arrives, but the rapid back to back requests can lead to increased wait times for other clients to utilise the servers resources.

3.2 Observations

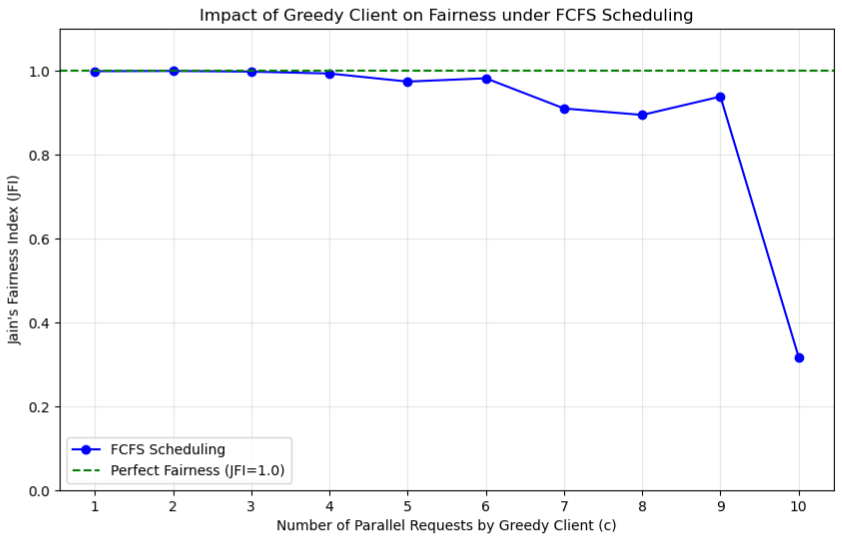

Figure 3: Jain's Fairness Index vs Number of back to back requests by a single client in a system of 10 Clients

1. **Trend in Graph**

   (a) **Decrease**: As the number of back to back requests by a single client increases, the Jain's Fairness Index decreases. However here the decrease is not very significant, in fact the handling of server seem almost fair.

   (b) **Reason**: Ideally, the fairness index should decrease as one client monopolizes server resources, leading to increased wait times for other clients. However, for small number of back to back requests, the server can still manage to handle requests from other clients in a timely manner, leading to a relatively high fairness index.

2. **Aymptotic Behaviour of Jain's Fairness Index**

   (a) **For larger c values**: For larger number of back to back requests, the fairness index would decrease more significantly as the greedy client would dominate server resources, leading to increased wait times for other clients. For $c = 100$ we obtained a JFI score of 0.25.
   (b) **Reason**: From the formula of JFI, if we take the limit of one client getting all the resources and others getting none, the JFI would tend to $1/n_{clients}$, where $n_{clients}$ is the number of clients in the topology. This is because both the numerator and denominator would be dominated by the resources allocated to the greedy client.

3. **Impact on Other Clients**
   (a) **Increased Wait Times**: Other clients experience increased wait times as the greedy client monopolizes server resources.
   (b) **Reason**: The server has to process multiple requests from the greedy client before it can attend to requests from other clients, leading to increased wait times.

4 Part 4: When the Server Enforces Fairness

4.1 Implementation Details

1. **Server Behavior**: The server implements a fairness mechanism to ensure that no single client can monopolize server resources. This is achieved by round-robin scheduling of client requests.

2. **Client Requests**: Clients send requests to the server specifying the file to be processed and the chunk size. The client reads the entire file in chunks of the specified size and sends each chunk to the server for word counting.

3. **Server Response**: The server processes each chunk received from the clients in a round-robin manner, counts the words, and sends the count back to the respective client.

4.2 Observations

1. **Trend in Graph**
   (a) **Stable JFI**: As the number of back to back requests by a single client increases, the Jain's Fairness Index remains relatively stable and high.
   (b) **Reason**: The round-robin scheduling ensures that each client gets a fair share of server resources, preventing any single client from monopolizing the server.
   (c) Slight decrease in JFI: There is a slight decrease in JFI as the number of back to back requests increases, but it is not significant.
   (d) Reason: This is because even with round-robin scheduling, a client sending a large number of back to back requests can still lead to increased wait times for other

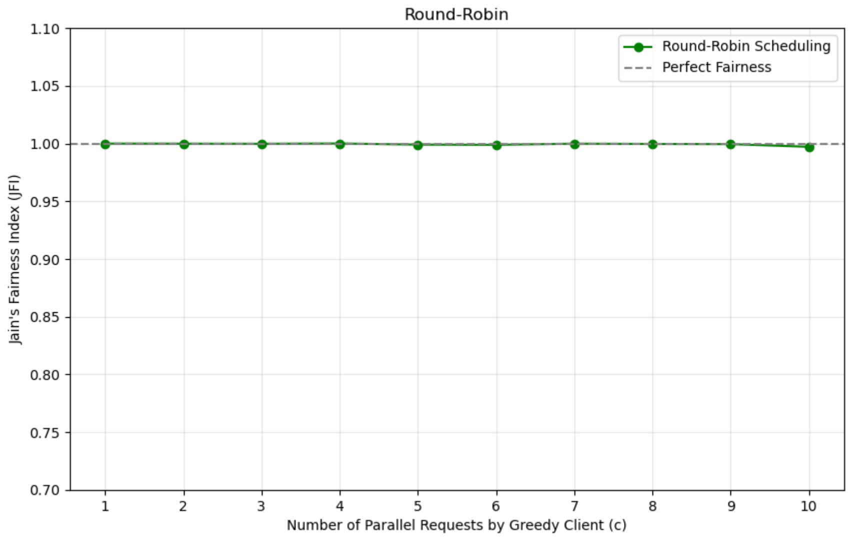

Figure 4: Jain's Fairness Index vs Number of back to back requests by a single client in a system of 10 Clients with server enforcing fairness with round robin scheduling

clients, but the impact is mitigated by the fairness mechanism. (Explained in Limitations of Round Robin Scheduling)

2. **Impact on Other Clients**
   (a) **Reduced Wait Times**: Other clients experience reduced wait times as the server enforces fairness.
   (b) **Reason**: The round-robin scheduling ensures that the server attends to requests from all clients in a timely manner, reducing wait times.
3. **Effectiveness of Fairness Mechanism**
   (a) **High JFI**: The Jain's Fairness Index remains high even with increasing back to back requests from a single client.
   (b) **Reason**: This indicates that the fairness mechanism is effective in ensuring equitable resource allocation among clients.
4. **Limitations of Round Robin Scheduling**
   (a) How can client still monopolize server resources: If a client sends a very large number of back to back requests, it can still lead to increased wait times for other clients, even with round-robin scheduling.
   (b) Reason: This is because the server has to process each request in turn, and a large number of requests from one client can lead to increased overall processing time, affecting the responsiveness for other clients.