COMP3334 – COMPUTER NETWORKS PROGRAMMING PROJECT, SPRING 2025

Load Balancing for Content Providers

08.05.2025

This project is due at 23:50 on **May 30th, 2025**

Project Demonstrations will be announced later

READ CAREFULLY !!!

This project will be done by a group of two students

1. Objective of this project

a. Experience team work and remote pair programming

This is a **team project**. Project teams were formed and announced at the Blackboard for this project. Pair programming (https://en.wikipedia.org/wiki/Pair_programming) has became a widely used methodology in many software companies. This project will give you a possibility to experience this technique.

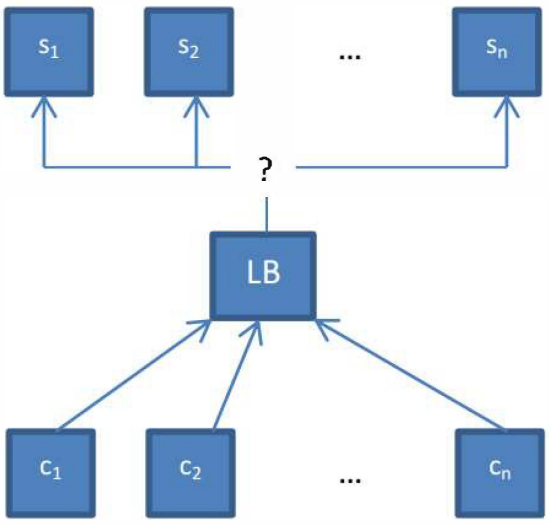
b. Self-directed learning of socket programming to create network applications

An introduction to socket programming was given in the PS section. However, the aim of this project is to improve your self-directed learning skills which is one of the key talents for a computer engineer. A couple of resources about the topic is provided below. You can also search for the related resources. However, you cannot take a ready solutions as your project implementation. You must spend some time to implement your own project. If you use any method (some piece of code) that is taken from the Internet, please state the resource clearly.

https://uwaterloo.ca/centre-for-teaching-excellence/catalogs/tip-sheets/self-directed-learning-four-step-process

2. Project Description

In many Internet services an important problem occurs when a server is overwhelmed with traffic and requests. For this project, your task is to solve this problem by creating a load balancer. The load balancer is an intermediate machine between clients and multiple servers which handle the user requests. To maximize the efficient use of servers, load balancer distributes the load (user requests) efficiently among different servers.



There are several ways in which **load balancing** can be performed. For example, requests that are expected to be quick may be handled by one server, while longer requests could be handled by another one. Another method is just randomly assigning a server to handle each request by taking the current load of the servers into account. There are many methods that can be used for load balancing.

3. Load Balancing Software

You can check the following commercial load balancer software. They also explain different methods for load balancing:

https://avinetworks.com/what-is-load-balancing/
https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques/
https://www.imperva.com/learn/availability/load-balancing-algorithms/

4. Which Load Balancing Method?

The following page explains different group of load balancing techniques:
https://www.cloudflare.com/learning/performance/types-of-load-balancing-algorithms/

In this project, you will implement one "**Dynamic load balancing algorithm"** and one **"Static load balancing algorithm".** When the server registers to a load balancer, it will choose the static or the dynamic strategy.

5. Requirements

Each server must support f our types of requests:
1. Directory listing (what content that server has: this is likely a fast request)
2. File transfer (transferring of a chosen file from server to client who is interested in: this could be fast or slow depending on the file size)
3. Computation (keeping server busy: this will be medium to slow)
4. Video streaming (constant bit rate for a given time)

The computation task will be simulated to make the assignment easier to test. The user/client should be able to send a computation request with a duration of a given number of seconds (e.g. 150 sec.). During this time, the server will remain "busy" and unable to handle new requests. In real life, this may be something like computing an RSA key (for encryption) or some other intensive request on a server.

The load balancer should be able to handle the **addition** or **removal** of servers at any time to handle increased or decreased load on the system. This means that servers must send "join" message to the load balancer when they want to serve client requests and also servers must either send a "goodbye" message to the load balancer when the server terminates, or the load balancer considers a server dead if a timeout occurs on a communication attempt.

The communication should occur as follows:
1. Server registers itself with load balancer (sends "`join`" message) and chooses a static or dynamic balancing method (example message types can be `join -v static` or `join -v dynamic`)
2. Client connects to load balancer with request
3. Load balancer estimates time to service request
4. Load balancer selects server to handle request
5. Server and client setup connection together and connection to load balancer and client is terminated (server and client directly communicate with each other, the load balancer only determines which server will serve for which client request)
6. At any time if server terminates, it should attempt to notify the load balancer, if this is not possible the load balancer should detect a dead server (through timeout)

6. Your programs

Implement the client, server and load balancing portion of the architecture. For grading purposes, there should be three executable files named "`client`", "`server`" and "`loadBalancer`". You can specify ports using command line arguments. Be sure to document (in a README file) which method you chose. You may use any transport protocol libraries in your project (such as TCP or UDP); choose the appropriate one according to your communication needs. You must **construct the packets** (data message, request message, register message, join message, leave message etc.) and related acknowledgements (join accepted message etc.) yourself, and interpret the incoming packets yourself.

As it is stated before, there are many methods that can be chosen for load balancing; therefore, make sure to document your method and include justification for your choice. For the highest grade, experiment with different load balancing techniques and provide some insight on which may perform best under certain circumstances.

7. Submitting your project

You should submit your project via Blackboard. You should submit a .zip file, which contains your code.

This .zip file should unpack to have the following structure

```
studentID/
README
client.java
server.java
loadBalancer.java
....
```

The plain-text (or can be a Word or PDF file) **README** file should be also included in your project**.** In this file, you should describe your load balancing strategy, high-level approach, a list of properties/features of your design that you think is good to mention, the challenges you faced, and an overview of how you tested your code**.**

README content
(1)Load Balancing Strategies you have implemented (one static, one dynamic)
(2)High Level Approach (which protocols that you use (TCP/UDP), which mechanisms you implement in applicaiton layer, a list of properties/features of your design)
(3)Challenges you have faced
(4)Testing
(5)How to run project