



# Fire Forecast

Socket programming

Name: Marina Magdy Rezkalla



## 1.0 Design:

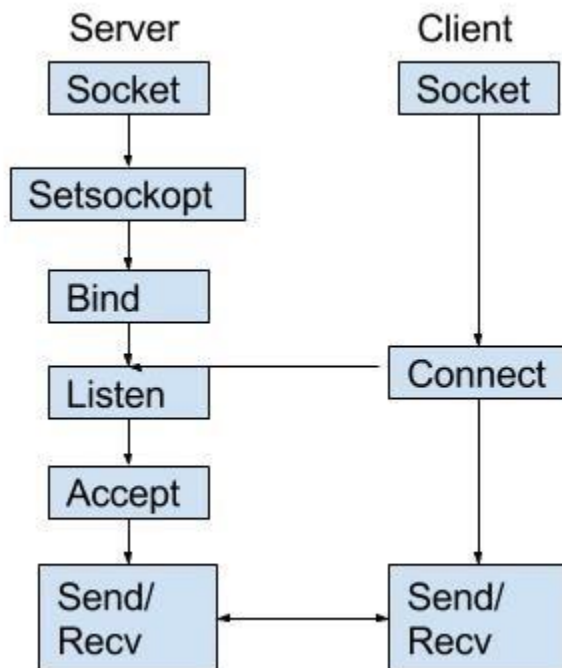
As the sensory node in the forest is the server and is sending temperatures every second for the client which calculates the average over time and the accumulation over time and prints them each 5 seconds that's why I used socket programming using C++.

Connection type:

I chose TCP connection:

Using: `SOCK_STREAM`

TCP is connection-oriented and enables two-way communication between two endpoints after the three-way handshake. TCP is reliable because the protocol ensures that all data is fully transmitted and can be assembled by the receiver in the correct order.



## 2.0 Requirements:

### 2.1 Utilize an automated build system

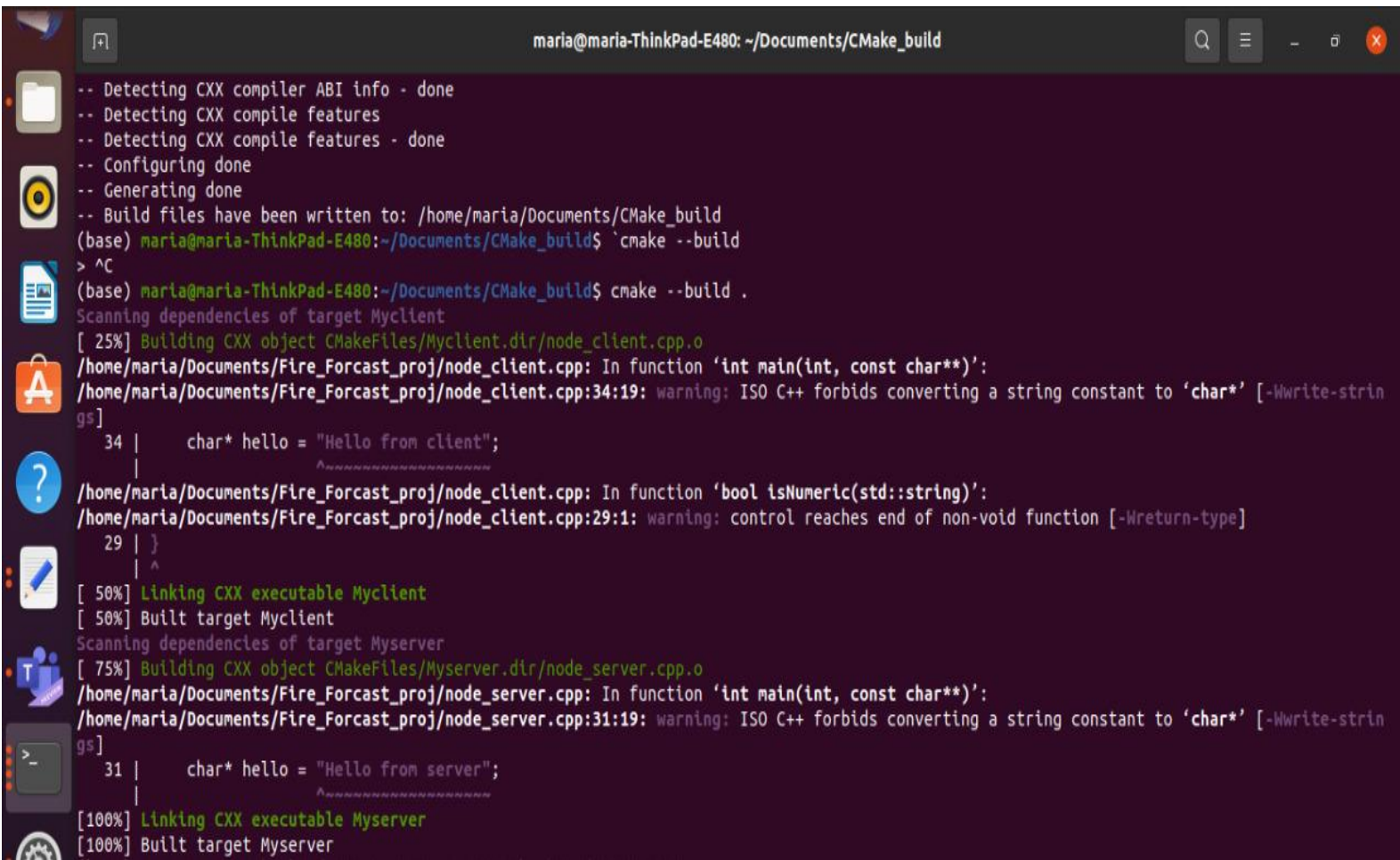
#### Build using CMake:

CMake is an open-source build system generator for software projects that allows developers to specify build parameters in a simple, portable, text file format. This file is then used by CMake to generate project files for native build tools including Integrated Development Environments (IDEs) such as Microsoft Visual Studio or Apple's Xcode, as well as UNIX, Linux, NMake, and Ninja. CMake handles the difficult aspects of building software such as cross-platform builds, system introspection, and user customized builds, in a simple manner that allows users to easily tailor builds for complex hardware and software systems.

```

(base) maria@maria-ThinkPad-E480:~/Documents/CMake_build$ cmake ../Fire_Forecast_proj
-- Setting build type to 'Debug' as none was specified.
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/maria/Documents/CMake_build
(base) maria@maria-ThinkPad-E480:~/Documents/CMake_build$ cmake --build
> ^C
(base) maria@maria-ThinkPad-E480:~/Documents/CMake_build$ cmake --build .
Scanning dependencies of target Myclient
[ 25%] Building CXX object CMakeFiles/Myclient.dir/node_client.cpp.o
/home/maria/Documents/Fire_Forecast_proj/node_client.cpp: In function 'int main(int, const char**)':
/home/maria/Documents/Fire_Forecast_proj/node_client.cpp:34:19: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
   34 |     char* hello = "Hello from client";
       |
/home/maria/Documents/Fire_Forecast_proj/node_client.cpp: In function 'bool isNumeric(std::string)':
/home/maria/Documents/Fire_Forecast_proj/node_client.cpp:29:1: warning: control reaches end of non-void function [-Wreturn-type]
   29 | }
       | ^
[ 50%] Linking CXX executable Myclient
[ 50%] Built target Myclient
Scanning dependencies of target Myserver
[ 75%] Building CXX object CMakeFiles/Myserver.dir/node_server.cpp.o
/home/maria/Documents/Fire_Forecast_proj/node_server.cpp: In function 'int main(int, const char**)':
/home/maria/Documents/Fire_Forecast_proj/node_server.cpp:31:19: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]

```



```
maria@maria-ThinkPad-E480: ~/Documents/CMake_build
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/maria/Documents/CMake_build
(base) maria@maria-ThinkPad-E480:~/Documents/CMake_build$ cmake --build
> ^C
(base) maria@maria-ThinkPad-E480:~/Documents/CMake_build$ cmake --build .
Scanning dependencies of target Myclient
[ 25%] Building CXX object CMakeFiles/Myclient.dir/node_client.cpp.o
/home/maria/Documents/Fire_Forecast_proj/node_client.cpp: In function 'int main(int, const char**)':
/home/maria/Documents/Fire_Forecast_proj/node_client.cpp:34:19: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
   34 |     char* hello = "Hello from client";
       |
/home/maria/Documents/Fire_Forecast_proj/node_client.cpp: In function 'bool isNumeric(std::string)':
/home/maria/Documents/Fire_Forecast_proj/node_client.cpp:29:1: warning: control reaches end of non-void function [-Wreturn-type]
   29 | }
       | ^
[ 50%] Linking CXX executable Myclient
[ 50%] Built target Myclient
Scanning dependencies of target Myserver
[ 75%] Building CXX object CMakeFiles/Myserver.dir/node_server.cpp.o
/home/maria/Documents/Fire_Forecast_proj/node_server.cpp: In function 'int main(int, const char**)':
/home/maria/Documents/Fire_Forecast_proj/node_server.cpp:31:19: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
   31 |     char* hello = "Hello from server";
       |
[100%] Linking CXX executable Myserver
[100%] Built target Myserver
```

2.2 The code shall use VCS:

I used GitHub and created a repo: <https://github.com/marina-magdy/Fire-Forecast-socket-programming>

The benefits of GitHub:

It's used for **storing, tracking, and collaborating on software projects**. It makes it easy for developers to share code files and collaborate with fellow developers on open-source projects. GitHub also serves as a social networking site where developers can openly network, collaborate, and pitch their work.

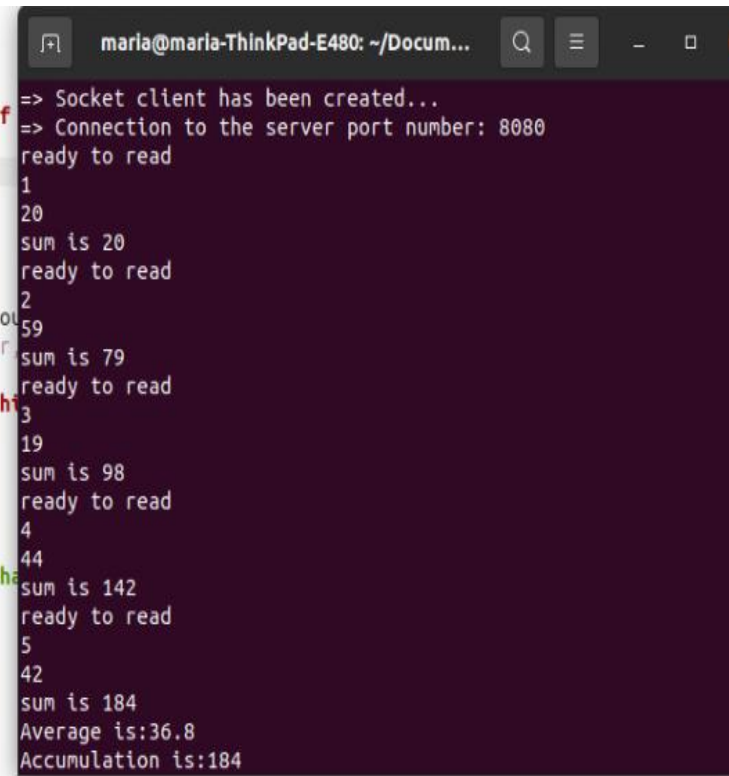
## 2.3 The code shall include tests:

The tests considered:

- 1- To print an error if socket isn't created or connection isn't established between client and server in both files.
- 2- To print an error in the client file if the received input isn't an integer.
- 3- To not proceed with calculations in the client file if value of recv function returns 0.

Test case:

Client:



```

f => Socket client has been created...
=> Connection to the server port number: 8080
ready to read
1
20
sum is 20
ready to read
2
59
sum is 79
ready to read
3
19
sum is 98
ready to read
4
44
sum is 142
ready to read
5
42
sum is 184
Average is:36.8
Accumulation is:184
```

The image shows a terminal window with a dark purple background. The window title is 'maria@maria-ThinkPad-E480: ~/Docum...'. The output of the program is as follows: '=> Socket client has been created...' followed by '=> Connection to the server port number: 8080'. Then, it enters a loop where it repeatedly prints 'ready to read', receives an integer input, calculates the sum, and prints it. The inputs are 1, 2, 3, 4, 5. The corresponding sums are 20, 79, 98, 142, 184. After the last input, it prints 'Average is:36.8' and 'Accumulation is:184'.

Server:

```
constant to 'char*' [-Wwrite-strings]
31 |     char* hello = "Hello from server";
    |
(base) maria@maria-ThinkPad-E480:~/Documents$ ./node_server
Socket server has been created...
Listening for clients...
the connection is established between client and server, and they are ready to transfer data
Temperature sent 20
Temperature sent 59
Temperature sent 19
Temperature sent 44
Temperature sent 42
Temperature sent 47
Temperature sent 47
Temperature sent 58
Temperature sent 55
Temperature sent 29
Temperature sent 15
Temperature sent 41
Temperature sent 30
Temperature sent 29
Temperature sent 39
(base) maria@maria-ThinkPad-E480:~/Documents$
```