Sensor network simulation

This document explains how and why the simulation works.

Methodology
- Reading thread
- Balloon thread
- Fault detection thread
- Sensors
  - Seismic reading
  - Seismic reading comparison
  - Sensor algorithm

Results
- metrics.txt
- alert_conclusive.txt
- alert_inconclusive.txt

Analysis
- Check between neighbour
- Check between sender and balloon

Methodology

The sensor network is formed as a M by N Cartesian grid of sensors, also called nodes. Each node is responsible for:

- generating sensor data which consists of: timestamp, location(coordinates), magnitude, depth
- comparing the results of adjacent nodes to its own results

The base station process is responsible for listening to any user input in the console. It compares the user input to the user-defined sentinel value. If the sentinel value was entered by the user then it triggers the shutdown procedure of the program. It is also responsible for creating 3 separate threads. These threads have separate purposes and responsibilities.

Reading thread

The reading thread polls for new data in the sensor network. In the event there is new data available it does the following routine:

1. Receives and unpacks the sensor data from the sensor and its neighbour
2. Fetches the balloon data from the shared buffer
3. Checks whether the balloon data matches the sensor or neighbour data
4. Updates the metrics
5. Writes the conclusive or inconclusive results to their corresponding files

The reading thread is also responsible for sending the shutdown messages to the sensor network to properly shutdown the network.

Balloon thread

The balloon thread is responsible for periodically generating sensor data to simulate the balloon sensor. It then fills the shared buffer with the generated data.

We created a struct that stores all the sensor data so that our implementation can be consistent throughout the project. The struct consists of a timestamp, coordinates, magnitude, and depth.

The base station has a communication error handler taking an error handler procedure. This routine is thread-safe, may be safely used by multiple threads without the need for any user-provided thread locks. The procedure hence provides the error code of the and terminates the system when a node throws an error.

Fault detection thread

The fault detection thread periodically receives messages from the sensors. If it does not receive a message from one sensor, it counts, if this count reaches a certain amount, this means that an error occurred for this particular sensor (errors are generated randomly for the simulation) and that the whole simulation has to end. So when a fault is detected, the fault detection thread tells the others and sensors to exit and exit itself.

Sensors

Each sensor is an MPI node, it has its own rank within the MPI_COMM_WORLD communicator, and is part of another communicator, that is, a cartesian grid of 2 dimensions, each of size M and N (given by the user at runtime).

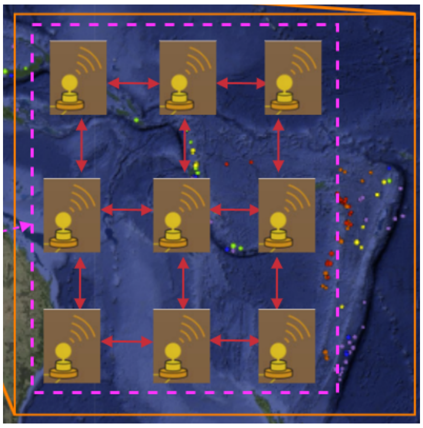
Seismic reading

A sensor periodically generates pseudo random seismic data.

This data is comprised of :

Timestamp, the exact time at which the reading was made.

Coordinates, latitude and longitude pseudo randomly generated within the range [-15, -45] for the latitude and [150, 180] for the longitude.
Each sensor covers an area depending on its cartesian coordinates and the number of sensors in the network, this means that sensor at position [0, 0] in a 9 nodes network will randomly generate coordinates between -15 and -25 for the latitude and between 150 and 160 for the longitude.



Magnitude, the strength of the recorded seismic activity (between 0 and 9.5).

Depth, length in meters down in the ground at which the seismic activity was recorded (between 0 and 900).

Seismic reading comparison

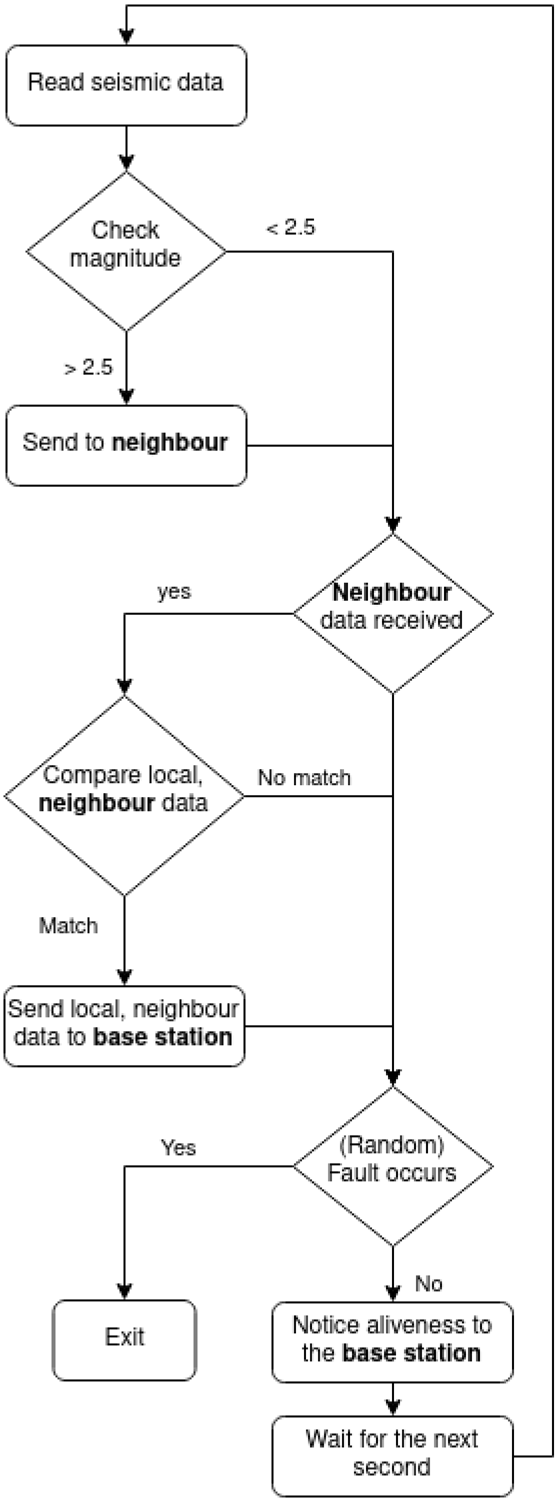
To ensure that a seismic reading is not a false positive, wecompare the data of two sensors.

This comparison is first based upon magnitude, if the two magnitudes are above 2.5 the data may be relevant.

Then the distance between the coordinates is calculated according to the functions provided in the subject, if this distance is less than 500km, then the seismic activity is considered relevant.

Sensor algorithm

The logic of the sensor is divided in several phases, repeated at every iteration of a loop, happening every second :


Read seismic data
Check magnitude
< 2.5
> 2.5
Send to **neighbour**
Neighbour data received
yes
Compare local, **neighbour** data
No match
Match
Send local, neighbour data to **base station**
(Random) Fault occurs
Yes
No
Exit
Notice aliveness to the **base station**
Wait for the next second

Results

The execution of our sensor network simulation results in 3 output files :

metrics.txt

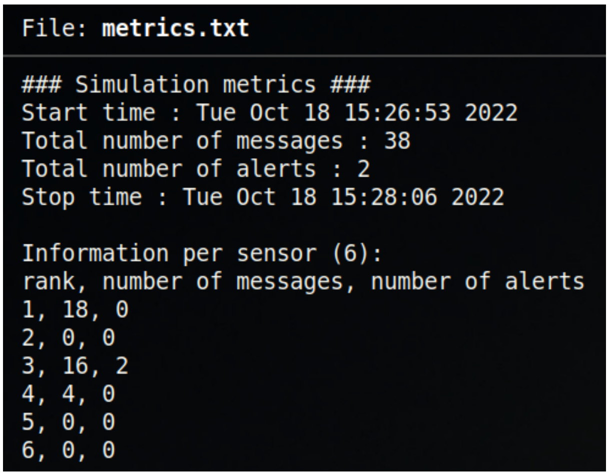
This file contains data about the overall simulation, from the point of view of the base station.

Start and stop time of the simulation are written in readable date format.

Total number of messages, this corresponds to the overall number of sensor readings that were sent from sensor to base station, this number increases by 2 every time a sensor has data in accordance with its immediate neighbour (as it will then send his and the neighbour's reading to the base station).

Total number of alerts is a subset of the total number of messages, as it is the number of times where the sensor reading received from the sensor is in accordance with the sensor reading from the balloon (it can be the reading from the sensor itself or from its neighbour).

Then, a **table of the number of messages and alerts per sensor**.



```
File: metrics.txt

### Simulation metrics ###
Start time : Tue Oct 18 15:26:53 2022
Total number of messages : 38
Total number of alerts : 2
Stop time : Tue Oct 18 15:28:06 2022

Information per sensor (6):
rank, number of messages, number of alerts
1, 18, 0
2, 0, 0
3, 16, 2
4, 4, 0
5, 0, 0
6, 0, 0
```

alert_conclusive.txt

This file contains a table of sensor readings that have been confirmed to announce an earthquake, that is, readings that have an immediate neighbour with a reading with magnitude superior to 2.5, and is closer than 500km from it and the same properties with the balloon reading.

The recorded information is as follows :

Sensor type, either

sensor : the one that sent the reading to the base station
neighbour : the one that sent to the sensor that sent to the base station
balloon : the balloon

Hour, exact hour, minute and seconds at which the sensor reading was read.

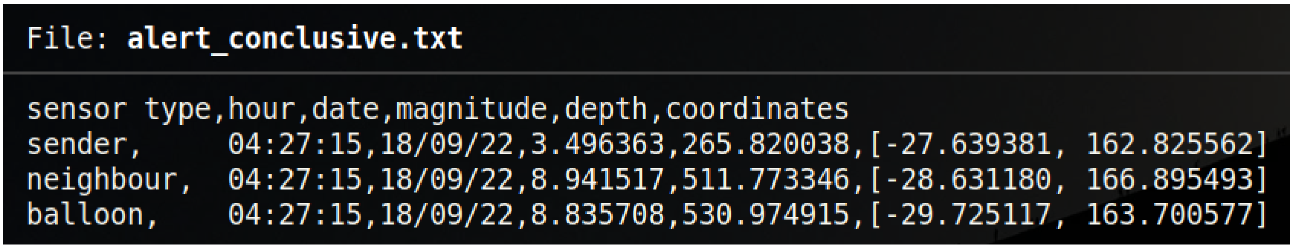
Date, exact day, month and year at which the sensor reading was read.

Magnitude, strength of the recorded earthquake.

Depth, length in meters down in the ground at which the earthquake occurred.

Coordinates, latitude and longitude at which the earthquake occurred.

The data is written in the file such that it can be interpreted as CSV.



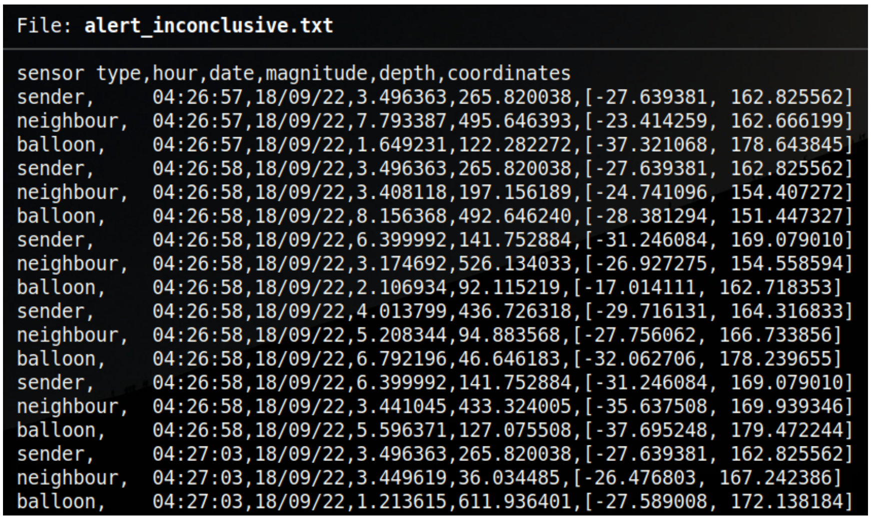
```
File: alert_conclusive.txt

sensor type,hour,date,magnitude,depth,coordinates
sender,     04:27:15,18/09/22,3.496363,265.820038,[-27.639381, 162.825562]
neighbour,  04:27:15,18/09/22,8.941517,511.773346,[-28.631180, 166.895493]
balloon,    04:27:15,18/09/22,8.835708,530.974915,[-29.725117, 163.700577]
```

alert_inconclusive.txt

This file contains a table of sensor readings that have been discarded, that is, readings that have an immediate neighbour with a reading with magnitude superior to 2.5, and is closer than 500km from it but not the same properties with the balloon reading.

The recorded information is written in the exact same format as of alert_conclusive.txt

The data is written in the file such that it can be interpreted as CSV.



```
File: alert_inconclusive.txt

sensor type,hour,date,magnitude,depth,coordinates
sender,     04:26:57,18/09/22,3.496363,265.820038,[-27.639381, 162.825562]
neighbour,  04:26:57,18/09/22,7.793387,495.646393,[-23.414259, 162.666199]
balloon,    04:26:57,18/09/22,1.649231,122.282272,[-37.321068, 178.643845]
sender,     04:26:58,18/09/22,3.496363,265.820038,[-27.639381, 162.825562]
neighbour,  04:26:58,18/09/22,3.408118,197.156189,[-24.741096, 154.407272]
balloon,    04:26:58,18/09/22,8.156368,492.646240,[-28.381294, 151.447327]
sender,     04:26:58,18/09/22,6.399992,141.752884,[-31.246084, 169.079010]
neighbour,  04:26:58,18/09/22,3.174692,526.134033,[-26.927275, 154.558594]
balloon,    04:26:58,18/09/22,2.106934,92.115219,[-17.014111, 162.718353]
sender,     04:26:58,18/09/22,4.013799,436.726318,[-29.716131, 164.316833]
neighbour,  04:26:58,18/09/22,5.208344,94.883568,[-27.756062, 166.733856]
balloon,    04:26:58,18/09/22,6.792196,46.646183,[-32.062706, 178.239655]
sender,     04:26:58,18/09/22,6.399992,141.752884,[-31.246084, 169.079010]
neighbour,  04:26:58,18/09/22,3.441045,433.324005,[-35.637508, 169.939346]
balloon,    04:26:58,18/09/22,5.596371,127.075508,[-37.695248, 179.472244]
sender,     04:27:03,18/09/22,3.496363,265.820038,[-27.639381, 162.825562]
neighbour,  04:27:03,18/09/22,3.449619,36.034485,[-26.476803, 167.242386]
balloon,    04:27:03,18/09/22,1.213615,611.936401,[-27.589008, 172.138184]
```

—

For further runtime output, adding the macro DEBUG to the compilation, will give print debug information about the running program. Simply add '-D DEBUG' to line 26 of CMakeLists.txt

```
target_compile_options(${SENSOR_NETWORK} PRIVATE -Wall -Wextra
-D DEBUG)
```

Grid Size	Result Metrics
2*2	### Simulation metrics ### Start time : Tue Oct 18 22:55:23 2022 Total number of messages : 4 Total number of alerts : 0 Stop time : Tue Oct 18 22:55:43 2022 Information per sensor (4): rank, number of messages, number of alerts 1, 0, 0 2, 4, 0 3, 0, 0 4, 0, 0
3*3	### Simulation metrics ### Start time : Tue Oct 18 23:02:03 2022 Total number of messages : 26 Total number of alerts : 0 Stop time : Tue Oct 18 23:02:21 2022 Information per sensor (9): rank, number of messages, number of alerts 1, 0, 0 2, 2, 0 3, 4, 0 4, 0, 0 5, 0, 0 6, 5, 0 7, 3, 0 8, 0, 0 9, 9, 0
4*4	### Simulation metrics ### Start time : Tue Oct 18 23:04:04 2022 Total number of messages : 36 Total number of alerts : 0 Stop time : Tue Oct 18 23:04:26 2022 Information per sensor (16): rank, number of messages, number of alerts 1, 0, 0 2, 1, 0 3, 1, 0 4, 4, 0 5, 3, 0 6, 2, 0 7, 4, 0 8, 2, 0 9, 2, 0 10, 2, 0 11, 2, 0 12, 3, 0 13, 2, 0 14, 3, 0 15, 2, 0 16, 3, 0
5*5	### Simulation metrics ### Start time : Tue Oct 18 23:05:00 2022 Total number of messages : 32 Total number of alerts : 0 Stop time : Tue Oct 18 23:05:19 2022 Information per sensor (25): rank, number of messages, number of alerts

	1, 0, 0 2, 1, 0 3, 1, 0 4, 1, 0 5, 2, 0 6, 1, 0 7, 1, 0 8, 2, 0 9, 1, 0 10, 1, 0 11, 2, 0 12, 2, 0 13, 0, 0 14, 2, 0 15, 3, 0 16, 0, 0 17, 3, 0 18, 3, 0 19, 0, 0 20, 2, 0 21, 1, 0 22, 0, 0 23, 1, 0 24, 2, 0 25, 0, 0

Results from a 6 core CPU

Analysis

At first our hypothesis was that there would be a rough 1:3 ratio between the amount of conclusive vs inconclusive alerts. This seemed realistic at the time because we thought that if an earthquake occurred at one position the sensors would be able to detect it with relatively good precision. This would then allow us to determine whether the alert was conclusive or not by comparing data across the sensor network. This however was not the case and we found that (based on the data in the previous section) that there was approximately a 2:38 ratio between the conclusive and inconclusive alerts.

The results varied with each run through of the simulation because of the randomness. However we see a general trend in the number of messages received by the base station which increases with the number of processes (grid size). As a disclaimer, these results were only obtained from one run of the program and do not fully represent the trend between the metrics and grid size.

From the results we also see that the spread of messages received are not all equal between the processes themselves. For some runs of the simulation, there were a few processes that sent messages to the base station a lot more than the other processes. We see this with the 3*3 grid where the 9th process messaged the base station 9 times while most of the other processes did not send any messages.

We observe that depending on the thresholds that we use for acceptable sensor data matches we see varying conclusiveness between the balloon and sensor network. Using a higher threshold yields more conclusive results and alerts from the base station and using a lower threshold yields more inconclusive results and less alerts. The threshold can be changed in the file “sensor/sensor_data_cmp.c” at line 32. This is the threshold for the maximum coherent distance

Check between neighbour

When any sensor generates a magnitude greater than 2.5 it sends its data to its adjacent neighbours. If the neighbour also generates a magnitude greater than 2.5 and sends its data to the first sensor then this could be a conclusive event. To check whether the data generated by the first sensor and the neighbouring sensor is conclusive we must check if the sensors are geographically close enough together. This threshold is defined in “sensor/sensor_data_cmp.c” at line 32. In the event they are conclusive the data is then sent to the base station

Check between sender and balloon

Once the base station receives the data from the sender and the sender’s neighbour it then compares the data to the balloon sensor data. For this alert to be conclusive we check whether the balloon data matches the sensors’ data.

In conclusion, the cross checking between the sensor network and balloon sensor provides a realistic model. Most of the time we find that the balloon sensor does not match with the sensor network and thus models an inaccurate balloon sensor if it were to be deployed in real life. The fact that there are not many conclusive alerts models a sensor network system where the sensors produce a random output