HW2 Wet- Elections in the USA

Your Mission:

The government of the United States wishes to shift to electronic voting, and asked YOU to assist.

To reduce the pressure on voters on the day of the elections, the period of voting is going to be extended and each person can vote as many time as s/he likes such that the last vote is the one that counts.

In the American system, a few candidates (typically 2-3) are nominated for election. Each state is allocated a given number of electors. At the end of the election, the winning candidate at each state gets the support of all the electors of that state, and the elected president is the one who obtained the overall highest number of electors.

Yes, as a result, sometimes the elected president is not the one who got the highest number of votes. Also, there are some nuances and exceptions to this rule that we will ignore here.

Detailed Description:

- Each state is treated as an independent shard, handled by its group of dedicated servers – more than one in order to ensure high availability and fault-tolerance.
- The list of eligible voters (clients) for each state in maintained in a dedicated file accessible to the servers.
- The election committee publishes in a dedicated file the names of the candidates (a small number) before the election period starts.
- The election committee publishes the addresses of all the servers in the system.
- A voter can contact any server in the system – not only the servers that are responsible for his/her state. If a voter contacts a server that does not belong to the voter's state, the server will forward it to the correct shard by contacting one of the servers in the respective shard.
- There is an RPC method to start\stop the election period.
- To support status reports as well as a final count, the election committee must be able to know what the votes' count is for each candidate at each state and the electors distribution at any given time (without stopping the voting) through a corresponding RPC method.
- You can ignore all security and authentication issues.

- At the end of the elections the winner is announced.
- The winner at each state is the one with the most votes.
- You can see the electoral votes allocated to each state at the end of this file.

Before delivering a solution, it is important to have a structure. You should write down every step in the process and document the necessary steps to implement your solution.

Environment Assumptions:

1. Each shard has its own pre-known set of nodes. New nodes are not allowed to join after the system was started.
2. At any shard with n nodes, there may be up to f < n/2 faulty nodes. The failure pattern is benign, meaning that a node may crash (but cannot behave maliciously). A crashed node stops sending messages and does not recover during the current run.
3. You can assume that initially all nodes are alive.
4. The system is asynchronous but you are allowed to use any type of failure detector (as long you implement it correctly).
5. For simplicity, you can assume that each client knows the address of all servers in advance, so there is no need to maintain a naming mechanism.
6. The communication between clients (voters) to servers should be done by REST.
7. The communication between servers should be done by gRPC.
8. Use ZooKeeper for implementing FDs, atomic broadcast, membership service and other distributed abstractions.

General Guidelines:

1. You should develop your system using Java (of any version you wish).
2. Use the techniques and principles you have learned in the lectures and tutorials.
3. You can choose inputs\outputs' formats as you see fit.
4. Assume the syntax is correct – no need for fancy parsers and checking for input errors. No need for fancy UI.
5. No need to save results after the system was shut down (no need for DB services).
6. You can test your system locally on your machine:
    a. Using different processes.
    b. Multiple VM's
    c. Dockers (this would give you a 5 points bonus)
7. Start testing with smaller amount of states (2) and slowly increase.
8. If your computer cannot cope with 50 states, it is OK to present with a smaller amount such as 5-10.

Grading details:

- You should submit your project in pairs. If you cannot find a partner contact the course staff for help.
- You should create a detailed **external documentation** that describes how you solved the exercise and, in particular, explain and argue your design choices.
- The system performance and design, as well as creativity and innovation, will play a major role in the grading consideration.
- Each team will present their project to the course staff in a **live demo** (dates to be assigned later on).

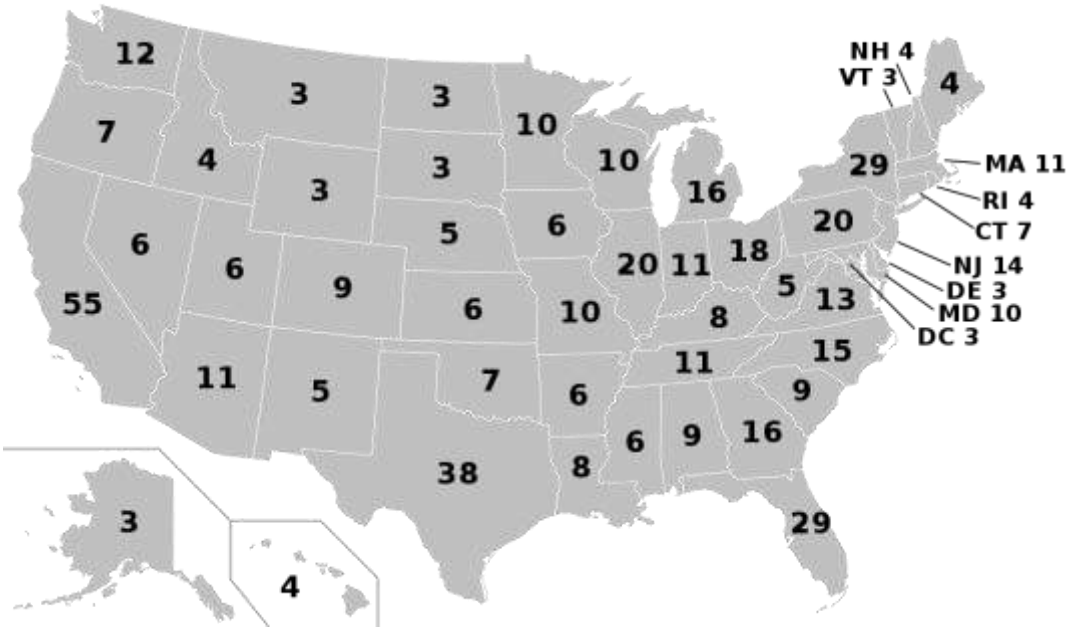
Submission:

Submission is in pairs only and using the electronic submission system.

You should submit hw2<name><name>.zip file containing:

1. project.pdf which includes:
   Documentation, analysis and justifications of your implementation.
2. src.zip which includes:
   All your system's source code.

Electoral votes allocated to each state:



Have a pleasant journey