

Performing Elections for OCP Applications

1. Description

This particular functionality aims to create and manage elections successfully. We have a dedicated server (to be thought of as a daemon running at some peer) which can assist the election handler by monitoring the execution of elections and computing the voting results.

2. Components

The server runs a particular protocol which consists of the following three states per election:

- The IDLE state: an election is at this state after its end.
- The REGISTRATION state: an election is in this state when voters are registering.
- The VOTING state: an election is in this state if a voting procedure is active.

All other components that participate in this functionality are clients which issue commands to the server. There are two different kinds of clients:

- Election Handlers: these are applications running on top of the OCP that use the server in order to perform conduction of elections efficiently. Commands that are sent to the server by an election handler include requesting/releasing the election server. They can send commands to the server of creating a new election. The information sent when creating a new election are the name (issue) of the election and a description of it, the voting method used to find the winner, the alternatives (candidates) that participate, the type of the election including open election (i.e. everybody can vote), registered voters election (i.e. voters need to be registered in order to vote), or a predefined electorate election (i.e. specific voters are allowed to vote).
- Voters: these are applications running on top of the OCP which request service from the server. They rank the candidates according to their preferences.

3. The protocol

The protocol has the following simple structure which consists of simple communication rounds. In each communication, the server waits for messages by a client (the election handler or the voter), processes them by performing the corresponding actions, and replies with another message. The messages that can be sent by the server are either OK or error messages (ERR).

There are two types of messages:

- Independent of election: these are messages independent of election id such as NEL and INF. These messages are independent of the state.
- Election dependent: these are messages dependent on election id and the first argument of them is the election id. These messages are dependent of the election state.

The messages that can be received by the server are presented in the following.

3.1 New election messages (NEL)

The message is issued by election handlers in order to create an election. When the type of election is open or predefined electorate then the voting procedure is started automatically. The message has the following format:

```
NEL#<issue> <description> <number of candidates> <cand_1>
<cand_2> ... <cand_m> <voting method> <who> <number of voters>
<voter id_1> <voter id_2> ... <voter id_m>
```

The server registers the corresponding election and replies with an OK message and the id of the election and the handler key. When the type of the election is open or predefined electorate then the election goes to VOTING state. Otherwise when the type of election is registered voters the election goes to REGISTRATION state.

3.2 Register messages (REG)

The message is issued by voters in order to register to the system and get their id.

```
REG#<election id>
```

Unless the election is in REGISTRATION state, the server replies with an error message (ERR). When in REGISTRATION state, the server registers the voter and replies with an OK message and the id of the voter.

3.3 Vote messages (VOT)

The message is issued by voters in order to vote in an election. The agent (voter) chooses the alternative(s) she prefers or ranks the alternatives according to her preference profile. The message has the following format:

```
VOT#<election id> <voter id> <number of candidates> <cand_1>
<cand_2>... <cand_m>
```

Unless the election is in VOTING state, the server replies with an error message (ERR). When in VOTING state, the server replies with an OK message.

3.4 Information request messages (INF)

This message is issued by voters in order to command the server to send them the available elections. The message has the following format:

```
INF#
```

The server replies with an OK message that contains the number of elections, election id, issue, description, number of candidates, candidates, voting method, type and voters ids (when type is predefined electorate) of the active elections.

3.5 End of Voting messages (END)

The message is issued by election handlers in order to command the server to end the current election and send them information on the result of an election. The message has the following format:

```
END#<election id> <handler key>
```

Unless the election is in VOTING state, the server replies with an error message (ERR). When in VOTING state, the server replies with an OK message and returns the results to election handler and set its state to IDLE.

3.6 Start voting messages (STV)

The message is issued by election handlers in order to start the voting procedure when the type of election is registered voters. The message has the following format:

```
STV#<election id> <handler key>
```

Unless the election is in REGISTRATION state, the server replies with an error message (ERR). When in REGISTRATION state, the server starts the corresponding election and election's state becomes VOTING and replies with an OK message.

3.7 Voting query messages (VRQ)

This message is issued by voters in order to command the server to send them information if they can participate in the specific election according the id provided. The message has the following format:

```
VRQ#<election id> <voter id>
```

Unless the election is in VOTING state, the server replies with an error message (ERR). When in VOTING state, the server replies with an OK message stating that the voter can vote for the election provided in the election id, otherwise replies with an ERR message stating that she can't vote.

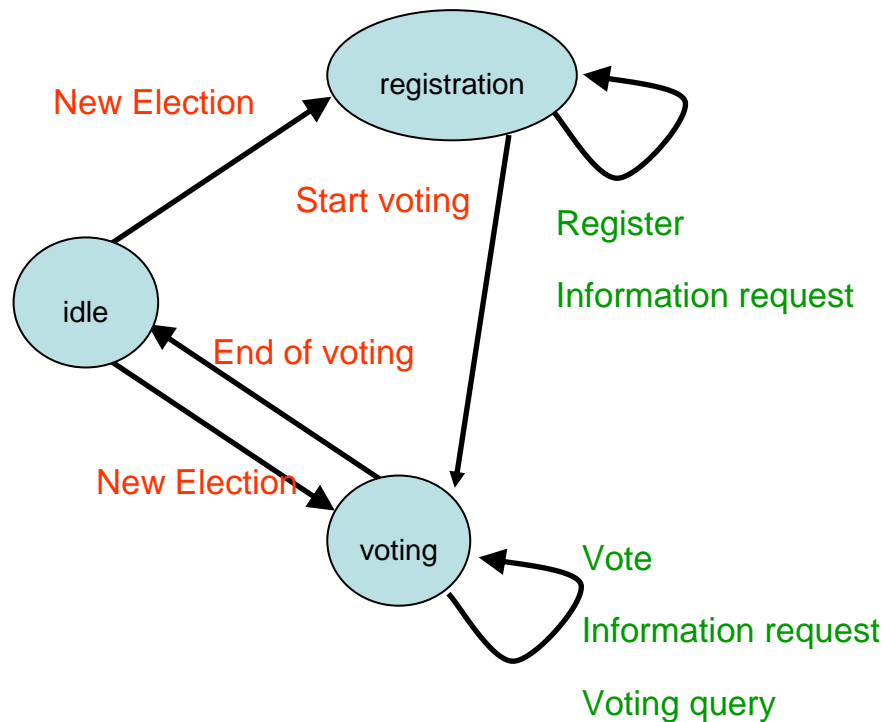


Figure 1. The protocol of the voting server. The diagram maps the procedure for every election. Red labels denote messages issued by election handlers, and green labels denote messages issued by voters. Arrows represent the change of state on success.

4. Description of Algorithms

In this section we are going to present a brief description of the algorithms that are used in order to find the preferred candidate.

4.1 Plurality

The voter must choose one candidate. The preferred candidate is assigned a score of 1 while the others 0. The candidate with the maximum score, i.e. the most votes, wins.

4.2 Approval

The voter must choose (approve of) as many of the candidates he wants. The preferred candidates are assigned a score of 1 while the others 0. The candidate with the maximum score, i.e. the most votes, wins.

4.3 Borda

This voting method is based on ranking the preferred candidates. Each voter ranks the candidates she prefers according to an increasing order, e.g. in her first preference she must insert 1, in her second preference 2 and so on. Then votes can be counted by giving each candidate a number of points equal to the number of candidates ranked lower than them. When there are m candidates, a candidate receives $m - 1$ points for a first preference, $m - 2$

for a second, and so on, with zero points for being ranked last. The candidate with the maximum points is the winner.

4.4 Tideman (Simplified Dodgson)

Also this method is based on ranking the preferred candidates. Each voter ranks the candidates she prefers according to an increasing order, e.g. in her first preference she must insert 1, in her second preference 2 and so on. Then we design the matrix of paired comparisons as follows. If a candidate e.g. a is above another candidate e.g. b in the preference of 3 voters then we fill the (a,b) cell of the matrix with 3. The (b,a) cell will have a value of $n-3$, with n being the number of voters. We do this for every candidate to fill the matrix of paired comparisons. We say that a candidate loses from another candidate if she is preferred to less than $n/2 + 1$ voters. So for every candidate (row of the matrix of paired comparisons) we sum the difference $(j,i) - (i,j)$, only for the cells (i,j) that have value less than $n/2 + 1$. Note that the sum is negative and the candidate with maximum score wins.

4.5 Random Dictator

In this method votes do not matter because a random candidate called the dictator is the winner of the election.

4.6 Veto

In this method the voter must choose the candidate she dislikes and sets a veto on her. The vetoed candidate is assigned a score of 1 while the others 0. The candidate with the minimum score is the winner of the election.