Digital Ecclesia: Towards an Online Direct-Democracy Framework

Dionysis Athanasopoulos
School of Engineering and Computer Science
Victoria University of Wellington, Wellington, New Zealand
dionysis.athanasopoulos@ecs.vuw.ac.nz

ABSTRACT

Citizens envision the transition from the representative democracy to the online direct democracy. Inspired by the ancient Athenians' direct democracy, we propose an initial version of the framework Digital Ecclesia. We model the Digital Ecclesia as a social network that offers dynamic and large-scale reachability of citizens. Citizens are dynamically notified to participate and vote on discussion topics of new working groups. To address scalability and privacy challenges, the architecture of the Digital Ecclesia is distributed, i.e. each node runs a local program with its own storage that executes the voting procedure in parallel with other nodes. Nodes communicate to each other via exchanging encrypted messages in a scalable manner. We model the voting procedure as a non-cooperative game and we specify an algorithm for employing the voting game in a distributed fashion. Finally, we conduct the preliminary evaluation of the algorithm on a corpus of real-world votes.

CCS CONCEPTS

•**Human-centered computing** → Collaborative and social computing systems and tools; •**Theory of computation** → Distributed algorithms;

KEYWORDS

Distributed message-passing model, game theory, scalability.

ACM Reference format:
Dionysis Athanasopoulos. 2018. Digital Ecclesia: Towards an Online Direct-Democracy Framework. In *Proceedings of 40th International Conference on Software Engineering: Software Track, Gothenburg, Sweden, May 27-June 3 2018 (ICSE-SEIS'18),* 4 pages.
DOI: 10.1145/3183428.3183432

1 INTRODUCTION

Citizens increasingly seek electronic ways to participate in the decision making, envisioning the transition from the representative democracy to the online direct democracy. Citizens' attempts have led to the adoption of online frameworks by public organizations (e.g. open ministry in Finland, participatory budgeting in Paris) [1].

Limitations of existing frameworks & emerging challenges. A framework supports the direct democracy if it enables the universal and direct citizens' participation [2], as was happening in Ecclesia (Greek: *εκκλησία του δήμου*) – the principal assembly of the ancient Athenians' direct democracy[1] [3]. Ecclesia was open to all citizens, regardless of their societal and financial class.

However, the existing online frameworks support the participatory democracy, since citizens use the frameworks on demand, i.e. citizens ask their participation in working groups. Thus, these frameworks do not support the *dynamic* and *large-scale* reachability of citizens by notifying them for participating in new working groups[2]. Though, time and message *scalability* challenges emerge if large-scale notification mechanisms are applied.

Moreover, the architecture of the existing (esp. e-voting[3]) frameworks raises *privacy* issues, since citizens access the frameworks via using simple user accounts. Consequently, the frameworks store citizens' votes on the server side, breaking citizens' privacy.

Contribution. To address the scalability and privacy challenges, we propose an initial version of the framework `Digital Ecclesia`. We model the Digital Ecclesia as a social network that offers *dynamic* and *large-scale reachability* of citizens [4]. In particular, network nodes, which offer high citizens' reachability, dynamically initiate the notification-message mechanism. The broadcast of messages is performed in a scalable manner, since nodes notify only their neighbours. To support voting privacy, the architecture of the Digital Ecclesia is *distributed* [5]. In particular, each node runs a local program with its own storage that executes the voting procedure in parallel with other nodes (a.k.a. locally stored votes). Nodes communicate to each other via exchanging encrypted messages.

As citizens' participation increases, the diversity of their votes grows, leading to vote conflicts. Ecclesia was resolving conflicts via compromising citizens' votes so that decisions were increasing the social welfare. Nash has modelled the social welfare as a mathematical function in the context of game theory [6]. Based on that, we define a *voting game on discussion topics* for making decisions that maximize the Nash social-welfare function.

According to [6], the requirements of a well-defined voting game are that citizens vote independently to each other and their votes are proportional to their preferences. To meet the requirement of the independent voting, we model the voting procedure as a non-cooperative game [7], since individuals' collaboration during voting is not usually permitted. On the contrary, cooperative games are based on the competition between groups of players. To meet the

ICSE-SEIS'18, Gothenburg, Sweden

978-1-4503-5661-9/18/05…$15.00
DOI: 10.1145/3183428.3183432

[1] It was the first recorded time in the history that the direct democracy was applied.
[2] We assume that citizens are always engaged to participate and vote when they receive notifications. This assumption is inspired by the Athenians' democracy in which anyone, notified for participation and not present in Ecclesia's meetings, was liable to a penalty.
[3] http://agoravoting.org

requirement that citizens' votes are proportional to their preferences, we adopt a metric that assesses the semantic similarity of voting topics. Finally, we specify an algorithm for employing the voting game in a distributed fashion and we conduct its preliminary evaluation on a corpus of real-world votes.

The rest of the paper is structured as follows. Section 2 defines the notion of the voting game. Section 3 specifies the distributed algorithm. Section 4 presents the preliminary evaluation of the algorithm. Section 5 describes related approaches. Section 6 summarizes our contribution and discusses its future directions.

2 VOTING GAME

A game generally consists of a set of players (e.g. voters). Players make choices with a pay-off that depends on their utility functions [8]. We assume voters in the Digital Ecclesia provide a numerical vote that belongs to a scale of discrete values (e.g. 5-star scale). We further assume that the utility function of a voter is proportional to his/her (probabilistic) voting preferences. We calculate the latter based on voters' past choices on similar voting topics. Overall, we define the notion of a voting game G for a topic p as follows.

*Definition 2.1 (**Voting game**).* A voting game is defined by the tuple $G = (T, N, R, U, W)$, where

- $T = \{t_1, t_2, \dots\}$ is a finite set of textual terms for a topic p;
- $N = \{1, 2, \dots\}$ is a finite set of voters[4];
- $R = \{1, 2, \dots\}$ is a finite set of numerical votes (voting space);
- $U_n(r) = k * pr_n(r)$ is the utility function of a voter n for a vote r and is proportional to his/her preferences pr_n with respect to the vote r (k is a constant of proportionality);
- $W(r) = \sum_{i=1}^{N} \log U_i(r)$ is the value of the social-welfare function for a vote r over the whole set of voters. □

*Definition 2.2 (**Voting preference**).* The preference pr_n of a voter n on a vote r equals to the percentage of the times that s/he has provided the vote r for a set s of similar topics over the total number of his/her votes for these topics:

1. $\forall r \in R, \quad pr_n(r) = \frac{v_n(r,s)}{v_n(s)} \in [0, 1]$,
   a. $v_n(r, s)$ is the number of times that a voter n has provided a vote r for a set s of similar topics;
   b. $v_n(s)$ is the total number of the votes that s/he has provided for a set s of similar topics;
   c. $s = \{p_1, p_2, \dots, p_{|s|}\} : \forall p_i \in s, \quad sim(p, p_i) > \theta \mid \theta \in [0, 1]$, is the set of the past topics p_i whose similarities to the current topic p are greater than a threshold θ;
   d. $sim(p_i, p_j) = \frac{\sum_{k=1}^{|T_1|} sim_t(t_k, f(t_k))}{|T_1|}$, $t_k \in T_1 \wedge f(t_k) \in T_2 \wedge |T_1| \leq |T_2|$ is the semantic similarity among the terms T_1 and T_2 of two topics p_i and p_j:
      i. $f : t_k \rightarrow t_m \mid \sum_{k=1}^{|T_1|} sim_t(t_k, f(t_k))$ is max;
      ii. $sim_t(t_k, t_m) = sim_{Lin}(t_k, t_m)$;
2. $\sum_{i=1}^{R_{max}} pr_n(r_i) = 1$, the sum of the voting preferences of a voter in the voting space R equals to 1. □

Concerning the metric $sim(p_i, p_j)$, it calculates the similarity between two topics as the average of the similarities sim_t of their most similar pairs of terms. The metric solves the assignment problem of the maximum-weighted matching in a bipartite graph [9], where the nodes correspond to terms and the edges to their similarities. Regarding the metric sim_t, it calculates the Lin's similarity of the corresponding WordNet terms [10].

[4]The framework assigns to each voter a unique identifier.

3 DISTRIBUTED FRAMEWORK

The Digital Ecclesia models the social network of citizens as a bidirectional graph of software components that communicate to each other via encrypted messages. Each component corresponds to a single citizen who holds authentication credentials for using the component. We call neighbours the nodes connected with a graph edge. To offer dynamic large-scale citizens' reachability and distributed employment of the voting game, we propose the distributed algorithm in Section 3.1, whose scalability is specified in Section 3.2.

3.1 Distributed Algorithm

Alg. 1 is triggered by a node (hereafter called source) that corresponds to a citizen who initiates a new working group. The algorithm starts the voting procedure only if the source node has high coverage of the network graph, i.e. a high number of nodes is reachable.

The algorithm consists of a *local program* for each node. The program performs local computations and sends/receives messages. Since nodes do not necessarily have the same computation clock, messages are sent asynchronously[5]. The types of sent/received messages are the following: $< reach >$, $< parent >$, $< source >$, $< vote >$, and $< utility >$ (Alg. 1 (1, 9, 11, 17, and 32)).

Reach messages. The source node initially sends a reach message to all neighbours (Alg. 1 (2-4)). Every node that receives the message stores its parent node, sends parent and source messages, and forwards the reach message to its neighbours (except its parent) (Alg. 1 (6-8)). Thus, a reach message is gradually forwarded to all nodes that are reachable from the source node (*broadcast* in message-passing systems [11]). To broadcast a reach message in a controlled way (i.e. messages do not wander in the network via different paths), a node does not forward messages that has already received (Alg. 1 (5)).

Source and parent messages. Since every node stores its parent and children nodes, a *minimum spanning-tree* [12] (rooted at the source node) of the network part, which is reachable from the source node, is finally built. Based on this tree, every source-message is gradually forwarded back to the source node (Alg. 1 (16)) – *convergecast* in message-passing systems [11]. When the source node receives a source message, the former increases the number of the received source-messages (Alg. 1 (13)). If the number is greater than a threshold ω, then the source node has high reachability[6] and sends vote messages to its children (Alg. 1 (14-15)). Finally, the source node does not send vote messages until a time limit has been expired (avoiding to send them multiple times).

Vote messages. When a node receives a vote message, then the node forwards the message to its children (Alg. 1 (18)). Following, the node asks by the corresponding citizen to vote for the current topic (Alg. 1 (21)) via firstly providing his/her credentials (Alg. 1

[5]For practical reasons, we consider upper bounds on message delays.

[6]We used the threshold 0.6 in the current research-prototype of the Digital Ecclesia.

Algorithm 1 Distributed Notification and Voting Game

// **Code for node** n_i, $1 \leq i \leq M$ (the total number of network nodes).

Input: n_r, θ, ω;

1: **upon receiving** $< reach, p, R >$ from neighbour n_j **do**
2: **if** $n_i = n_r$ **then** // *The node n_i is the source node n_r*
3: $parent \leftarrow n_i$;
4: SEND $< reach, p, R >$ to all neighbours
5: **else if** $parent$ = `null` **then**
6: $parent \leftarrow n_j$;
7: SEND $< parent >$ and $< source >$ to $parent$
8: SEND $< reach, p, R >$ to all neighbours except n_j
9: **upon receiving** $< parent >$ from neighbour n_j **do**
10: $children \leftarrow children \cup \{n_j\}$;
11: **upon receiving** $< source >$ **do**
12: **if** $n_i = n_r$ **then**
13: $++reached$;
14: **if** $reached > \omega$ and $expired$ = `true` **then**
15: SEND $< vote(p, R, \theta) >$ to $children$
16: **else** SEND $< source >$ to $parent$
17: **upon receiving** $< vote(p, R, \theta) >$ **do**
18: SEND $< vote(p, R, \theta) >$ to $children$
19: $correct \leftarrow$ AUTHENTICATE();
20: **if** $correct$ = `false` **then** RETURN();
21: $r \leftarrow$ VOTE(p, R, θ);
22: STORE(p, r);
23: **for all** $p_i \in (R_n - \{p\})$ **do**
24: **if** SIM$(p, p_i) > \theta$ **then**
25: $s \leftarrow s \cup \{p_i\}$;
26: $++v_n(s)$;
27: **if** $r = R_n(p_i)$ **then** $++v_n(r, s)$;
28: $pr_n(r) \leftarrow \frac{v_n(r,s)}{v_n(s)}$;
29: STORE$(pr_n(r))$;
30: $< utility(pr_n) > \leftarrow$ ENCRYPT(pr_n);
31: SEND $< utility(pr_n) >$ to $parent$
32: **upon receiving** $< utility(pr_n) >$ from node n **do**
33: **if** $n_i = n_r$ **then**
34: $pr_n \leftarrow$ DECRYPT$(< utility(pr_n) >)$;
35: $utilities$.ADD(pr_n);
36: **if** $expired$ = `true` or $|utilities| = reached$ **then**
37: **for all** $r \in R$ **do**
38: $W(r) \leftarrow \sum_{i=1}^{|utilities|} \log U_i(r)$;
39: **if** $W(r) > max$ **then**
40: $max \leftarrow W(r)$;
41: $r_{max} \leftarrow r$;
42: **else** SEND $< utility(pr_n) >$ to $parent$

(19)). If the authentication is not successful, then the execution of the node for this topic is stopped (Alg. 1 (20)). Otherwise, the current vote of the citizen is locally stored (Alg. 1 (22)). Following, the node updates the voting preferences of the citizen (Alg. 1 (23-29)) by applying Def. 2.2 and sends the updated utility-values to the parent of the node (Alg. 1 (31)). For privacy reasons, the utility values have been firstly *encrypted* (Alg. 1 (30)) by using the asymmetrical-cryptography technique [13].

Table 1: Statistics for the reported categories of products.

ID	Category	#consumers	#products
A_1	`Beauty`	1210271	249274
A_2	`Tools & Home Improvement`	1212468	260659
A_3	`Video Games`	826767	50210
A_4	`Toys & Games`	1342911	327698
A_5	`Health & Personal Care`	1851132	252331

Utility messages. When a node receives a utility message, then the node forwards the message to its parent. All utility values are gradually forwarded to the source node (*convergecast*). The source node decrypts each received utility-message (Alg. 1 (34)) and stores the utility values (Alg. 1 (35)). Before making a decision, the source node waits until a time limit has expired or all reachable nodes have sent utility messages (Alg. 1 (36)). Finally, the source node makes a decision that maximizes the social-welfare function (Alg. 1 (37-41)) by applying Def. 2.1.

3.2 Algorithm Scalability

In the broadcast of reach messages, the time complexity is $O(2 * N)$, since each node receives a reach message at most twice. The message complexity is $O(N)$, since every node sends a reach message. In the convergecast of source messages, all nodes send source messages to their parents using the minimum spinning-tree and hence, the time complexity is $O(\log N)$, where $\log N$ is the depth of the tree. The message complexity is $O(N)$, since every node sends a source message. In the broadcast of vote messages, the time and message complexity is $O(N)$, since every node receives a vote message exactly once. Similarly, in the convergecast of utility messages, the time complexity is $O(\log N)$ and the message complexity is $O(N)$. Thus, the overall (time and message) complexity of the algorithm is $O(7N + 2\log N) = O(N)$ that asymptotically scales in a *linear* way with the total number N of the nodes that are reachable from the source node. Note that this complexity concerns the execution of the distributed algorithm for one voting topic. In the case of the parallel execution of the algorithm for multiple topics, the complexity further scales with the number L of all topics, $O(L * N)$.

4 PRELIMINARY EVALUATION

Dataset. To evaluate the distributed algorithm on large-scale real-world votes, we used the publicly available (jmcauley.ucsd.edu/data/amazon) numerical ratings of millions of consumers on thousands of Amazon products [14]. We consider that each product corresponds to a voting topic and a rating corresponds to a vote. The products are organized in high-level categories that contain products similar to each other in terms of their descriptive terms. Due to lack of space, we indicatively present the results for the categories[7] of Table 1.

Evaluation methodology. We implemented the distributed algorithm and executed it for each product of the reported categories. At each execution, we configured the network to contain all consumers who rated an examined product. We also considered a topology of the network such that all nodes are reachable from at least one source node. We examined products with high percentages of conflicting

[7] We have reached to analogous conclusions for the remaining categories.

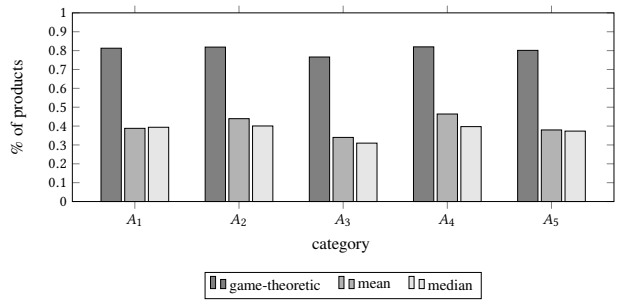

Figure 1: Decisions for products with conflicting ratings.

ratings. A product has a high percentage of conflicting ratings if there is no rating for the product that represents the majority of its consumers. We hereafter use the percentage `60%` as the minimum majority percentage[8]. We compared the decisions made by our algorithm against the aggregators of the arithmetic mean and median that are widely used for aggregating consumers' ratings [15].

Evaluation results. Fig. 1 depicts for each category the percentages of the products with both conflicting ratings and decisions that are proportional to the majority of past consumers' preferences. We observe that the percentages of the products, for which the decisions made by our algorithm are proportional to the majority of past consumers' preferences, range from the `76%` to the `81%` of the products. On the contrary, the corresponding percentages for the arithmetic mean (resp. median) range from the `33%` (resp. `30%`) to the `46%` (resp. `40%`) of the products. Concluding, the distributed algorithm is more effective than the classical aggregators in high percentages of the products of various categories.

5 RELATED WORK

The existing frameworks support the online participatory democracy. From an architecture perspective, we organize them into three categories: (i) centralized Web-based frameworks (e.g. [16]), (ii) cloud-based frameworks (e.g. `Open Town Hall` – opentownhall.com), and service-oriented frameworks (e.g. [2, 17, 18]). Going beyond the existing frameworks, we proposed the Digital Ecclesia that has distributed architecture in order to face scalability and privacy issues.

Among the key functions of online democracy [2], the existing frameworks have focused on proposal making & voting (e.g. `Agora Voting` – agoravoting.org), online collaboration & discussion (e.g. `Etherpad` – etherpad.org), and social communication [18]. We go beyond these frameworks via offering dynamic large-scale reachability of citizens and distributed employment of voting game.

From a privacy perspective, the existing frameworks can be accessed by citizens via firstly creating user accounts. Consequently, frameworks store citizens' votes on the server side, breaking their privacy. To support voting privacy, each node of the Digital Ecclesia runs a local program with its own storage that executes the voting procedure in parallel with other nodes. Nodes communicate to each other via exchanging encrypted messages.

[8] https://www.princeton.edu/~pcwcr/drafting/voting.html

6 CONCLUSIONS AND FUTURE WORK

To enable the online direct democracy, we proposed the framework Digital Ecclesia that offers dynamic large-scale reachability of citizens and distributed employment of voting game. We evaluated the proposed distributed algorithm on a corpus of real-world votes.

A future research-direction is to propose advanced security and privacy techniques. Moreover, the modelling of the voting procedure as a cooperative game could be investigated. Another direction is the consideration of social-dependability requirements (e.g. handling impersonation). Finally, advanced engagement mechanisms for citizens, who have already been notified for participation in the decision making, could be studied based on interdisciplinary collaboration with specialists who study human behaviour.

REFERENCES

[1] E. Bruno. Co-deciding with Citizens: Towards Digital Democracy at EU Level. Technical report, European Citizen Action Service, 2015.
[2] J. Holston, V. Issarny, and C. Parra. Engineering software assemblies for participatory democracy: The participatory budgeting use case. In *International Conference on Software Engineering*, pages 573–582, 2016.
[3] Nicholas C. Kyriazis and Emmanouil Marios L. Economou. *Democracy and Education: A History from Ancient Athens*. Springer International Publishing, 2015.
[4] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.
[5] A. S. Tanenbaum and M. van Steen. *Distributed systems - principles and paradigms, 2nd Edition*. Pearson Education, 2007.
[6] P. Coughlin and S. Nitzan. Electoral outcomes with probabilistic voting and nash social welfare maxima. *Journal of Public Economics*, 15(1):113–121, 1981.
[7] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.
[8] K. Binmore. Playing for real: A text on game theory. 2007.
[9] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, USA, 2009.
[10] George A. Miller. Wordnet: a lexical database for english. *ACM Communications*, 38(11):39–41, 1995.
[11] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.
[12] Gabriel Valiente. *Algorithms on Trees and Graphs*. Springer, 2002.
[13] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson Education, 3rd edition, 2002.
[14] J. McAuley and A. Yang. Addressing complex and subjective product-related queries with customer reviews. In *International Conference on World Wide Web*, pages 625–635, 2016.
[15] R. Di Pietro, M. Petrocchi, and A. Spognardi. A lot of slots – outliers confinement in review-based systems. In *International Conference of Web Information Systems Engineering*, pages 15–30, 2014.
[16] Matt Leighninger. Using Online Tools to Engage- and be Engaged by- the Public. Technical report, IBM Center for the Business of Government, 2011.
[17] A. Pathak, V. Issarny, and J. Holston. Appcivist – A service-oriented software platform for socially sustainable activism. In *International Conference on Software Engineering*, pages 515–518, 2015.
[18] R. Angarita, N. Georgantas, C. Parra, J. Holston, and V. Issarny. Leveraging the service bus paradigm for computer-mediated social communication interoperability. In *International Conference on Software Engineering*, 2017.