A Protocol for Distributed Voting in Urban Environments

Peter Danielis, Sylvia T. Kouyoumdjieva, and Gunnar Karlsson
ACCESS Linnaeus Center, School of Electrical Engineering
KTH Royal Institute of Technology, Stockholm, Sweden
Email: {pdanieli, stkou, gk}@kth.se

***Abstract*—Distributed aggregation algorithms have traditionally been applied to environments with no or rather low rates of node churn. The proliferation of mobile devices in recent years introduces high mobility and node churn to these environments, thus imposing a new dimension on the problem of distributed aggregation in terms of scalability and convergence speed. To address this, we present a distributed voting protocol for mobile device-to-device communication. We investigate a particular use case, in which pedestrians equipped with mobile phones roam around in an urban area and participate in a distributed yes/no poll, which has both spatial and temporal relevance to the community. The objective of our approach is to produce a precise mapping of the local estimate to the anticipated global voting result while preserving node privacy. Since mobile devices may have limited resources allocated for mobile sensing activities, we utilize D-GAP compression. We evaluate the proposed protocol via trace-driven simulations of realistic pedestrian behavior and demonstrate that it scales well with the number of nodes. Furthermore, in densely populated areas the local estimate of participants does not deviate by more than 2.5% from the global result and the achievable compression rate is at least 25%.**

I. Introduction

Distributed tasks and computations, e.g., to estimate the average of a set of values, are often conducted based on inputs supplied by collaborative users. Such aggregation functions are of high importance in large-scale distributed systems where there is a need to compute global system properties [1].

In this paper, we focus on a specific class of distributed tasks, namely distributed voting in the context of *urban polling*. Potential urban polling applications collect and process information on locally-relevant questions and provide users in a community with answers to them [2]. In general, the information obtained during a poll can be processed either in a centralized or in a decentralized manner. Centralized processing requires nodes to submit their votes to a central entity. However, this approach lacks scalability and poses privacy concerns as users might in general not want their votes to be seen by a central entity, be it trusted or not [3]. Contrary, decentralized (or distributed) processing requires nodes to compute local estimates of the result based on partial system knowledge. As opposed to conventional distributed processing scenarios where nodes are considered to be static or semi-static [4], in this work we examine scenarios, in which nodes exhibit high mobility. We refer to a node as a pedestrian carrying some device equipped with a wireless communication interface such as a mobile phone. We rely on device-to-device communication for disseminating votes among participants in the poll. Mobile nodes opportunistically exchange data whenever they come in direct communication range [5]. For urban polling, this data comprises voting information conveyed by means of broadcasts and nodes immediately update their local estimate upon message reception.

In the context of distributed voting in urban environments, a distributed voting protocol needs to comply to the following requirements: (1) to be scalable, (2) to have fast convergence and high accuracy, and (3) to preserve node privacy. Thus, in this work we present a distributed voting protocol for mobile device-to-device communication, which provides all of the above characteristics. The main contributions are:

- We show that our approach is suitable for operation in dynamic environments with high node mobility. It makes use of the benefits of D-GAP compression, which allows for *scalability* of the protocol [6]. Furthermore, our approach *preserves node privacy* by applying a cryptographic hash function to user identities.
- We perform extensive trace-driven simulations using realistic pedestrian mobility. We show that our approach scales well with the number of nodes in the system. Furthermore, our approach demonstrates both *fast convergence* and *high accuracy*, with local estimates deviating at most by 2.5 % from the global value in dense scenarios.
- Our approach achieves at least *25 % compression rate* in the investigated use case making it appropriate for execution on mobile devices with limited storage capabilities or with restrictions on the memory to be used.
- Our approach exhibits *low processing load* at the application layer as it requires only a fraction of the received broadcast messages (34 % in dense scenarios and even less in sparser scenarios) to be processed to achieve accurate local estimates.

II. Related Work

Distributed voting belongs to the class of distributed aggregation problems. Distributed aggregation in general comprises computations such as sum, average, minimum, or maximum over unreliable networks, in which no central entity is accessible or required. There are two main paradigms to address the aggregation problem, namely restarted and bookkeeping gossip-based and tree-based aggregation. Tree-based aggregation protocols have been shown to perform

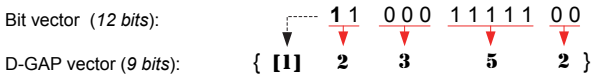


Fig. 1. An example of D-GAP compression. A bit vector of 12 bits is converted into an integer D-GAP vector of 9 bits. The leading bit in the D-GAP vector indicates if the vector starts with 0s or 1s.

poorly in dynamic environments with high levels of churn [4]. Most of the restarted gossip algorithms may count a node twice and do therefore not achieve a high accuracy. Further, bookkeeping protocols usually require up to thousands of rounds to converge, thus they are not suitable for scenarios with high dynamics. For further information, the interested reader is refered to [4].

III. Our Distributed Voting Protocol

In this section, we present a distributed voting protocol for mobile device-to-device communication.

A. The need for D-GAP compression

In the context of distributed voting, each node can cast a binary vote (0 or 1) to a poll. For nodes to be able to calculate the anticipated global vote, they need to keep track of the votes of other peers in their vicinity throughout their lifetime. We leverage the concept of *D-GAP compression* [6]. D-GAP compression provides a compressed representation of bit vectors in the form of integer vectors (later referred to as D-GAP vectors) and can be treated as a specialized variant of run length encoding. Each integer in a D-GAP vector represents the number of consecutive 0s or 1s that are present in the bit vector at a given position. Whether the integer corresponds to a sequence of 0s or 1s is determined by the leading bit of the D-GAP vector. A leading bit of 0 shows that the first integer corresponds to a number of consecutive 0s, followed by a number of consecutive 1s and so on; a leading bit of 1 indicates the opposite behavior. An example of converting a 12-bits bit vector into a 9-bits D-GAP vector is illustrated in Figure 1. We calculate the total number of bits required for representing the D-GAP vector, $\mathrm{N(DGAP)}$:

$$\mathrm{N(DGAP)} = \sum_{i=1}^{n} (\lfloor \log_2 d_i \rfloor + 1) \qquad (1)$$

where d_i is the integer representation at position i of the D-GAP vector, and n is the size of the vector.

B. Operations on D-GAP vectors

A D-GAP vector is solely a data structure. Hence, we define the following three operations for our approach that can be performed on two D-GAP vectors of arbitrary lengths: `merge`, `consolidate`, and `append`. For brevity, let us consider vectors of different lengths, DGAP_{min} and DGAP_{max}, denoting the shorter and the longer vector, respectively. Note that here vector length corresponds to the *expanded bit vector length* and is defined as $\mathrm{L(DGAP)} = \sum_{i=1}^{n} d_i$.

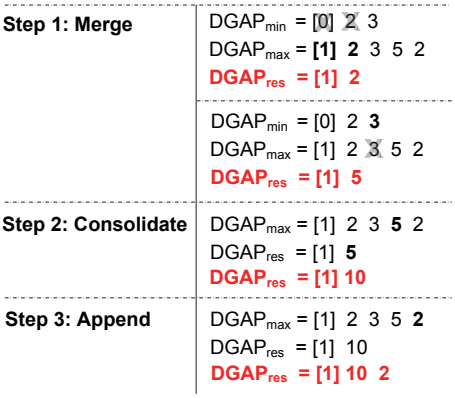


Fig. 2. Merging, consolidating, and appending D-GAP vectors. Bold integers denote positions under consideration, crossed integers are not considered, and the resulting vector is highlighted in red.

- The `merge` operation combines DGAP_{min} and DGAP_{max}[1:L(DGAP_{min})] vectors into a resulting vector DGAP_{res}. The `merge` operation is performed in an iterative manner until it reaches the end of DGAP_{min}.
- The `consolidate` operation combines the last position of DGAP_{res} with the next position of DGAP_{max} after the `merge` operation is performed. The `consolidate` operation is only performed if the integers at these two positions correspond to the same bit value.
- The `append` operation simply adds the remainder of DGAP_{max} to DGAP_{res}.

Observe that if the two input vectors are of equal lengths, the only operation that will be performed is `merge`. Figure 2 illustrates an example of all three operations.

C. Functional principles of our protocol

With our approach, each node locally keeps track of nodes it has has obtained knowledge of, either directly or via other peers, as well as of their votes. This information is presented in the form of two correlated D-GAP vectors. Whenever a node first casts a vote, it adds itself to the *disclosed-nodes vector*, and it adds its vote to the *votes vector*. Observe that due to privacy preservation reasons, the position, in which information is stored in each of the D-GAP vectors, is determined by a cryptographic hash function such as MD5, which is calculated over a unique identifier, e.g., the node's MAC address. For instance, if for a node, which casts a vote for 1, the cryptographic hash function returns a position of 25, both its disclosed-nodes vector and its votes vector would be initialized with $\{[0]\ 24\ 1\}$. If the same node casted a vote for 0, the votes vector would instead be initialized with $\{[0]\ 25\}$.

Each node periodically broadcasts a beacon containing its disclosed-nodes vector and its votes vector to peers in its vicinity. Whenever a node receives information from another peer, it immediately updates both its disclosed-nodes vector and its votes vector. The protocol consecutively executes the operations `merge`, `consolidate`, and `append` in order to update the local estimate. Thus, the disclosed-nodes vector and

the votes vector contain cumulative information of all nodes that have been disclosed over time, and their corresponding votes, even if these nodes have left the system. Furthermore, our protocol allows nodes to disclose peers that they have not encountered physically by propagating the knowledge accumulated by other participants in the system.

IV. Evaluation Scenario

In this section, we introduce the mobility scenario as well as the simulation setup and investigated performance metrics.

A. Mobility scenario

In order to realistically recreate pedestrian mobility, we use the Walkers traces [7] captured in Legion Studio [8], a commercial simulator initially developed for designing and dimensioning large-scale spaces via simulation of pedestrian behaviors. Each simulation run results in a trace file, containing a snapshot of the positions of all nodes in the system every 0.6 s. We considered an outdoor urban scenario, modeling the Östermalm area of central Stockholm. It consists of a grid of interconnected streets. Fourteen passages connect the observed area to the outside world. The active area, i.e., the total surface of the streets, is 5872 m^2. The nodes are constantly moving, hence the scenario can be characterized as a high mobility scenario. (More information can be found in [9].)

B. Simulation setup

In our evaluation scenario, we assume that all nodes carry devices and all are participating in the distributed poll in the area. Each node casts a binary vote $v = \{0, 1\}$ upon entry in the simulation, and votes are distributed according to a distribution $f(x)$ with a mean $\mathbb{E}(x)$. For the evaluation, we use an implementation of an opportunistic content distribution system in the OMNeT++ simulator [10]. Each simulation run is executed in synchronous rounds of 0.6 s. Nodes broadcast their disclosed-nodes vector and their votes vector at the beginning of each round. The transmission range is set to 10 m.

C. Performance metrics

We focus on evaluating the following performance metrics.

- *Deviation* (Δ): The deviation is a measure of the accuracy of our protocol:
$$\Delta = \left| \frac{\bar{x} - x}{x} \right| \tag{2}$$
where $\bar{x} = \mathbb{E}(\mathrm{DGAP}_{loc})$ is the local estimate, and x is the anticipated global result.
- *Compression ratio* (CR): The compression ratio is a measure of the efficiency and scalability of our protocol:
$$\mathrm{CR} = 1 - \frac{\mathrm{N(DGAP)}}{\mathrm{N(BVEC)}} \tag{3}$$
where $\mathrm{N(DGAP)}$ is calculated as per Eq. 1, and $\mathrm{N(BVEC)}$ is the number of bits required if the data were represented in the form of a bit vector. $\mathrm{N(BVEC)}$ can be calculated by summing up all d_i of the D-GAP vector (see Eq. 1).
- *Information overhead* (IO): The information overhead is a measure of the processing load reduction for a node in the system and therefore indicates scalability as well:
$$\mathrm{IO} = 1 - \frac{\mathrm{n(BRC)}}{\mathrm{N(BRC)}} \tag{4}$$
where $\mathrm{N(BRC)}$ is the total number of broadcast messages received by a node throughout its lifetime in the system, and $\mathrm{n(BRC)} \subset \mathrm{N(BRC)}$ is the number of broadcast messages that were used for updating the local estimate.

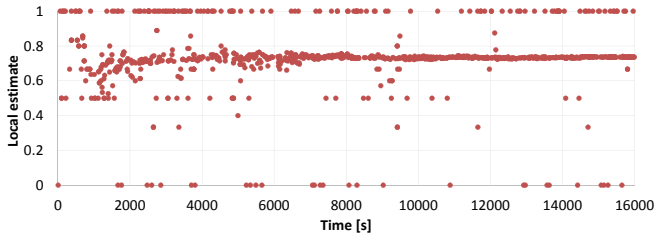
V. Simulation Results

In this section, we investigate simulation results for the Östermalm scenario for the arrival rates λ = {0.0025, 0.005 0.01, 0.07, 0.15, 0.30} nodes/s each for a single trace. We assume that votes are deterministically distributed, with a mean $\mathbb{E}(x)$ = 0.75, i.e., 75 % of all nodes vote for one and 25 % vote for zero. In this case, the first node entering the system votes for zero whereas the following three nodes vote for one. After that, this distribution continues for all further nodes.

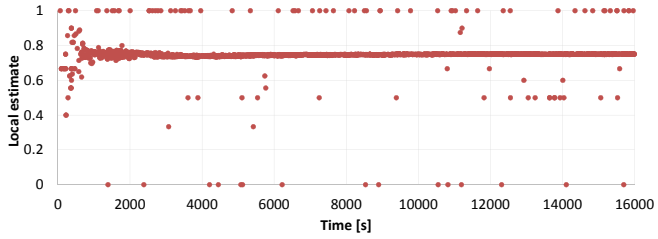
Figures 3(a)-(b) show the local estimates of all nodes over time for sparsely populated scenarios, i.e., λ = {0.005, 0.01} nodes/s. We see that the convergence of the local estimates towards the global result is strongly dependent on the population density. As the arrival rate increases, a trend can be seen towards convergence, and at λ = 0.01 nodes/s nodes are clearly able to locally estimate the global result. Still, for any of the low arrival rates, some nodes do not gain sufficient knowledge about other nodes or even do not disclose anyone so that their local estimates remain 0 or 1 (see Figures 3(a)-(b)). As the arrival rate further increases, nodes converge earlier to the global result, and outliers disappear. We have omitted results for λ = {0.07, 0.15, 0.3} nodes/s due to space constraints.

Generally, our results show that disclosing approximately 30 % of nodes is sufficient for achieving precise estimates. Thus, we conclude that even in dynamically changing environments there is a correlation between the percentage of disclosed nodes and the convergence to the global result.

We further evaluate the performance of the system in steady state, i.e., once the average number of nodes in the area stays unchanged albeit the arrivals and departures in the system. We then aggregate results from 1000 nodes. Table I shows the average and maximum deviation Δ including 95 % confidence intervals (CIs) as well as the information overhead IO in the steady state depending on the arrival rate λ. Note that minimum deviation values are omitted as they are zero in all cases. These results clearly show that the denser the scenario, the smaller the deviation Δ, i.e., the accuracy increases. As the arrival rate increases, both the average and the maximum deviation are steadily decreasing. Finally, for the most densely populated scenario, λ = 0.3 nodes/s, the maximum deviation never exceeds 2.43 %. Moreover, the information overhead IO is between 93 % and 94 % for λ = {0.005, 0.01, 0.07} nodes/s, which results from the fact that often the same nodes meet



(a)



(b)

Fig. 3. Local estimates of all nodes: (a) λ = 0.005 nodes/s and (b) λ = 0.01 nodes/s.

TABLE I
AVERAGE AND MAXIMUM DEVIATION Δ, AND INFORMATION OVERHEAD IO WITH DIFFERENT ARRIVAL RATES λ.

ARRIVAL RATE λ [NODES/S]	AVG. Δ [%]	MAX. Δ [%]	IO [%]
0.005	14.91±1.19	100	93.30±0.4
0.01	7.19±0.73	100	94.05±0.2
0.07	2.06±0.08	6.3	93.98±0.3
0.15	1.01±0.05	3.07	89.55±0.2
0.3	0.79±0.04	2.43	66.28±0.5

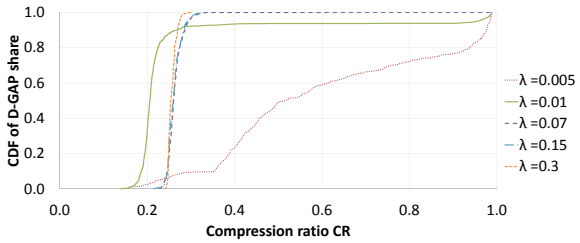


Fig. 4. CDF of the compression ratio CR for storing votes under different arrival rates λ.

again and do not exchange new information. On the one hand, this underlines the low processing load on application layer and thus our protocol's scalability in sparse scenarios. On the other hand, as shown above, the accuracy is not very high for the sparse scenarios. In denser scenarios, the information overhead amounts to lower values of IO = 89.55 % and 66.28 % for λ = 0.15 nodes/s and λ = 0.3 nodes/s, respectively. Due to the higher population density, it is more probable that new nodes meet, which then exchange new information and thus the information overhead decreases.

Figure 4 shows the cumulative distribution functions (CDFs) of the compression ratio CR when storing the D-GAP vectors with votes across different arrival rates λ. The achievable compression ratio CR is higher in case of using D-GAP for storing disclosed nodes than that for storing votes. For each disclosed node, a 1 is set in the D-GAP, which results in long sequences of consecutive 1s and increases compression. On the other hand, depending on the voting distribution and its mean value, sequences of consecutive 1s may be shorter in the D-GAP vector for storing votes resulting in a lower CR. With the increase of the arrival rate, the compression ratio also increases, and for λ = 0.3 nodes/s, the average compression ratio amounts to 91.52 %. This results from the fact that most nodes in the system have been disclosed, thus almost all bits in the D-GAP are set to 1. The compression ratio of the D-GAP for storing votes approaches 25 % as the arrival rate increases, Figure 4, which depends on the chosen voting distribution and its mean value $\mathbb{E}(x)$ = 0.75. Different voting distributions have been studied but have been omitted due to space constraints.

VI. CONCLUSION

In this paper, we presented a distributed voting protocol in the context of urban polling, which is suitable for environments, in which nodes exhibit high mobility. It relies on device-to-device communication to exchange voting information. The proposed protocol preserves privacy and exhibits high convergence speed, accuracy, and scalability. Prospectively, we will investigate the use case of counting the nodes that are currently in a system in more detail.

ACKNOWLEDGMENT

The authors would like to thank the German Research Foundation (DFG) (research fellowship, GZ: DA 1687/2-1) for their financial support.

REFERENCES

[1] S. Gambs, R. Guerraoui, H. Harkous, F. Huc, and A.-M. Kermarrec, "Scalable and secure polling in dynamic distributed networks," in *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*, Oct 2012, pp. 181–190.

[2] L. Koeman, V. Kalnikaité, and Y. Rogers, ""everyone is talking about it!": A distributed approach to urban voting technology and visualisations," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 3127–3136.

[3] Y. Benkaouz, R. Guerraoui, M. Erradi, and F. Huc, "A distributed polling with probabilistic privacy," in *Reliable Distributed Systems (SRDS), 2013 IEEE 32nd International Symposium on*, Sept 2013, pp. 41–50.

[4] L. Nyers and M. Jelasity, "A comparative study of spanning tree and gossip protocols for aggregation," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 16, pp. 4091–4106, 2015, cpe.3549.

[5] Ó. Helgason, E. A. Yavuz, S. Kouyoumdjieva, L. Pajevic, and G. Karlsson, "A mobile peer-to-peer system for opportunistic content-centric networking," in *Proc. ACM SIGCOMM MobiHeld workshop*, 2010.

[6] A. Kuznetsov, "D-gap compression," 2002. [Online]. Available: http://bmagic.sourceforge.net/dGap.html

[7] S. T. Kouyoumdjieva, Ó. R. Helgason, and G. Karlsson, "CRAWDAD data set kth/walkers (v. 2014-05-05)," Downloaded from http://crawdad.org/kth/walkers/, May 2014.

[8] "Legion Studio," http://www.legion.com/.

[9] Ó. Helgason, S. T. Kouyoumdjieva, and G. Karlsson, "Opportunistic communication and human mobility," *Mobile Computing, IEEE Transactions on*, vol. 13, no. 7, pp. 1597–1610, July 2014.

[10] Ó. R. Helgason and K. V. Jónsson, "Opportunistic networking in OMNeT++," in *Proc. SIMUTools, OMNeT++ workshop*, 2008.