INTERVAL CONSENSUS: FROM QUANTIZED GOSSIP TO VOTING

Florence Bénézit, Patrick Thiran, Martin Vetterli.

EPFL, School of IC, CH-1015 Lausanne, Switzerland

ABSTRACT

We design distributed and quantized average consensus algorithms on arbitrary connected networks. By construction, quantized algorithms cannot produce a real, analog average. Instead, our algorithm reaches consensus on the quantized interval that contains the average. We prove that this consensus in reached in finite time almost surely. As a byproduct of this convergence result, we show that the majority voting problem is solvable with only 2 bits of memory per agent.

Index Terms— Consensus, quantization, gossip, voting.

1. INTRODUCTION

Distributed average consensus algorithms have been designed to solve distributed coordination problems in networks. Most of them, however, rely on the exchange of analog values and on infinitely precise memory at each node. Kashyap, Başar and Srikant suggested in [1] an average consensus algorithm over integers, which is a quantized version of pairwise gossip [2]. In their setting, nodes initially measure some integer values, and the two integers framing the average of these values are denoted by L and $L+1$. When convergence is reached, some nodes have state L, the remaining nodes have state $L+1$, and the overall average is preserved. This quasi-consensus was called "quantized consensus".

Kar and Moura [3] and Aysal, Coates and Rabbat [4] designed probabilistic algorithms that are able to reach a true consensus. In [4], it is shown that if nodes use probabilistic quantization at each iteration, then all the states converge to a common but random quantization level. These probabilistic algorithms compute an unbiased estimate of the average of the initial data, but with a non zero variance, hence the precision of the results is not guaranteed.

On the contrary, in the first algorithm we mentioned [1], if a node reaches state L, it concludes that the average lies in $[L-1, L+1]$. A node with state $L+1$ outputs the interval $[L, L+2]$. Both conclusions are correct, but the nodes did not reach an *interval consensus*. This absence of consensus may lead the nodes to take two different and uncoordinated decisions.

Our work focuses on constructing quantized distributed averaging algorithms that reach interval consensus. One particular instance of this problem, called the *voting problem*, appeared several decades ago, and no simple and exact solution has been found so far. The problem is simple: nodes initially vote for Yes (1) or No (0), and they want to know the majority opinion. The number n of nodes is chosen odd in order to always have a strict majority. The voting problem has numerous applications in distributed computing, system-level diagnosis, distributed database management, fault-local mending, etc [5]. It was shown in [6], that the problem (with an extra synchronization constraint) has no solution if nodes have 1 bit memory states. There is an obvious link between the averaging problem and the voting problem. If one is able to compute the average of the initial votes (0's and 1's), then comparing the average to 0.5 is sufficient to deduce the majority vote. Note that the algorithm of Kashyap et. al [1] solves the voting problem if the quantization step is smaller than $1/(2n+1)$, which requires to code states on order $O(\log n)$ bits. The interesting question now is whether the voting problem has a solution with states of size $O(1)$, indeed larger than 1 bit, yet smaller than $O(\log n)$ bits. As a direct consequence of our work, we show that 2 bits are sufficient to solve the voting problem.

The remainder of this paper is organized as follows: Section 2 states the interval consensus problem. We describe our algorithm in Section 3, and we prove that it converges correctly in finite time in Section 4. Finally, Sections 5 and 6 list some further work and conclusions.

2. PROBLEM STATEMENT

At time $t = 0$, n nodes measure some quantized values $(x_1[0], x_2[0], \dots, x_n[0])$, where $\mathbb{R}$ has been uniformly quantized with step Δ. We denote by x_{ave} the average of the n measurements:

$$x_{ave} = \frac{1}{n}\sum_{i=1}^{n} x_i[0].$$

The nodes can communicate through a connected network $\mathcal{G}$ and we are given an ordered subset of quantization levels $\Theta_1 < \Theta_2 < \ldots < \Theta_r$ called thresholds. The goal is to design a quantized distributed algorithm such that nodes can tell whether the average x_{ave} is smaller than Θ_1, or between Θ_1 and Θ_2, or between Θ_2 and Θ_3, or ..., or larger than Θ_r. At each step of the algorithm, nodes can store a limited number of bits as their current state, and neighboring

nodes can exchange their states. Based on their final state, all the nodes should reach a consensus on the interval $[\Theta_i, \Theta_{i+1}]$ which contains x_{ave}. To simplify, we assume that x_{ave} can not be threshold level ($x_{ave} \neq \Theta$). The case $x_{ave} = \Theta$ will be briefly discussed in Section 5.

Throughout the paper, we will discuss three examples where nodes initially vote for 0 or 1. For each of these problems, we specify how the parameters should be chosen:

1. *The voting problem.* Are there more 0's or 1's? Parameters: $\Delta = 0.5$, $\Theta = 0.5$, 3 quantization levels: $\{0, 0.5, 1\}$, n odd.

2. *The large majority voting problem.* Is there a large majority winner? A large majority winner gets more than 2/3 of the votes. Parameters: $\Delta = 1/3$, $\Theta_1 = 1/3$, $\Theta_2 = 2/3$, 4 quantization levels: $\{0, 1/3, 2/3, 1\}$, $n/3 \notin \mathbb{N}$.

3. *The quorum checking problem.* Have at least $2/3$ of the nodes voted for 1? Parameters: $\Delta = 1/3$, $\Theta = 2/3$, 4 quantization levels $\{0, 1/3, 2/3, 1\}$, $n/3 \notin \mathbb{N}$.

3. OUR QUANTIZED CONSENSUS ALGORITHM

3.1. A gossip based algorithm

Just as in [1], our algorithm is a quantized version of the pairwise gossip algorithm. Similarly to gossip, at the beginning of every round of our algorithm, an undirected edge of the communication network is randomly selected and its two end-nodes exchange their states. We denote by p_e the probability that edge e is chosen. The most common way of selecting edges is to assign a random exponential clock to each node. When their clock activates, nodes wake up and choose a neighbor uniformly at random among their neighbors. In that setting, if i and j are neighbors, edge $e = (i, j)$ is chosen with positive probability $p_e = (1/nd_i) + (1/nd_j)$, where d_i and d_j are the degrees of nodes i and j.

In pairwise gossip, nodes update their states to the average of the two states. By iterating this update rule over the successively chosen edges, all the states progressively converge to the average of the initial states. Our goal is to modify this simple averaging rule so that the states are quantized and so that the nodes reach an interval consensus.

3.2. Threshold values are split into two states

In order to achieve our goal, we assign *two states* to each threshold level Θ while all the other quantization levels are represented by one state only. An ordinary state will be denoted by its quantization value. The two states with threshold value Θ are distinguished by Θ^- and Θ^+. We order the set of states: if the quantization level of state x is smaller than the quantization level of state y, we write $x \prec y$. Also, for any threshold level Θ, we adopt the following convention: $\Theta^- \prec \Theta^+$. As a result, for example, the voting problem functions with 4 states, coded by the two bits we announced. The four states are ordered: $0 \prec 0.5^- \prec 0.5^+ \prec 1$. We adopt all the natural ordering vocabulary, which we adapt to $\prec$: min, max, $=$, $\preceq$, $\succ$, $\succeq$. In particular the notion of *consecutive states* is crucial. In previous example, 0 and 0.5^- are consecutive states. So are 0.5^- and 0.5^+. But 0 and 0.5^+ are *not* consecutive states.

Definition 1 (Convergence) *A quantized gossip algorithm has converged iff all the nodes have either equal or consecutive states.*

In other words, an algorithm has converged when there are two consecutive states x and y such that every state in the network is equal to x or y. Suppose that we have run a converging algorithm that *preserves average*, and that the average x_{ave} is in $[\Theta_1, \Theta_2]$, then necessarily, the two converging states are Θ_1^+, Θ_2^- or quantized levels between these two. Individually, each node knows in which interval x_{ave} is. Even a node with state Θ_1^+ makes the correct decision, because the $+$ sign tells it that $x_{ave} \geq \Theta_1$. Thanks to the threshold state splitting, we are able to locally decide common intervals.

3.3. Properties of our algorithm

In this section, we list the desirable properties of our algorithm. They are sufficient for the algorithm to converge, but not necessary. If nodes i and j are activated at time t:

- **Conservation property** The average should be preserved: $x_i[t+1] + x_j[t+1] = x_i[t] + x_j[t]$.

- **Contraction property** $x_i[t+1]$ and $x_j[t+1]$ should be either equal or consecutive states. Furthermore, if $x_i[t] = x_j[t]$ then $x_i[t+1] = x_j[t+1] = x_i[t] = x_j[t]$.

- **Mixing property** If $x_i[t] \preceq x_j[t]$, then $x_i[t+1] \succeq x_j[t+1]$. In particular, if $x_i[t]$ and $x_j[t]$ are consecutive states, then states are swapped: $x_i[t+1] = x_j[t]$ and $x_j[t+1] = x_i[t]$.

The algorithm in [1] has similar properties, which we have adapted to our setting. Our three properties imply two other properties we will use in Section 4:

Consequence 1: The conservation property together with the contraction property imply that:

$$\min(x_i[t], x_j[t]) \preceq x_i[t+1] \preceq \max(x_i[t], x_j[t]). \qquad (1)$$

Consequence 2: The mixing property admits a converse, which we call the **swapping property**: if two states are swapped, then they are equal or consecutive. Indeed, otherwise they would contract.

3.4. Explicit formulation of our algorithm

To simplify, we consider that quantization levels are centered at 0, i.e. they can be written $k\Delta$, with k integer. At time t, an edge is randomly chosen. We denote by i the activated node with smaller state and by j the other activated node: $x_i[t] \preceq x_j[t]$. Nodes i and j update their states according to the following rules:

$$\begin{aligned} x_i[t+1] &= \left\lceil \frac{x_i[t]+x_j[t]}{2\Delta} \right\rceil \Delta. \\ x_j[t+1] &= \left\lfloor \frac{x_i[t]+x_j[t]}{2\Delta} \right\rfloor \Delta. \end{aligned}$$

When $x_i[t+1]$ or $x_j[t+1]$ is equal to a threshold value Θ, we need to specify whether they are equal to Θ^+ or Θ^-. There are four cases:

- If $x_i[t] = x_j[t]$, then $x_i[t+1] = x_j[t+1] = x_i[t]$.
- If $x_i[t] \neq x_j[t]$ and $x_i[t+1] = x_j[t+1] = \Theta$, then $x_i[t+1] = \Theta^+$ and $x_j[t+1] = \Theta^-$.
- If only $x_i[t+1] = \Theta$, then $x_i[t+1] = \Theta^-$.
- If only $x_j[t+1] = \Theta$, then $x_j[t+1] = \Theta^+$.

It is easy to check that the three properties hold for this algorithm. In Fig.1, we show the update rules for the example problems of Section 2. By the conservation property, if the algorithm converges, then the network reaches consensus on the interval containing $\mathbf{x}_{ave}$ (as in Fig. 2, where a 2 bits voter simulation converges). In next section, we prove that our algorithm converges in finite time with probability 1.

4. CONVERGENCE THEOREM

Theorem 1 *Let T be the first time the algorithm has converged. If the updating rules follow all the properties of Section 3.3, then* $\mathbb{P}[T < \infty] = 1$.

Proof: Let $\overline{x}[t] = \max_i x_i[t]$ and $\underline{x}[t] = \min_i x_i[t]$. Equation (1) implies that $\{\overline{x}[t]\}_{t\geq 0}$ is an integer non increasing sequence and $\{\underline{x}[t]\}_{t\geq 0}$ is an integer non decreasing sequence. The first sequence being larger than the second, both sequences are bounded so that they admit limits in finite time, which we denote by $\overline{x}_\infty$ and $\underline{x}_\infty$. Let t_0 be the smallest iteration at which all the states in the network are larger than or equal to $\underline{x}_\infty$ and smaller than or equal to $\overline{x}_\infty$. For any $t \geq t_0$, let $\overline{\mathcal{M}}[t]$ (respectively, $\underline{\mathcal{M}}[t]$) be the set of nodes with state $\overline{x}_\infty$ (respectively, $\underline{x}_\infty$) at time t. Both sets have a non increasing number of elements. Therefore there is a time t_1 after which their cardinalities remain constant.

Let $\overline{m}[t_1]$ be a node in $\overline{\mathcal{M}}[t_1]$, and $\underline{m}[t_1]$ be another node in $\underline{\mathcal{M}}[t_1]$. We are now going to construct recursively two sequences of nodes $\{\overline{m}[t]\}_{t\geq t_1}$ and $\{\underline{m}[t]\}_{t\geq t_1}$. For any $t \geq t_1$,

$$\overline{m}[t+1] = \begin{cases} i[t] & \text{if } \overline{m}[t] \text{ and } i[t] \text{ communicate at time } t. \\ \overline{m}[t] & \text{if } \overline{m}[t] \text{ does not communicate at time } t. \end{cases}$$

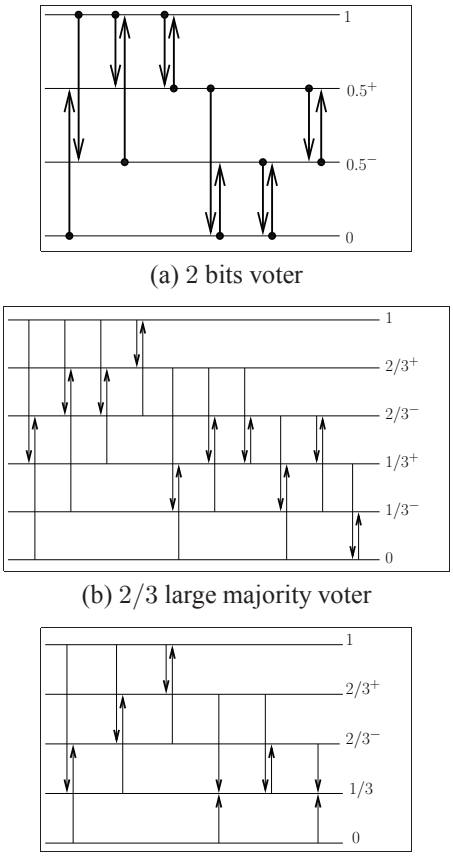


Fig. 1. Update rules for the simple voter, the $2/3$ large majority voter and the $2/3$ quorum checker. The figures show all the different pairwise updates possible. Easy consecutive states swaps are not shown in (b) and (c). For example, Fig (a) reads as follows: $(0,1)$ is updated to $(0.5^+, 0.5^-)$, $(1, 0.5^-)$ is updated to $(0.5^+, 1)$, etc.

$$\underline{m}[t+1] = \begin{cases} i[t] & \text{if } \underline{m}[t] \text{ and } i[t] \text{ communicate at time } t. \\ \underline{m}[t] & \text{if } \underline{m}[t] \text{ does not communicate at time } t. \end{cases}$$

We prove recursively that node $\overline{m}[t]$ has state $\overline{x}_\infty$ at time t for every $t \geq t_1$. It is true at time t_1 by construction of node $\overline{m}[t_1]$. Suppose that it is true until time t. Then, if $\overline{m}[t]$ communicates with a node $i[t]$ at time t, the two nodes will update their states. In order to keep the number of nodes in $\overline{\mathcal{M}}[t]$ constant, one of the two nodes should update its state to $\overline{x}_\infty$. By the Mixing property, we know that $i[t] = \overline{m}[t+1]$ has a state larger than the state of $\overline{m}[t]$ at time $t+1$. Therefore $\overline{m}[t+1]$ has state $\overline{x}_\infty$ at time $t+1$. If $\overline{m}[t]$ does not communicate at time t, then its state does not change and $\overline{m}[t+1]$ has state $\overline{x}_\infty$ at time $t+1$. For the same reasons, for any $t \geq t_1$, node $\underline{m}[t]$ has state $\underline{x}_\infty$ at time t.

Let T_{meet} be the first time at which $\overline{m}$ and $\underline{m}$ are the two end-nodes of the randomly chosen edge, or in other words the

first time when $\overline{m}$ and $\underline{m}$ communicate with each other:

$$T_{meet} = \min\{t : \overline{m}[t] \text{ and } \underline{m}[t] \text{ communicate at time } t.\}$$

By definition of the sequence $\{\overline{m}[t]\}_{t\geq t_1}$ and $\{\underline{m}[t]\}_{t\geq t_1}$, $\overline{m}[T_{meet}]$ and $\underline{m}[T_{meet}]$ swap states at time T_{meet}. By the swapping property, we conclude that, at time T_{meet}, $\underline{x}_\infty$ and $\overline{x}_\infty$ are either equal or consecutive states, which implies that the algorithm has converged. In other words, the convergence time T is smaller than T_{meet}: $T \leq T_{meet}$. The last step of the proof is to show that $\mathbb{P}[T_{meet} < \infty] = 1$, which implies that $\mathbb{P}[T < \infty] = 1$.

We consider the joint process $\{\overline{m}[t], \underline{m}[t]\}_{t\geq t_1}$. Note that for any t, $\overline{m}[t] \neq \underline{m}[t]$. This process is a Markov chain over the state space E (here the states are pair of nodes in the network):

$$E = [1, n]^2 \setminus \{(1, 1), (2, 2), \ldots, (n, n)\}.$$

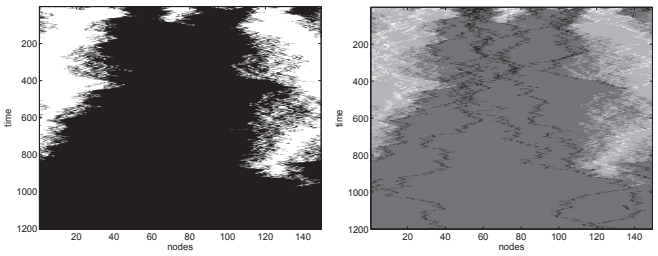
The communication network being connected, $\{\overline{m}[t], \underline{m}[t]\}_{t\geq t_1}$ is irreducible. Noting that E is a finite set, we can conclude that the Markov chain is positive recurrent. For any pair $e = (i, j)$ of neighboring nodes in the communication network, the Markov chain $\{\overline{m}[t], \underline{m}[t]\}_{t\geq t_1}$ admits a positive transition probability $p = p_e$ from state (i, j) to state (j, i). The first time such a transition occurs is equal to T_{meet}. To finish the proof, it is sufficient to notice that any transition with positive probability of an irreducible Markov chain over a finite state space is used in finite time with probability 1. This particular point of Markov chain theory is proved in Appendix. □

5. FURTHER WORK

Further work should investigate the convergence time of these algorithms, which depends on the number n of nodes, the network topology, the quantization step size and also the average x_{ave}. Another point to study is the behavior of the algorithm when x_{ave} can be equal to a threshold level Θ. If $x_{ave} = \Theta$, convergence states are Θ^+ and Θ^-. To detect this situation, a node with state Θ^+ should gather neighboring states and check if there is any Θ^- state. If approximatively $n/2$ nodes have state Θ^-, then a Θ^+ node finds a Θ^- node with high probability as long as it gathers $O(\log n)$ states. Therefore, computing the distribution of the repartition of Θ^+ and Θ^- among nodes, and showing that it is well concentrated around $n/2$ would solve the interval consensus problem, even when x_{ave} is a threshold level.

6. CONCLUSION

We have designed a class of quantized gossip algorithms that reach interval consensus in finite time. We proved that the voting problem can be solved with 2 bits states, and that the large majority voting problem and the quorum checking problem can be solved with 3 bits states.



(a) One bit displayed (b) Two bits displayed

Fig. 2. The 2 bits voting algorithm. The algorithm was run on a circular network of 149 nodes, displayed here on a line. The successive state configurations are shown with time increasing down the page (149 iterations per line). In (a), black pixels represent states 0 and 0.5^-, white pixels represent states 0.5^+ and 1. In (b), the four states are distinguished with white (0), light gray (0.5^-), dark gray (0.5^+) and black (1). There were initially 3 more black states than white states.

7. APPENDIX

Theorem 2 *Let E be a finite set. Any transition with positive probability of an irreducible Markov chain over E is used in finite time with probability* 1.

Proof: Let $\{X_n\}_{n\geq 0}$ be an irreducible Markov chain over E. We consider the Markov chain $\{Y_n\}_{n\geq 0} = \{(X_n, X_{n+1})\}_{n\geq 0}$ over the state space of positive transition edges of chain $\{X_n\}_{n\geq 0}$. $\{Y_n\}_{n\geq 0}$ is irreducible as well, and the number of positive transition edges of chain $\{X_n\}_{n\geq 0}$ is finite. Therefore $\{Y_n\}_{n\geq 0}$ is positive recurrent and it reaches any of its states in finite time with probability 1.

8. REFERENCES

[1] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, 2007.

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms : Design, analysis and applications," in *Proc. INFOCOM, 2005*.

[3] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Quantized data," *CoRR*, 2007.

[4] T. C. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus using probabilistic quantization," *Statistical Signal Processing, 2007. IEEE/SP 14th Workshop*.

[5] B. Hardekopf et al., "Secure and fault-tolerant voting in distributed systems," *Proc. IEEE Aerospace Conf., 2001*.

[6] M. Land and R. K. Belew, "No perfect two-state cellular automata for density classification exists," *Phys. Rev. Lett.*, vol. 74, no. 25, pp. 5148–5150, Jun 1995.