**Report on “A Poly-logarithmic Gossip Algorithm for Plurality Consensus”**

**L**et us consider the case of an asynchronous distributed network which has n nodes and each of them choosing from a bunch of options {1, 2, …., k}. When every node is capable of selecting an option, our aim is to come up with the option that is selected by the most of the nodes. This process of selecting an option from many options and making every node to know about the majority option is called plurality consensus. This basic agreement problem, also called as proportionate agreement, exists in different systems - for e.g., in biological systems, social networks, chemical reaction networks sensor, peer – to – peer networks, etc. The efficiency of an algorithm developed to acknowledge plurality consensus problem depends on two cases namely the time complexity and the message complexity.

In this following discussion, we will take the example of a robotic swarm which are moving in accordance with hand gestures. These robots will achieve plurality consensus after some (minimum possible) rounds. Initially, in the first stage, we will assume that the robots are not in a sync in following these rounds but they all will remember the number of rounds. This will make them to execute the algorithm for so and so rounds, after which the consensus is reached. We then implement a procedure to make the robots execute the algorithm in a synchronous manner and hence limit the memory usage per robot. At the end of the whole process, the robotic swarm would end up as a connected graph. We will try to look into the possibilities that can make it a complete graph.

Let us take a swarm of robots and all the robots have to coordinate depending on the ‘hand gestures’ given by the human. Let us consider there are 8 different gestures. And for the analysis purpose, let the total number of robots be 128 and it is a decentralized case, where there is no leader. Also consider that each robot is fed with the prerequisite data that indicates the information about different gestures so that every robot will pick the data from data base and match it with the shown gesture until it confirms with one.

In real time situation, make note that all the robots will not predict the same gesture (which is shown by the person) as the orientation of each robot has been different w.r.t the hand and hence the gestures shown by the hand will be seen from a different perspective by each of the robots. This created the problem called plurality where each robot has a peculiar opinion. There’s the case where all the robots have predicted the same opinion. The probability of this case is not zero but is very less. If this doesn’t happen, there are multiple opinions out of which only 1 opinion has to be selected (by everyone). This selected opinion has to be the opinion with high priority. There can also be an option where a robot remains undecided. Hence, we can have 9 different options to choose for each robot i.e., {0,1,…,8}

**First Stage**: Achieving plurality consensus in minimum number of rounds.

When we consider a case where every robot has to communicate with every other robot, the time taken would be O(n). Our aim is to minimize this time complexity to a polylogarithmic order of n (n refers the number of robots in the swarm). For this, we can go with a standard reading protocol where for each opinion *i*, we would have at least one robot that has some estimation for the frequency of this opinion in the swarm. Here the time complexity can be *O(log n)*. But, if we assume the number of opinions as ‘k’ (not including the undecided opinion), the message complexity would be *O(log k)* as, for every option, we have to take ‘log n’ bits into consideration. Rather than going for this, if every robot chooses 3 other random robots and and adopts the majority opinion among these, each robot do not need to remember much information and hence the algorithm uses local memory of order *log k* bits but this will take very long time for the consensus.

In the above cases, either time or message complexity can be improved at the cost of the other. Hence, we opt Undecided State Dynamics to enhance both time and memory. For that, we make use of an algorithm as given below.

The algorithm works in phases, each with R rounds where R= O (log k). Round 1 of each phase *(Gap Amplification Phase)*

* + - *A decided robot keeps its opinion only if it contacts a robot with the same opinion*
    - *Undecided robotss remains undecided*

Round 2 to R of each phase *(Healing phase)*

* + - *Decided robots keep their opinion*
    - *An undecided robot v that contacts a decided robot u adopts u’s opinion.*

This meeting and exchange of information is not a one-time task but can go on for many times in each phase. The robots which are decided can be turned to undecided at the end of round 1 of each phase. Though the number of decided robots decrease, the gap between the robots holding the largest opinion and the second largest opinion will increase at the end of round 1 in each phase. That is the reason, this phase is considered the *gap amplification phase*. The other rounds in each phase are used to improve the amount of decided robots hence reducing the undecided robot count. Hence, this phase is considered as *healing phase.*

Let us consider 64 of the 128 robots have decided with one opinion (largest opinion) and the second largest opinion is because of 32 robots of the remaining 64 robots. And let all the remaining 32 robots have decided on the remaining 6 decisions and the also have undecided robots among them. In this scenario, let pi be the fraction of the robots holding the options i= {1,2,… 8}. And let p1 is for the largest opinion, p2 for the second largest, and p8 be the fraction of robots with least opinion. Then, 1- (p1 + p2 + p3 …+ p8) would be the fraction of undecided robots. Let p1/pi ≥ 1+(1/n). Taking these assumptions into consideration, in the above mentioned case,

p1 = 64/128 = ½ and p2 = 32/128 = ¼

p1/p2 = 2 which is more than 1+1/128 = 1.0078.

By the end of the first round in every phase, pi gets squared. So, the ratio of p1 to p2 also gets squared. But the number of robots that became undecided is increased. For example, in the above case, let

p1 = 40/128, and p2 = 10/128

Both have decreased from their initial values, but the ratio becomes 4, which is square of the previous value. So. At the end of round 1 of each phase, we can observe the relative change in p2 is more than that of p1, so the gap between the number of robots with first largest opinion and that with the second largest opinion has been increased and hence the name *gap amplification*.

There would be no change in this ratio 4 from round 2 to R, except that the number of robots with different opinions are increased. So, we have an increase in p1 and p2 which on the other hand decreases the undecided robot count.

As the ratio p1/pi has started with 1+1/n, its will reach 2 within *O(log n)* phases, and from there to n in *O(log log n)* additional phases. At this point, the ratio passes n, which indicates that p1=1 i.e., every robot has the same opinion (which is the majority opinion). And from there, with additional *O(log n/log k)* phases, the system reaches plurality consensus. In these rounds, it is known to every robot that plurality consensus has reached.

**Second Stage:** Introducing synchronicity in between robots

In the above discussion, the robots involved are not synchronous. They are moving, interacting with other robots and changing their opinions for the stipulated phases as in the algorithm. We presume (also prove) w.h.p that plurality consensus is reached after *O (log n log k)* phases. So, irrespective to the other robots, every robot will execute these phases in asynchronous manner. Because of this, every robot has to remember the round number which is *modulo R = O (log k).* This increases the memory usage at every robot by *log log k + O(1)* bitswhich is an additional memory with the existing *log k* bits used to remember its opinion. Keeping these robots in sync will decrease this extra burden of remembering rounds.

For this task, we ask some of the nodes to keep time (choose by the flip of a coin), for which, the memory size *log k + O (1)* is sufficient. These are called *clock- nodes.* All the robots will not participate in time keeping and hence the remaining robots, called *game players* will follow the algorithm as mentioned above. These game players take the help of clock nodes in order to implement the phases (of algorithm) in a synchronous manner. There are three subtleties to this idea

* *Game players do not know the time and cannot perform algorithm by themselves nor can they receive time from clock nodes* (as it increases memory bits). Hence, we make clock nodes to report time to the game players in integer values {0,1,2,3}.
* *Game players do not know the phase numbers on their own but should rely on clock nodes*.
* *Every clock node, at some point, has to forget it’s time keeping responsibilities and try to learn the plurality opinion*.

Keeping in mind about these three delicacies, we try to develop different algorithms for game players and clock nodes to meet the time complexity requirements.

Let’s discuss the game player algorithm first and then go the clock node algorithm. Game player algorithm interpolates the *gap amplification (GA) protocol* (the one presented as previous algorithm) and *Undecided state protocol* (where clock nodes goes down). At the time when clock nodes goes down and not following their time keeping responsibilities, we cannot follow GA protocol. As p1 is very less than 1, GA protocol has a dominant role. As p1 value reaches 1, it indicates that consensus is about to reach and hence clock nodes have not much work to do and we no longer need the power of GA protocol.

In the GA protocol what we follow in the perspective of this algorithm, the phases which were discussed in the *Stage 1* algorithm are divided into 4 phases. Two of them are time buffer phases included between the gap amplification phase and healing phase. Time buffer will provide sufficient time for a robot to go undecided if it came across a robot with different opinion. *So, we are now referring to the protocol described by phases {0, 1, 2, 3} as gap amplification protocol*.

A game player node goes out of the GA protocol only if it meets a clock node that has moved to end game (end game is a state of clock node explained in its algorithm). Then the game players move out of GA Protocol to Undecided protocol. Each of the game players has the flexibility to come back to GA Protocol if it meets a clock node that reports phase 0.

The major challenge of the algorithm for clock nodes is to recognize whether the game players have reached the plurality consensus or not. We take the number of undecided nodes as reference for this. If the clock nodes come across at least one of the undecided nodes directly or indirectly (through gossip), they keep on with their time keeping duty. This assumption is fair enough as long as p1 is far away from 1.

During the *long phase* (combination of 4 phases in GA protocol), when the clock node has no information about the undecided nodes, it goes to a phase called *end-game* where it neglects all its time keeping duties and take the opinion of the game player that it last met. This is an additional phase we add to what the clock nodes report to the game players. If the consensus is reached, there is no other undecided robot that come across the clock node and hence the clock node continues to be in end-game phase and adopts the plurality option. When it comes across undecided node, it come back from end-game phase, keeping track of time, and reporting one of the phases {0,1,2,3} to the game players.

Hence, by the end of *O(log n log k)* rounds, we achieve plurality consensus with *log k + O (1)* bits memory needed at every robot.

**Complete Graph vs. Connected Graph**

The robot swarm we discussed above forms a connected graph at the end where every robot is capable of contacting with every other robot directly or through the other robots. The graph is not a complete graph where each robot is directly connected to the other robot. As we have a total of n robots, we need n (n-1)/2 edges to make it a connected graph i.e., *O(n).* For the example we have taken, where there are 128 robots, we need 8128 edges where all the 128 robots are in direct contact with the other 127 robots directly.

Once the graph is completely connected, there will be ease in navigation from one robot to the other robot. Breadth first search or Depth first search algorithms will come handy in this case. But reaching plurality will still be a complicated process when it has to come with minimum time and message complexity. Those can run at the order of *O(n+e)* (e is the number of edges).

In this case, we need a random process where robots of connected graph reach consensus through pairwise interactions. There is no specific minimum time to reach the consensus in this case, but can be approximated using different models, according to John Haslegrave and Mate Puljiz in their paper ‘*Reaching consensus on a connected graph’*

**Appendix: Algorithms for game player robots(node) and clock nodes**

