Time-based Consensus

Hasan Heydari, Guthemberg Silvestre, Nicolas Larrieu, Alain Pirovano

Time-based Consensus

Hasan Heydari, Guthemberg Silvestre, Nicolas Larrieu, and Alain Pirovano

ENAC, Université de Toulouse, France
{heydari,silvestre,nicolas.larrieu,alain.pirovano}@enac.fr

Abstract. Reaching consensus is fundamental in distributed computing. For each execution of a consensus algorithm, there is no difference between the proposed values by different nodes with respect to their proposed times. By presenting a realistic application scenario related to distributed asynchronous mobile robots in dynamic environments, we argue some safety-critical, real-time systems require reaching consensus on the *newest proposed values* when the old proposed values may not be valid anymore. Afterward, we formulate a new type of consensus problem called time-based consensus, which requires to take into account the times of proposed values. Finally, to tackle such a consensus problem, we determine an essential characteristic which should be considered.

Keywords: Consensus algorithm · Distributed mobile robot · Safety-critical application · Real-time system · Dynamic environment.

1 Introduction

Reaching consensus is a primitive of distributed computing [3]. Consensus, informally, refers to an agreement among a group of nodes in which each node proposes a value, and the goal is to agree on exactly one value. There are several reasons why, in distributed systems, consensus is required, like to agree on who is the leader, to agree on who gets access to a shared resource, synchronizing nodes' clocks, to agree on an ordering of events/operations among nodes, or achieving formation control [2].

There are different types of consensus problems. Each type is presented for specific purposes and has its own characteristics but satisfying agreement and termination properties is common among all the types. Based on validity property, which means the decided value is one of the proposed values, the consensus problems can be divided in two types. The first type satisfies validity property which is the case in distributed data stores. For the other type of consensus problems, validity property is stated differently or not defined. For example, validity property for average consensus [1] and max-min consensus [8], which are mostly used in robotics, is not defined. In this paper, we focus on a subset of the first type which is subject to FLP [4] (it is leader-based and should provide strong consistency in addition to satisfying validity).

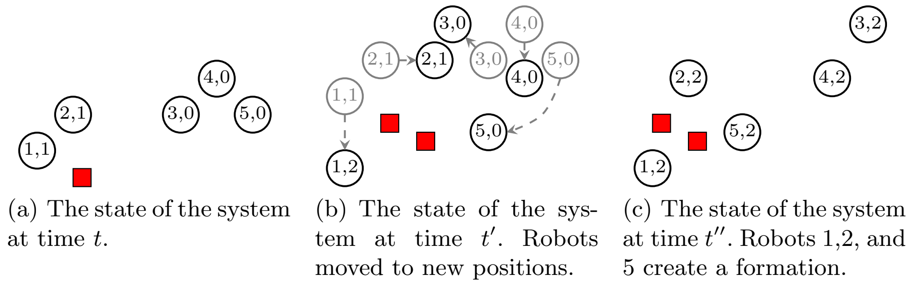
Formally, a consensus algorithm which is subject to FLP is correct when it satisfies three properties– *agreement*, *termination*, and *validity* [3]. Also, it has some characteristics– each node proposes exactly one value, all the proposed values should be taken into account to reach consensus, there is no difference between the proposed values with respect to their proposed times because the system's model is asynchronous and failures eventually occur, and if the nodes want to propose other values, they have to execute the algorithm again. Paxos [6] and Raft [7] are two well-known examples for such an algorithm.

For some safety-critical, real-time systems, the nodes have to consider the times of proposed values and reach consensus on the *newest proposed values*. Note that determining which proposed values are new is a challenging problem in asynchronous distributed systems when occurring failures are possible. In this paper, we formulate a new type of consensus problem called time-based consensus, which requires to take into account the times of proposed values. The structure of the paper is as follows. In Section 2, we argue that having such a time-based consensus is crucial for some safety-critical, real-time systems, like distributed asynchronous mobile robots in dynamic environments. We explain the limitations of consensus algorithms that lead to not reaching consensus on the newest proposed values in Section 3. Finally, (i) we formulate time-based consensus problem and (ii) finish the paper by presenting ongoing works to tackle such a problem.

2 Motivation

There are some safety-critical, real-time systems in which consensus is required, and in the process of reaching consensus, it is important to consider the times of proposed values. For an instance of such systems, suppose that there are n mobile heterogeneous robots located in a burned building in which some persons need help to rescue (Fig. 1(a)). The robots have two responsibilities– detecting and counting the persons and rescue some of them by creating a formation. Since the robots are mobile, and the environment is dynamic $\big($which means if there are n_t detected persons at time t, it is possible that at time t' $(t' > t)$, there are $n_{t'}$ $(n_t \neq n_{t'})$ detected ones due to finding new alive ones, dying some of them, etc. $\big)$, the number of detected persons can be different for each robot and is not constant during the rescue process (Fig. 1(b)). Control formation means that some of the robots create a determined formation around the detected persons (Fig. 1(c)). After creating a formation, they spread fire extinguishers to rescue the persons.

To create a formation for rescuing p persons, n_p robots is required. Robots for creating the formation need to reach consensus on the number of detected persons. Its reason is two-fold. First, some of the robots do not know the correct number of detected persons (Fig. 1(b)). Thus, when a robot detects a new person, it needs to broadcast the number of persons. Second, if more than n_p robots are allocated, it is not optimized. Note that the number of robots is limited, and here, taking into account optimization is crucial because what robots are doing is a critical task, and the remaining non-allocated robots can continue the rescue process. Therefore, the robots need to know the exact number of persons to decide how many of them have to participate in creating the formation.



(a) The state of the system at time t.

(b) The state of the system at time t'. Robots moved to new positions.

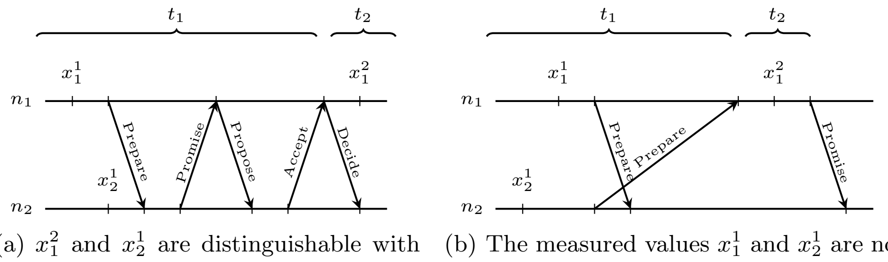
(c) The state of the system at time t''. Robots 1,2, and 5 create a formation.

Fig. 1. Circles and squares correspond to robots and persons respectively. The first and second digits written in each circle are its unique identifier and detected persons respectively. $t < t' < t''$.

3 Consensus Algorithms' Limitations

Suppose that two nodes n_1 and n_2 execute a consensus algorithm (for the sake of generality, suppose that the algorithm is Paxos) and are measuring/sensing a *critical data*. In what follows, by presenting two scenarios, we show Paxos' limitations that lead to not reaching consensus on the newest measured values. Consider x_1^i and x_2^j ($i, j \in \mathbb{N}$) are the measured values by n_1 and n_2 respectively. n_1 and n_2 measure two values– x_1^1 and x_2^1. n_1 by sending a prepare message to n_2 starts a consensus execution. In the first scenario, suppose that they reach consensus on x_1^1. Then, n_1 measures a new value, x_1^2 ($x_1^2 \neq x_2^1$). It is clear that if they execute the algorithm for another time, they can reach consensus on x_2^1 while it is not correct (i.e, they have to reach consensus on x_1^2 because it is the newest measured value). This scenario is depicted in Fig. 2(a).

In another scenario which is depicted in Fig. 2(b), n_1 measures a value like x_1^2 ($x_1^2 \neq x_2^1$) in the time period t_2 started after receiving some prepare message of n_2. They can reach consensus on x_2^1 which is not correct. Indeed, what we can do to distinguish the measured values with respect to their proposed times is using the executions and phases of a consensus algorithm.



(a) x_1^2 and x_2^1 are distinguishable with respect to their proposed times but Paxos does not consider it.

(b) The measured values x_1^1 and x_2^1 are not distinguishable with respect to their proposed times for n_1 but x_1^2 and x_2^1 are.

Fig. 2. Executing Paxos on two nodes.

4 Time-based Consensus and Ongoing Works

To tackle the problem explained in the previous sections, we model the system by a dynamic asynchronous distributed system with n nodes, where each node is a mobile robot. Each node has a sensor, which can be used to measure/sense some *critical data*, and can send (receive) messages to (from) the other nodes located in its communication range. Crash and link failures are possible. A correct node and link is a non-failed one. Being dynamic, here, means the set of correct nodes and links between a correct node and other correct nodes are not constant during the rescue process. Here, we formulate time-based consensus. When an algorithm satisfies three following properties, we say it can solve time-based consensus.

- *Termination.* Each node eventually should decide a value.
- *Agreement.* The decided values of all nodes should be the same. **The decided value is the last distinguishable proposed value (or among the last distinguishable proposed values) before deciding a value.**
- *Validity.* **Each node can propose one or more than one values before deciding a value by all nodes.** The decided value should be proposed by at least one node.

To tackle time-based consensus, we are using Paxos as a baseline because most consensus algorithms which are subject to FLP are variants of Paxos [5]. By changing some of its phases and adding an (some) additional phase(s) to it, we want to present an algorithm for time-based consensus. One of the essential characteristics of this algorithm is when a leader proposes a value measured at pth phase of eth execution to some node, and the measured value of the node was measured at p'th phase of e'th execution, the node cannot propose its measured value where $e' < e$ or $(e' = e) \wedge (p' < p)$.

References

1. Acciani, F., Frasca, P., Heijenk, G., Stoorvogel, A.A.: Achieving robust average consensus over lossy wireless networks. IEEE Transactions on Control of Network Systems **6**, 127–137 (2019)
2. Alonso-Mora, J., Montijano, E., Nägeli, T., Hilliges, O., Schwager, M., Rus, D.: Distributed multi-robot formation control in dynamic environments. Autonomous Robots (2018)
3. Attiya, H., Welch, J., Zomaya, A.Y.: Distributed Computing. John Wiley & Sons (2004)
4. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. Journal of the ACM **32**, 374–382 (1985)
5. Hao, Z., Yi, S., Li, Q.: Edgecons: Achieving efficient consensus in edge computing networks (2018)
6. Lamport, L.: Paxos made simple. ACM SIGACT News **32** (2001)
7. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm (2014)
8. Shi, G., Xia, W., Johansson, K.H.: Convergence of max–min consensus algorithms. Automatica **62**, 11–17 (2015)