paxos-algorithm

Khanh Nguyen

January 2022

1 The Paxos Algorithm

In Paxos algorithm, there are two type of machines: *proposer* and *acceptor* where each *proposer* holds a *value* and the goal of Paxos algorithm is to reach a consensus in the set of *acceptor* on a single *value*.

Let assign a partition of natural numbers to each *proposer* (each partition is a disjoint subset of natural numbers).

The algorithm below describes the simple version of Paxos algorithm [1]

1.1 *proposer*

1.1.1 phase 1: prepare

input: a *value* the *proposer* intended to propose
1 choose a number from its partition, namely *proposal_number*
2 broadcast the prepare message to all *acceptor*

PREPARE_REQUEST {*proposal_number*}

3 if it receives responses from a majority of *acceptor*, do **phase 2**

1.1.2 phase 2: accept

4 receives

PREPARE_RESPONSE {{*proposal_number*, *value*} or *Null*}

5 if all PREPARE_RESPONSE contain *Null*, pick the input *value* the *proposer* intended to propose. Else, pick the *value* corresponding to the highest *proposer_number*.
6 broadcast the accept message to all *acceptor*

ACCEPT_REQUEST {*proposal_number*, *value*}

1.2 *acceptor*

an *acceptor* holds an accepted proposal {*proposal_number*, *value*} in its stable storage.

1.2.1 on receiving prepare request

input: PREPARE_REQUEST {*proposal_number*} **1** if the accepted proposal is not set or *proposal_number* is greater than the accepted proposal, reply with the accepted proposal, namely *promise* PREPARE_RESPONSE {{*proposal_number*, *value*} or *Null*}

1.2.2 on receiving accept request

input: ACCEPT_REQUEST {*proposal_number*, *value*} **1** set the accepted proposal with the input, namely *accept*

2 The Proof of Correctness

If the proposal $\{m, u\}$ is accepted by majority of *acceptor*, the algorithm reaches the consensus on value u.

Theorem 1 *If the proposal* $\{m,\ u\}$ *is accepted by majority of acceptor, any issued proposal* $\{n,\ v\}$ *with* $m < n$ *has* $v = u$

Since all proposals accepted by majority of *acceptor* must be issued at some point, there cannot be any two proposals accepted by majority of *acceptor* with two different values. A straight forward corollary of theorem 1

Corollary 1 *All proposals accepted by majority of acceptor has the same value.*

By guaranteeing corollary 1, *acceptor* can notify the consensus to *proposer* by replying if they accept the accept request. If *proposer* receives accept responses from the majority of *acceptor*, it can confirm the consensus.

2.1 The proof the theorem 1

The theorem 1 can be proved by induction.

Lemma 1 (induction step) *Let* $\{m,\ u\}$ *be the first proposal accepted by majority of acceptor. Let* $\{n,\ v\}$ *be an issued proposal with* $m < n$*. Suppose that, for all* $k \in [m, n)$*, the proposal with* k *has value* u*, then* $v = u$

Let M_1 be the set of *acceptor* that accepted $\{m, u\}$, proposal $\{n, v\}$ is issued by some *proposer p* hence its prepare request must be replied from some majority set of *acceptor*, namely M_2.

Let an *acceptor* $q \in M_1 \cap M_2$, since $m < n$, from step 1 of 1.2.1, we know that *acceptor q* must accept $\{m, u\}$ before receiving prepare request of n.

Furthermore, *acceptor q* returns to the prepare request with the proposal $\{k, u^*\}$ where it is the latest proposal *acceptor q* accepts.

We can establish the relation among m, k, and n: $m \leq k < n$, by the induction step assumption, we can conclude that $u^* = u$

From the perspective of *proposal p*, it receives responses from M_2 where there is at least one non-$Null$ proposal. The maximum *proposal_number* from the responses set is greater than or equal to k, one of the responses, namely k^*.

We can further establish the relation between k^* and n. Since, *proposer p* receives k^* from an *acceptor* that accepted a proposal with *proposal_number* k^*, from step 1 of 1.2.1, $k^* < n$

Therefore, $m \leq k \leq k^* < n$, that implies *proposer p* picks the *value* corresponding to the highest *proposer_number* k^* which is u. Or, $v = u$

References

[1] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.