Egalitarian Paxos: Proof of correctness

Alexis Le Glaunec

January 3, 2021

Definition 1 (Pre-Accept)**.**

$$preaccept(D, c, Q, b) \triangleq \forall p \in Q, \Diamond(deps(c) = D \\ \land vbal = b = 0 \\ \land status(c) = "preaccepted")$$

Definition 2 (Accept)**.**

$$accept(D, c, Q, b) \triangleq \forall p \in Q, \Diamond(deps(c) = D \\ \land vbal = b \\ \land status(c) = "accepted")$$

Definition 3 (Vote)**.**

$$vote(D, c, p, b) \triangleq \Diamond(p.deps(c) = D \\ \land p.vbal = b > 0 \\ \land p.status(c) = "accepted")$$

Definition 4 (Committable)**.**

$$committable(D, c, Q, b) \triangleq preaccept(D, c, Q, b) \lor accept(D, c, Q, b)$$

Definition 5 (Committed)**.**

$$committed(D, c) \triangleq \exists p \in Replicas, \Diamond(deps(c) = D \\ \land status(c) = "committed")$$

Definition 6 (Executed)**.**

$$executed(D, c) \triangleq \exists p \in Replicas, \Diamond(deps(c) = D \\ \land status(c) = "executed")$$

Property 1. $committed(D, c) \implies \exists\, b, Q\ committable(D, c, Q, b)$

Proof. Let's suppose $committed(D, c)$. We consider $cleader \in Replicas$, the set of replicas, the leader of the command first to have $deps(c) = D \land status(c) = "committed"$. Therefore $committed(D, c) \implies \Diamond Phase1Fast(cleader, i, Q) \lor \Diamond Phase2Finalize(cleader, i, Q)$.

Case 1. $\lozenge Phase1Fast(cleader, i, Q)$

$Phase1Fast(cleader, i, Q) \implies \lozenge StartPhase1(c, cleader, Q, i, 0, oldMsg)$ and a $StartPhase1$ postcondition is $vbal = b$. Thus we have $vbal = b = 0$. Moreover one of $Phase1Fast$ preconditions is $\forall p \in Q, (p.deps(c) = cleader.deps(c))$. As $\lozenge Phase1Fast \implies \forall p \in q, p.status(c) = "preaccepted"$, we conclude that $preaccept(D, c, Q, b)$ and therefore $committable(D, c, Q, b)$.

Case 2. $\lozenge Phase2Finalize(cleader, i, Q)$

This is a similar case, replacing $p.status = "preaccepted"$ with $p.status = "accepted"$ and $\lozenge Phase2Finalize \implies \lozenge Phase1Slow(cleader, i, Q)$. Therefore as $vbal = b$ is a postcondition of $Phase1Slow$, we conclude that $accept(D, c, Q, b)$ and therefore $committable(D, c, Q, b)$.

■

Property 2.

$$vote(D, c, p, b) \implies \forall b' < b,\ (committable(D', c, Q', b') \implies D = D')$$

Proof. By induction on b, the ballot number.
Induction hypothesis:
$vote(D, c, p, b) \implies \forall b' < b,\ (committable(D', c, Q', b') \implies D = D')$

By definition, the base case is true. Let's consider a ballot $b > 0$ as we suppose $vote(D, c, p, b)$. At a recovery step, only $\lozenge PrepareFinalize$ leads to $vote$. Let's suppose $\exists b_M < b$ the highest ballot with $committable(D_M, c, Q_M, b_M)$ and by induction $\forall b' < b_M,\ committable(D', c, Q', b') \implies D = D'$. Replica p is part of quorum Q. Let's define $R = Q \cap Q_M$ and by definition $R \neq \emptyset$ hence $\exists r \in R$. If $vote(D_r, c, r, b) < vote(D'_r, c, r, b_M)$, it is in contradiction with the ballot order $b < b_M$ that is a precondition of $PrepareFinalize$ therefore impossible. Thus $vote(D_r, c, r, b) > vote(D'_r, c, r, b_M)$, and by definition of b_M, $\nexists q, vote(D_q, c, q, b_q)$. Therefore we take $D = D_M$ as b_M is the highest ballot lower than b possibly $committable$, and the induction hypothesis is verified. ■

Property 3. $committable(D, c, Q, b) \wedge committable(D', c, Q', b') \implies D = D'$

Proof. We suppose $committable(D, c, Q, b) \wedge committable(D', c, Q', b')$.

Case 1. $b = b'$

Case 1.i. $b = 0$

At ballot $b = 0$, $\exists! cleader, \lozenge Propose(c, cleader)$ as $Propose$ precondition $c \notin proposed$ is completed with $Propose$ postcondition $proposed' = proposed \cup \{c\}$. Thus $cleader$ proposes only once a set of dependences D at a quorum Q. Therefore, there is only $committable(D, c, Q, 0)$ with D and Q uniques.

Case 1.ii. $b > 0$

There can be several leaders recovering a command at the same time. However, as ballots are totally ordered by lexicographical order on $(ballot, replica)$ and that EPaxos is a majority-based protocol, only one $cleader$ has committable(D,c,Q,b). And as for the previous case, $SendPrepare$ preconditions guarantee that $cleader$ will propose a unique set of dependencies D to a quorum Q.

Case 2. $b > b'$

By induction on b, the ballot number.

Induction hypothesis:

$$committable(D, c, Q, b) \implies (\forall b' < b, committable(D', c, Q', b') \implies D' = D)$$

By definition of the induction hypothesis, $b > 0$ and the base case is true. In the recovery step, *committable* is accessible only through $\Diamond PrepareFinalize$. We define *replies* the set of replies from a quorum Q to the new leader *cleader*, and consider the different cases regarding *replies* content.

Case 2.i. $\exists\ com \in replies\ with\ com.status \in \{"committed", "executed"\}$

As *executed* $\implies$ *committed*, we conclude by property 1 that $committable(D_M, c, Q_M, b_M)$ happened and take $D = D_M$. Therefore by induction comes the result.

Case 2.ii. $\exists\ acc \in replies\ with\ acc.status = "accepted"$

Since we have $acc \in replies$, it means that $\exists p \in Q, \exists b_M < b, vote(D_M, c, p, b_M)$. By property 2, we have $\forall b'' < b_M,\ (committable(D'', c, Q'', b'') \implies D'' = D_M)$. Hence by induction, as we choose $D = D_M = D''$, it verifies the induction hypothesis.

Case 2.iii. $\forall msg \in replies, msg.status \notin \{"accepted", "committed", "executed"\}$

Then if $committable(D', c, Q', b')$, necessarily b'=0 by definition of *pre-accept*. Let's consider $R = Q \cap Q'$. By definition of a quorum, $R \neq \emptyset$.

Case 2.iii.a. $\forall p, q \in R,\ p.deps(c) = q.deps(c) = D'$

Therefore we choose $D = D'$.

Case 2.iii.b. $\exists p, q \in R,\ p.deps(c) \neq q.deps(c)$

It is in contradiction with $committable(D', c, Q', b')$. Hence, such b' doest not exist and there is no constraint on D.

Therefore the induction hypothesis is verified in any case, hence comes the result. ■

Invariant 1.

$$committed(D, c) \land committed(D', c) \implies D = D'$$

Proof. Direct by combining properties 1 and 3. ■

Definition 7 (Sent)**.**

$$\exists m \in Sent \iff \Diamond(\exists m \in sentMsg)$$

Definition 8 (Seen)**.**

$$\begin{aligned} seen(D, c, b, p) \triangleq \Diamond(&\exists m \in Sent,\ m.type \in \{"preaccept", "preaccept - reply", "try - preaccept - reply"\} \\ &\land m.src = p \\ &\land m.cmd = c \\ &\land (m.type \neq try - preaccept - reply \implies m.deps = D) \\ &\land (m.type = preaccept \implies b = 0 \lor m.ballot = b)) \end{aligned}$$

Property 4.

$$vote(D, c, p, b) \implies \exists Q',\ \forall p \in Q',\ seen(D_p, c, b, p) \wedge (D = \bigcup_{p \in Q'} D_p)$$

Proof. ∎

Property 5.

$$commitable(D, c, Q, b) \implies \exists Q',\ \forall p \in Q',\ seen(D_p, c, p, b) \wedge (D = \bigcup_{p \in Q'} D_p)$$

Proof. By induction on b, the ballot number.

<u>Induction hypothesis:</u>

$commitable(D, c, Q, b) \implies \exists Q',\ \forall p \in Q',\ seen(D_p, c, p, b) \wedge (D = \bigcup_{p \in Q'} D_p)$

<u>Base case:</u> $b = 0$

Case 1. $preaccept(D, c, Q, 0)$
Let's consider the initial leader of the command *cleader* and $\Diamond StartPhase1(c, cleader, Q, i, b, \{\})$. The message is different depending on the nature of $p \in Q$:

Case 1.i. $p = cleader$
Therefore p sends a message m with $m.src = p, m.cmd = c, m.deps = \{rec.inst : rec \in cmdLog[cleader]\}$ and $m.type = preaccept$.

Case 1.ii. $p \neq cleader$
Therefore p replies to m with m_r having $m_r.src = p, m_r.cmd = c, m_r.deps = m.deps \cup (t.inst : t \in cmdLog[p] \backslash \{m.inst\})$ and $m_r.type = preaccept - reply$.

Hence, $\exists Q,\ \forall p \in Q,\ seen(D_p, c, p, b) \wedge (D = \bigcup_{p \in Q} D_p)$

Case 2. $accept(D, c, Q, 0)$
Therefore $\Diamond Phase1Slow(cleader, i, Q)$ and as a consequence, like for the previous case, *cleader* sent a message in $\Diamond StartPhase1$ and $\forall p \in Q, p \neq cleader$, p replied in $\Diamond Phase1Reply$. Thus *cleader* sends a message m with $m.deps = \cup\{m_r.deps : m_r \in replies\}$ where replies is the union of all m_r from the previous case. Hence, by going through the two calls developed in the case above, *seen* condtion is checked. And as *cleader* proposes $D = m.deps$, we have $D = \bigcup_{p \in Q'} D_p$.

<u>Induction step:</u> $b > 0$

For c to be *committable*, there must be $\Diamond PrepareFinalize(replica, i, Q)$.

Case 1. $\exists\ com \in replies\ with\ com.status \in \{"committed", "executed"\}$
Therefore, $\exists b' < b, committable(D', c, Q', b')$. Taking $D = D'$, by induction, we have that $\exists Q',\ \forall p \in Q',\ seen(D_p, c, p, b) \wedge (D = \bigcup_{p \in Q'} D_p)$.

Case 2. $\exists\, acc \in replies \text{ with } acc.status = \text{"}accepted\text{"}$

By definition, we have $vote(D, c, p, b)$. Hence by property 4, $\exists Q'$, $\forall p \in Q'$, $seen(D_p, c, p, b) \wedge (D = \bigcup_{p \in Q'} D_p)$.

Case 3. $\forall msg \in replies, msg.status \notin \{\text{"}accepted\text{"}, \text{"}committed\text{"}, \text{"}executed\text{"}\}$

Let's define $preaccepts \triangleq \{msg \in replies : msg.status = \text{"}preaccepted\text{"}\}$.

Case 3.i. $(|preaccepts| \geq |Q|-1) \wedge (\forall m_1, m_2 \in preaccepts, m_1.deps = m_2.deps) \wedge (\forall m \in preaccepts : m.src \neq i[1])$

Hence we have $|Q|-1$ replicas p with $seen(D, c, b', p)$ without the leader, and $b' = 0$ by definition of *pre-accept*. As the initial leader picked the set of dependencies, it also had $seen(D, c, b', cleader)$. Therefore $\forall p \in Q, seen(D, c, b', p)$ and the induction hypothesis is true.

Case 3.ii. $(|Q|-1 > |preaccepts| \geq |Q|/2) \wedge (\forall m_1, m_2 \in preaccepts, m_1.deps = m_2.deps) \wedge (\forall m \in preaccepts : m.src \neq i[1])$

$committable(D, c, Q, b) \implies \Diamond FinalizeTryPreAccept(cleader, i, Q)$.

Let's define $tprs \triangleq \{msg \in sentMsg : msg.type = \text{"}try-preaccept-reply\text{"} \wedge msg.dst = cleader \wedge msg.inst = i \wedge msg.ballot = rec.ballot\}$.

To be committable, we have either:

- $\forall tpr \in tprs : tpr.status = \text{"}OK\text{"}$ which means that $\forall p \in Q, seen(D, c, p, b)$ and D is chosen to be committable.
- $\exists tpr \in tprs : tpr.status \in \{\text{"}accepted\text{"}, \text{"}committed\text{"}, \text{"}executed\text{"}\}$, hence we initiate $StartPhase1$. We deal with this case in the following case.

Case 3.iii. Remaining cases

In these cases, $committable \implies StartPhase1$. It can be reduced to the base case with $Phase1Slow$ (we can only call *StartPhase1* on a slow quorum at a ballot $b > 0$). As the base case verifies the induction hypothesis, this case verifies it too.

Therefore the induction hypothesis is verified in any case, hence comes the result.

∎

Property 6.

$$\Box(seen(\_, c, p, \_)) \implies \Box(c \in cmdLog[p])$$

Proof. ∎

Property 7.

$$\Box(seen(D, c, p, b)) \implies \Box(p.deps[c] \subseteq D)$$

Proof. ∎

Property 8.

$$seen(D, c, p, b) \wedge seen(D', c', p, b') \implies c \in D' \vee c' \in D$$

Proof. Let's suppose, without loss of generality, $seen(D_p, c, p, b) < seen(D'_p, c', p, b')$. By property 6, let's consider the first time when $\Box(c \in cmdLog[p])$. Then let's consider the first time when $\Box seen(D'_p, c', p, b')$. It can happen in 3 different cases according to the type of message:

Case 1. $m.type = "preaccept"$

It means that p is the leader of the command. As $c \in cmdLog[p]$, necessarily $c \in D'_p$ because p sends a message m with $m.deps = \{rec.inst : rec \in cmdLog[p]\}$.

Case 2. $m.type = "preaccept - reply"$

It means that p is a follower. As $c \in cmdLog[p]$, necessarily $c \in D'_p$ because p receives a message msg and reply with a message m with $m.deps = msg.deps \cup \{t.inst : t \in cmdLog[p]\} \backslash \{msg.inst\}$.

Case 3. $m.type = "try - preaccept - reply"$

■

Invariant 2.

$$committed(D, c) \wedge committed(D', c') \implies c \in D' \text{ } or \text{ } c' \in D$$

Proof. Direct by combining properties 1, 5 and 8. ■