Egalitarian Paxos: Proof of correctness

Alexis Le Glaunec

January 9, 2021

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

$$\begin{aligned} preaccept(D, c, Q, b) \triangleq \forall p \in Q, \Diamond(&p.deps(c) = D \\ &\wedge p.vbal = b = 0 \\ &\wedge p.status(c) = \text{"}preaccepted\text{"}) \end{aligned}$$

Definition 2 (Accept)**.**

$$\begin{aligned} accept(D, c, Q, b) \triangleq \forall p \in Q, \Diamond(&p.deps(c) = D \\ &\wedge p.vbal = b \\ &\wedge p.status(c) = \text{"}accepted\text{"}) \end{aligned}$$

Definition 3 (Vote)**.**

$$\begin{aligned} vote(D, c, p, b) \triangleq \Diamond(&p.deps(c) = D \\ &\wedge p.vbal = b > 0 \\ &\wedge p.status(c) = \text{"}accepted\text{"}) \end{aligned}$$

Definition 4 (Committable)**.**

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

Definition 5 (Committed)**.**

$$\begin{aligned} committed(D, c) \triangleq \exists p \in Replicas, \Diamond(&p.deps(c) = D \\ &\wedge p.status(c) = \text{"}committed\text{"}) \end{aligned}$$

Definition 6 (Executed)**.**

$$\begin{aligned} executed(D, c) \triangleq \exists p \in Replicas, \Diamond(&p.deps(c) = D \\ &\wedge p.status(c) = \text{"}executed\text{"}) \end{aligned}$$

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 $deps(c) = D \wedge status(c) =$ "$committed$". Therefore $committed(D, c) \implies \Diamond Phase1Fast(cleader, i, Q) \vee \Diamond Phase2Finalize(cleader, i, Q)$.

Case 1. $\Diamond Phase1Fast(cleader, i, Q)$
$\Diamond Phase1Fast(cleader, i, Q) \implies \Diamond StartPhase1(c, cleader, Q, i, 0, oldMsg)$ 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 $\Diamond Phase1Fast \implies \forall p \in q, p.status(c) = "preaccepted"$, we conclude that $preaccept(D, c, Q, b)$ and therefore $committable(D, c, Q, b)$.

Case 2. $\Diamond Phase2Finalize(cleader, i, Q)$
This is a similar case, replacing $p.status = "preaccepted"$ with $p.status = "accepted"$ and $\Diamond Phase2Finalize \implies \Diamond Phase1Slow(cleader, i, Q)$. 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.
<u>Induction hypothesis:</u>
$vote(D, c, p, b) \implies \forall b' < b,\ (committable(D', c, Q', b') \implies D = D')$

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 $\Diamond PrepareFinalize$ 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 $b = 0$, $\exists! cleader, \Diamond Propose(c, cleader)$ 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.

<u>Induction hypothesis:</u>

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

By definition of the induction hypothesis, $b > 0$ and the base case is true. In the recovery step, *committable* is accessible only through $\lozenge PrepareFinalize$. 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.

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

Proof. Direct by combining properties 1 and 3. ■

Definition 7 (Sent)**.**

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

Definition 8 (Seen)**.**

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

Property 4.

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

Proof. We split into 2 cases depending on *preaccept* or *accept*.

Case 1. $preaccept(D, c, Q, 0)$
$preaccept(D, c, Q, 0) \implies \Diamond StartPhase1(c, cleader, Q, i, b, \{\})$.
Let's consider the initial leader of the command *cleader*. 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. $accept(D, c, Q, b)$
In that case, $vote(D, c, cleader, b) \implies accept(D, c, Q, b)$.
By induction on b, the ballot number.

Induction hypothesis:

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

Base case: $b = 0$

$vote(D, c, cleader, 0) \implies \Diamond Phase1Slow(cleader, i, Q)$.
As a consequence, like for the $preaccept(D, c, Q, 0)$ 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 $D = \bigcup_{p \in Q'} D_p$.

Induction step: $b > 0$

$vote(D, c, cleader, b) \wedge (b > 0) \implies \Diamond PrepareFinalize(cleader, i, Q)$.

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. $\exists\ acc \in replies\ with\ acc.status = "accepted"$
By induction, the result holds.

Case 2.ii. $\forall msg \in replies, msg.status \notin \{"accepted", "committed", "executed"\}$
Let's define $preaccepts \triangleq \{msg \in replies : msg.status = "preaccepted"\}$.

Case 2.ii.a. $(|preaccepts| \geq |Q| - 1) \wedge (\forall m_1, m_2 \in preaccepts, m_1.deps = m_2.deps) \wedge (\forall m \in preaccepts : m.src \neq i[1])$
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 2.ii.b. $(|Q|-1 > |preaccepts| \geq |Q|/2) \wedge (\forall m_1, m_2 \in preaccepts, m_1.deps = m_2.deps) \wedge (\forall m \in preaccepts : m.src \neq i[1])$
$committable(D, c, Q, b) \implies \Diamond FinalizeTryPreAccept(cleader, i, Q)$.
Let's define $tprs \triangleq \{msg \in sentMsg : msg.type = "try-preaccept-reply" \wedge msg.dst = cleader \wedge msg.inst = i \wedge msg.ballot = rec.ballot\}$.
To be committable, we have either:

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

Therefore the induction hypothesis is verified in any case.
Hence as $committable(D, c, Q, b) \cap vote(D, c, cleader, b) \implies accept(D, c, Q, b)$, we have $accept(D, c, Q, b) \implies \exists Q',\ \forall p \in Q',\ seen(D_p, c, p, b) \wedge (D = \bigcup_{p \in Q'} D_p)$.

As $committable(D, c, Q, b) \implies preaccept(D, c, Q, b) \cup accept(D, c, Q, b)$, hence comes the result. ■

Property 5.

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

Proof. For the 3 sorts of message type, the message is sent after modifying $cmdLog[p]$ accordingly to the message content, hence the result. ■

Property 6.

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

Proof. ■

Property 7.

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

Proof. Let's suppose, without loss of generality, $seen(D, c, p, b) < seen(D', c', p, b')$. By property 6, let's consider a time when $\Box(c \in cmdLog[p])$. Then let's consider the first time when $\Box seen(D', c', p, b')$. 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 $c \in D'$ 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 $c \in D'$ 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. $m.type = "try - preaccept - reply"$
If $c \in D'$, the set proposed by the leader of the recovery, then the result is verified. Otherwise, if $c \notin D'$ then $c' \in p.deps[c]$ and as $seen(D, c, p, b)$, by property 8 we conclude that $p.deps[c] \subseteq D$ hence $c' \in D$.

■

Invariant 2.

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

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