PBFT [23] is a partially synchronous protocol for Byzantine state machine replication.

Below we informally describe the protocol for the case when $n = 3f + 1$. It is not hard to modify the protocol for the more general case $n > 3f + 1$. In our description, we assume transactions are proposed in units called batches.

Normal-case operations. We first describe the normal-case operations of the PBFT protocol, where all messages are signed by the sender.

1. The leader of the current view proposes a tuple ("propose", v, ℓ, batch) to all nodes where v denotes the view number and ℓ denotes the sequence number.
2. When an honest node hears ("propose", v, ℓ, batch), if it has not sent a prepare message for (v, ℓ), it multicasts ("prepare", v, ℓ, batch).
3. When an honest node collects ("prepare", v, ℓ, batch) from $2f + 1$ distinct nodes for the same $(v, \ell, \mathsf{batch})$ tuple, it multicasts ("commit", v, ℓ, batch). Further, the honest node now considers $\mathsf{prepared}(v, \ell, \mathsf{batch}) := 1$.
4. When an honest node first collects ("commit", v, ℓ, batch) from $2f + 1$ distinct nodes for the same $(v, \ell, \mathsf{batch})$ tuple; or when it first collects ("commited", v, ℓ, batch) from $f + 1$ distinct nodes for the same $(v, \ell, \mathsf{batch})$ tuple: the node considers $\mathsf{lcommitted}(v, \ell, \mathsf{batch}) := 1$ and multicasts ("commited", v, ℓ, batch). Here $\mathsf{lcommitted}$ is short for "locally committed".

The normal-case protocol satisfies the following important properties:

- *Agreement.* If two honest nodes each believes that $\mathsf{prepared}(v, \ell, \mathsf{batch}) := 1$ and $\mathsf{prepared}(v, \ell, \mathsf{batch}') := 1$ respectively, then $\mathsf{batch} = \mathsf{batch}'$.
- *Liveness under an honest leader.* If the leader is honest and no honest node has timed out since start of the latest view, then any batch submitted by an honest node will be locally committed by all honest nodes in $O(1)$ atomic time steps.
- *Ample proofs of preparedness.* If at least one honest node considers $\mathsf{lcommitted}(v, \ell, \mathsf{batch}) := 1$, then at least $f + 1$ honest node considers $\mathsf{prepared}(v, \ell, \mathsf{batch}) := 1$.
  If an honest node believes that $\mathsf{prepared}(v, \ell, \mathsf{batch}) := 1$, then it can produce $2f+1$ signed prepare messages that led to this belief. We refer to the collection of these $2f + 1$ prepare messages a *proof-of-preparedness*.

Notice that an immediate corrolary of the agreement property is that if two honest nodes each believes that $\mathsf{lcommitted}(v, \ell, \mathsf{batch}) := 1$ and $\mathsf{lcommitted}(v, \ell, \mathsf{batch}') := 1$ respectively, then $\mathsf{batch} = \mathsf{batch}'$. However, the normal-case operation does not guarantee, under a potentially corrupt leader, that if one honest node thinks $\mathsf{lcommitted}(v, \ell, \mathsf{batch}) := 1$, other honest nodes will necessarily think $\mathsf{lcommitted}(v, \ell, \mathsf{batch}) := 1$. This therefore motivates the view change protocol.

View change. The normal-case protocol alone does not guarantee liveness when the leader is corrupt. To guarantee liveness even when the leader is corrupt, a view change protocol is invoked upon timeouts.

Henceforth, we assume that a node's output log consists of the maximal sequence of locally committeed batches with increasing and consecutive sequence numbers.

- If a node observes some transaction but the transaction does not get included in the output log after a certain timeout, the node will complain and request a view change. When announcing a view change request, an honest node attaches a proof-of-preparedness for every batch it is prepared for. The same timeout parameter used for a while, and then if after n view changes, things are still bad, then everyone doubles their timeout value. In the partially synchronous model, when the timeout backs off to $\Theta(\delta)$ and the leader is honest, liveness ensues. It is not hard to see that this allows us to ensure $O(n\delta)$ worst-case response time where δ is the actual maximum network delay.
- If an honest node hears $f+1$ valid view change requests for a new view v', it will echo the view change request by multicasting a view change message itself for view v', and including in the view change message a proof-of-preparedness for any batch it is prepared for. This step is necessary for ensuring liveness of the view change: recall that in the previous view, if at least $f+1$ honest nodes locally committed some $(v, \ell, \mathsf{batch})$, then all honest nodes will soon commit $(v, \ell, \mathsf{batch})$. However, it is possible that, say, only f honest nodes locally committed some $(v, \ell, \mathsf{batch})$ — in this case, $f+1$ nodes will complain. But the f honest nodes who locally committed $(v, \ell, \mathsf{batch})$ may be happy and do not initiate a complaint. Thus, we require that an honest node also complains if at least $f+1$ complaints (i.e., view change requests) have been received — among these $f+1$ complaints, at least one must come from an honest node. Note also that when things are good, the adversary cannot hamper progress by forcing view changes since the adversary controls only f nodes.
- When the leader for the new view v' collects $2f+1$ valid view change requests, the set of $2f+1$ valid view change requests together form a *new-view* message. The leader then proposes the new-view message to all nodes. When an honest node receives the new-view message, For every $(v, \ell, \mathsf{batch})$ with a valid proof-of-preparedness contained in the new-view message, the node acts as if it has just received a ("propose", v', ℓ, batch) message, therefore multicasts a prepare message for the tuple, and continues as in the normal-case operations.

Due to the "ample proofs of preparedness" property of the normal-case operation, the following property holds: If an honest node believes that $\mathsf{lcommitted}(v, \ell, \mathsf{batch}) = 1$, then at least one valid proof-of-preparedness will be included in any valid new-view message. This ensures that if at least one honest node believes that $\mathsf{lcommitted}(v, \ell, \mathsf{batch}) = 1$, the tuple $(v, \ell, \mathsf{batch})$ is guaranteed to carry over to the new view, and therefore if other honest nodes locally commits $(v, \ell, \mathsf{batch}')$ in the new view, it holds that $\mathsf{batch} = \mathsf{batch}'$.

Finally, as long as the new leader is honest and no honest node has timed out yet in the new view, then liveness ensues for the new view.