MODULE *ZenWithTerms*

EXTENDS *Naturals*, *FiniteSets*, *Sequences*, *TLC*

CONSTANTS *Values*

CONSTANTS *Nodes*

CONSTANTS
Join,
PublishRequest,
PublishResponse,
Commit

VARIABLE *messages*

VARIABLE *descendant*

VARIABLE *initialConfiguration*
VARIABLE *initialValue*
VARIABLE *initialAcceptedVersion*

VARIABLE *currentTerm*
VARIABLE *lastCommittedConfiguration*
VARIABLE *lastAcceptedTerm*
VARIABLE *lastAcceptedVersion*
VARIABLE *lastAcceptedValue*
VARIABLE *lastAcceptedConfiguration*
VARIABLE *joinVotes*
VARIABLE *startedJoinSinceLastReboot*
VARIABLE *electionWon*
VARIABLE *lastPublishedVersion*
VARIABLE *lastPublishedConfiguration*
VARIABLE *publishVotes*

$Terms \triangleq Nat$

$Versions \triangleq Nat$

$ValidConfigs \triangleq \text{SUBSET } (Nodes) \setminus \{\{\}\}$

$InitialVersions \triangleq Nat$

$IsQuorum(votes, config) \triangleq Cardinality(votes \cap config) * 2 > Cardinality(config)$

$IsElectionQuorum(n, votes) \triangleq$
$\land IsQuorum(votes, lastCommittedConfiguration[n])$
$\land IsQuorum(votes, lastAcceptedConfiguration[n])$

$IsPublishQuorum(n, votes) \triangleq$
$\land IsQuorum(votes, lastCommittedConfiguration[n])$
$\land IsQuorum(votes, lastPublishedConfiguration[n])$

$Init \triangleq \land messages = \{\}$
$\land descendant = \{\}$
$\land initialConfiguration \in ValidConfigs$
$\land initialValue \in Values$
$\land initialAcceptedVersion \in [Nodes \rightarrow InitialVersions]$
$\land currentTerm = [n \in Nodes \mapsto 0]$
$\land lastCommittedConfiguration = [n \in Nodes \mapsto \{\}]$ empty *config*
$\land lastAcceptedTerm = [n \in Nodes \mapsto 0]$
$\land lastAcceptedVersion = initialAcceptedVersion$
$\land lastAcceptedValue \in \{[n \in Nodes \mapsto v] : v \in Values\}$ all agree on initial value
$\land lastAcceptedConfiguration = [n \in Nodes \mapsto lastCommittedConfiguration[n]]$
$\land joinVotes = [n \in Nodes \mapsto \{\}]$
$\land startedJoinSinceLastReboot = [n \in Nodes \mapsto \text{FALSE}]$
$\land electionWon = [n \in Nodes \mapsto \text{FALSE}]$
$\land lastPublishedVersion = [n \in Nodes \mapsto 0]$
$\land lastPublishedConfiguration = [n \in Nodes \mapsto lastCommittedConfiguration[n]]$
$\land publishVotes = [n \in Nodes \mapsto \{\}]$

$SetInitialState(n) \triangleq$
$\land lastAcceptedConfiguration[n] = \{\}$ not already bootstrapped

$\land\ Assert(lastAcceptedTerm[n] = 0,\ \text{“lastAcceptedTerm should be 0”})$

$\land\ Assert(lastCommittedConfiguration[n] = \{\},\ \text{“lastCommittedConfiguration should be empty”})$

$\land\ Assert(lastPublishedVersion[n] = 0,\ \text{“lastPublishedVersion should be 0”})$

$\land\ Assert(lastPublishedConfiguration[n] = \{\},\ \text{“lastPublishedConfiguration should be empty”})$

$\land\ Assert(electionWon[n] = \text{FALSE},\ \text{“electionWon should be FALSE”})$

$\land\ Assert(joinVotes[n] = \{\},\ \text{“joinVotes should be empty”})$

$\land\ Assert(publishVotes[n] = \{\},\ \text{“publishVotes should be empty”})$

$\land\ lastAcceptedConfiguration' = [lastAcceptedConfiguration\ \text{EXCEPT}\ ![n] = initialConfiguration]$

$\land\ lastAcceptedValue' = [lastAcceptedValue\ \text{EXCEPT}\ ![n] = initialValue]$

$\land\ lastCommittedConfiguration' = [lastCommittedConfiguration\ \text{EXCEPT}\ ![n] = initialConfiguration]$

$\land\ Assert(lastAcceptedTerm[n] = 0,\ \text{“lastAcceptedTerm should be 0”})$

$\land\ Assert(lastAcceptedConfiguration'[n] \neq \{\},\ \text{“lastAcceptedConfiguration should be non-empty”})$

$\land\ Assert(lastCommittedConfiguration'[n] \neq \{\},\ \text{“lastCommittedConfiguration should be non-empty”})$

$\land\ \text{UNCHANGED}\ \langle descendant,\ initialConfiguration,\ initialValue,\ messages,\ lastAcceptedTerm,\ lastAcceptedVers$
$lastPublishedVersion,\ lastPublishedConfiguration,\ electionWon,\ joinVotes,\ publishVotes,$
$startedJoinSinceLastReboot,\ currentTerm,\ initialAcceptedVersion\rangle$

$HandleStartJoin(n,\ nm,\ t)\ \triangleq$

$\land\ t > currentTerm[n]$

$\land\ \text{LET}$

$joinRequest\ \triangleq\ [method \mapsto Join,$

$source \mapsto n,$

$dest \mapsto nm,$

$term \mapsto t,$

$laTerm \mapsto lastAcceptedTerm[n],$

$laVersion \mapsto lastAcceptedVersion[n]]$

IN

$\land\ currentTerm' = [currentTerm\ \text{EXCEPT}\ ![n] = t]$

$\land\ lastPublishedVersion' = [lastPublishedVersion\ \text{EXCEPT}\ ![n] = 0]$

$\land\ lastPublishedConfiguration' = [lastPublishedConfiguration\ \text{EXCEPT}\ ![n] = lastAcceptedConfiguration[n]]$

$\land\ startedJoinSinceLastReboot' = [startedJoinSinceLastReboot\ \text{EXCEPT}\ ![n] = \text{TRUE}]$

$\land\ electionWon' = [electionWon\ \text{EXCEPT}\ ![n] = \text{FALSE}]$

$\land\ joinVotes' = [joinVotes\ \text{EXCEPT}\ ![n] = \{\}]$

$\land\ publishVotes' = [publishVotes\ \text{EXCEPT}\ ![n] = \{\}]$

$\land\ messages' = messages \cup \{joinRequest\}$

$\land\ \text{UNCHANGED}\ \langle lastCommittedConfiguration,\ lastAcceptedConfiguration,\ lastAcceptedVersion,$
$lastAcceptedValue,\ lastAcceptedTerm,\ descendant,\ initialConfiguration,\ initialValue,\ initialAcceptedVersion\rangle$

$HandleJoin(n,\ m)\ \triangleq$

$\land\ m.method = Join$

$\land\ m.term = currentTerm[n]$

$\land\ startedJoinSinceLastReboot[n]$

```
$\land \lor m.laTerm < lastAcceptedTerm[n]$
$\lor \land m.laTerm = lastAcceptedTerm[n]$
$\land m.laVersion \leq lastAcceptedVersion[n]$
$\land lastAcceptedConfiguration[n] \neq \{\}$ must be bootstrapped
$\land joinVotes' = [joinVotes$ EXCEPT $![n] = @ \cup \{m.source\}]$
$\land electionWon' = [electionWon$ EXCEPT $![n] = IsElectionQuorum(n, joinVotes'[n])]$
$\land$ IF $electionWon[n] = $ FALSE $\land electionWon'[n]$
THEN

$\land lastPublishedVersion' = [lastPublishedVersion$ EXCEPT $![n] = lastAcceptedVersion[n]]$
ELSE
UNCHANGED $\langle lastPublishedVersion \rangle$
$\land$ UNCHANGED $\langle lastCommittedConfiguration, currentTerm, publishVotes, messages, descendant,$
$lastAcceptedVersion, lastAcceptedValue, lastAcceptedConfiguration,$
$lastAcceptedTerm, startedJoinSinceLastReboot, lastPublishedConfiguration,$
$initialConfiguration, initialValue, initialAcceptedVersion \rangle$


$HandleClientValue(n, t, v, val, cfg) \triangleq$
$\land electionWon[n]$
$\land lastPublishedVersion[n] = lastAcceptedVersion[n]$ means we have the last published value / *config* (useful for *CAS* op
$\land t = currentTerm[n]$
$\land v > lastPublishedVersion[n]$
$\land cfg \neq lastAcceptedConfiguration[n] \Rightarrow lastCommittedConfiguration[n] = lastAcceptedConfiguration[n]$ only al
$\land IsQuorum(joinVotes[n], cfg)$ only allow reconfiguration if we have a quorum of (join) votes for the new *config*
$\land$ LET
$publishRequests \triangleq \{[method \mapsto PublishRequest,$
$source \mapsto n,$
$dest \mapsto ns,$
$term \mapsto t,$
$version \mapsto v,$
$value \mapsto val,$
$config \mapsto cfg,$
$commConf \mapsto lastCommittedConfiguration[n]] : ns \in Nodes\}$
$newEntry \triangleq [prevT \mapsto lastAcceptedTerm[n],$
$prevV \mapsto lastAcceptedVersion[n],$
$nextT \mapsto t,$
$nextV \mapsto v]$
$matchingElems \triangleq \{e \in descendant :$
$\land e.nextT = newEntry.prevT$
$\land e.nextV = newEntry.prevV\}$
$newTransitiveElems \triangleq \{[prevT \mapsto e.prevT,$
$prevV \mapsto e.prevV,$
$nextT \mapsto newEntry.nextT,$
$nextV \mapsto newEntry.nextV] : e \in matchingElems\}$
```

```
IN
$\land\ descendant' = descendant \cup \{newEntry\} \cup newTransitiveElems$
$\land\ lastPublishedVersion' = [lastPublishedVersion$ EXCEPT $![n] = v]$
$\land\ lastPublishedConfiguration' = [lastPublishedConfiguration$ EXCEPT $![n] = cfg]$
$\land\ publishVotes' = [publishVotes$ EXCEPT $![n] = \{\}]$   $publishVotes$ are only counted per publish version
$\land\ messages' = messages \cup publishRequests$
$\land$ UNCHANGED $\langle startedJoinSinceLastReboot, lastCommittedConfiguration, currentTerm, electionWon,$
$lastAcceptedVersion, lastAcceptedValue, lastAcceptedTerm, lastAcceptedConfiguration,$
$joinVotes, initialConfiguration, initialValue, initialAcceptedVersion\rangle$


$HandlePublishRequest(n, m) \triangleq$
$\land\ m.method = PublishRequest$
$\land\ m.term = currentTerm[n]$
$\land\ (m.term = lastAcceptedTerm[n]) \Rightarrow (m.version > lastAcceptedVersion[n])$
$\land\ lastAcceptedTerm' = [lastAcceptedTerm$ EXCEPT $![n] = m.term]$
$\land\ lastAcceptedVersion' = [lastAcceptedVersion$ EXCEPT $![n] = m.version]$
$\land\ lastAcceptedValue' = [lastAcceptedValue$ EXCEPT $![n] = m.value]$
$\land\ lastAcceptedConfiguration' = [lastAcceptedConfiguration$ EXCEPT $![n] = m.config]$
$\land\ lastCommittedConfiguration' = [lastCommittedConfiguration$ EXCEPT $![n] = m.commConf]$
$\land$ LET
$response \triangleq [method \mapsto PublishResponse,$
$source \mapsto n,$
$dest \mapsto m.source,$
$term \mapsto m.term,$
$version \mapsto m.version]$
IN
$\land\ messages' = messages \cup \{response\}$
$\land$ UNCHANGED $\langle startedJoinSinceLastReboot, currentTerm, descendant, lastPublishedConfiguration,$
$electionWon, lastPublishedVersion, joinVotes, publishVotes, initialConfiguration,$
$initialValue, initialAcceptedVersion\rangle$


$HandlePublishResponse(n, m) \triangleq$
$\land\ m.method = PublishResponse$
$\land\ electionWon[n]$
$\land\ m.term = currentTerm[n]$
$\land\ m.version = lastPublishedVersion[n]$
$\land\ publishVotes' = [publishVotes$ EXCEPT $![n] = @ \cup \{m.source\}]$
$\land$ IF
$IsPublishQuorum(n, publishVotes'[n])$
 THEN
LET
$commitRequests \triangleq \{[method \mapsto Commit,$
$source \mapsto n,$
```

$dest \mapsto ns,$
$term \mapsto currentTerm[n],$
$version \mapsto lastPublishedVersion[n]] : ns \in Nodes\}$
IN
$\land\ messages' = messages \cup commitRequests$
ELSE
UNCHANGED $\langle messages \rangle$
$\land$ UNCHANGED $\langle startedJoinSinceLastReboot,\ lastCommittedConfiguration,\ currentTerm,\ electionWon,\ descen$
$lastAcceptedVersion,\ lastAcceptedValue,\ lastAcceptedTerm,\ lastAcceptedConfiguration,$
$lastPublishedVersion,\ lastPublishedConfiguration,\ joinVotes,\ initialConfiguration,$
$initialValue,\ initialAcceptedVersion \rangle$

$HandleCommit(n,\ m)\ \triangleq$
$\land\ m.method = Commit$
$\land\ m.term = currentTerm[n]$
$\land\ m.term = lastAcceptedTerm[n]$
$\land\ m.version = lastAcceptedVersion[n]$
$\land\ (electionWon[n] \Rightarrow lastAcceptedVersion[n] = lastPublishedVersion[n])$
$\land\ lastCommittedConfiguration' = [lastCommittedConfiguration$ EXCEPT $![n] = lastAcceptedConfiguration[n]]$
$\land$ UNCHANGED $\langle currentTerm,\ joinVotes,\ messages,\ lastAcceptedTerm,\ lastAcceptedValue,\ startedJoinSinceLas$
$electionWon,\ lastAcceptedConfiguration,\ lastAcceptedVersion,\ lastPublishedVersion,\ publishVotes,$
$lastPublishedConfiguration,\ initialConfiguration,\ initialValue,\ initialAcceptedVersion \rangle$

$RestartNode(n)\ \triangleq$
$\land\ joinVotes' = [joinVotes$ EXCEPT $![n] = \{\}]$
$\land\ startedJoinSinceLastReboot' = [startedJoinSinceLastReboot$ EXCEPT $![n] =$ FALSE$]$
$\land\ electionWon' = [electionWon$ EXCEPT $![n] =$ FALSE$]$
$\land\ lastPublishedVersion' = [lastPublishedVersion$ EXCEPT $![n] = 0]$
$\land\ lastPublishedConfiguration' = [lastPublishedConfiguration$ EXCEPT $![n] = lastAcceptedConfiguration[n]]$
$\land\ publishVotes' = [publishVotes$ EXCEPT $![n] = \{\}]$
$\land$ UNCHANGED $\langle messages,\ lastAcceptedVersion,\ currentTerm,\ lastCommittedConfiguration,\ descendant,$
$lastAcceptedTerm,\ lastAcceptedValue,\ lastAcceptedConfiguration,\ initialConfiguration,$
$initialValue,\ initialAcceptedVersion \rangle$

$Next\ \triangleq$
$\lor\ \exists\, n \in Nodes : SetInitialState(n)$
$\lor\ \exists\, n,\ nm \in Nodes : \exists\, t \in Terms : HandleStartJoin(n,\ nm,\ t)$
$\lor\ \exists\, m \in messages : HandleJoin(m.dest,\ m)$
$\lor\ \exists\, n \in Nodes : \exists\, t \in Terms : \exists\, v \in Versions : \exists\, val \in Values : \exists\, vs \in ValidConfigs : HandleClientValue(n,\ t,$
$\lor\ \exists\, m \in messages : HandlePublishRequest(m.dest,\ m)$
$\lor\ \exists\, m \in messages : HandlePublishResponse(m.dest,\ m)$
$\lor\ \exists\, m \in messages : HandleCommit(m.dest,\ m)$

$\lor \exists\, n \in \mathit{Nodes} : \mathit{RestartNode}(n)$

$\mathit{SingleNodeInvariant} \triangleq$
$\forall\, n \in \mathit{Nodes} :$
$\land\ \mathit{lastAcceptedTerm}[n] \leq \mathit{currentTerm}[n]$
$\land\ \mathit{electionWon}[n] = \mathit{IsElectionQuorum}(n,\ \mathit{joinVotes}[n])$ cached value is consistent
$\land\ \text{IF}\ \ \mathit{electionWon}[n]\ \ \text{THEN}\ \ \mathit{lastPublishedVersion}[n] \geq \mathit{lastAcceptedVersion}[n]\ \ \text{ELSE}\ \ \mathit{lastPublishedVersion}[n] =$
$\land\ \mathit{electionWon}[n] \Rightarrow \mathit{startedJoinSinceLastReboot}[n]$
$\land\ \mathit{publishVotes}[n] \neq \{\} \Rightarrow \mathit{electionWon}[n]$

$\mathit{OneMasterPerTerm} \triangleq$
$\forall\, m1,\ m2 \in \mathit{messages} :$
$\land\ m1.\mathit{method} = \mathit{PublishRequest}$
$\land\ m2.\mathit{method} = \mathit{PublishRequest}$
$\land\ m1.\mathit{term} = m2.\mathit{term}$
$\Rightarrow m1.\mathit{source} = m2.\mathit{source}$

$\mathit{LogMatching} \triangleq$
$\forall\, m1,\ m2 \in \mathit{messages} :$
$\land\ m1.\mathit{method} = \mathit{PublishRequest}$
$\land\ m2.\mathit{method} = \mathit{PublishRequest}$
$\land\ m1.\mathit{term} = m2.\mathit{term}$
$\land\ m1.\mathit{version} = m2.\mathit{version}$
$\Rightarrow m1.\mathit{value} = m2.\mathit{value}$

$\mathit{CommittedPublishRequest}(mp) \triangleq$
$\land\ mp.\mathit{method} = \mathit{PublishRequest}$
$\land\ \exists\, mc \in \mathit{messages} :$
$\land\ mc.\mathit{method} = \mathit{Commit}$
$\land\ mp.\mathit{term} = mc.\mathit{term}$
$\land\ mp.\mathit{version} = mc.\mathit{version}$

$\mathit{DescendantRelationIsStrictlyOrdered} \triangleq$
$\forall\, d \in \mathit{descendant} :$
$\land\ d.\mathit{prevT} \leq d.\mathit{nextT}$
$\land\ d.\mathit{prevV} < d.\mathit{nextV}$

$\mathit{DescendantRelationIsTransitive} \triangleq$
$\forall\, d1,\ d2 \in \mathit{descendant} :$
$d1.\mathit{nextT} = d2.\mathit{prevT} \land d1.\mathit{nextV} = d2.\mathit{prevV}$
$\Rightarrow [\mathit{prevT} \mapsto d1.\mathit{prevT},\ \mathit{prevV} \mapsto d1.\mathit{prevV},\ \mathit{nextT} \mapsto d2.\mathit{nextT},\ \mathit{nextV} \mapsto d2.\mathit{nextV}] \in \mathit{descendant}$

$NewerOpsBasedOnOlderCommittedOps \triangleq$
$\forall m1, m2 \in messages :$
$\land CommittedPublishRequest(m1)$
$\land m2.method = PublishRequest$
$\land m2.term \geq m1.term$
$\land m2.version > m1.version$
$\Rightarrow [prevT \mapsto m1.term, prevV \mapsto m1.version, nextT \mapsto m2.term, nextV \mapsto m2.version] \in descendant$

$CommittedValuesDescendantsFromCommittedValues \triangleq$
$\forall m1, m2 \in messages :$
$\land CommittedPublishRequest(m1)$
$\land CommittedPublishRequest(m2)$
$\land \lor m1.term \neq m2.term$
$\lor m1.version \neq m2.version$
$\Rightarrow$
$\lor [prevT \mapsto m1.term, prevV \mapsto m1.version, nextT \mapsto m2.term, nextV \mapsto m2.version] \in descendant$
$\lor [prevT \mapsto m2.term, prevV \mapsto m2.version, nextT \mapsto m1.term, nextV \mapsto m1.version] \in descendant$

$CommittedValuesDescendantsFromInitialValue \triangleq$
$\exists v \in InitialVersions :$
$\land \exists n \in Nodes : v = initialAcceptedVersion[n]$
$\land \exists votes \in \text{SUBSET } (initialConfiguration) :$
$\land IsQuorum(votes, initialConfiguration)$
$\land \forall n \in votes : initialAcceptedVersion[n] \leq v$
$\land \forall m \in messages :$
$CommittedPublishRequest(m)$
$\Rightarrow$
$[prevT \mapsto 0, prevV \mapsto v, nextT \mapsto m.term, nextV \mapsto m.version] \in descendant$

$CommitHasQuorumVsPreviousCommittedConfiguration \triangleq$
$\forall mc \in messages : mc.method = Commit$
$\Rightarrow (\forall mprq \in messages : (\land mprq.method = PublishRequest$
$\land mprq.term = mc.term$
$\land mprq.version = mc.version)$

$\Rightarrow IsQuorum(\{mprs.source : mprs \in \{mprs \in messages : \land mprs.method = PublishResponse$
$\land mprs.term = mprq.term$
$\land mprs.version = mprq.version$
$\}\}, mprq.commConf))$

$P2bInvariant \triangleq$
$\forall mc \in messages : mc.method = Commit$
$\Rightarrow (\forall mprq \in messages : mprq.method = PublishRequest$

$\Rightarrow (mprq.term > mc.term \Rightarrow mprq.version > mc.version))$

$StateConstraint \triangleq$
$\land \forall n \in Nodes : \textsc{if}\ currentTerm[n] \leq 1\ \textsc{then}\ lastPublishedVersion[n] \leq 2\ \textsc{else}\ lastPublishedVersion[n] \leq 3$
$\land Cardinality(messages) \leq 15$